



# Developing Solutions for Microsoft Azure

Exam Ref **AZ-204**

Santiago Fernández Muñoz

# **Ref. De examen AZ-204**

## **Desarrollo de soluciones para**

## **Microsoft Azure**

**Santiago Fernández Muñoz**



# Contenido de un vistazo

## Introducción

Importante: cómo utilizar este libro para estudiar para el examen

**Capítulo 1 Desarrollar la infraestructura de Azure como solución informática de servicio**

**Capítulo 2 Desarrollar para almacenamiento de Azure**

**Capítulo 3 Implementar la seguridad de Azure**

**Capítulo 4 Supervisar, solucionar problemas y optimizar las soluciones de Azure**

**Capítulo 5 Conectarse y consumir servicios de Azure y servicios de terceros**

Índice

# Contenido

[Introducción](#)

[Organización de este libro](#)

[Certificaciones de Microsoft](#)

[Errata, actualizaciones y soporte de libros](#)

[Mantente en contacto](#)

[Importante: cómo utilizar este libro para estudiar para el examen](#)

[Capítulo 1 Desarrollar la infraestructura de Azure como solución informática de servicio](#)

[Habilidad 1.1: implementar soluciones que utilicen máquinas virtuales \(VM\)](#)

[Aprovisionar máquinas virtuales](#)

[Configurar máquinas virtuales para acceso remoto](#)

[Crear plantillas ARM](#)

[Cree imágenes de contenedor para soluciones con Docker](#)

[Publicar una imagen en Azure Container Registry](#)

[Ejecutar contenedores con Azure Container Instance](#)

[Habilidad 1.2: Crear aplicaciones web de Azure App Service](#)

[Crear una aplicación web de Azure App Service](#)

[Habilitar el registro de diagnósticos](#)

[Implementar código en una aplicación web](#)

[Configure los ajustes de la aplicación web, incluidos SSL, API y cadenas de conexión](#)

Implementar reglas de escalado automático, incluido el escalado automático programado y el escalado por métricas operativas o del sistema.

Habilidad 1.3: Implementar funciones de Azure

Implementar enlaces de entrada y salida para una función

Implementar activadores de funciones mediante el uso de operaciones de datos, temporizadores y webhooks

Implementar funciones duraderas de Azure

Resumen del capítulo

Experimento mental

Respuestas del experimento mental

Capítulo 2 Desarrollar para almacenamiento de Azure

Habilidad 2.1: Desarrollar soluciones que utilicen almacenamiento Cosmos DB

Seleccione la API adecuada para su solución

Implementar esquemas de particiones

Interactuar con los datos usando el SDK apropiado

Establecer el nivel de coherencia adecuado para las operaciones

Crear contenedores de Cosmos DB

Implemente la programación del lado del servidor, incluidos los procedimientos almacenados, los desencadenantes y las notificaciones de cambios de fuentes

Habilidad 2.2: Desarrollar soluciones que usen Blob Storage

Mover elementos en Blob Storage entre cuentas de almacenamiento o contenedores

Establecer y recuperar propiedades y metadatos

Interactuar con los datos usando el SDK apropiado

Implementar el archivo y la retención de datos

Implementar almacenamiento en caliente, frío y de archivo

Resumen del capítulo

Experimento mental

Respuestas del experimento mental

Capítulo 3 Implementar la seguridad de Azure

Habilidad 3.1: Implementar la autenticación y autorización de usuarios

Implementar la autenticación OAuth2

Cree e implemente firmas de acceso compartido

Registre aplicaciones y use Azure Active Directory para autenticar usuarios

Controlar el acceso a los recursos mediante el uso de controles de acceso basados en roles (RBAC)

Habilidad 3.2: Implementar soluciones seguras en la nube

Proteja los datos de configuración de la aplicación mediante la configuración de la aplicación y la API de KeyVault

Administre claves, secretos y certificados mediante la API de KeyVault

Implementar identidades administradas para recursos de Azure

Resumen del capítulo

Experimento mental

Respuestas del experimento mental

## Capítulo 4 Supervisar, solucionar problemas y optimizar las soluciones de Azure

Habilidad 4.1: Integrar el almacenamiento en caché y la entrega de contenido dentro de las soluciones

Desarrollar código para implementar CDN en soluciones

Configure las políticas de caducidad y caché para las cachés de FrontDoor, CDN y Redis

Almacenar y recuperar datos en Azure Redis Cache

Habilidad 4.2: Soluciones de instrumentos para respaldar el monitoreo y el registro

Configurar la instrumentación en una aplicación o servicio mediante Application Insights

Analice los datos de registro y solucione problemas de soluciones mediante Azure Monitor

Implementar alertas y pruebas web de Application Insights

Implementar código que maneje fallas transitorias

Resumen del capítulo

Experimento mental

Respuestas del experimento mental

Capítulo 5 Conectarse y consumir servicios de Azure y servicios de terceros

Habilidad 5.1: Desarrollar una aplicación lógica de servicio de aplicaciones

Crear una aplicación lógica

Cree un conector personalizado para Logic Apps

Crea una plantilla personalizada para Logic Apps

Habilidad 5.2: Implementar la gestión de API

Crea una instancia de APIM

Configurar la autenticación para las API

Definir políticas para API

Habilidad 5.3: Desarrollar soluciones basadas en eventos

Implementar soluciones que usen Azure Event Grid

Implementar soluciones que usan Azure Notification Hubs

Implementar soluciones que usen Azure Event Hub

Habilidad 5.4: Desarrollar soluciones basadas en mensajes

Implementar soluciones que utilicen Azure Service Bus

Implementar soluciones que usen colas de Azure Queue Storage

Resumen del capítulo

Experimento mental

Respuestas del experimento mental

Índice

## Sobre el Autor



Comencé mi carrera como instructor de Linux y Windows. Al mismo tiempo, también comencé a aprender lenguajes de programación de secuencias de comandos como bash y VBS que eran útiles para mi trabajo. Durante ese período de mi carrera, me di cuenta de que los lenguajes de scripting eran útiles, pero no eran suficientes para satisfacer todas mis necesidades, así que comencé a aprender otros lenguajes como Java, PHP y finalmente C #.

He trabajado como consultor de tecnologías de Microsoft durante los últimos 14 años, y durante los últimos 6 años, he consultado sobre tecnologías relacionadas con Azure. He participado en diferentes tipos de proyectos, sirviendo en una variedad de capacidades desde desarrollador .NET hasta arquitecto de soluciones. Ahora estoy enfocado en desarrollar soluciones de IoT industriales personalizadas para mi empresa y mis clientes.

# Introducción

---

La mayoría de los libros adoptan un enfoque de muy bajo nivel, que le enseña cómo usar clases individuales y realizar tareas detalladas. A través de este libro, revisamos las principales tecnologías que ofrece Microsoft para implementar diferentes tipos de soluciones en Azure. Desde los enfoques más clásicos y conservadores que usan máquinas virtuales de Azure hasta las últimas tecnologías, implementando patrones basados en eventos o mensajes con Azure Event Grid o Azure Service Bus, este libro revisa los conceptos básicos para desarrollar la mayoría de los tipos de soluciones usando los servicios de Azure. El libro también proporciona ejemplos de código para ilustrar cómo implementar la mayoría de los conceptos cubiertos en las diferentes secciones.

Este libro está dirigido a aquellos profesionales que planean aprobar el examen AZ-204. Este libro cubre todas las áreas temáticas principales que se encuentran en el examen, pero no cubre todas las preguntas del examen. Solo el equipo de examen de Microsoft tiene acceso a las preguntas del examen, y Microsoft agrega periódicamente nuevas preguntas al examen, lo que hace imposible cubrir preguntas específicas. Debe considerar este libro como un complemento de su experiencia relevante en el mundo real y otros materiales de estudio. Si encuentra un tema en este libro con el que no se siente completamente cómodo, use la opción "¿Necesita más revisión?" enlaces en el texto para encontrar más información y tomarse el tiempo para investigar y estudiar el tema. Hay gran información disponible en MSDN y TechNet y en blogs y foros.

## ORGANIZACIÓN DE ESTE LIBRO

---

Este libro está organizado por la lista de "Habilidades medidas" publicada para el examen. La lista "Habilidades medidas" está disponible para cada examen en el sitio web de Microsoft Learn: <http://aka.ms/examlist>. Cada capítulo de este libro corresponde a un área temática principal de la lista, y las tareas técnicas de cada área temática determinan la organización de un capítulo. Si un examen cubre seis áreas temáticas principales, por ejemplo, el libro contiene seis capítulos.

## CERTIFICACIONES DE MICROSOFT

---

Las certificaciones de Microsoft lo distinguen al demostrar su dominio de un amplio conjunto de habilidades y experiencia con los productos y tecnologías actuales de Microsoft. Los exámenes y las certificaciones correspondientes se desarrollan para validar su dominio de las competencias críticas a medida que diseña y desarrolla, o implementa y brinda soporte, soluciones con productos y tecnologías de Microsoft tanto en las instalaciones como en la nube. La certificación aporta una variedad de beneficios para el individuo y para los empleadores y las organizaciones.

### ***Más información Todas las certificaciones de Microsoft***

Para obtener información sobre las certificaciones de Microsoft, incluida una lista completa de las certificaciones disponibles, visite <http://www.microsoft.com/learn>.

¡Vuelve a menudo para ver qué hay de nuevo!

## ERRATA, ACTUALIZACIONES Y SOPORTE DE LIBROS

---

Hemos hecho todo lo posible para garantizar la precisión de este libro y su contenido complementario. Puede acceder a las actualizaciones de este libro, en forma de lista de erratas enviadas y sus correcciones relacionadas, en

<MicrosoftPressStore.com/ExamRefAZ204/errata> .

Si descubre un error que aún no está en la lista, envíenoslo en la misma página.

Para obtener ayuda e información adicional sobre libros, visite

<http://www.MicrosoftPressStore.com/Support> .

Tenga en cuenta que el soporte de productos para software y hardware de Microsoft no se ofrece a través de las direcciones anteriores. Para obtener ayuda con el software o hardware de Microsoft, vaya a <http://support.microsoft.com> .

## MANTENTE EN CONTACTO

---

¡Sigamos con la conversación! Estamos en Twitter:

<http://twitter.com/MicrosoftPress>

## Importante: cómo utilizar este libro para estudiar para el examen

Los exámenes de certificación validan su experiencia en el trabajo y su conocimiento del producto. Para medir su preparación para realizar un examen, utilice esta Referencia de examen para comprobar su comprensión de las habilidades evaluadas por el examen. Determina los temas que conoces bien y las áreas en las que necesitas más experiencia. Para ayudarlo a actualizar sus habilidades en áreas específicas, también hemos proporcionado "¿Necesita más revisión?" punteros, que le dirigen a información más detallada fuera del libro.

El Examen Ref no sustituye la experiencia práctica. Este libro *no* está diseñado para enseñarle nuevas habilidades.

Le recomendamos que complete la preparación de su examen utilizando una combinación de materiales de estudio y cursos disponibles. Obtenga más información sobre la capacitación presencial disponible y encuentre cursos en línea gratuitos y eventos en vivo en <http://microsoft.com/learn>. Las pruebas de práctica oficiales de Microsoft están disponibles para muchos exámenes en <http://aka.ms/practicetests>.

Este libro está organizado por la lista de "Habilidades medidas" publicada para el examen. La lista de "Habilidades medidas" para cada examen está disponible en el sitio web de Microsoft Learn: <http://aka.ms/examlist>.

Tenga en cuenta que esta referencia de examen se basa en esta información disponible públicamente y en la experiencia del autor. Para salvaguardar la integridad del examen, los autores no tienen acceso a las preguntas del examen.

# Capítulo 1. Desarrollar la infraestructura de Azure como una solución informática de servicio

Hoy en día, la computación en la nube es una realidad consolidada que cualquier empresa o profesional debe considerar a la hora de desarrollar o mantener productos nuevos o existentes. Cuando planea desarrollar o implementar una aplicación, puede elegir entre dos modelos principales de servicios en la nube, Infraestructura como servicio (IaaS) o Plataforma como servicio (PaaS), y cada modelo tiene sus pros y sus contras. Si decide utilizar el modelo IaaS, tiene un control más granular sobre la infraestructura que respaldará su aplicación.

Sin embargo, una vez finalizada la implementación en el entorno de producción, debe mantenerla. Este mantenimiento significa que también debe asignar el presupuesto para el soporte de la infraestructura, y debe contar con personal capacitado para realizar este mantenimiento.

Gracias a las tecnologías en la nube, puede reducir drásticamente estos requisitos de implementación y planificación de la infraestructura mediante la implementación de su software en un servicio administrado conocido como Plataforma como servicio (PaaS). Hacerlo significa que solo debe preocuparse por su código y cómo interactúa con otros servicios en Azure. Los productos PaaS como Azure App Service o Azure Functions lo liberan de preocuparse por configuraciones de alta disponibilidad o tolerantes a fallas porque el servicio provisto por Azure ya administra estas cosas.

Este capítulo revisa cómo trabajar con las opciones que Azure pone a su disposición para desarrollar sus soluciones basadas en el modelo IaaS. El capítulo también cubre las soluciones PaaS que proporciona Azure, que le permiten concentrarse en su código y olvidarse de la infraestructura subyacente.

## Importante ¿Ha leído la página xvii?

Contiene información valiosa sobre las habilidades que necesita para aprobar el examen.

## **Habilidades cubiertas en este capítulo:**

- Habilidad 1.1: implementar soluciones que utilicen máquinas virtuales (VM)
- Habilidad 1.2: Crear aplicaciones web de Azure App Service
- Habilidad 1.3: Implementar funciones de Azure

## **HABILIDAD 1.1: IMPLEMENTAR SOLUCIONES QUE UTILICEN MÁQUINAS VIRTUALES (VM)**

---

Una de las principales características del modelo IaaS es el mayor nivel de control que ofrece a la hora de desplegar la infraestructura necesaria para su aplicación. Normalmente, necesita trabajar con este modelo porque necesita más control sobre los diferentes elementos de su aplicación. Con IaaS, implementa sus máquinas virtuales, donde implementará todos los componentes necesarios para su solución.

Azure le proporciona todo el hardware y la configuración subyacentes necesarios para que su máquina virtual (VM) se ejecute correctamente. Sin embargo, aún debe administrar todas las tareas administrativas relacionadas con el sistema operativo de la máquina virtual, como la instalación de actualizaciones del sistema operativo o parches de seguridad. Microsoft administra la configuración requerida para proporcionar tolerancia a fallas para el hardware físico que admite su VM. Pero si necesita que su aplicación o solución de software esté altamente disponible, debe administrar la configuración de las VM que alojan su aplicación.

### **Esta habilidad cubre cómo**

- Aprovisionar máquinas virtuales
- Configurar máquinas virtuales para acceso remoto
- Crear plantillas ARM
- Cree imágenes de contenedor para soluciones con Docker
- Publicar una imagen en Azure Container Registry
- Ejecutar contenedores con Azure Container Instance

## *Aprovisionar máquinas virtuales*

La implementación de una máquina virtual en Azure es un proceso sencillo, pero aún debe pensar en algunos puntos clave si desea lograr el mejor equilibrio entre los costos y sus requisitos. Quizás la decisión más obvia es qué sistema operativo debe utilizar. La buena noticia es que Azure es totalmente compatible con Windows, Windows Server y las principales distribuciones de Linux.

### **Nota Sistemas operativos compatibles**

Puede revisar la lista completa de sistemas operativos compatibles que se usarán en máquinas virtuales de Azure en las siguientes direcciones URL:

- Windows: <https://support.microsoft.com/en-us/help/2721672/microsoft-server-software-support-for-microsoft-azure-virtual-machines>
- Linux: <https://docs.microsoft.com/en-us/azure/virtual-machines/linux/endorsed-distros>

Todos estos sistemas operativos Windows y Linux están preinstalados y disponibles en Azure Marketplace como imágenes de máquina virtual. Además de estas imágenes de VM predeterminadas, también encontrará otras imágenes en el mercado de otros proveedores que contienen soluciones preconfiguradas que pueden adaptarse mejor a sus necesidades.

Una vez que haya elegido su sistema operativo, debe decidir otros aspectos esenciales de la VM:

- **Nombre** Introduzca el nombre de la máquina virtual. Los nombres pueden tener hasta 15 caracteres.
- **Ubicación** Seleccione la región geográfica donde implementa su VM. Azure tiene varios centros de datos distribuidos por todo el mundo que están agrupados en regiones geográficas. La elección de la región o ubicación incorrecta puede tener efectos adversos.
- **Tamaño** Designe la cantidad de recursos que asignará a sus máquinas virtuales. Estos recursos incluyen la cantidad de memoria, la potencia de procesamiento, la cantidad de tarjetas de interfaz de red virtual (NIC) que puede conectar a su VM y la capacidad total de almacenamiento que estará disponible para su VM.
- **Límites** Cada suscripción tiene límites de cuota predeterminados. Estos límites pueden afectarlo al implementar

nuevas máquinas virtuales. De forma predeterminada, cada suscripción tiene un límite de 20 máquinas virtuales por región. Sin embargo, puede aumentar este límite poniéndose en contacto con el servicio de soporte de Azure.

- **Extensiones** Las extensiones le permiten automatizar algunas tareas o configuraciones una vez que la implementación de su máquina virtual se completa con éxito. Algunas de las extensiones más comunes son
  - Ejecutar scripts personalizados
  - Implementar y administrar configuraciones
  - Recopilar datos de diagnóstico
- **Recursos relacionados** Cuando implementa una máquina virtual, debe pensar en la cantidad y el tipo de almacenamiento, por ejemplo, si esta VM estará conectada a Internet y necesitará una IP pública o qué tipo de tráfico puede ir hacia o desde el VM. Algunos de estos recursos relacionados, descritos en la siguiente lista, son obligatorios para implementar una VM.
  - **Grupo de recursos** Cada máquina virtual debe estar contenida en un grupo de recursos. Puede crear un nuevo grupo de recursos o reutilizar uno existente.
  - **Cuenta de almacenamiento** Los discos virtuales que necesita la máquina virtual son archivos .vhf almacenados como blobs de página en una cuenta de almacenamiento. Según los requisitos de rendimiento de su máquina virtual, puede utilizar cuentas de almacenamiento estándar o premium. Si configura discos administrados al implementar una máquina virtual, Azure maneja la cuenta de almacenamiento automáticamente y no aparecerá en la configuración de la máquina virtual.
  - **Red virtual** Para poder comunicarse con el resto del mundo, su nueva máquina virtual debe estar conectada a una red virtual.
  - **Interfaz de red** Como en el mundo físico, su VM necesita una interfaz de red para conectarse a la red virtual para enviar y recibir información.

Una vez que haya reunido toda la información que necesita para implementar su VM, estará listo para la implementación. Tiene varias formas de realizar esta tarea:

- Usando el portal de Azure
- Usando PowerShell
- Uso de la CLI de Azure
- Programáticamente usando REST API o C #

En general, cuando desee implementar una nueva máquina virtual, debe seguir estos pasos:

1. Cree un grupo de recursos para la máquina virtual. También puede utilizar un grupo de recursos existente para esta máquina virtual.
2. Crea una red virtual. Si está utilizando Azure Portal, puede hacer esto mientras crea la máquina virtual. Para PowerShell y la CLI de Azure, debe especificar la red virtual. Sin embargo, si aún no existe una red virtual, se crea una automáticamente.
3. Cree una NIC virtual. Si usa Azure Portal, PowerShell o la CLI de Azure, no es necesario que lo haga porque el proceso de implementación crea automáticamente la NIC.
4. Crea la máquina virtual.

El siguiente ejemplo muestra cómo crear una aplicación de consola .NET Core simple para crear una máquina virtual:

1. Abra el código de Visual Studio. Necesita tener instalada la extensión Omnishare.
2. Crea una carpeta para tu proyecto.
3. En la ventana de la terminal, cambie el directorio de trabajo a la carpeta que creó en el paso anterior y escriba el siguiente comando.

```
nueva consola dotnet
```

4. Instale el paquete nuget Microsoft.Azure.Management.Fluent.  
[Haga clic aquí para ver la imagen del código](#)

```
dotnet agregar paquete Microsoft.Azure.Management.Fluent
```

5. Cree un archivo vacío llamado **azureauth.properties**. Agregue el contenido que se muestra en el [Listado 1-1](#) al archivo. Reemplace las variables con los valores de su suscripción de Azure.

6. Reemplace el contenido del archivo Program.cs con el contenido del [Listado 1-2](#). El [Listado 1-2](#) muestra cómo crear una máquina virtual con discos administrados en su suscripción de Azure.

**Listado 1-1** azureauth.properties

[Haga clic aquí para ver la imagen del código](#)

---

```
suscripción = <subscription-id>

client = <client-id>

key = <client-secret>

inquilino = <tenant-id>

managementURI = https://management.core.windows.net/

baseURL = https://management.azure.com/

authURL = https://login.windows.net/

graphURL = https://graph.windows.net/
```

**Listado 1-2** Program.cs

[Haga clic aquí para ver la imagen del código](#)

---

```
// dotnet core 2.2

usando System;

utilizando Microsoft.Azure.Management.Compute.Fluent;

utilizando Microsoft.Azure.Management.Compute.Fluent.Models;

utilizando Microsoft.Azure.Management.Fluent;

utilizando
Microsoft.Azure.Management.ResourceManager.Fluent;
```

```
utilizando
Microsoft.Azure.Management.ResourceManager.Fluent.Core;

espacio de nombres ch1_1_1

{

    programa de clase

    {

        static void Main (cadena [] argumentos)

        {

            // Crea el cliente de gestión. Esto se utilizará
            para todas las operaciones.

            // que realizaremos en Azure.

            var credentials =
SdkContext.AzureCredentialsFactory

                .FromFile ("./
azureauth.properties");




            var azure = Azure.Configure ()

                .WithLogLevel
(HttpLoggingDelegatingHandler.Level.Basic)

                .Authenticate (credenciales)

                .WithDefaultSubscription ();




            // Primero que nada, necesitamos crear un grupo
            de recursos donde agregaremos todos

            //Los recursos
```

```

// necesario para la máquina virtual

var groupName = "az204-ResourceGroup";

var vmName = "az204VMTesting";

var location = Region.USWest2;

var vNetName = "az204VNET";

var vNetAddress = "172.16.0.0/16";

var subnetName = "az204Subnet";

var subnetAddress = "172.16.0.0/24";

var nicName = "az204NIC";

var adminUser = "azureadminuser";

var adminPassword = "Pa $$ w0rd! 2019";


Console.WriteLine ($ "Creando grupo de recursos
{groupName} ...");

var ResourceGroup = azure.ResourceGroups.Define
( nombre_de_grupo )

    .WithRegion ( ubicación )

    .Crear();


// Todas las máquinas virtuales deben estar
conectadas a una red virtual.

Console.WriteLine ($ "Creando red virtual
{vNetName} ...");

var network = azure.Networks.Define ( vNetName )

    .WithRegion ( ubicación )

```

```
        .WithExistingResourceGroup ( nombre_de_grupo )
    )

        .WithAddressSpace ( vNetAddress )

        .WithSubnet ( subnetName , subnetAddress )

        .Crear();

    }

    // Cualquier máquina virtual necesita una
    interfaz de red para conectarse al

    // red virtual

    Console.WriteLine ($ "Creando interfaz de red
{nicName} ...");

    var nic = azure.NetworkInterfaces.Define (
nicName )

        .WithRegion ( ubicación )

        .WithExistingResourceGroup ( nombre_de_grupo
    )

        .WithExistingPrimaryNetwork ( red )

        .WithSubnet ( nombre_subred )

        .WithPrimaryPrivateIPAddressDynamic ()

        .Crear();

    }

    // Crea la máquina virtual

    Console.WriteLine ($ "Creando máquina virtual
{vmName} ...");

    azure.VirtualMachines.Define ( vmName )

        .WithRegion ( ubicación )
```

```

        .WithExistingResourceGroup ( nombre_de_grupo
    )

        .WithExistingPrimaryNetworkInterface ( nic )

            .WithLatestWindowsImage
("MicrosoftWindowsServer", "WindowsServer",
"2012-R2-Datacenter")

        .WithAdminUsername ( adminUser )

        .WithAdminPassword ( adminPassword )

        .Con nombreDeEquipo ( nombreVm )

            .WithSize
(VirtualMachineSizeTypes.StandardDS2V2)

        .Crear();

    {

}

}

```

### **Tenga en cuenta los requisitos de la aplicación**

Para ejecutar todos los ejemplos de este libro, debe tener una suscripción a Azure. Si no tiene una suscripción a Azure, puede crear una suscripción gratuita para probar el código de este libro.

Además, debe crear una aplicación de Azure AD y una entidad de seguridad en su suscripción de Azure. Debe configurar estos elementos para otorgar, crear y modificar privilegios a su aplicación. Siga las instrucciones de este procedimiento para crear la aplicación de Azure AD y la entidad de seguridad. Consulte <https://docs.microsoft.com/en-us/azure/active-directory/develop/howto-create-service-principal-portal>.

Como puede ver en el [Listado 1-2](#), debe crear cada uno de los recursos relacionados y requeridos por separado y luego proporcionar todas las

dependencias requeridas al cliente de administración de Azure que crea la VM.

Antes de proceder a implementar una nueva máquina virtual, también debe tener en cuenta otras consideraciones que afectarían la implementación. Por ejemplo, si su aplicación o solución de software debe tener una alta disponibilidad, normalmente utilizará un equilibrador de carga. Si sus máquinas virtuales utilizan un equilibrador de carga, debe colocar sus máquinas virtuales que alojan la aplicación en un conjunto de disponibilidad. El uso de un conjunto de disponibilidad garantiza que ninguna máquina virtual del mismo conjunto de disponibilidad se coloque en el mismo hardware. La colocación de las máquinas virtuales en diferentes equipos asegura que las VM no se reinicen al mismo tiempo debido a las actualizaciones de software en los servidores que ejecutan la VM. Una máquina virtual solo se puede agregar a un conjunto de disponibilidad durante la creación de la VM. Si olvida agregar la VM a un conjunto de disponibilidad,

**¿Necesita más revisión? Administre la disponibilidad de sus máquinas virtuales**

Puede encontrar más información sobre cómo administrar la disponibilidad de sus máquinas virtuales revisando el artículo en <https://docs.microsoft.com/en-us/azure/virtual-machines/windows/manage-availability>.



### **Sugerencia para el examen**

La creación de una máquina virtual es un proceso sencillo, pero aún debe planificar la implementación. Debe considerar si necesita hacer que la aplicación esté altamente disponible alojada en la máquina virtual, o si necesita escalar hacia arriba y hacia abajo la cantidad de VM asociadas a un balanceador de carga. En tales casos, debe recordar crear el conjunto de disponibilidad antes de crear las VM.

### **Configurar máquinas virtuales para acceso remoto**

La sección anterior revisó cómo crear una nueva máquina virtual mediante programación. Cuando creó la VM del ejemplo anterior, no pudo acceder a su nueva VM. La razón de esto es que no configuró una dirección IP pública que pueda usar para acceder de forma remota a la VM.

Como puede imaginar, el simple hecho de agregar una IP pública a su VM puede conducir a una exposición excesiva a Internet. Puede controlar esa exposición utilizando los cortafuegos proporcionados por el sistema operativo. Pero el mantenimiento de todos los firewalls para varias máquinas virtuales puede ser una tarea que requiere mucho tiempo. Azure le proporciona grupos de seguridad de red como mecanismo para filtrar el tráfico entre sus recursos de Azure y diferentes redes, incluida Internet. Si necesita configurar el acceso remoto para su VM, debe agregar una regla de seguridad a un grupo de seguridad de red asociado con su VM.

De forma predeterminada, cualquier máquina virtual que implemente con una imagen de máquina virtual de Azure tiene habilitado el protocolo de acceso remoto correspondiente, es decir, Protocolo de escritorio remoto o RDP para máquinas virtuales Windows y Secure Shell o SSH para máquinas virtuales Linux.

Siguiendo el ejemplo del [Listado 1-2](#), debe agregar tres elementos adicionales al código para tener acceso remoto a su VM:

- **Una IP pública** Puede configurar una IP estática o dinámica. La IP estática tiene costos asociados, por lo que este ejemplo usa una IP dinámica.
- **Un grupo de seguridad de red** Lo necesita para administrar las reglas de seguridad que permiten o deniegan el acceso a la VM.
- **Una regla de seguridad** Debe crear una regla de seguridad para permitir el acceso a la máquina virtual mediante el protocolo remoto adecuado. Para este ejemplo, debe permitir el tráfico al puerto TCP / 3389. Este es el puerto para el protocolo de escritorio remoto.

[El Listado 1-3](#) muestra las modificaciones en negrita que necesita hacer en su código para configurar el acceso remoto a su VM durante la implementación.

**Listado 1-3** Program.cs modificado

[Haga clic aquí para ver la imagen del código](#)

---

```
// dotnet core 2.2
```

```
using el sistema;
```

```
utilizando Microsoft.Azure.Management.Compute.Fluent;
utilizando Microsoft.Azure.Management.Compute.Fluent.Models;
utilizando Microsoft.Azure.Management.Network.Fluent;
utilizando Microsoft.Azure.Management.Fluent;
utilizando
Microsoft.Azure.Management.ResourceManager.Fluent;
utilizando
Microsoft.Azure.Management.ResourceManager.Fluent.Core;
utilizando Microsoft.Azure.Management.Network.Fluent.Models;

espacio de nombres ch1_1_2

{
    programa de clase

    {
        static void Main (cadena [] argumentos)

        {

            // Crea el cliente de gestión. Esto se utilizará
            para todas las operaciones.

            // que realizaremos en Azure.

            var credentials =
SdkContext.AzureCredentialsFactory

                .FromFile ("./
azureauth.properties");

            var azure = Azure.Configure ()

                .WithLogLevel
(HttpLoggingDelegatingHandler.Level.Basic)
```

```

        .Authenticate (credenciales)

        .WithDefaultSubscription () ;



    // Primero que nada, necesitamos crear un grupo
    de recursos donde agregaremos todos

    //Los recursos

    // necesario para la máquina virtual

    var groupName = "az204-ResourceGroup";

    var vmName = "az204VMTesting";

    var location = Region.USWest2;

    var vNetName = "az204VNET";

    var vNetAddress = "172.16.0.0/16";

    var subnetName = "az204Subnet";

    var subnetAddress = "172.16.0.0/24";

    var nicName = "az204NIC";

    var adminUser = "azureadminuser";

    var adminPassword = "Pa $$ w0rd! 2019";

var publicIPName = "az204publicIP";

var nsgName = "az204VNET-NSG";



    Console.WriteLine ($ "Creando grupo de recursos
{groupName} ...");

    var resourceGroup = azure.ResourceGroups.Define
(groupName)

        .WithRegion (ubicación)

```

```
        .Crear();  
  
        // Todas las máquinas virtuales deben estar  
        conectadas a una red virtual.  
  
        Console.WriteLine ($ "Creando red virtual  
{vNetName} ...");  
  
        var network = azure.Networks.Define (vNetName)  
            .WithRegion (ubicación)  
            .WithExistingResourceGroup (groupName)  
            .WithAddressSpace (vNetAddress)  
            .WithSubnet (nombre de subred, dirección de  
subred)  
            .Crear();  
  
        // Necesita una IP pública para poder conectarse  
        a la VM desde Internet  
  
        Console.WriteLine ($ "Creando IP pública  
{publicIPName} ...");  
  
        var publicIP = azure.PublicIPAddresses.Define  
(publicIPName)  
            .WithRegion (ubicación)  
            .WithExistingResourceGroup (groupName  
        ) .Create ();  
  
        // Necesita un grupo de seguridad de red para  
        controlar el acceso a VM
```

```
Console.WriteLine ($ "Creando grupo de seguridad  
de red {nsgName} . . ." );  
  
var nsg = azure.NetworkSecurityGroups.Define  
(nsgName)  
  
.WithRegion (ubicación)  
  
.WithExistingResourceGroup (groupName  
) .Create ();  
  
  
// Necesita una regla de seguridad para permitir  
el acceso a la VM desde  
  
// Internet  
  
Console.WriteLine ($ "Creando una regla de  
seguridad para permitir el control remoto  
acceso " );  
  
nsg.Update ()  
  
.DefineRule (" Allow-RDP ")  
  
.AllowInbound ()  
  
.FromAnyAddress ()  
  
.FromAnyPort ()  
  
.ToAnyAddress ()  
  
.ToPort (3389)  
  
.WithProtocol (SecurityRuleProtocol.Tcp)  
  
.WithPriority (100 )  
  
.WithDescription ("Permitir-RDP")  
  
.Attach ()
```

```
    .Apply () ;

    // Cualquier máquina virtual necesita una
    interfaz de red para conectarse al

    // red virtual

    Console.WriteLine ($ "Creando interfaz de red
{nicName} ..." );

    var nic = azure.NetworkInterfaces.Define
(nicName)

        .WithRegion (ubicación)

        .WithExistingResourceGroup (groupName)

        .WithExistingPrimaryNetwork (red)

        .WithSubnet (nombre de subred)

        .WithPrimaryPrivateIPAddressDynamic ()

        .WithExistingPrimaryPublicIPAddress
(publicIP)

        .WithExistingNetworkSecurityGroup (nsg)

        .Crear () ;

    // Crea la máquina virtual

    Console.WriteLine ($ "Creando máquina virtual
{vmName} ..." );

    azure.VirtualMachines.Define (vmName)

        .WithRegion (ubicación)

        .WithExistingResourceGroup (groupName)

        .ConExistingPrimaryNetworkInterface (nic)
```

```

        .WithLatestWindowsImage
("MicrosoftWindowsServer", "WindowsServer",
"2012-R2-Datacenter")

        .WithAdminUsername (adminUser)

        .WithAdminPassword (adminPassword)

        .Con nombreDeEquipo (nombreVm)

        .WithSize
(VirtualMachineSizeTypes.StandardDS2V2)

        .Crear();

}

}

```

El código que usó para crear y habilitar el acceso remoto a la VM es una buena manera de comprender la relación entre los diferentes componentes necesarios para implementar una VM. Debe comprender estas relaciones si necesita implementar o reconfigurar una máquina virtual mediante PowerShell o la CLI de Azure. Después de haber creado la VM usando el código modificado en el [Listado 1-3](#), use el siguiente procedimiento para verificar que todo esté funcionando correctamente:

1. Abra Azure Portal (<https://portal.azure.com>).
2. En el cuadro de texto Buscar recursos en la parte superior media del portal, escriba **az204VMTesting**.
3. En la lista de resultados, haga clic en el nombre de la máquina virtual.
4. En la página de la máquina virtual az204VMTesting, haga clic en Redes en la sección Configuración.

5. En la lista de reglas de seguridad del Grupo de seguridad de red, que se muestra en la [Figura 1-1](#), asegúrese de que haya una regla llamada Allow-RDP.

Priority	Name	Port	Protocol	Source	Destination	Action	...
100	Allow-RDP	3389	TCP	Any	Any	<span style="color: green;">Allow</span>	...
65000	AllowVnetInBound	Any	Any	VirtualNetwork	VirtualNetwork	<span style="color: green;">Allow</span>	...
65001	AllowAzureLoadBalancerInBound	Any	Any	AzureLoadBalancer	Any	<span style="color: green;">Allow</span>	...
65500	DenyAllInBound	Any	Any	Any	Any	<span style="color: red;">Deny</span>	...

**Figura 1-1** Grupo de seguridad de red de la máquina virtual

1. En la página az204VMTTesting, haga clic en Conectar en la sección Configuración en la lista de navegación en el lado izquierdo de la página.
2. En la página Conectar, asegúrese de que la pestaña RDP esté seleccionada.
3. Asegúrese de que la opción Dirección IP pública esté seleccionada en el menú desplegable Dirección IP.
4. Haga clic en el botón Descargar archivo RDP.
5. Una vez que se haya descargado el archivo RDP, haga doble clic en el archivo RDP para abrir la sesión remota con su VM. Debe proporcionar la contraseña configurada en el código del [Listado 1-3](#).

Al configurar el acceso remoto en una máquina virtual, debe considerar si esa máquina virtual debe ser accesible desde Internet o si puede restringir el acceso. Como regla general, debe evitar configurar el acceso remoto desde Internet para las máquinas virtuales de producción. Para estas máquinas virtuales, debe considerar la implementación de una red privada virtual y el acceso remoto a la máquina virtual utilizando la IP privada en lugar de una IP pública. De esta manera, puede usar la IP pública para publicar solo el servicio alojado en la VM, como un servicio IIS, mientras obtiene acceso remoto a la VM utilizando la IP privada.

*¿Necesita más revisión? Grupos de seguridad de red*

Puede encontrar más información sobre cómo administrar grupos de seguridad de red mediante Azure Portal, Azure PowerShell o la CLI de Azure revisando los siguientes artículos:

- <https://docs.microsoft.com/en-us/azure/virtual-network/security-overview>
- <https://docs.microsoft.com/en-us/azure/virtual-network/tutorial-restrict-network-access-to-resources>



### *Sugerencia para el examen*

Debe considerar cuidadosamente cuándo configurar una máquina virtual para el acceso remoto desde Internet. En general, no debe configurar el acceso remoto a través de IP públicas en máquinas virtuales de producción. Para esos casos, debe implementar una red privada virtual y conectarse a su VM utilizando su IP privada.

### *Crear plantillas ARM*

Una de las ventajas más importantes del uso de IaaS de Azure es el nivel de automatización que puede lograr al implementar nuevos servicios, recursos o infraestructura. Una de las principales razones por las que puede hacer esto es porque Microsoft le proporciona Azure Resource Manager (ARM), que es el servicio de implementación y administración en Azure. El servicio ARM se encarga de crear, actualizar y eliminar los diferentes servicios que puedes implementar en tu suscripción. Puede interactuar con todas las acciones que ofrece el servicio ARM utilizando la misma API. Gracias a esta misma API, independientemente del mecanismo que utilice (portal, PowerShell, CLI de Azure, API Rest o SDK de cliente), obtendrá un comportamiento y un resultado coherentes al interactuar con ARM.

Cuando trabaja con Azure Resource Manager, hay algunos conceptos y términos que debe comprender claramente:

- **Recurso** Estos son los elementos que puede administrar en Azure.
- **Grupo de recursos** Este es un contenedor que usa para almacenar recursos. Puede utilizar cualquier criterio de agrupación para sus recursos, pero debe recordar que cualquier recurso individual debe estar incluido en un grupo de recursos. También

puede usar grupos de recursos para administrar diferentes niveles de acceso de administración a diferentes grupos de usuarios.

- **Proveedor de recursos** Un recurso es un servicio que ofrece diferentes tipos de recursos de Azure y administra el ciclo de vida del recurso. Por ejemplo, el servicio a cargo de proporcionar recursos de máquinas virtuales es el proveedor de Microsoft.Compute. También puede utilizar el proveedor de Microsoft.Storage para las cuentas de almacenamiento o Microsoft.Network para todos los recursos de red.
- **Plantilla de Resource Manager** Este es el archivo que debe proporcionar a la API de ARM cuando desee implementar uno o más recursos en un grupo de recursos o suscripción. Este archivo está escrito en JavaScript Object Notation (JSON).

La principal ventaja de usar plantillas ARM es que tiene la definición de todos los recursos que desea implementar en una estructura consistente. Esto le permite reutilizar la misma plantilla para implementar el mismo grupo de recursos en diferentes suscripciones, grupos de recursos o regiones. Algunos escenarios comunes en los que puede aprovechar las plantillas ARM son implementaciones de planes de recuperación ante desastres, configuraciones de alta disponibilidad o escenarios de aprovisionamiento automático (como escenarios de implementación continua). En el siguiente fragmento de código, puede ver la estructura más básica de una plantilla ARM.

[Haga clic aquí para ver la imagen del código](#)

```
{  
  "$schema":  
    "https://schema.management.azure.com/schemas/2015-01-  
    01/deploymentTemplate.  
  
  "json # ",  
  
  "contentVersion": "",  
  
  "parámetros": {},  
  
  "variables": {},  
  
  "funciones": []},
```

```
"recursos": [ ] ,  
"salidas": {}  
}
```

En lo que respecta a la estructura de la plantilla ARM, solo las secciones \$ schema, contentVersion y resources deben estar presentes en una plantilla válida. A continuación se muestra una breve descripción de cada sección en una plantilla:

- **\$ esquema** Esta sección obligatoria establece el esquema JSON que describe la versión de la plantilla que usará en el archivo. Puede elegir entre dos esquemas diferentes según el tipo de implementación:
  - **Implementaciones de grupos de recursos** Debe usar <https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#>.
  - **Implementaciones de suscripción** Debe usar <https://schema.management.azure.com/schemas/2018-05-01/subscriptionDeploymentTemplate.json#>.
- **contentVersion** En esta sección obligatoria, establece un valor que puede usar para proporcionar su número de versión interno a la plantilla, como 1.0.0. Este número de versión solo es significativo para usted; Azure no lo usa. Normalmente, cambia el número de versión cuando realiza cambios significativos en la plantilla.
- **parámetros** Esta es una sección opcional que puede utilizar para establecer los valores proporcionados al Administrador de recursos cuando realiza una implementación. Puede utilizar parámetros de plantilla personalizables para diferentes implementaciones sin cambiar el contenido de la plantilla.
- **variables** Esta sección opcional contiene los valores que reutilizará en toda la plantilla. Utiliza variables para mejorar la usabilidad y legibilidad de la plantilla.
- **funciones** Puede utilizar esta sección opcional para definir las funciones que se utilizarán en la plantilla.

- **recursos** Esta es una sección obligatoria que contiene todos los recursos que la plantilla implementará o actualizará.
- **salidas** Esta sección opcional define los valores que Resource Manager debe devolver una vez finalizada la implementación.

El Listado 1-4 muestra la plantilla ARM que necesita usar para implementar nuevas VM con la misma configuración. Puede modificar los valores de los parámetros según sus necesidades.

**Listado 1-4** plantilla ARM para implementar una máquina virtual

Haga clic aquí para ver la imagen del código

---

{

```

"$ esquema":
"https://schema.management.azure.com/schemas/2015-01-
01/deploymentTemplate.

json # ,

"contentVersion": "1.0.0.0",

"parámetros": {

  "virtualNetworks_az204VNET_name": {

    "defaultValue": "az204demoVNET",
    "tipo": "cadena"

  },
  "networkInterfaces_az204NIC_name": {

    "defaultValue": "az204demoNIC",
    "tipo": "cadena"

  },
  "virtualMachines_az204VMTesting_name": {

```

```
        "defaultValue": "az204demoVM",
        "tipo": "cadena"
    },
    "subnets_az204Subnet_name": {
        "defaultValue": "az204demoSubnet",
        "tipo": "cadena"
    },
    "virtualMachines_az204VMTesting_id": {
        "defaultValue": "[concat (parámetros
('virtualMachines_az204VMTesting_name'),
 '_OSDisk1_1')] ",
        "tipo": "cadena"
    },
    "virtualMachines_adminUser": {
        "defaultValue": "azureadminuser",
        "tipo": "cadena"
    },
    "virtualMachines_adminpassword": {
        "defaultValue": "Pa $$ w0rd",
        "tipo": "cadena de seguridad"
    }
},
"variables": {
```

```
        "osDiskName":  
    "_OSDisk1_1_39c654d89d88405e968db84b722002d1"  
  
    },  
  
    "recursos": [  
  
        {  
  
            "type": "Microsoft.Compute / virtualMachines",  
  
            "nombre": "[parámetros  
('virtualMachines_az204VMTesting_name')]",  
  
            "apiVersion": "2018-06-01",  
  
            "ubicación": "westus2",  
  
            "etiquetas": {},  
  
            "escala": nulo,  
  
            "propiedades": {  
  
                "hardwareProfile": {  
  
                    "vmSize": "Standard_DS2_v2"  
  
                },  
  
                "storageProfile": {  
  
                    "imageReference": {  
  
                        "publisher":  
"MicrosoftWindowsServer",  
  
                        "oferta": "WindowsServer",  
  
                        "sku": "2012-R2-Datacenter",  
  
                        "versión": "más reciente"  
  
                    },  
  
                    "osDisk": {  
  
                }  
            }  
        }  
    ]  
}
```

```
        "osType": "Windows",
        "nombre": "[concat (parámetros
('virtualMachines_az204VMTesting_'
                     nombre '), variables ('
osDiskName '))] ",
        "createOption": "FromImage",
        "almacenamiento en caché": "Leer y
escribir"
    },
    "dataDisks": []
},
"osProfile": {
    "computerName": "[parámetros (
'vertualMachines_az204VMTesting_name')] ",
    "adminUsername": "azureadminuser",
    "adminPassword": "Pa $$ w0rd",
    "windowsConfiguration": {
        "provisionVMAgent": verdadero,
        "enableAutomaticUpdates": verdadero
    },
    "secretos": [],
    "allowExtensionOperations": verdadero
},
"networkProfile": {
```

```
        "interfaces de red": [
            {
                "id": "[resourceId ('Microsoft.Network/networkInterfaces',
                parámetros
                    ('networkInterfaces_az204NIC_name'))] ",
                "propiedades": {
                    "primario": verdadero
                }
            }
        ]
    }
},

"depende de": [
    "[resourceId ('Microsoft.Network/networkInterfaces',
    parámetros (
        'networkInterfaces_az204NIC_name'))] "
]

},
{
    "tipo": "Microsoft.Network/networkInterfaces",
    "nombre": "[parámetros
('networkInterfaces_az204NIC_name')]",
    "apiVersion": "2018-10-01",
    "ubicación": "westus2",
```

```
        "etiquetas": { },
        "escala": nulo,
        "propiedades": {
            "ipConfigurations": [
                {
                    "nombre": "principal",
                    "propiedades": {
                        "privateIPAllocationMethod": "Dinámico",
                        "subred": {
                            "id": "[resourceId('Microsoft.Network/virtualNetworks/subnets',
                                parámetros ('virtualNetworks_az204VNET_name')),
                                parámetros ('subnets_az204Subnet_name'))] ",
                            },
                    },
                    "primario": verdadero,
                    "privateIPAddressVersion": "IPv4"
                }
            ],
            "dnsSettings": {
                "dnsServers": [],
                "ApplicationDnsServers": []
            }
        }
    ]
}
```

```
        } ,  
        "enableAcceleratedNetworking": falso,  
        "enableIPForwarding": falso,  
        "primario": verdadero,  
        "tapConfigurations": []  
    } ,  
    "depende de": [  
  
    "[resourceId ('Microsoft.Network/virtualNetworks/subnets',  
    parámetros ('virtualNetworks_  
az204VNET_name '), parámetros (' subnets_az204Subnet_name  
' ))]"  
]  
},  
{  
    "tipo": "Microsoft.Network/virtualNetworks",  
    "nombre": "[parámetros  
( 'virtualNetworks_az204VNET_name ') ]",  
    "apiVersion": "2018-10-01",  
    "ubicación": "westus2",  
    "etiquetas": {},  
    "escala": nulo,  
    "propiedades": {  
        "resourceGuid": "145e7bfc-8b00-48cf-8fa1-  
082448a30bae",
```

```
    "espacio de dirección": {

        "addressPrefixes": [
            "172.16.0.0/16"
        ]

    } ,

    "dhcpOptions": {

        "dnsServers": []
    } ,

    "subredes": [
        {
            "nombre": "[parámetros
('subnets_az204Subnet_name') ]",

            "propiedades": {

                "addressPrefix": "172.16.0.0/24"
            }
        }
    ] ,

    "virtualNetworkPeerings": [],
    "enableDdosProtection": false,
    "enableVmProtection": false
} ,

"depende de": []

},

{
```

```

        "tipo": "Microsoft.Network/virtualNetworks/subnets",

        "nombre": "[concat (parámetros ('virtualNetworks_az204VNET_name'), '/',
    parámetros ('subnets_az204Subnet_name'))] ",

        "apiVersion": "2018-10-01",

        "escala": nulo,

        "propiedades": {

            "addressPrefix": "172.16.0.0/24"

        },

        "depende de": [

    "[resourceId ('Microsoft.Network/virtualNetworks',
    parámetros ('virtualNetworks_az204VNET_name'))] "

]

}
]
}

```

Este ejemplo tiene algunas características interesantes. Ha definido los parámetros y variables que utilizará en toda la plantilla. Si observa cualquier definición de parámetro, puede ver que tiene tres elementos: parameterName, defaultValue y type. El elemento tipo es casi autoexplicativo; establece el tipo de valor que contendrá este parámetro. Los tipos permitidos son string, securestring, int, bool, object, secureObject y array. El parámetroName también es bastante sencillo y es cualquier identificador JavaScript válido que represente el nombre del parámetro. Sin embargo, ¿por qué usar un elemento defaultValue en lugar

de un elemento de valor? Utiliza defaultValue porque cuando define un parámetro en el archivo de plantilla, los únicos componentes necesarios son parameterName y type. El valor del parámetro se proporciona durante el proceso de implementación. Si no proporciona un valor para un parámetro que definió en la plantilla, se utilizará el valor predeterminado en su lugar. Debes tener en cuenta que este elemento es opcional.

Puede proporcionar valores a los parámetros que defina para su plantilla utilizando la línea de comando o creando un archivo con los valores que desea proporcionar a cada parámetro. El siguiente ejemplo muestra el contenido de un archivo de parámetros para la plantilla mostrada anteriormente en el [Listado 1-4](#) :

[Haga clic aquí para ver la imagen del código](#)

```
{  
  "$ esquema":  
    "https://schema.management.azure.com/schemas/2015-01-  
    01/deploymentParameters.  
  
  json # ",  
  
  "contentVersion": "1.0.0.0",  
  
  "parámetros": {  
  
    "virtualNetworks_az204VNET_name": {  
  
      "valor": "az204demoVNET"  
  
    },  
  
    "networkInterfaces_az204NIC_name": {  
  
      "valor": "az204demoNIC"  
  
    },  
  
    "virtualMachines_az204VMTesting_name": {  
  
      "valor": "az204demoVM"  
  
    },  
  }  
}
```

```

    "subnets_az204Subnet_name": {
        "valor": "az204demoSubnet"
    },
    "virtualMachines_az204VMTesting_id": {
        "valor": "[concat (parámetros (
            'virtualMachines_az204VMTesting_name') ,
            '_OSDisk1_1_39c654d89d88405e968db84b722002d1')] "
    },
    "virtualMachines_adminUser": {
        "valor": "azureadminuser"
    },
    "virtualMachines_adminpassword": {
        "valor": "Pa $$ w0rd"
    }
}
}

```

Cuando define el valor de un parámetro, también puede utilizar funciones para construir valores dinámicos. Si echas un vistazo al parámetro virtualMachines\_az204VMTesting\_id, puedes ver que su valor está establecido en una función. En este caso, la función devuelve una cadena que es el resultado de agregar la cadena \_OSDisk1\_1\_39c654d89d88405e968db84b722002d1 al valor del parámetro virtualMachines\_az204VMTesting\_name.

Hay muchas funciones predefinidas que puede utilizar en su plantilla. Incluso puede definir sus funciones personalizadas para esas piezas complicadas de código que se repiten en su plantilla. Cuando

trabaje con funciones personalizadas, tenga cuidado con algunas limitaciones:

- Las funciones personalizadas no pueden acceder a las variables de la plantilla, aunque puede pasárselas como un parámetro de su función.
- Su función personalizada no puede acceder a los parámetros de la plantilla; en su lugar, solo tienen acceso a los parámetros que defina en su función.
- Los parámetros de su función personalizada no pueden tener valores predeterminados.
- Su función personalizada no puede llamar a otras funciones personalizadas; solo puede llamar a funciones predefinidas.
- No puede utilizar la función predefinida `reference()`.

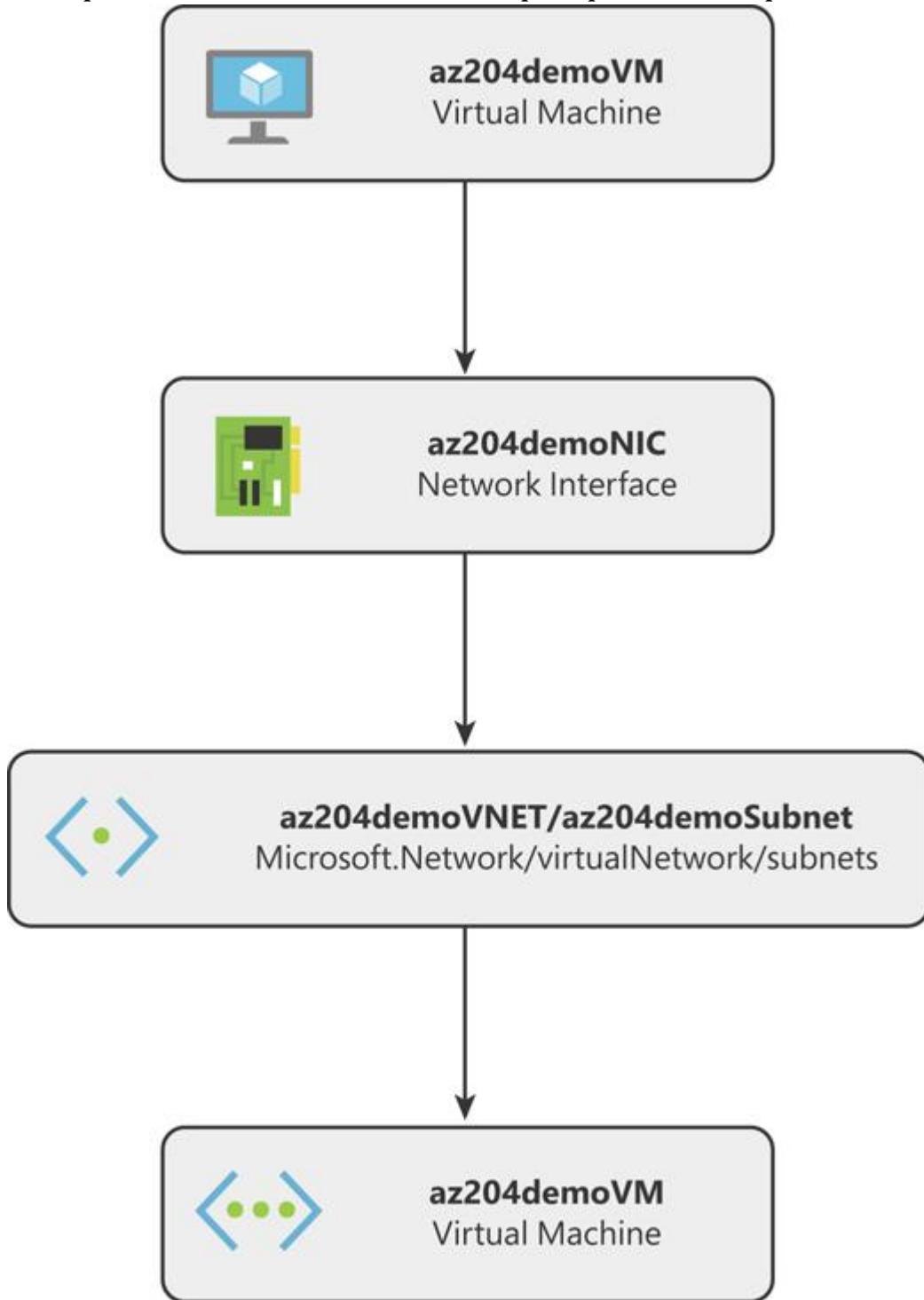
#### Referencia de plantilla de *nota*

Cuando trabaja con plantillas ARM, es útil consultar la referencia de la plantilla para cada tipo de recurso que está configurando. Puede revisar la referencia de la plantilla completa en <https://docs.microsoft.com/en-us/azure/templates/>. También puede revisar la referencia completa de funciones predefinidas en <https://docs.microsoft.com/en-us/azure/azure-resource-manager/resource-group-template-functions>.

Cuando hablé inicialmente sobre los recursos que necesita para implementar una máquina virtual, le expliqué que hay algunos recursos que necesita para que la máquina virtual se ejecute correctamente. Por ejemplo, necesita al menos un disco virtual para almacenar el sistema operativo. También necesita una red virtual para conectar la máquina virtual con el mundo, y necesita una tarjeta de interfaz de red virtual para conectar la máquina virtual a la red virtual. Todas esas dependencias se definen en una plantilla ARM utilizando el elemento `dependsOn` de cada tipo de recurso. Este elemento acepta una lista de nombres de recursos, separados por comas, que definen los recursos que deben implementarse antes de que se pueda implementar el recurso. Como práctica recomendada para evitar la ambigüedad, debe hacer referencia a cualquier recurso que coloque en el elemento `dependsOn` mediante el espacio de nombres y el tipo de su proveedor.

Si revisa el ejemplo, la red virtual `virtualNetworks_az204VNET_name` debe implementarse antes de que se pueda implementar `subnets_az204Subnet_name` (consulte la [Figura 1-2](#)). El elemento

`dependOn` es obligatorio porque los recursos definidos en la plantilla no se implementan en el mismo orden que aparece en la plantilla.



**Figura 1-2** Dependencias de recursos

*Tenga en cuenta los elementos secundarios frente a las dependencias*

Algunos recursos pueden contener otros recursos como elementos secundarios. Esta relación padre-hijo no es lo mismo que una relación de dependencia. La relación padre-hijo no garantiza que el padre se implemente antes que sus hijos. Debe utilizar el elemento dependOn para garantizar el orden de implementación correcto .

Una vez que esté satisfecho con su plantilla, puede implementarla en su suscripción de Azure mediante PowerShell, Azure CLI, Azure Cloud Shell o REST API. Otra característica interesante es que puede almacenar sus archivos JSON de plantilla en una ubicación remota. Esta ubicación remota debe estar disponible públicamente. Si su plantilla contiene información que no debería ser pública, puede proporcionar esa información como un parámetro en línea durante la implementación. Si prefiere que su plantilla no esté abierta al mundo, también puede almacenar su plantilla en una cuenta de almacenamiento y protegerla mediante un token SAS.

El siguiente comando muestra cómo implementar la plantilla de ejemplo utilizando el archivo de plantilla az204-template.json y el archivo de propiedades az204-parameters.json.

[Haga clic aquí para ver la imagen del código](#)

```
#!/bin/bash

Implementación de la plantilla de la CLI #Azure

az group create --name AZ204-ResourceGroup --location "West
US"

implementación de grupo az crear \
--nombre AZ204DemoDeployment \
--grupo de recursos AZ204-ResourceGroup \
--template-file az204-template.json \
--parameters @ az204-parameters.json
```

El comando anterior crea el grupo de recursos denominado AZ204-ResoureGroup en la región oeste de EE. UU. Luego crea una nueva implementación llamada AZ204DemoDeployment que generará los recursos definidos en la plantilla az204-template.json usando los valores proporcionados en el archivo de parámetros llamado az204-parameters.json. Tenga en cuenta el uso del símbolo @ delante del

archivo de parámetros. Este símbolo @ es requerido por el comando az group deployment create.



### *Sugerencia para el examen*

Las plantillas ARM son herramientas poderosas que le permiten crear funciones personalizadas para automatizar algunas acciones repetidas. Cuando cree su función personalizada, recuerde las limitaciones al llamar a funciones predefinidas u otras funciones personalizadas. También debe tener en cuenta la visibilidad de las variables y los parámetros de la plantilla cuando se trabaja con funciones personalizadas.

### *Cree imágenes de contenedor para soluciones con Docker*

Con la evolución de la tecnología y la aparición de la nube, es necesario hacer frente a otros desafíos que presenta esta evolución técnica. Uno de estos requisitos es la capacidad de implementar piezas de software de manera confiable y rápida. Las tecnologías de virtualización fueron una de las claves para hacer posible este tipo de implementación rápida y confiable.

Sin embargo, en el contexto de la virtualización del sistema operativo utilizando máquinas virtuales, uno de los principales inconvenientes es el hecho de que tiene un conjunto completo de binarios, bibliotecas y recursos que están duplicados entre máquinas virtuales. Aquí es donde la contenedorización proporciona un enfoque diferente para implementar piezas de software en múltiples servidores de manera confiable y rápida.

Un contenedor es una pieza de software que empaqueta su código y todas sus dependencias en un solo paquete que puede ser ejecutado directamente por el entorno informático. Cuando se ejecuta un contenedor, utiliza una copia de solo lectura de las bibliotecas compartidas del sistema operativo que su código necesita para ejecutarse. Esto reduce la cantidad requerida de recursos que un contenedor necesita para ejecutar su código en comparación con ejecutar el mismo código en una máquina virtual. La tecnología de contenedores nació inicialmente en entornos Linux, pero también se ha adaptado al entorno Microsoft Windows. Hay varias implementaciones de tecnología de contenedores en el ecosistema de Linux, pero los contenedores Docker son los más utilizados.

Cuando traslada la tecnología de contenedores a un entorno empresarial, escalar de forma dinámica y automática es un problema, al igual que con las máquinas virtuales. Hay varias soluciones disponibles en el mercado, como Docker Swarm, DC / OS o Kubernetes. Todas estas soluciones son soluciones de orquestación que escalan e implementan automáticamente sus contenedores en los recursos disponibles.

Azure proporciona varios servicios que le permiten implementar su aplicación en un contenedor. No importa si decide utilizar Azure Kubernetes Services, Service Fabric, Azure Web Apps for Containers, Azure Container Registry o Azure Container Instances; todos estos servicios utilizan la misma implementación de tecnología de contenedores, Docker.

Antes de que pueda implementar su aplicación en cualquiera de estos servicios, debe colocar su aplicación en un contenedor creando una imagen de su contenedor. Una imagen de contenedor es un paquete que contiene todo lo que necesita (código, bibliotecas, variables de entorno y archivos de configuración) para ejecutar su aplicación. Una vez que tenga las imágenes de su contenedor, puede crear instancias de la imagen para ejecutar el código, cada una de las cuales es un contenedor. Si necesita realizar modificaciones en uno de sus contenedores, debe modificar la definición de la imagen y volver a implementar el contenedor. En general, cualquier cambio que realice en un contenedor no se mantiene durante los reinicios. Si necesita asegurarse de que parte de la información de su contenedor no se elimine cuando un contenedor se reinicia, debe usar puntos de montaje externos, conocidos como volúmenes.

Cuando crea su imagen de contenedor, debe definir los requisitos de su aplicación, que se colocan en un archivo llamado Dockerfile. Este Dockerfile contiene la definición y los requisitos necesarios para crear su imagen de contenedor. Utilice el siguiente procedimiento de alto nivel para crear una imagen:

1. **Cree un directorio para la nueva imagen.** Este directorio contiene su archivo Docker, su código y cualquier otra dependencia que necesite incluir en la imagen, y que no está disponible en una imagen separada.
2. **Cree el Dockerfile.** Este archivo contiene la definición de su imagen. [El Listado 1-5](#) muestra un ejemplo de un Dockerfile funcional.

**3. Abra una línea de comando.** Utiliza esta línea de comando para ejecutar los comandos de Docker.

**4. Crea la imagen del contenedor.** Utilice el comando docker build para crear la imagen. Cuando crea una imagen, debe agregar una etiqueta para identificar la imagen y la versión. Si no establece un número de versión, la ventana acoplable asigna automáticamente el valor predeterminado *más reciente*. Debe proporcionar la ruta de la carpeta que contiene el Dockerfile. Este comando tiene la siguiente estructura:

[Haga clic aquí para ver la imagen del código](#)

```
docker build --tag = <tag_name> [: <version>] <dockerfile_dir>
```

**5. Enumere la imagen recién creada.** Una vez que Docker termine de descargar todas las dependencias para su imagen, puede asegurarse de que su imagen se haya creado ejecutando este comando:

```
imagen acoplable ls
```

#### **Listado 1-5 Ejemplo de Dockerfile**

[Haga clic aquí para ver la imagen del código](#)

---

```
# Use un tiempo de ejecución oficial de Python como imagen principal

DESDE python: 2.7-delgado

# Establecer el directorio de trabajo en / app
WORKDIR / aplicación

# Copie el contenido del directorio actual en el contenedor en / app
COPIAR . / aplicación
```

```
# Instale los paquetes necesarios especificados en  
requirements.txt  
  
EJECUTAR pip install --trusted-host pypi.python.org -r  
requirements.txt  
  
  
# Poner el puerto 80 a disposición del mundo fuera de este  
contenedor  
  
EXPONER 80  
  
  
# Definir una variable de entorno  
  
ENV NAME Mundo  
  
  
  
# Ejecute app.py cuando se inicie el contenedor  
  
CMD ["python", "app.py"]
```

### ***¿Necesita más revisión? Prácticas recomendadas para escribir archivos Docker***

Cuando esté escribiendo su Dockerfile, debe tener en cuenta algunas de las mejores prácticas detalladas en [https://docs.docker.com/develop/develop-images/dockerfile\\_best-practices/](https://docs.docker.com/develop/develop-images/dockerfile_best-practices/).

Para aplicaciones complejas, crear una imagen para cada componente de la aplicación puede convertirse en una tarea complicada. Para escenarios en los que necesita definir y ejecutar varios contenedores, puede usar Docker Compose. También puede pensar en Docker Compose como la definición de sus imágenes para un entorno de producción. Si su aplicación se compone de varias imágenes, puede definir la relación entre esas imágenes y cómo se exponen al mundo externo. 24 Con Docker Compose también puede establecer los límites de los recursos asignados a cada contenedor cuando se ejecuta y defina qué sucede si falla un contenedor asociado con un servicio.

Un servicio en el mundo Docker es cada una de las piezas que forman parte de su aplicación. Un servicio tiene una relación de uno a uno con una imagen. Es importante recordar que un servicio puede tener varias

instancias de la misma imagen, lo que significa que puede tener varios contenedores. El archivo docker-compose.yaml contiene las definiciones de las relaciones y los requisitos necesarios para ejecutar su aplicación.

#### *¿Necesita más revisión? Ejemplo completamente funcional*

Puede ejecutar un ejemplo completamente funcional en su entorno local revisando las instrucciones publicadas por Microsoft en <https://docs.microsoft.com/en-us/azure/aks/tutorial-kubernetes-prepare-app> .



#### *Sugerencia para el examen*

Las modificaciones que realiza en un contenedor mientras se está ejecutando no persisten si reinicia el contenedor. Si necesita realizar cambios en el contenido de un contenedor, debe modificar el contenedor de imágenes y luego volver a implementar el contenedor. Si necesita que su contenedor guarde información que debe conservarse durante los reinicios, debe usar volúmenes.

#### *Publicar una imagen en Azure Container Registry*

El propósito principal de crear una imagen es hacer que su código sea altamente portátil e independiente del servidor que ejecuta su código. Para lograr este objetivo, su imagen debe ser accesible por todos los servidores que pueden ejecutar su imagen. Por lo tanto, necesita almacenar su imagen en un servicio de almacenamiento centralizado.

Azure Container Registry (ACR) es la implementación de Microsoft de un servicio de registro de Docker, basado en la definición de Docker Registry 2.0. Con este servicio de registro de Docker administrado, puede almacenar de forma privada sus imágenes para su posterior distribución a servicios de contenedor, como Azure Managed Kubernetes Service. También puede utilizar ACR para crear sus imágenes sobre la marcha y automatizar la creación de la imagen en función de las confirmaciones de su código fuente.

Antes de que pueda cargar una imagen en su registro de contenedor privado, debe etiquetar la imagen. Para hacer esto, debe incluir el nombre de su registro de contenedor privado en la etiqueta. Utilizará la estructura de nombres <acr\_name>.azurecr.io / [repository\_name] [:versión]. La siguiente lista desglosa cada parte de la etiqueta:

- **acr\_name** Este es el nombre que le dio a su registro.

- **repository\_name** Este es un nombre opcional para un repositorio en su registro. ACR le permite crear repositorios multinivel dentro del registro. Si desea utilizar un repositorio personalizado, simplemente ponga su nombre en la etiqueta.
- **versión** Ésta es la versión que usa para la imagen.

Utilice el siguiente procedimiento para enviar su imagen a su registro ACR. Estos pasos asumen que ya creó un Azure Container Registry e instaló la última CLI de Azure:

1. Inicie sesión en su suscripción de Azure.

```
az login
```

2. Inicie sesión en su registro con este comando:

[Haga clic aquí para ver la imagen del código](#)

```
az acr login --- nombre <acr_name>
```

3. Etiquete la imagen que desea cargar en el registro con este comando:

[Haga clic aquí para ver la imagen del código](#)

```
etiqueta acoplable foobar <acr_name> .azurecr.io /  
<repository_name> / <image_name>
```

4. Empuje la imagen al registro usando este comando:

[Haga clic aquí para ver la imagen del código](#)

```
docker push <acr_name> .azurecr.io / <repository_name> /  
<image_name>
```

Cuando Docker termine de enviar su imagen al registro, puede explorar los repositorios en su registro, como se muestra en la [Figura 1-3](#), para verificar que se haya cargado correctamente.

The screenshot shows the Azure Container Registry interface for the 'az204demo' repository. The left sidebar has a tree view with 'Overview', 'Activity log', 'Access control (IAM)', 'Tags', 'Quick start', and 'Events' under 'Container registry'. Under 'Settings', there are 'Access keys', 'Encryption (Preview)', 'Identity (Preview)', 'Firewalls and virtual networks (...)', 'Private endpoint connections (P...)', 'Locks', and 'Export template'. Under 'Services', 'Repositories' is selected. The main pane shows a search bar with 'Search (Cmd + /)' and a 'Refresh' button. Below that is a 'Search to filter repositories ...' field. A 'Repositories ↑↓' section lists 'develop/foobar'.

**Figura 1-3** Examinar el repositorio de contenedores

La siguiente sección revisa cómo ejecutar el contenedor desde la imagen que ya ha enviado al registro.



### *Sugerencia para el examen*

Un registro de contenedor es útil no solo para almacenar las imágenes de contenedor, sino también para automatizar la implementación de contenedores en los servicios de contenedor de Azure. Cualquier servicio de entrega continua, como Azure Pipelines, necesitaría un registro de contenedor para implementar las imágenes del contenedor.

## *Ejecutar contenedores con Azure Container Instance*

Una vez que haya creado su imagen y la haya puesto a disposición de los servicios de Azure empujándola al registro de su contenedor, es el momento de ejecutar el contenedor en cualquiera de los servicios que Azure le ofrece. Siga este procedimiento de alto nivel:

1. Cree tantas imágenes como necesite su aplicación para ejecutarse correctamente.
2. Cargue o envíe las imágenes de su aplicación a un registro de contenedor.
3. Implemente la aplicación.

Cuando desee crear una imagen en el servicio Azure Container Instance (ACI) desde su Azure Container Registry (ACR), debe autenticarse antes de poder extraer la imagen de su ACR. Con fines de demostración, el siguiente procedimiento utiliza la autenticación de la cuenta de administrador para mostrar cómo crear y ejecutar un contenedor en ACI:

1. Inicie sesión en el shell de la nube de Azure (<https://shell.azure.com>).
2. En el Selector de Shell, seleccione Bash.
3. Abra el editor en línea haciendo clic en el ícono de llave a la derecha del Selector de conchas.
4. Utilice el script del [Listado 1-6](#) para crear una contraseña principal de servicio y crear un contenedor a partir de sus imágenes en el registro.

**Listado 1-6** Creación de una contraseña principal de servicio

[Haga clic aquí para ver la imagen del código](#)

---

```
#! / bin / bash
```

```
# Alguna definición de variable útil para el script
ACR_NAME = az204demo
SP_NAME = az204demo_sp
```

```
IMAGE_TAG = az204demo.azurecr.io / Develop / foobar:  
último  
  
GRUPO_RECURSOS = AKSdemo-RG  
  
APP_NAME = foobar  
  
APP_DNS_NAME = prueba  
  
  
#Obtenga el ID de registro. Necesitará este ID para  
crear la autorización para el  
  
# principal del servicio  
  
ACR_ID = $(az acr show --name $ ACR_NAME --query id --  
output tsv)  
  
  
#Obtenga el servidor de inicio de sesión ACR  
  
ACR_SERVER = $(az acr show --name $ ACR_NAME --query  
loginServer --output tsv)  
  
  
#Obtenga la contraseña principal del servicio.  
Otorgaremos privilegios de extracción al servicio.  
  
#principal  
  
echo "Generando contraseña principal de servicio"  
  
SP_PASS = $(az ad sp create-for-rbac --name http://$  
SP_NAME --scopes $ ACR_ID  
  
--role acrpull --query password --output tsv)  
  
  
#Obtenga el ID de la aplicación asociado a la entidad de  
servicio
```

```

SP_ID = $(az ad sp show --id http://$SP_NAME --query
appId --output tsv)

echo "ID de entidad de servicio: $SP_ID"

echo "Contraseña principal de servicio: $SP_PASS"

#Crear el contenedor en el servicio de instancia de
contenedor

az container create --resource-group $RESOURCE_GROUP --
name $APP_NAME --image

$IMAGE_TAG --cpu 1 --memoria 1 --registry-login-server
$ACR_SERVER --registry-username

$SP_ID - contraseña de registro $SP_PASS - etiqueta de
nombre de dns $APP_DNS_NAME --ports 80

```

5. En la esquina superior derecha del editor en línea, debajo de la información del usuario, haga clic en el ícono de puntos suspensivos y luego haga clic en Guardar. Proporcione un nombre para el guión.
6. En Azure Cloud Shell, ejecute el script escribiendo el siguiente comando en el shell bash:

```
sh <your_script_name>
```

Una vez que haya ejecutado este procedimiento, puede acceder a su contenedor buscando el nombre de su contenedor en el portal de Azure. También puede acceder a la aplicación que colocó en este contenedor ingresando la URL del contenedor en un navegador. La URL de este contenedor tendrá el formato <APP\_DNS\_NAME>. <region>.azurecontainer.io, según el valor de la variable APP\_DNS\_NAME que proporcionó en el script anterior.



*Sugerencia para el examen*

Puede usar varios mecanismos de autenticación, como un inicio de sesión individual con Azure AD, una cuenta de administrador o una entidad de servicio. La autenticación con Azure AD es un buen enfoque para su entorno de desarrollo y prueba. El uso de la cuenta de administrador está deshabilitado de forma predeterminada y no se recomienda para entornos de producción porque debe ingresar la contraseña de la cuenta de administrador en su código. Para los entornos de producción, la forma recomendada de extraer imágenes es utilizar entidades de servicio para la autenticación con ACR.

## HABILIDAD 1.2: CREAR APLICACIONES WEB DE AZURE APP SERVICE

---

Azure App Service es una solución de plataforma como servicio (PaaS) que Microsoft ofrece para ayudarlo a desarrollar sus aplicaciones, back-end de aplicaciones móviles o API REST sin preocuparse por la infraestructura subyacente.

Utiliza la mayoría de los lenguajes de programación más populares, .NET, .NET Core, Java, Ruby, Node.js, PHP o Python, además de su plataforma preferida (Linux o Windows). Azure App Service le proporciona capacidades de infraestructura de nivel empresarial, como equilibrio de carga, seguridad, ajuste de escala automático y administración automatizada. También puede incluir Azure App Service en su ciclo de vida de implementación continua gracias a la integración con GitHub, Docker Hub y Azure DevOps.

### Esta habilidad cubre cómo

- Crear una aplicación web de Azure App Service
- Habilitar el registro de diagnósticos
- Implementar código en una aplicación web
- Configure los ajustes de la aplicación web, incluidos SSL, API y cadenas de conexión
- Implementar reglas de escalado automático, incluido el escalado automático programado y el escalado por métricas operativas o del sistema.

## *Crear una aplicación web de Azure App Service*

Azure App Service es un servicio PaaS basado en HTTP que le permite implementar sus aplicaciones back-end móviles o web o API REST en la nube. El uso de Azure App Services le permite desarrollar su aplicación en cualquiera de los lenguajes más populares del momento, como .NET, .NET Core, Java, Ruby, Node.js, PHP o Python. Azure App Services también le ofrece la flexibilidad de trabajar con cualquiera de sus plataformas favoritas: Windows, Linux o contenedores basados en Linux. La ventaja de utilizar Azure App Services no se limita solo a las diferentes opciones de desarrollo. También se integra bastante bien con diferentes plataformas de implementación e integración continua.

Cuando planea crear un Servicio de aplicaciones de Azure, hay algunos conceptos sobre el rendimiento de su aplicación que debe comprender. Cada App Service necesita recursos para ejecutar su código. Las máquinas virtuales son la base de estos recursos. Aunque Azure proporciona automáticamente la configuración de bajo nivel para ejecutar estas máquinas virtuales, aún debe proporcionar información de alto nivel. Un plan de App Service administra el grupo de máquinas virtuales que alojan su aplicación web.

Puede pensar en un plan de App Service como una granja de servidores que se ejecuta en un entorno de nube. Esto también significa que no está limitado a ejecutar un único App Service en un plan de App Service. Puede compartir los mismos recursos informáticos entre varios servicios de aplicaciones que implemente en el mismo plan de servicios de aplicaciones.

Cuando crea un nuevo plan de App Service, debe proporcionar la siguiente información:

- **Región** Esta es la región donde implementa el plan de App Service. Todos los servicios de aplicaciones de este plan de servicios de aplicaciones se ubican en la misma región que el plan de servicios de aplicaciones.
- **Número de instancias** Este es el número de máquinas virtuales agregadas a su plan de App Service. Tenga en cuenta que la cantidad máxima de instancias que puede configurar para su plan de App Service depende del nivel de precios que

seleccione. Puede escalar el número de instancias de forma manual o automática.

- **Tamaño de las instancias** El tamaño de la VM que se usa en el plan de App Service se configura.
- **Plataforma del sistema operativo** Esto controla si su aplicación web se ejecuta en máquinas virtuales Linux o Windows. Dependiendo del sistema operativo, tiene acceso a diferentes niveles de precios. Tenga en cuenta que una vez que haya seleccionado la plataforma del sistema operativo, no podrá cambiar el sistema operativo del App Service sin volver a crearlo.
- **Nivel de precios** Esto establece las características y capacidades disponibles para su plan de App Service y cuánto paga por el plan. Para las máquinas virtuales de Windows, dos niveles de precios básicos utilizan máquinas virtuales compartidas: F1 y D1. Este nivel compartido no está disponible para máquinas virtuales Linux. Cuando usa los niveles de precios básicos, su código se ejecuta junto con el código de otros clientes de Azure.

Cuando ejecuta un App Service en un plan de App Service, todas las instancias configuradas en el plan ejecutan el código correspondiente a su aplicación. Esto significa que si tiene cinco máquinas virtuales, cualquier aplicación que implemente en App Service se ejecutará en cada una de las cinco máquinas virtuales. Otras operaciones relacionadas con App Service, como ranuras de implementación adicionales, registros de diagnóstico, copias de seguridad o WebJobs, también se ejecutan utilizando los recursos de cada máquina virtual en el plan de App Service.

Azure App Service también le brinda la capacidad de integrar la autenticación y autorización de su aplicación web, API REST, un back-end de aplicación móvil o incluso Azure Functions. Puede utilizar diferentes proveedores de autenticación conocidos, como Azure, Microsoft, Google, Facebook y Twitter, para autenticar a los usuarios en su aplicación. También puede utilizar otros mecanismos de autenticación y autorización en sus aplicaciones. Sin embargo, al utilizar este módulo de seguridad, puede proporcionar un nivel razonable de seguridad a su aplicación con cambios de código mínimos o incluso nulos.

Hay situaciones en las que su aplicación puede requerir acceso a recursos en su infraestructura local y App Service le brinda dos enfoques diferentes:

- **Integración de VNet** Esta opción está disponible solo para los niveles de precios Estándar, Premium o PremiumV2. Esta integración permite que su aplicación web acceda a recursos en su red virtual. Si crea una VPN de sitio a sitio con su infraestructura local, puede acceder a sus recursos privados desde su aplicación web.
- **Conexiones híbridas** Esta opción depende de Azure Service Bus Relay y crea una conexión de red entre App Service y un extremo de la aplicación. Esto significa que las conexiones híbridas permiten el tráfico entre combinaciones específicas de puerto y host TCP.

El siguiente procedimiento muestra cómo crear un plan de App Service y cargar una aplicación web simple basada en .NET Core usando Visual Studio 2019. Asegúrese de haber instalado ASP.NET y la carga de trabajo de desarrollo web y de haber instalado las últimas actualizaciones.

1. Abra Visual Studio 2019 en su computadora.
2. En la ventana Inicio de Visual Studio 2019, en la columna denominada Comenzar, haga clic en el vínculo Continuar sin código en la parte inferior de la columna.
3. Haga clic en el menú Herramientas y elija Obtener herramientas y funciones. Verifique que la sección ASP.NET And Web Development In The Web & Cloud esté marcada.
4. En la ventana de Visual Studio 2019, haga clic en Archivo> Nuevo> Proyecto para abrir la ventana Nuevo proyecto.
5. En la ventana Crear un nuevo proyecto, seleccione C # en el menú desplegable debajo del cuadro de texto Buscar plantillas en la parte superior derecha de la ventana.
6. En el menú desplegable Todos los tipos de proyectos, seleccione Web.
7. En la lista de plantillas en el lado derecho de la ventana, seleccione Aplicación web ASP.NET Core.
8. En la ventana Configure Your New Project, complete los siguientes pasos:
  1. Seleccione un nombre para el proyecto.
  2. Ingrese una ruta para la ubicación de la solución.

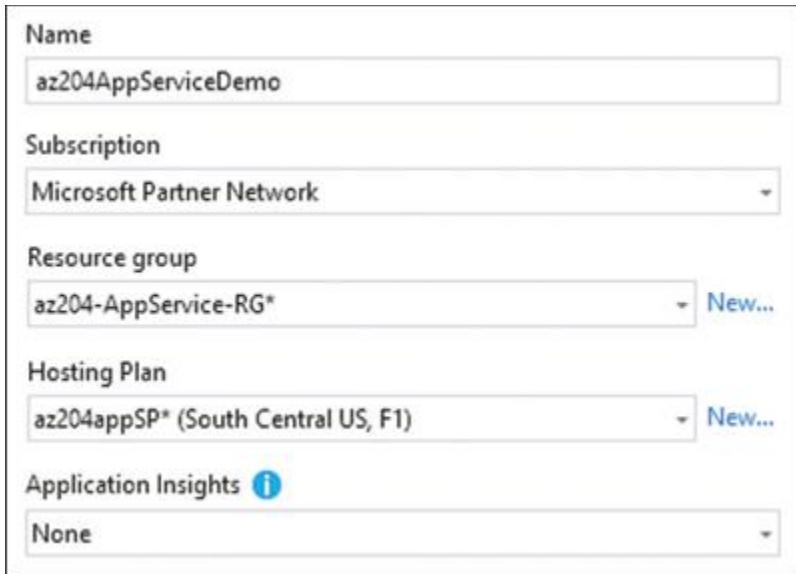
3. En el menú desplegable Solución, seleccione Crear una nueva solución.
  4. Ingrese un nombre para la solución.
9. Haga clic en el botón Crear en la esquina inferior derecha de la ventana Configure su nuevo proyecto. Esto abre la ventana Crear una nueva aplicación web ASP.NET Core.
10. En la ventana Crear una nueva aplicación web ASP.NET Core, asegúrese de que los siguientes valores estén seleccionados en los dos menús desplegables en la parte superior de la ventana:
0. .NET Core
  1. ASP.NET Core 3.1
11. Seleccione Aplicación web en el área Plantillas de proyecto en el centro de la ventana.
12. Desmarque la opción Configurar para HTTPS en la parte inferior derecha de la ventana.
13. Haga clic en el botón Crear en la esquina inferior derecha de la ventana Crear una nueva aplicación web ASP.NET Core.

En este punto, ha creado una aplicación web ASP.NET Core simple. Puede ejecutar esta aplicación en su entorno local para asegurarse de que la aplicación se esté ejecutando correctamente antes de publicar la aplicación en Azure.

Ahora necesita crear el grupo de recursos y el plan de App Service que hospeda App Service en Azure:

1. En su ventana de Visual Studio 2019, asegúrese de haber abierto la solución de la aplicación web que desea publicar en Azure.
2. En el lado derecho de la ventana de Visual Studio, en la ventana del Explorador de soluciones, haga clic con el botón derecho en el nombre del proyecto.
3. En el menú contextual, haga clic en Publicar. Esto abre la ventana Elija un destino de publicación.
4. En la ventana Elija un destino de publicación, asegúrese de que App Service esté seleccionado en la lista de destinos disponibles en el lado izquierdo de la ventana.

5. En la sección Azure App Service, en el lado derecho de la ventana, asegúrese de que la opción Crear nueva opción esté seleccionada.
6. En la esquina inferior derecha de la ventana, haga clic en el botón Crear perfil, que abre la ventana Crear servicio de aplicaciones.
7. En la ventana Crear App Service, agregue una nueva cuenta de Azure. Esta cuenta debe tener suficientes privilegios en la suscripción para crear nuevos grupos de recursos, servicios de aplicaciones y un plan de servicios de aplicaciones.
8. Una vez que haya agregado una cuenta válida, puede configurar los ajustes para publicar su aplicación web, como se muestra en la Figura 1-4.



**Figura 1-4** Creación de un servicio de aplicación

9. En el cuadro de texto Nombre de la aplicación, ingrese un nombre para el Servicio de la aplicación. De forma predeterminada, este nombre coincide con el nombre que le dio a su proyecto.
10. En el menú desplegable Suscripción, seleccione la suscripción en la que desea crear el Servicio de aplicaciones.
11. En el menú desplegable Grupo de recursos, seleccione el grupo de recursos en el que desea crear App Service y el plan de App Service. Si necesita crear un nuevo grupo de recursos, puede

hacerlo haciendo clic en el enlace Nuevo en el lado derecho del menú desplegable.

12. A la derecha del menú desplegable Plan de alojamiento, haga clic en el enlace Nuevo para abrir la ventana Configurar plan de alojamiento.
13. En la ventana Configurar plan de alojamiento, escriba un nombre para el plan de App Service en el cuadro de texto Plan de App Service.
14. Seleccione una región del menú desplegable Ubicación.
15. Seleccione un tamaño de máquina virtual en el menú desplegable Tamaño.
16. Haga clic en el botón Aceptar en la esquina inferior derecha de la ventana. Esto cierra la ventana Configurar plan de alojamiento.
17. En la esquina inferior derecha de la ventana Crear App Service, haga clic en el botón Crear. Esto inicia la creación de los recursos necesarios y la carga del código en App Service.
18. Una vez finalizado el proceso de publicación, Visual Studio abre su navegador web predeterminado con la URL del App Service recién implementado. Esta URL tendrá la estructura https://<your\_app\_service\_name>.azurewebsites.net.

Dependiendo del nivel de precios que seleccionó, algunas funciones están habilitadas, como configurar dominios personalizados o configurar conexiones SSL para sus aplicaciones web. Para la implementación de producción, debe usar los niveles de precios Estándar o Premium. A medida que sus necesidades de función cambien, puede elegir diferentes niveles de precios. Puede comenzar utilizando el nivel gratuito, F1, en las primeras etapas de su implementación y luego aumentar a un nivel S1 o P1 si necesita hacer copias de seguridad de su aplicación web o necesita usar ranuras de implementación.

Incluso si los niveles de precios premium no se ajustan a los requisitos de su computadora, aún puede implementar un entorno dedicado y aislado, llamado nivel de precios aislado. Este nivel le proporciona máquinas virtuales dedicadas que se ejecutan sobre redes virtuales dedicadas donde puede alcanzar el nivel máximo de capacidades de escalamiento horizontal. Tenga en cuenta que no puede usar el nivel compartido D1 para implementar un plan de Servicio de aplicaciones de Linux.



### *Sugerencia para el examen*

Dado que Azure App Service no admite las mismas características para Linux y Windows, no puede combinar aplicaciones de Windows y Linux en el mismo grupo de recursos en la misma región. Para obtener más información sobre las limitaciones de los servicios de aplicaciones de Linux, revise el siguiente artículo: <https://docs.microsoft.com/en-us/azure/app-service/containers/app-service-linux-intro>.

### *Habilitar el registro de diagnósticos*

La resolución de problemas y el diagnóstico del comportamiento de una aplicación es una operación fundamental en el ciclo de vida de cada aplicación. Esto es especialmente cierto si está desarrollando su aplicación. Azure App Service le proporciona algunos mecanismos para habilitar el registro de diagnósticos en diferentes niveles que pueden afectar su aplicación:

- **Diagnóstico del servidor web** Estos son registros de mensajes generados desde el propio servidor web. Puede habilitar tres tipos diferentes de registros:
  - **Registro detallado de errores** Este registro contiene información detallada para cualquier solicitud que dé como resultado un código de estado HTTP 400 o superior. Cuando ocurre un error 400, se genera un nuevo archivo HTML que contiene toda la información sobre el error. Se genera un archivo HTML separado para cada error. Estos archivos se almacenan en el sistema de archivos de la instancia en la que se ejecuta la aplicación web. Se pueden almacenar un máximo de 50 archivos de error. Cuando se alcanza este límite, los 26 archivos más antiguos se eliminan automáticamente del sistema de archivos.
  - **Seguimiento de solicitudes fallidas** Este registro contiene información detallada sobre solicitudes fallidas al servidor. Esta información contiene un rastro de los componentes de IIS que participaron en el procesamiento de la solicitud. También contiene el tiempo que tarda cada componente de IIS. Estos registros se almacenan en el

sistema de archivos. El sistema crea una nueva carpeta para cada nuevo error, aplicando las mismas políticas de retención que para el registro detallado de errores.

- Registro del **servidor web** Este registro registra la información de transacciones HTTP para las solicitudes realizadas al servidor web. La información se almacena utilizando el formato de archivo de registro extendido W3C. Puede configurar políticas de retención personalizadas para estos archivos de registro. De forma predeterminada, estos registros de diagnóstico nunca se eliminan, pero están restringidos por el espacio que pueden usar en el sistema de archivos. La cuota de espacio predeterminada es de 35 MB.
- **Diagnóstico de aplicaciones** Puede enviar un mensaje de registro directamente desde su código al sistema de registro. Utiliza el sistema de registro estándar del idioma que utiliza en su aplicación para enviar mensajes a los registros de diagnóstico de la aplicación. Esto es diferente de Application Insights porque los diagnósticos de la aplicación son solo información registrada que te registras desde tu aplicación. Si desea que su aplicación envíe registros a Application Insights, debe agregar el SDK de Application Insights a su aplicación.
- **Diagnóstico de implementación** Este registro se habilita automáticamente para usted y recopila toda la información relacionada con la implementación de su aplicación. Por lo general, usa este registro para solucionar fallas durante el proceso de implementación, especialmente si utiliza scripts de implementación personalizados.

Puede habilitar los diferentes registros de diagnóstico, que se muestran en la [Figura 1-5](#), mediante Azure Portal. Cuando habilita el registro de aplicaciones, puede seleccionar el nivel de registro de errores que se registrará en los archivos. Estos niveles de error son

- **Desactivado** No se registran errores.
- **Se registran las categorías Error** Crítico y Error.
- **Advertencia** Registra las categorías Advertencia, Error y Crítico.

- **La información** registra las categorías de registro Información, Advertencia, Error y Crítico.
- **Detallado** Registra todas las categorías de registro (seguimiento, depuración, información, advertencia, error y crítico).



**Figura 1-5** Habilitación del registro de diagnósticos

Cuando configura el registro de la aplicación, puede establecer la ubicación para almacenar los archivos de registro. Puede elegir entre guardar los registros en el sistema de archivos o usar Blob Storage. El almacenamiento de registros de aplicaciones en el sistema de archivos está destinado a fines de depuración. Si habilita esta opción, se deshabilitará automáticamente después de 12 horas. Si necesita habilitar el registro de la aplicación durante un período más largo, debe guardar los archivos de registro en Blob Storage. Cuando configura el registro de la aplicación para almacenar los archivos de registro en Blob Storage, también puede proporcionar un período de retención en días. Cuando los archivos de registro son más antiguos que el valor configurado para el período de retención, los archivos se eliminan automáticamente. De forma predeterminada, no hay un período de retención establecido.

Si configura el registro de la aplicación o del servidor web para almacenar los archivos de registro en el sistema de archivos, el sistema crea la siguiente estructura para los archivos de registro:

- **/ LogFiles / Application** / Esta carpeta contiene los archivos de registro del registro de la aplicación.
- **/ LogFiles / W3SVC ##### /** Esta carpeta contiene los archivos de los seguimientos de solicitudes fallidas. La carpeta contiene un archivo XSL y varios archivos XML. Los archivos XML contienen la información de seguimiento real, mientras que el archivo XSL proporciona la funcionalidad de formato y filtrado para el contenido almacenado en los archivos XML.
- **/ LogFiles / DetailErrors** / Esta carpeta contiene los archivos \*.htm relacionados con los registros detallados de errores.
- **/ LogFiles / http / RawLogs** / Esta carpeta contiene los registros del servidor web en formato de registro extendido W3C.
- **/ LogFiles / Git** Esta carpeta contiene el registro generado durante la implementación de la aplicación. También puede encontrar archivos de implementación en la carpeta D:\home\site\deployments.

Necesita esta estructura de carpetas cuando desee descargar los archivos de registro. Puede usar dos mecanismos diferentes para descargar los archivos de registro: FTP / S o CLI de Azure. El siguiente comando muestra cómo descargar archivos de registro al directorio de trabajo actual:

[Haga clic aquí para ver la imagen del código](#)

```
az webapp log download --resource-group <nombre del grupo de recursos> --name <nombre de la aplicación>
```

Los registros de la aplicación *<nombre de la aplicación>* se comprimen automáticamente en un archivo llamado webapp\_logs.zip. Luego, este archivo se descarga en el mismo directorio donde ejecutó el comando. Puede utilizar el parámetro opcional --log-file para descargar los archivos de registro en una ruta diferente en un archivo zip diferente.

Hay situaciones en las que es posible que deba ver los registros de su aplicación casi en tiempo real. Para estas situaciones, App Service le

proporciona flujos de registro. Con la transmisión, puede ver los mensajes de registro a medida que se guardan en los archivos de registro. Cualquier archivo de texto almacenado en la carpeta D:\home\LogFiles\ también se muestra en el flujo de registro. Puede ver las secuencias de registros mediante el visor incrustado en Azure Portal, en el elemento Log Stream en la sección de supervisión de su App Service. También puede usar el siguiente comando de la CLI de Azure para ver los registros de su aplicación o servidor web en transmisión:

Haga clic aquí para ver la imagen del código

```
az webapp log tail --resource-group <nombre del grupo de recursos> --name <nombre de la aplicación>
```

#### *¿Necesita más revisión? Integrar registros con Azure Monitor*

También puede enviar la información de diagnóstico desde sus servicios de aplicaciones de Windows o Linux a Azure Monitor. En el momento de escribir este artículo, esta función se encuentra en versión preliminar. Puede obtener más información sobre cómo integrar sus registros de Azure App Service con Azure Monitor revisando el siguiente artículo: <https://azure.github.io/AppService/2019/11/01/App-Service-Integration-with-Azure-Monitor.html>.



#### *Sugerencia para el examen*

Cuando planee configurar el registro de la aplicación, debe considerar que no todos los códigos de los lenguajes de programación pueden escribir la información del registro en Blob Storage. Puede usar Blob Storage solo con registros de aplicaciones .NET. Si usa Java, PHP, Node.js o Python, debe usar la opción del sistema de archivos de registro de la aplicación.

#### *Implementar código en una aplicación web*

Como parte del ciclo de vida de desarrollo típico de su aplicación, hay un punto en el que debe implementar su código en un servicio de aplicaciones de Azure. La sección "[Crear una aplicación web de Azure App Service](#)" que aparece anteriormente en este capítulo revisa cómo implementar el código directamente desde Visual Studio 2019. Esta sección explica cómo implementar su código utilizando otras alternativas más adecuadas para la implementación continua o los flujos de trabajo de integración continua.

Cuando está desarrollando su aplicación web, necesita probar su código tanto en su entorno local como en entornos de desarrollo o prueba que son similares al entorno de producción. Comenzando con el nivel de precios estándar, Azure App Service le proporciona las ranuras de implementación. Estas ranuras son implementaciones de su aplicación web que residen en el mismo App Service de su aplicación web. Una ranura de implementación tiene su configuración y nombre de host. Puede utilizar estas ranuras de implementación adicionales para probar su código antes de pasar a la ranura de producción. El principal beneficio de usar estas ranuras de implementación es que puede intercambiar estas ranuras sin ningún tiempo de inactividad. Incluso puede configurar un intercambio automático de las ranuras utilizando Auto Swap.

Cuando planea implementar su aplicación web en un App Service, Azure le ofrece varias opciones:

- **Archivos ZIP o WAR** Cuando desee implementar su aplicación, puede empaquetar todos sus archivos en un paquete ZIP o WAR. Con el servicio Kudu, puede implementar su código en App Service.
- **FTP** Puede copiar los archivos de su aplicación directamente en App Service utilizando el punto final FTP / S configurado de forma predeterminada en App Service.
- **Sincronización en la nube** Impulsado por el motor de implementación de Kudu, este método le permite tener su código en una carpeta de OneDrive o Dropbox, y sincroniza esa carpeta con App Service.
- **Implementación continua** Azure puede integrarse con repositorios de GitHub, BitBucket o Azure Repos para implementar las actualizaciones más recientes de su aplicación en App Service. Dependiendo del servicio, puede usar el servidor de compilación de Kudu o Azure Pipelines para implementar un proceso de entrega continua. También puede configurar la integración manualmente con otros repositorios en la nube como GitLab.
- **Su repositorio de Git local** Puede configurar su App Service como un repositorio remoto para su repositorio de Git local y enviar su código a Azure. Luego, el servidor de compilación de

Kudu compila automáticamente su código y lo implementa en el Servicio de aplicaciones.

- **Plantilla ARM** Puede usar Visual Studio y una plantilla ARM para implementar su código en un App Service.

*Nota Kudu*

Kudu es la plataforma que se encarga de las implementaciones de Git en Azure App Service. Puede encontrar información más detallada en su sitio de GitHub en <https://github.com/projectkudu/kudu/wiki>.

El siguiente ejemplo muestra cómo implementar su código en una aplicación web mediante Azure Pipelines. Para este ejemplo, debe implementar su código en un repositorio de Azure Repos. Si aún no tiene su código en un repositorio de Azure Repos, puede usar el siguiente artículo para crear un nuevo repositorio: <https://docs.microsoft.com/en-us/azure/devops/repos/git/creatingrepo>.

1. Abra Azure Portal (<https://portal.azure.com>).
2. En el cuadro de texto de búsqueda en la parte superior de Azure Portal, escriba el nombre de su App Service.
3. En la lista de resultados debajo del cuadro de texto de búsqueda, haga clic en su Servicio de aplicaciones.
4. En la hoja de App Service, en el menú del lado izquierdo de la página, en la sección Implementación, haga clic en Centro de implementación.
5. En el Centro de implementación, que se muestra en la [Figura 1-6](#), en el paso de control de código fuente, haga clic en Azure Repos.
6. En la parte inferior de la página, haga clic en Continuar.
7. En el paso Compilar proveedor, haga clic en Azure Pipelines (versión preliminar) y haga clic en Continuar en la parte inferior de la página.
8. En el paso Configurar, en la sección Código, seleccione su organización en el menú desplegable Organización de Azure DevOps.
9. En el menú desplegable Proyecto, seleccione el proyecto con el repositorio que contiene el código que desea implementar en Azure App Service.

10. En el menú desplegable Repositorio, seleccione el repositorio que contiene su código.
11. En el menú desplegable Rama, seleccione la rama que desea implementar.
12. En la sección Compilar, en el menú desplegable Marco de aplicación web, seleccione el marco apropiado para su código.
13. Haga clic en el botón Continuar en la parte inferior de la página.
14. En el paso Resumen, revise los detalles de la configuración.
15. En la parte inferior de la página, haga clic en Finalizar.

En este punto, ha configurado una canalización de Azure en su Azure Repo que implementa automáticamente su código en Azure App Service. Cuando realiza una confirmación en la rama que seleccionó en el ejemplo anterior, Azure Pipeline usa automáticamente el código en la última confirmación. Una vez que haya configurado la implementación continua para su Azure App Service, puede revisar el estado de las diferentes implementaciones en el Centro de implementación de su Azure App Service.

The screenshot shows the 'Deployment Center' section of the Azure App Service portal. At the top, there's a gear icon and the title 'Deployment Center'. Below it, a sub-header reads: 'App Service Deployment Center enables you to choose the location of your code as well as options for build and deployment to'. A horizontal line with two numbered circles (1 and 2) connects 'SOURCE CONTROL' and 'BUILD PROVIDER'. Under 'Continuous Deployment (CI / CD)', there are two cards: one for 'Azure Repos' and one for 'GitHub'. The 'Azure Repos' card has a dashed border and contains the text: 'Configure continuous integration with an Azure Repo, part of Azure DevOps Services (formerly known as VSTS.)'. The 'GitHub' card contains the text: 'Configure continuous integration with a GitHub repo.' and a note below it says 'Not Authorized'.

**Figura 1-6** Habilitación del registro de diagnósticos

*¿Necesita más revisión? Ejemplos de implementación de servicios de aplicaciones*

Puede revisar ejemplos de los diferentes tipos de implementaciones siguiendo los casos publicados en los siguientes artículos:

- Implemente archivos ZIP o WAR: <https://docs.microsoft.com/en-us/azure/app-service/deploy-zip>
- Implementar mediante sincronización en la nube: <https://docs.microsoft.com/en-us/azure/app-service/deploy-content-sync>
- Implementar desde Git local: <https://docs.microsoft.com/en-us/azure/app-service/deploy-local-git>

Cuando implementa su código en un servicio de aplicaciones de Azure, puede hacerlo en diferentes ranuras de implementación. Una ranura de implementación es una aplicación en vivo que es diferente de la aplicación principal. Cada ranura de implementación tiene su propio nombre de host y grupo de configuraciones. Por lo general, utiliza las distintas ranuras como entorno de ensayo para realizar pruebas. Puede cambiar entre las diferentes ranuras sin perder solicitudes. Las ranuras

de implementación están disponibles solo para los niveles de servicios de aplicaciones estándar, premium y aislado.

#### *¿Necesita más revisión? Ranuras de implementación*

Puede obtener más información sobre cómo trabajar con ranuras de implementación revisando el siguiente artículo: <https://docs.microsoft.com/en-us/azure/app-service/deploy-staging-slots>.



#### *Sugerencia para el examen*

Puede usar diferentes mecanismos para implementar su código en un servicio de aplicaciones de Azure. Si decide utilizar sistemas de implementación continua para implementar su código, como Azure Repos o GitHub, recuerde que debe autorizar su sistema de implementación continua antes de poder realizar cualquier implementación.

#### *Configure los ajustes de la aplicación web, incluidos SSL, API y cadenas de conexión*

Una vez que haya creado su aplicación App Service, puede administrar los diversos parámetros que pueden afectar su aplicación. Puede acceder a esta configuración en el menú Configuración en la sección Configuración en la hoja de App Service. Los parámetros disponibles están agrupados por las siguientes cuatro categorías principales de configuración:

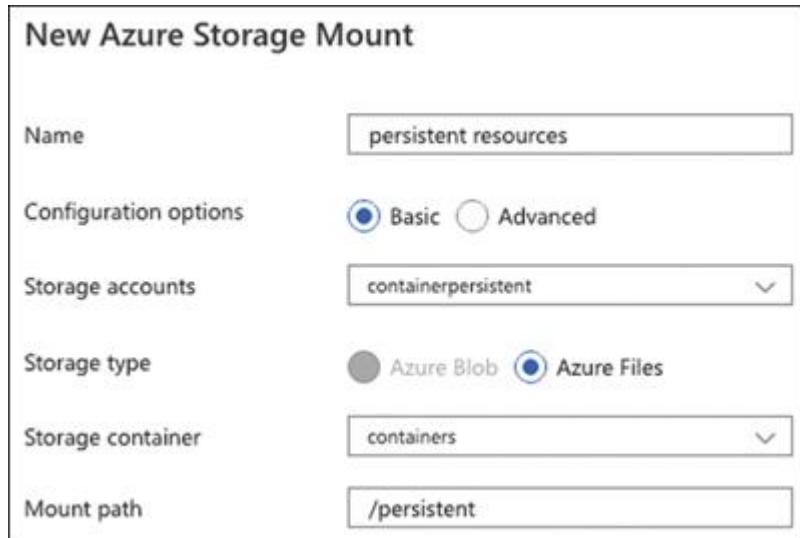
- **Configuración de la aplicación** Puede configurar las variables de entorno que se pasan a su código. El uso de esta configuración equivale a establecer las mismas variables en la sección `<appSettings>` en los archivos Web.config o appsettings.json en un proyecto ASP.NET o ASP.NET Core. Si establece una variable en esta sección que coincide con una variable en los archivos Web.config o appsettings.json, el valor de las variables en los archivos de configuración se reemplazará con el valor en la configuración de la aplicación web de Azure. Estas configuraciones siempre se cifran en reposo, es decir, cuando se almacenan.
- **Cadenas de conexión** Utilice esta sección para configurar las cadenas de conexión para la base de datos que su código necesita utilizar. Esto es similar a usar la sección `<connectionString>` en los

archivos Web.config o appsettings.json en proyectos ASP.NET o ASP.NET Core.

- **Configuración general** Estas configuraciones están relacionadas con el entorno y la plataforma en la que se ejecuta su aplicación. Puede controlar los siguientes elementos:
  - **Configuración de la pila** Usted configura la pila y la versión utilizada para ejecutar su aplicación.
    - **Pila** Puede elegir entre .NET Core, .NET, Java, PHP y Python.
    - **Versión** Ésta es la versión de la pila que eligió en la configuración anterior.
  - **Configuración de la plataforma** Esta sección controla las diferentes configuraciones relacionadas con la plataforma que ejecuta su código:
    - **Plataforma** Esta configuración controla si su aplicación se ejecuta en una plataforma de 32 o 64 bits.
    - **Versión de canalización administrada** Configura el modo de canalización de IIS. Debe establecer esto en *clásico* si necesita ejecutar una aplicación heredada que requiere una versión anterior de IIS.
    - **Estado de FTP** Configura la posibilidad de usar FTP o FTPS para implementar su aplicación web en Azure App Service. De forma predeterminada, los protocolos FTP y FTPS están habilitados.
    - **Versión HTTP** Esto habilita el protocolo HTTPS / 2.
    - **Web Sockets** Si su aplicación usa SignalR o socket.io, necesita habilitar los Web Sockets.
    - **Always On** Habilitar esta configuración significa que su aplicación siempre está cargada. De forma predeterminada, la aplicación se descarga si está inactiva durante algún tiempo. Puede configurar este tiempo de espera inactivo en el archivo de proyecto host.json. El valor predeterminado para App Service es 30 minutos.

- **Afinidad de ARR** Habilitar esta configuración garantiza que las solicitudes de los clientes se enruten a la misma instancia durante la duración de la sesión. Esta configuración es útil para aplicaciones con estado, pero puede afectar negativamente a las aplicaciones sin estado.
- **Depuración** Habilite las opciones de depuración remota para que pueda conectarse directamente desde su IDE a Azure App Service para depurar sus aplicaciones ASP.NET, ASP.NET Core o Node.js. Esta opción se apaga automáticamente después de 48 horas.
- **Certificados de cliente entrantes** Si necesita autenticación SSL mutua para su aplicación, debe habilitar esta opción.
- **Documentos predeterminados** Esta configuración configura qué página web se muestra en la URL raíz de su aplicación. Puede establecer una lista de diferentes documentos predeterminados, donde la primera coincidencia válida se muestra en la URL raíz de su sitio web.
- **Asignaciones de ruta** La configuración de esta sección depende del tipo de sistema operativo que elija para su Azure App Service:
  - **Aplicaciones de Windows (sin contener)** Estas configuraciones son similares a las que puede encontrar en IIS:
    - **Asignaciones de manejadores** Puede configurar procesadores de scripts personalizados para diferentes extensiones de archivo.
    - **Aplicaciones y directorios virtuales** Esta configuración le permite agregar directorios o aplicaciones virtuales adicionales a su App Service.
    - **Aplicaciones en contenedores** Puede configurar los puntos de montaje que se adjuntan a los contenedores durante la ejecución. Puede adjuntar hasta cinco archivos de Azure o puntos de montaje de blobs por aplicación. La figura

1-7 muestra el cuadro de diálogo para configurar un punto de montaje de archivos de Azure.



**Figura 1-7** Habilitación del registro de diagnósticos

Una vez que haya creado una configuración de aplicación o una variable de cadena de conexión, puede acceder a estos valores desde su código utilizando variables de entorno. El siguiente fragmento de código muestra cómo acceder a una configuración de aplicación llamada testing-var1 y una cadena de conexión llamada testing-connsql1 desde una página PHP:

Haga clic aquí para ver la imagen del código

```
<? php  
    $ testing_var1 = getenv ('APPSETTING_testing-var1')  
  
    $ cadena_conexión = getenv ('SQLAZURECONNSTR_testing-  
connsql1')  
  
?>
```

Como puede ver en el fragmento de código anterior, debe anteponer la cadena APPSETTING\_ al nombre de la variable de configuración de su aplicación. En el caso de las cadenas de conexión, la cadena que debe anteponer al nombre de la cadena de conexión depende del tipo que configure en la cadena de conexión en Azure Portal:

- **Bases de datos SQL** SQLAZURECONNSTR\_
- **SQL Server** SQLCONNSTR\_

- **MySQL MYSQLCONNSTR\_**
- **PostgreSQL POSTGRESQLCONNSTR\_**
- **CUSTOMCONNSTR\_personalizado**

Para las aplicaciones ASP.NET, también puede acceder a la configuración de la aplicación y las cadenas de conexión mediante el ConfigurationManager tradicional. Si decide utilizar ConfigurationManager, no es necesario que anteponga ninguna cadena al nombre de la configuración de la aplicación o la cadena de conexión. El siguiente fragmento de código muestra cómo acceder a la configuración de la aplicación o la cadena de conexión desde el código ASP.NET:

[Haga clic aquí para ver la imagen del código](#)

```
System.Configuration.ConfigurationManager.AppSettings  
["testing-var1"]  
  
System.Configuration.ConfigurationManager.ConnectionStrings  
["testing-connsql1"]
```

Cuando configura una aplicación web de Azure para un entorno de producción, normalmente necesita proteger las conexiones con la aplicación web. También debe hacer que su aplicación web de Azure esté disponible a través de su propio dominio en lugar del dominio azurewebsites.net predeterminado. Puede hacerlo configurando el dominio personalizado y la configuración de SSL.

Utilice el siguiente procedimiento para configurar los ajustes de SSL para una aplicación web existente. Recuerde que la configuración de SSL solo está disponible para niveles de precios B1 o superiores:

1. Abra Azure Portal (<https://portal.azure.com>).
2. En el cuadro de texto Buscar recursos, escriba el nombre de su aplicación web de Azure.
3. En la lista de resultados, haga clic en el nombre de su aplicación web de Azure.
4. En la página de la aplicación web de Azure, en la lista de navegación del lado izquierdo de la página, haga clic en Dominio personalizado en la sección Configuración.
5. Haga clic en el botón Agregar dominio personalizado en el medio de la página Dominios personalizados.

6. En la hoja Agregar dominio personalizado en el lado derecho de la página, en el cuadro de texto Dominio personalizado, escriba un nombre para su dominio.
7. Haga clic en el botón validar y siga las instrucciones para validar su dominio.
8. Una vez que haya validado su dominio personalizado, haga clic en el botón Agregar dominio personalizado en la hoja Agregar dominio personalizado.
9. Haga clic en Configuración de TLS / SSL en la sección Configuración en el lado izquierdo de la página de la aplicación web de Azure.
10. En la página Configuración de TLS / SSL, en las secciones Enlaces TLS / SSL, haga clic en Agregar enlace TLS / SSL. Tenga en cuenta que necesita el certificado pfx adecuado para configurar este enlace. Puede importar un certificado existente o comprar uno nuevo.
11. En la hoja Enlace TLS / SSL, en el menú desplegable Dominio personalizado, seleccione el dominio personalizado que agregó en el paso 8.
12. En el menú desplegable Huella digital de certificado privado, seleccione un certificado válido para su dominio personalizado.
13. En el menú desplegable TLS / SSL Type, seleccione SNI SSL.
14. En la parte inferior de la hoja Enlace TLS / SSL, haga clic en el botón Agregar enlace.

#### **Nota Azure Storage en el servicio de aplicaciones**

En el momento de redactar este documento, el uso de Azure Storage en App Services es una característica que se encuentra en versión preliminar y no debe usarse en entornos de producción.

#### **¿Necesita más revisión? Configurar los ajustes de la aplicación**

Puede revisar más detalles sobre cómo configurar las diferentes configuraciones en su aplicación web de Azure revisando el artículo en <https://docs.microsoft.com/en-us/azure/app-service/configure-common> .



**Sugerencia para el examen**

Recuerde que la configuración que configura en la sección Configuración de la aplicación sobrescribe los valores que configura en `<appSettings>` o `<connectionStrings>` en sus archivos Web.config o appsettings.json.

*Implementar reglas de escalado automático, incluido el escalado automático programado y el escalado por métricas operativas o del sistema.*

Uno de los mayores desafíos a los que se enfrenta cuando implementa su aplicación en un entorno de producción es asegurarse de proporcionar suficientes recursos para que su aplicación tenga el rendimiento esperado. Determinar la cantidad de recursos que debe asignar es la gran pregunta cuando se trata de configurar los recursos para su aplicación. Si asigna demasiados recursos, su aplicación funcionará bien durante los picos de uso, pero potencialmente está desperdiando recursos. Si asigna menos recursos, está ahorrando recursos, pero es posible que su aplicación no funcione bien durante los picos de uso. Otro problema con el rendimiento de la aplicación es queEs difícil anticipar cuándo puede ocurrir un pico de uso intenso. Esta afirmación es especialmente cierta para las aplicaciones que tienen patrones de uso impredecibles.

Afortunadamente, Azure proporciona un mecanismo para abordar este problema. Puede asignar dinámicamente más recursos a su aplicación cuando los necesite. El ajuste de escala automática es la acción de agregar o quitar recursos automáticamente a un servicio de Azure y proporcionar la potencia informática necesaria para su aplicación en cada situación. Una aplicación puede escalar de dos formas diferentes:

- **Verticalmente** Usted agrega más potencia informática agregando más memoria, recursos de CPU e IOPS a la aplicación. Al final del día, su aplicación se ejecuta en una máquina virtual. No importa si usa una máquina virtual IaaS, Azure App Service o Azure Service Fabric, está usando máquinas virtuales bajo el capó. Escalar verticalmente una aplicación significa pasar de una VM más pequeña a una VM más grande y agregar más memoria, CPU e IOPS. El escalado vertical requiere detener el sistema mientras la máquina virtual cambia de tamaño. Este tipo de escalado también se conoce como "escalado hacia arriba y hacia abajo".

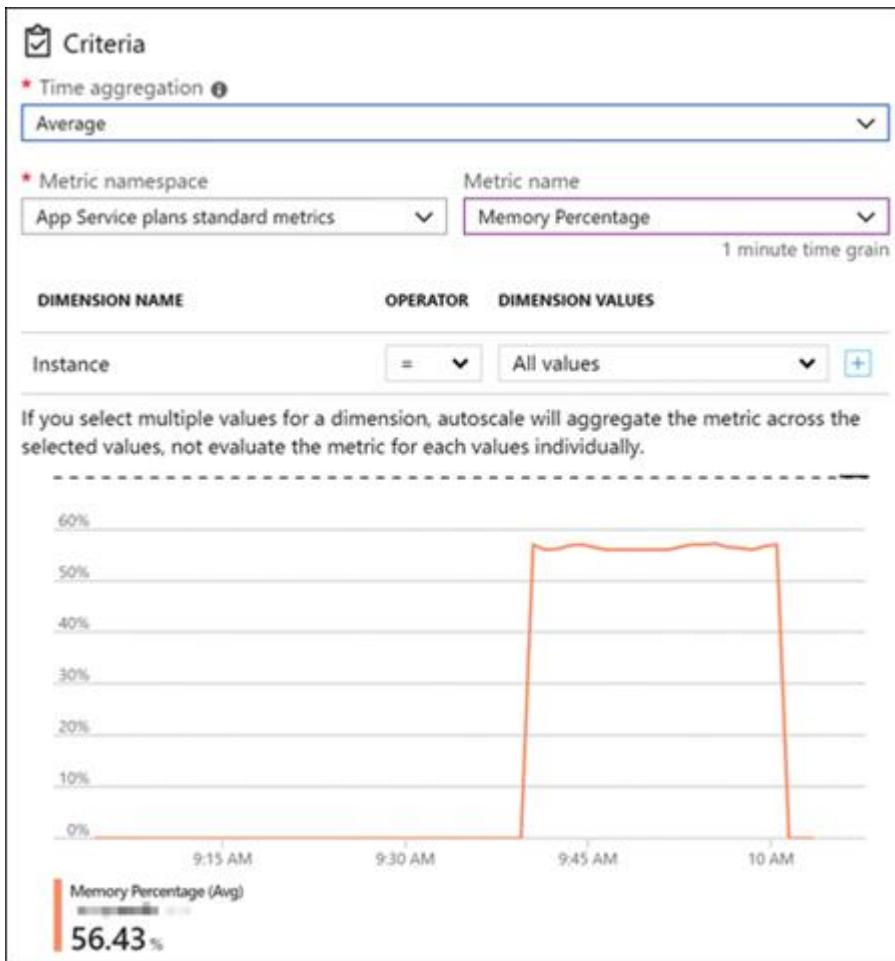
- **Horizontalmente** También puede escalar su aplicación creando o eliminando instancias de su aplicación. Cada instancia de su aplicación se ejecuta en una máquina virtual que forma parte de un conjunto de escalado de máquinas virtuales. El servicio de Azure correspondiente administra automáticamente las máquinas virtuales en el conjunto de escalado. Todas estas instancias de su aplicación funcionan juntas para brindar el mismo servicio. La ventaja de escalar horizontalmente es que la disponibilidad de su aplicación no se ve afectada porque no es necesario reiniciar todas las instancias de su aplicación que brindan el servicio. Este tipo de escalado también se conoce como "escalado hacia afuera y hacia adentro".

Cuando trabaja con el escalado automático, nos referimos al escalado horizontal porque el escalado vertical requiere la interrupción del servicio mientras Azure Resource Manager cambia el tamaño de la máquina virtual. Por ese motivo, el escalado vertical no es adecuado para el escalado automático.

El ajuste de escala automático se configura en función de algunos criterios que debe cumplir su aplicación para proporcionar el nivel de rendimiento adecuado. Configure estos criterios en Azure mediante reglas de ajuste de escala automática. Una regla define qué métrica debe usar Azure Monitor para realizar el ajuste de escala automática. Cuando esa métrica alcanza la condición configurada, Azure realiza automáticamente la acción configurada para esa regla. La acción típica que puede pensar que debería realizar la regla es agregar o eliminar una máquina virtual al conjunto de escalado, pero también puede realizar otras acciones como enviar un correo electrónico o realizar una solicitud HTTP a un webhook. Puede configurar tres tipos diferentes de reglas cuando trabaja con las reglas de ajuste de escala automática:

- **Basado en el tiempo** Azure Monitor ejecuta la regla de ajuste de escala automática según una programación. Por ejemplo, si su aplicación requiere más recursos durante la primera semana del mes, puede agregar más instancias y reducir la cantidad de recursos durante el resto del mes.
- **Basado en métricas** Configure el umbral para métricas estándar, como el uso de la CPU, la longitud de la cola HTTP o el porcentaje de uso de memoria, como se muestra en la [Figura 1-8](#).

- **Basado en personalización** Puede crear sus métricas en su aplicación, exponerlas mediante Application Insight y utilizarlas para las reglas de ajuste de escala automático.



**Figura 1-8** Configuración de una regla de autoescala basada en métricas

Solo puede usar el mecanismo de ajuste de escala automático integrado con un grupo limitado de tipos de recursos de Azure:

- **Máquinas virtuales de Azure** Puede aplicar el ajuste de escala automático mediante el uso de conjuntos de escalado de máquinas virtuales. Todas las máquinas virtuales de un conjunto de escalas se tratan como un grupo. Al utilizar el ajuste de escala automático, puede agregar máquinas virtuales al conjunto de escalado o eliminarlas.
- **Azure Service Fabric** Cuando crea un clúster de Azure Service Fabric, define diferentes tipos de nodos. Un conjunto de escalado de máquina virtual diferente admite cada tipo de nodo

que defina en un clúster de Azure Service Fabric. Puede aplicar el mismo tipo de reglas de ajuste de escala automático que utiliza en un conjunto de escalado de máquina virtual estándar.

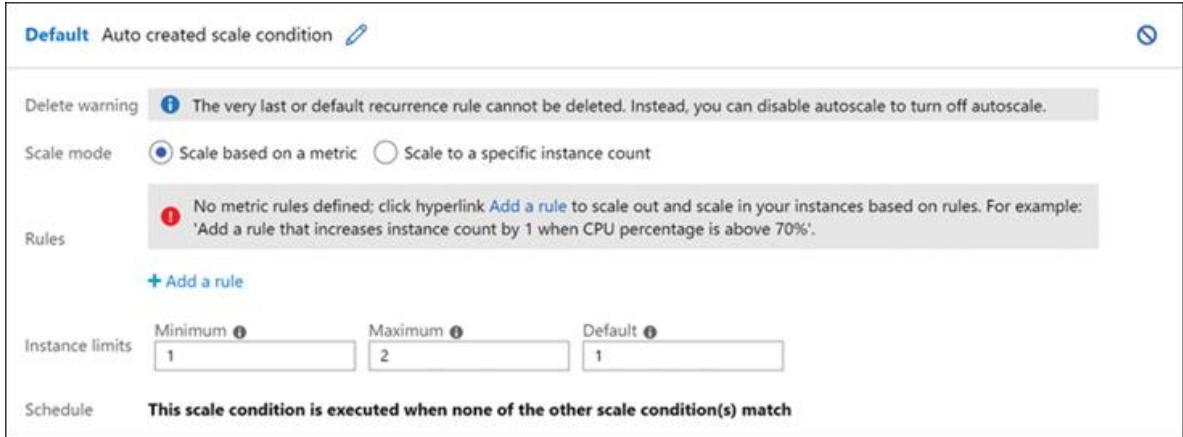
- **Servicio de aplicaciones de Azure** Este servicio tiene capacidades de ajuste de escala automática integradas que puede usar para agregar o quitar instancias al Servicio de aplicaciones de Azure. Las reglas de escala automática se aplican a todas las aplicaciones dentro de Azure App Service.
- **Servicios en la nube de Azure** Este servicio tiene capacidades de ajuste de escala automática integradas que puede usar para agregar o quitar recursos a los roles en el Servicio en la nube de Azure.

Cuando trabaja con la característica de escala automática en uno de los servicios de Azure admitidos, define una condición de perfil. Una condición de perfil define la regla que configuras para agregar o quitar recursos. También puede definir las instancias permitidas predeterminadas, mínimas y máximas para este perfil. Cuando define un mínimo y un máximo, su servicio no puede disminuir o crecer más allá de los límites que define en el perfil. También puedes configurar el perfil para escalar en función de un programa o en función de los valores de métricas integradas o personalizadas. Puede usar el siguiente procedimiento para agregar una regla de escala automática basada en métricas a Azure App Service. Esta regla agrega una instancia al plan de Azure App Service cuando el porcentaje promedio de uso de CPU es superior al 80 por ciento más de 10 minutos:

1. Abra Azure Portal (<https://portal.azure.com>).
2. En el cuadro de texto de búsqueda en la parte superior de Azure Portal, escriba el nombre de su Azure App Service.
3. Haga clic en el nombre de su Azure App Service en la lista de resultados.
4. En la hoja de Azure App Service, en el menú de navegación del lado izquierdo de la hoja, haga clic en la opción Escalado horizontal (plan de App Service) en la sección Configuración.
5. En la hoja Scale-Out (App Service Plan), en la pestaña Configurar, haga clic en el botón Custom Autoscale. Las reglas de

escala automática están disponibles solo para los planes de App Service que son de tamaño estándar o más grandes.

6. En la hoja Scale-Out (App Service Plan), en la pestaña Configurar, en la ventana Condición de escala creada automáticamente predeterminada que se muestra en la [Figura 1-9](#), haga clic en el enlace Agregar una regla.



**Figura 1-9** Configuración de una regla de autoescala basada en métricas

7. En el panel Regla de escala, en la sección Criterios, asegúrese de que el valor de Porcentaje de CPU esté seleccionado en el menú desplegable Nombre de métrica.
8. Asegúrese de que el valor Mayor que esté seleccionado en el menú desplegable Operador.
9. Escriba el valor **80** en el cuadro de texto Umbral métrico para activar la acción de escala.
10. En la sección Acción, asegúrese de que el valor del recuento de instancias esté establecido en **1**.
11. Haga clic en el botón Agregar en la parte inferior del panel.
12. En la hoja Scale-Out (App Service Plan), en la condición de perfil predeterminado, establezca el límite máximo de instancias en **3**.
13. Haga clic en el botón Guardar en la esquina superior izquierda de la hoja.

*Note Scale-Out / Scale-In*

El procedimiento anterior muestra cómo agregar una instancia al plan de App Service (está escalando horizontalmente el plan de App Service) pero no quita la instancia adicional una vez que el porcentaje de CPU cae por debajo del umbral configurado. Debe agregar una regla de ampliación para eliminar las instancias adicionales una vez que no sean necesarias. Configure una regla de escalamiento interno de la misma manera que lo hizo para la regla de escalamiento horizontal. Simplemente configure el menú desplegable Operación en el valor Disminuir recuento hasta.

Puede usar diferentes patrones de autoescala comunes, según la configuración que he revisado hasta ahora:

- **Escale según la CPU** Escale su servicio (Azure App Service, VM Scale Set o Cloud Service) según la CPU. Debe configurar las reglas de escalabilidad horizontal y vertical para agregar y eliminar instancias al servicio. En este patrón, también establece un número mínimo y máximo de instancias.
- **Escale de manera diferente los días de semana** que los **fines de semana** Utilice este patrón cuando espera que el uso principal de su aplicación ocurra en los días de semana. Configura la condición de perfil predeterminada con un número fijo de instancias. Luego, configura otra condición de perfil para reducir el número de instancias durante los fines de semana.
- **Escale de manera diferente durante las vacaciones** Utilice la escala según el patrón de la CPU. Aún así, agrega una condición de perfil para agregar instancias adicionales durante las vacaciones o los días que son importantes para su negocio.
- **Escale según métricas personalizadas** Utilice este patrón con una aplicación web compuesta por tres capas: front-end, back-end y niveles de API. El front-end de un nivel de API se comunica con el nivel de back-end. Usted define sus métricas personalizadas en la aplicación web y las expone a Azure Monitor mediante Application Insights. Luego, puede usar estas métricas personalizadas para agregar más recursos a cualquiera de las tres capas.



### *Sugerencia para el examen*

El ajuste de escala automática le permite asignar recursos a su aplicación de manera eficiente. Las reglas de escala automática para agregar más instancias a su aplicación no eliminan esas instancias cuando no se

cumple la condición de la regla. Como práctica recomendada, si crea una regla de escalamiento horizontal para agregar instancias a un servicio, debe crear la regla de escalamiento hacia adentro opuesta para eliminar la instancia. Esto asegura que los recursos se asignen de manera eficiente a su aplicación.

#### *¿Necesita más revisión? Prácticas recomendadas de autoescala*

Puede encontrar más información sobre las mejores prácticas al configurar las reglas de autoescala revisando el artículo en <https://docs.microsoft.com/en-us/azure/azure-monitor/platform/autoscale-best-practices>.

#### *¿Necesita más revisión? Consideraciones de diseño de aplicaciones*

El simple hecho de agregar más recursos a su aplicación no garantiza que su aplicación funcione bien. Su aplicación debe conocer los nuevos recursos para aprovecharlos. Puede revisar algunas consideraciones de diseño de aplicaciones revisando el artículo en <https://docs.microsoft.com/en-us/azure/architecture/best-practices/auto-scaling#related-patterns-and-guidance>.

## HABILIDAD 1.3: IMPLEMENTAR FUNCIONES DE AZURE

---

Basado en Azure App Service, Azure Functions le permite ejecutar fragmentos de código que resuelven problemas particulares dentro de toda la aplicación. Usas estas funciones de la misma manera que puedes usar una clase o función dentro de tu código. Es decir, su función obtiene alguna entrada, ejecuta el fragmento de código y proporciona una salida.

La gran diferencia entre Azure Functions y otros modelos de servicios de aplicaciones es que con Azure Functions (con el nivel de precios de Consumo), se le cobra por segundo solo cuando su código se está ejecutando. Si usa App Service, se le cobrará por hora cuando el App Service Plan se esté ejecutando, incluso si no se está ejecutando ningún código. Debido a que Azure Functions se basa en App Service, también puede decidir ejecutar su función de Azure en su plan de App Service si ya tiene otros servicios de aplicaciones en ejecución.

### **Esta habilidad cubre cómo**

- Implementar enlaces de entrada y salida para una función
- Implementar activadores de funciones mediante el uso de operaciones de datos, temporizadores y webhooks

- Implementar funciones duraderas de Azure

### *Implementar enlaces de entrada y salida para una función*

Cuando escribe una función en su código, esa función puede requerir datos como información de entrada para realizar el trabajo que está escribiendo. La función también puede producir información de salida como resultado de las operaciones realizadas dentro de la función. Cuando trabaja con Azure Functions, es posible que también necesite estos flujos de datos de entrada y salida.

El enlace usa Azure Functions para conectar su función con el mundo externo sin codificar la conexión a los recursos externos. Una función de Azure puede tener una combinación de enlaces de entrada y salida, o no puede tener ningún enlace. Los enlaces pasan datos a la función como parámetros.

Aunque los desencadenantes y las vinculaciones están estrechamente relacionados, no debe confundirlos. Los disparadores son los eventos que hacen que la función comience su ejecución; Los enlaces son como la conexión a los datos necesarios para la función. Puedes ver la diferencia en este ejemplo:

Un servicio de publicador envía un evento (a un Event Grid que lee una nueva imagen que se ha cargado en Blob Storage) a una cuenta de Azure Storage. Tu función necesita leer estoImagen, procesarla y colocar cierta información en un documento de Cosmos DB. Cuando la imagen se ha procesado, su función también envía una notificación a la interfaz de usuario mediante SignalR.

En este ejemplo, puede encontrar un activador, un enlace de entrada y dos enlaces de salida:

- **Desencadenador** La cuadrícula de eventos debe configurarse como desencadenante de la función de Azure.
- **Enlace de entrada** Su función necesita leer la imagen que se cargó en Blob Storage. En este caso, debe usar Blob Storage como enlace de entrada.
- **Enlaces de salida** Su función necesita escribir un documento de Cosmos DB con los resultados del procesamiento de la imagen. Debe utilizar el enlace de salida de Cosmos DB. Su función

también necesita enviar una notificación a la interfaz de usuario mediante el enlace de salida de SignalR.

Dependiendo del lenguaje que use para programar su función de Azure, la forma en que declara un enlace cambia:

- **C #** Declaras enlaces y disparadores decorando métodos y parámetros.
- **Otro** Actualice el archivo de configuración function.json.

Al definir un enlace para funciones de lenguaje que no son de C #, debe definir su enlace utilizando los siguientes atributos mínimos requeridos:

- **tipo** Esta cadena representa el tipo de enlace. Por ejemplo, usaría eventHub cuando use un enlace de salida para Event Hub.
- **direction** Los únicos valores permitidos son in para enlaces de entrada y out para enlaces de salida. Algunas fijaciones también admiten la dirección especial inout.
- **nombre** La función utiliza este atributo para vincular los datos en la función. Por ejemplo, en JavaScript, la clave en una lista de clave-valor es un atributo.

Dependiendo del enlace específico que esté configurando, podría haber algunos atributos adicionales que deban definirse.

#### *Nota vinculaciones admitidas*

Para obtener una lista completa de enlaces compatibles, consulte el artículo en <https://docs.microsoft.com/en-us/azure/azure-functions/functions-triggers-bindings#supported-bindings>.

Antes de poder utilizar un enlace en su código, debe registrarlo. Si usa C # para sus funciones, puede hacerlo instalando el paquete NuGet apropiado. Para otros idiomas, debe instalar el paquete con el código de extensión utilizando la utilidad de línea de comandos func. El siguiente ejemplo instala la extensión de Service Bus en su entorno local para proyectos que no son de C #:

[Haga clic aquí para ver la imagen del código](#)

```
instalación de extensiones func -paquete  
Microsoft.Azure.WebJobs.ServiceBus
```

Si está desarrollando su función de Azure mediante Azure Portal, puede agregar los enlaces en la sección Integrar de su función. Cuando agrega un

enlace que no está instalado en su entorno, verá el mensaje de advertencia que se muestra en la [Figura 1-10](#). Puede instalar la extensión haciendo clic en el enlace Instalar.



**Figura 1-10** Instalación de una extensión de enlace

*¿Necesita más revisión? Instalar manualmente extensiones de enlace desde Azure Portal*

Cuando desarrolle su función de Azure con el portal de Azure, puede usar el editor estándar o el editor avanzado. Cuando usa el editor avanzado, puede editar directamente el archivo de configuración function.json. Si agrega nuevos enlaces con el editor avanzado, debe instalar manualmente las nuevas extensiones de enlace que agregó a function.json. Puede revisar el siguiente artículo para instalar manualmente extensiones de enlace desde Azure Portal en <https://docs.microsoft.com/en-us/azure/azure-functions/install-update-binding-extensions-manual>.

Si decide programar su función de Azure con C #, realice la configuración de los enlaces mediante decoradores para la función y los parámetros. El archivo function.json se construye automáticamente en función de la información que proporciona en su código. [El Listado 1-7](#) muestra cómo configurar enlaces de entrada y salida usando decoradores de parámetros.

#### **Listado 1-7** Configuración de enlaces de entrada y salida

[Haga clic aquí para ver la imagen del código](#)

---

```
// C # ASP.NET Core  
  
using el sistema;  
  
using System.IO;  
  
utilizando Microsoft.Azure.WebJobs;
```

```
utilizando Microsoft.Extensions.Logging;
utilizando
Microsoft.Azure.WebJobs.Extensions.SignalRService;
utilizando Microsoft.Azure.WebJobs.Extensions.EventGrid;
utilizando Microsoft.Azure.EventGrid.Models;
usando System.Threading.Tasks;

espacio de nombres Company.Functions

{
    clase estática pública BlobTriggerCSharp
    {
        [FunctionName ("BlobTriggerCSharp")]
        ejecución de tarea estática pública (
            [EventGridTrigger] EventGridEvent
eventGridEvent,
[Blob ("{data.url}", FileAccess.Read, Connection =
"ImagesBlobStorage")] Transmisión
imageBlob,
[CosmosDB (
    databaseName: "GIS",
    collectionName: "Processed_images",
    ConnectionStringSetting =
"CosmosDBConnection")] documento dinámico,
```

```

[SignalR (HubName = "notificaciones")] IAsyncCollector<SignalRMessage> signalRMessages,
    ILogger log)

{
    document = new {Description =
eventGridEvent.Topic,
id = Guid.NewGuid ()};

    log.LogInformation ($ "C # Función de activación de blobs
Evento procesado \ n Tema: {eventGridEvent.

Tema} \ n Asunto: {eventGridEvent.Subject} ");

    return signalRMessages.AddAsync (
        nuevo SignalRMessage
    {
        Target = "newMessage",
        Argumentos = nuevo []
{eventGridEvent.Subject}

    });
}
}

```

Repasemos las partes del Listado 1-7 que están relacionadas con la configuración de enlace. En este ejemplo, configuró un enlace de entrada y dos enlaces de salida. El parámetro imageBlob se configura como un enlace de entrada. Ha decorado el parámetro con el atributo Blob, que toma los siguientes parámetros:

- **Ruta** El valor {data.url} configura la ruta de los blobs que se pasan a la función. En este caso, está usando una expresión de enlace que se resuelve en la ruta completa del blob en Blob Storage.
- **Modo de acceso a blob** En este ejemplo, accede al blob en modo de solo lectura.
- **Conexión** Esto establece la cadena de conexión a la cuenta de almacenamiento donde se almacenan los blobs. Este parámetro establece el nombre de la configuración de la aplicación que contiene la cadena de conexión real.

También ha configurado dos enlaces de salida, aunque los ha configurado de forma diferente. El primer enlace de salida se configura utilizando la palabra clave out en la definición del parámetro. Al igual que hizo con el parámetro de entrada, configuró el documento de parámetro de salida mediante utilizando un atributo de parámetro. En este caso, usó el atributo CosmosDB. Utiliza los siguientes parámetros para configurar este enlace de salida:

- **databaseName** Establece la base de datos en la que guarda el documento que crea durante la ejecución de la función.
- **collectionName** Establece la colección en la que guarda el documento generado.
- **ConnectionStringSetting** Establece el nombre de la variable de configuración de la aplicación que contiene la cadena de conexión real para la base de datos. No debe poner aquí la cadena de conexión real.

Establecer un valor para este enlace de salida es tan simple como asignar un valor al documento de parámetros. También puede configurar enlaces de salida utilizando la declaración de retorno de la función. En el ejemplo, configura el segundo enlace de salida de esta manera.

El parámetro de función signalRMessages es su segundo enlace de salida. Como puede ver en el [Listado 1-7](#), no agregó la palabra clave out a este parámetro porque puede devolver múltiples valores de salida. Cuando necesite devolver varios valores de salida, debe usar los tipos ICollector o IAsyncCollector con el parámetro de enlace de salida, como lo hizo con signalRMessages. Dentro de la función, agrega los valores necesarios a la colección signalRMessages y usa esta colección como el valor de retorno de la función. Usó el atributo de parámetro

SignalR para configurar este enlace de salida. En este caso, solo usó un parámetro para configurar el enlace de salida.

- **HubName** Este es el nombre del concentrador de SignalR al que envía sus mensajes.
- **ConnectionStringSetting** En este caso, no usó este parámetro, por lo que usa su valor predeterminado AzureSignalRConnectionString. Como vio en los otros enlaces, este parámetro establece el nombre de la variable de configuración de la aplicación que contiene la cadena de conexión real SignalR.

Cuando configura enlaces o desencadenadores, hay situaciones en las que necesita asignar el desencadenador o el enlace a una ruta o elemento generado dinámicamente. En estas situaciones, puede utilizar expresiones vinculantes. Usted define una expresión vinculante envolviendo su expresión entre llaves. Puede ver un ejemplo de una expresión de enlace que se mostró anteriormente en el [Listado 1-7](#). La ruta que configuras para el enlace de entrada contiene la expresión de enlace {data.url}, que se resuelve en la ruta completa del blob en Blob Storage. En este caso, EventGridTrigger envía una carga útil JSON al enlace de entrada que contiene el atributo data.url.

#### *¿Necesita más revisión? Encuadernación de patrones de expresión*

Puede obtener más información sobre más patrones de expresión de enlace revisando este artículo sobre los patrones de expresión de enlace de Azure Functions en Microsoft Docs en <https://docs.microsoft.com/en-us/azure/azure-functions/functions-bindings-expressions-patterns>.

La forma en que configura los enlaces para su código depende del idioma que usó para su función de Azure. En el ejemplo anterior, revisa cómo configurar la entrada y enlaces de salida usando C # y decoraciones de parámetros. Si usa cualquiera de los otros idiomas admitidos en su función de Azure, la forma en que configura los enlaces de entrada y salida cambia.

El primer paso al configurar enlaces en lenguajes distintos de C # es modificar el archivo de configuración function.json. [El Listado 1-8](#) muestra el function.json equivalente para la configuración de enlace realizada en el [Listado 1-7](#). Una vez que haya configurado sus enlaces, puede escribir su código para acceder a los enlaces que configuró. [El Listado 1-9](#) muestra un ejemplo escrito en JavaScript para usar enlaces en su código.

**Listado 1-8** Configurando enlaces de entrada y salida en function.json

Haga clic aquí para ver la imagen del código

---

```
{  
    "inhabilitado": falso,  
    "vinculaciones": [  
        {  
            "nombre": "eventGridEvent",  
            "tipo": "eventGridTrigger",  
            "dirección": "en"  
        },  
        {  
            "nombre": "imageBlob",  
            "type": "blob",  
            "connection": "ImagesBlobStorage",  
            "direction": "in",  
            "ruta": "{data.url}"  
        },  
        {  
            "nombre": "documento",  
            "tipo": "cosmosDB",  
            "direction": "out",  
            "databaseName": "GIS",  
            "collectionName": "Imágenes_procesadas",  
            "connectionStringSetting": "CosmosDBConnection",  
            "id": "f53a2a2d-1a2c-4a2e-8a2d-1a2c4a2e8a2d"  
        }  
    ]  
}
```

```

    "createIfNotExist": verdadero
  },
{
  "nombre": "signalRMessages",
  "tipo": "señalR",
  "direction": "out",
  "hubName": "notificaciones"
}
]
}

```

**Listado 1-9** Uso de enlaces en JavaScript

Haga clic aquí para ver la imagen del código

---

```

// NodeJS. Index.js

const uuid = require ('uuid / v4');

module.exports = función asíncrona (contexto,
eventGridEvent) {

  contexto.log ('La función de activación de JavaScript
Event Grid procesó una solicitud.');

  contexto.log ("Asunto:" + eventGridEvent.subject);

  contexto.log ("Hora:" + eventGridEvent.eventTime);

  contexto.log ("Datos:" + JSON.stringify
(eventGridEvent.data));

  contexto.bindings.document = JSON.stringify ({
    id: uuid (),
    
```

```

    Descripción: eventGridEvent.topic

} ) ;

context.bindings.signalRMessages = [ {

    "target": "newMessage",

    "argumentos": [eventGridEvent.subject]

} ] ;

context.done () ;

} ;

```

Los listados 1-8 y 1-9 representan el código equivalente en JavaScript al código del código C # que se muestra en el listado 1-7 . Lo más importante es que los atributos de nombre en las definiciones vinculantes que se muestran en el Listado 1-8 corresponden a las propiedades del objeto de contexto que se muestra en el Listado 1-9 . Por ejemplo, creó un enlace de salida de Cosmos DB y asignó el documento de valor al atributo de nombre en la definición de enlace en el Listado 1-8 . En su código JavaScript, accede a este enlace de salida utilizando context.bindings.document.



### *Sugerencia para el examen*

Recuerde que debe instalar las extensiones en su entorno local antes de poder utilizar enlaces o desencadenadores. Puede usar el comando de línea de comandos func desde las herramientas de la CLI de funciones de Azure.

*Implementar activadores de funciones mediante el uso de operaciones de datos, temporizadores y webhooks*

Cuando crea una función de Azure, esa función se ejecuta en función de los eventos que suceden en el mundo externo. Algunos ejemplos incluyen

- Ejecutar una función periódicamente
- Ejecutar una función cuando algún otro proceso carga un archivo en Blob Storage o envía un mensaje a un almacenamiento en cola
- Ejecutar una función cuando llega un correo electrónico a Outlook

Los desencadenadores gestionan de forma programática todos estos eventos.

Puede configurar los desencadenadores de funciones de la misma manera que configura los enlaces de entrada o salida, pero debe prestar atención a algunos detalles adicionales cuando se trata de desencadenadores. Configura un disparador para escuchar eventos específicos. Cuando ocurre un evento, el objeto disparador puede enviar datos e información a la función.

Puede configurar tres tipos diferentes de disparadores:

- **operación de datos** El disparador se inicia en función de los nuevos datos que se crean, actualizan o agregan al sistema. Los sistemas compatibles son Cosmos DB, Event Grid, Event Hub, Blob Storage, Queue Storage y Service Bus.
- **temporizadores** Utilice este tipo de disparador cuando necesite ejecutar su función según una programación.
- **webhooks** Utiliza HTTP o activadores de webhooks cuando necesita ejecutar su función en función de una solicitud HTTP.

Los activadores envían datos a la función con información sobre el evento que provocó el inicio del activador. Esta información depende del tipo de disparador. [El Listado 1-10](#) muestra cómo configurar un disparador de operación de datos para Cosmos DB.

**Listado 1-10** Configuración de un disparador de Cosmos DB

[Haga clic aquí para ver la imagen del código](#)

---

```
// C # ASP.NET Core

using System.Collections.Generic;
utilizando Microsoft.Azure.Documents;
```

```
utilizando Microsoft.Azure.WebJobs;
utilizando Microsoft.Azure.WebJobs.Host;
utilizando Microsoft.Extensions.Logging;
espacio de nombres Company.Function

{
    clase estática pública CosmosDBTriggerCSharp
    {
        [FunctionName ("CosmosDBTriggerCSharp")]
        ejecución vacía estática pública ([CosmosDBTrigger (
            databaseName: "databaseName",
            collectionName: "collectionName",
            ConnectionStringSetting = "AzureWebJobsStorage",
            LeaseCollectionName = "arrendamientos",
            CreateLeaseCollectionIfNotExists = true)] IReadOnlyList
<Documento> entrada, registro de ILogger)

        {
            if (input! = null && input.Count> 0)

            {
                log.LogInformation ("Documentos modificados"
+ input.Count);

                log.LogInformation ("ID del primer
documento" + entrada [0] .Id);

                log.LogInformation ("Documento modificado:"
+ entrada [0]);
            }
        }
    }
}
```

```
    }  
  
}  
  
}  
  
}
```

## Trabajo *importante* con la colección de arrendamientos

En el momento de escribir este artículo, el disparador de Cosmos DB no admite el trabajo con una colección de concesiones particionada. Microsoft está eliminando la capacidad de crear una colección sin particiones mediante Azure Portal. Aún puede crear sus colecciones no particionadas utilizando SDK. El disparador de Cosmos DB requiere una segunda colección para almacenar concesiones sobre particiones. Ambas colecciones, las concesiones y la colección que desea supervisar, deben existir antes de que se ejecute su código. Para asegurarse de que la colección de concesión se crea correctamente como una colección sin particiones, no cree la colección mediante Azure Portal y establezca el parámetro de desencadenador CreateLeaseCollectionIfNotExists en true.

Al igual que con los enlaces, debe instalar el paquete NuGet correspondiente con la extensión adecuada para trabajar con desencadenadores. En este caso, debe instalar el paquete Microsoft.Azure.WebJobs.Extensions.CosmosDB. Usó el atributo de parámetro CosmosDBTrigger para configurar el desencadenador con los siguientes parámetros:

- **databaseName** Este es el nombre de la base de datos que contiene la colección que este disparador debe monitorear.
- **collectionName** Este es el nombre de la colección que este disparador debe monitorear. Esta colección debe existir antes de que se ejecute su función.
- **ConnectionStringSetting** Este es el nombre de la variable de configuración de la aplicación que contiene la cadena de conexión a la base de datos de Cosmos DB. Si desea depurar su función en su entorno local, debe configurar esta variable en el archivo local.settings.json y asignar el valor de la cadena de conexión a su base de datos CosmosDB de desarrollo. Azure Functions Core Tools usa este archivo local.settings.json para almacenar la configuración de la aplicación, las cadenas de conexión y la configuración de forma local y no se cargará automáticamente en Azure cuando publique su función de Azure.

- **LeaseCollectionName** Este es el nombre de la colección que se usa para almacenar concesiones sobre particiones. De forma predeterminada, esta colección se almacena en la misma base de datos que el nombre de la colección. Si necesita almacenar esta colección en una base de datos separada, use el parámetro leaseDatabaseName o leaseConnectionStringSetting si necesita almacenar la base de datos en una cuenta de Cosmos DB separada.
- **CreateLeaseCollectionIfNotExists** Esto crea la colección de arrendamiento establecida por el parámetro LeaseCollectionName si no existe en la base de datos. La colección de arrendamiento debe ser una colección sin particiones y debe existir antes de que se ejecute la función.

El desencadenador de Cosmos DB supervisa los documentos nuevos o actualizados en la base de datos que configura en los parámetros del desencadenador. Una vez que el disparador detecta un cambio, pasa los cambios detectados a la función usando un `IReadOnlyList<Document>`. Una vez que tenga la información proporcionada por el disparador en la lista de entrada, puede procesar la información dentro de su función. Si ha habilitado la integración de Application Insight, debería poder ver los mensajes de registro de su función, como se muestra en la [Figura 1-11](#).

```

6/4/2020 0:00:44 - TRACE
Executed 'az204function' (Succeeded, Id=06144f30-0439-4fc2-81c5-371d5462bd74)
Severity level: Informational

6/4/2020 0:00:44 - TRACE
Modified document: { "id": "2", "name": "New Testing Document AZ204", "_rid":
Severity level: Informational

6/4/2020 0:00:44 - TRACE
First document Id 2
Severity level: Informational

6/4/2020 0:00:44 - TRACE
Documents modified 1
Severity level: Informational

6/4/2020 0:00:44 - TRACE
Executing 'az204function' (Reason='New changes on collection az204collection')
Severity level: Informational

```

**Figura 1-11** Ver registros de funciones de Azure en Application Insight

*Nota Versión 1.0 versus versión 2.0 versus versión 3.0*

Cuando trabaja con Azure Functions, puede elegir entre las versiones 1.0, 2.0 y 3.0. La principal diferencia entre las versiones 1.0 y las otras versiones es que solo puede desarrollar y alojar Azure Functions 1.0 en Azure Portal o en equipos con Windows. Las

funciones 2.0 y 3.0 se pueden desarrollar y alojar en todas las plataformas compatibles con .NET Core. La función de Azure que usa afecta a los paquetes de extensión que necesita instalar al configurar desencadenadores y enlaces. Revise la descripción general de las versiones en tiempo de ejecución de Azure Functions en <https://docs.microsoft.com/en-us/azure/azure-functions/functions-versions> .

Cuando trabaja con activadores de temporizador y webhooks, la principal diferencia entre ellos y un activador de operaciones de datos es que no necesita instalar el paquete de extensión que admite el activador de forma explícita.

Los activadores del temporizador ejecutan su función según un horario. Este horario se configura utilizando una expresión CRON que es interpretada por la biblioteca NCronTab. Una expresión CRON es una cadena compuesta de seis campos diferentes con esta estructura:

[Haga clic aquí para ver la imagen del código](#)

```
{segundo} {minuto} {hora} {día} {mes} {día de la semana}
```

Cada campo puede tener valores numéricos que sean significativos para el campo:

- **segundo** Representa los segundos en un minuto. Puede asignar valores de 0 a 59.
- **minuto** Representa los minutos en una hora. Puede asignar valores de 0 a 59.
- **hora** Representa las horas de un día. Puede asignar valores de 0 a 23.
- **día** Representa los días de un mes. Puede asignar valores de 1 a 31.
- **mes** Representa los meses de un año. Puede asignar valores del 1 al 12. También puede usar nombres en inglés, como enero, o puede usar abreviaturas del nombre en inglés, como enero. Los nombres no distinguen entre mayúsculas y minúsculas.
- **día de la semana** Representa los días de la semana. Puede asignar valores de 0 a 6, donde 0 es el domingo. También puede usar nombres en inglés, como Monday, o puede usar abreviaturas del nombre en inglés, como Mon. Los nombres no distinguen entre mayúsculas y minúsculas.

Todos los campos deben estar presentes en una expresión CRON. Si no desea proporcionar un valor a un campo, puede utilizar el carácter de asterisco \*. Esto significa que la expresión usa todos los disponibles valores para ese campo. Por ejemplo, la expresión CRON \* \* \* \* \* significa que el disparador se ejecuta cada segundo, cada minuto, cada hora, cada día y cada mes del año. También puede utilizar algunos operadores con los valores permitidos en los campos:

- **Rango de valores** Utilice el operador de guion (-) para representar todos los valores disponibles entre dos límites. Por ejemplo, la expresión 0 10-12 \* \* \* significa que la función se ejecuta a las hh: 10: 00, hh: 11: 00 y hh: 12: 00 donde hh significa cada hora. Es decir, se ejecuta tres veces por hora.
- **Conjunto de valores** Utilice el operador de coma (,) para representar un conjunto de valores. Por ejemplo, la expresión 0 0 11,12,13 \* \* \* significa que la función se ejecutará tres veces al día, todos los días, una vez a las 11:00:00, una segunda vez a las 12:00:00 y finalmente a las 13:00:00.
- **Intervalo de valores** Utilice el operador de barra inclinada (/) para representar un intervalo de valores. La función se ejecuta cuando el valor del campo es divisible por el valor que pones en el lado derecho del operador. Por ejemplo, la expresión \* / 5 \* \* \* \* ejecutará la función cada cinco segundos.

Los listados 1-11 y 1-12 muestran cómo configurar un disparador de temporizador y cómo usar el disparador con código JavaScript.

**Listado 1-11** Configurando un disparador de temporizador en function.json

Haga clic aquí para ver la imagen del código

---

```
{  
  "inhabilitado": falso,  
  "vinculaciones": [  
    {  
      "nombre": "myTimer",  
      "type": "timerTrigger",
```

```

    "direction": "in",
    "horario": "0 * / 5 * * *",
    "useMonitor": verdadero,
    "runOnStartup": verdadero
}
]
}

```

**Listado 1-12** Usando un disparador de temporizador con JavaScript

Haga clic aquí para ver la imagen del código

---

```

// NodeJS. Archivo index.js

module.exports = función asíncrona (contexto, myTimer) {

    var timeStamp = nueva fecha ().toISOString ();

    if (myTimer.isPastDue)

    {
        context.log ('¡JavaScript se está retrasando!');

    }

    context.log ('Última ejecución del disparador del
temporizador de JavaScript:', myTimer.ScheduleStatus.

Último);

    context.log ('Ejecución siguiente del disparador del
temporizador de JavaScript:', myTimer.ScheduleStatus.

Próximo);

};


```

Tal como lo hizo cuando configuró los enlaces en la sección anterior, cuando configura un desencadenador para lenguajes que no son de C #, debe agregarlos al archivo de configuración function.json. Configuras tus disparadores en la sección de enlaces. [El listado 1-11](#) muestra las propiedades apropiadas para configurar un disparador de temporizador:

- **nombre** Este es el nombre de la variable que usa en su código JavaScript para acceder a la información del disparador.
- **tipo** Este es el tipo de disparador que está configurando. En este ejemplo, el valor del disparador del temporizador es timerTrigger.
- **dirección** Esto siempre se incluye en un disparador.
- **horario** Ésta es la expresión CRON utilizada para configurar la programación de ejecución de su función. También puede utilizar una expresión TimeSpan.
- **useMonitor** Esta propiedad monitorea la programación incluso si se reinicia la instancia de la aplicación de función. El valor predeterminado de esta propiedad es verdadero para todas las programaciones con una periodicidad superior a un minuto. El seguimiento de las incidencias del cronograma asegurará que el cronograma se mantenga correctamente.
- **runOnStartup** Esto indica que la función debe invocarse tan pronto como comience el tiempo de ejecución. La función se ejecutará después de que la aplicación de la función se active después de estar inactiva debido a la inactividad o si la aplicación de la función se reinicia debido a cambios en la función. No se recomienda establecer este parámetro en verdadero en entornos de producción porque puede dar lugar a tiempos de ejecución impredecibles de su función.

#### *Nota Funciones de solución de problemas en su entorno local*

Mientras desarrolla sus funciones de Azure, debe solucionar los problemas de su código en su entorno local. Si usa desencadenadores que no son HTML, debe proporcionar un valor válido para el atributo AzureWebJobsStorage en el archivo local.settings.json.

Utiliza expresiones de TimeSpan para especificar el intervalo de tiempo entre las invocaciones de la función. Si la ejecución de la función tarda más que el intervalo especificado, la función se invoca inmediatamente después de que finaliza la invocación anterior. Las expresiones TimeSpan

son cadenas con el formato hh: mm: ss donde hh representa horas, mm representa minutos y ss representa segundos. Las horas en una expresión TimeSpan deben ser inferiores a 24. La expresión TimeSpan 24:00:00 significa que la función se ejecutará todos los días. 02:00:00 significa que la función se invocará cada dos horas. Puede usar expresiones TimeSpan solo en Azure Functions que se ejecutan en App Service Plans. Es decir, no puede usar expresiones TimeSpan cuando usa el nivel de precios de Consumo.

Utiliza desencadenadores HTTP para ejecutar su función de Azure cuando un proceso externo realiza una solicitud HTTP. Esta solicitud HTTP puede ser una solicitud regular utilizando cualquiera de los métodos HTTP disponibles o un webhook. Una devolución de llamada web o webhook es una solicitud HTTP realizada por sistemas de terceros o aplicaciones web externas, o como resultado de un evento generado en el sistema externo. Por ejemplo, si usa GitHub como repositorio de código, GitHub puede enviar un webhook a su función de Azure cada vez que se abre una nueva solicitud de extracción.

Cuando crea una función de Azure mediante desencadenadores HTTP, el tiempo de ejecución publica automáticamente un punto de conexión con la siguiente estructura:

[Haga clic aquí para ver la imagen del código](#)

```
http:// < your_function_app >.azurewebsites.net / api / <  
your_function_name >
```

Esta es la URL o el punto final que necesita usar cuando llame a su función usando una solicitud HTTP regular o cuando configure un webhook externo para invocar su función. Puede personalizar la ruta de este punto final utilizando las propiedades de configuración adecuadas. Esto significa que también puede implementar API sin servidor mediante desencadenadores HTTP. Incluso puede proteger el acceso a los puntos finales de su función solicitando autorización para cualquier solicitud realizada a su API mediante la autenticación / autorización de App Service. [El listado 1-13](#) muestra cómo configurar un disparador HTTP con un punto final personalizado.

**Listado 1-13** Configurando un disparador HTTP

[Haga clic aquí para ver la imagen del código](#)

---

```
// C # ASP.NET Core

utilizando System.Security.Claims;
usando el sistema;
usando System.IO;
usando System.Threading.Tasks;
utilizando Microsoft.AspNetCore.Mvc;
utilizando Microsoft.Azure.WebJobs;
utilizando Microsoft.Azure.WebJobs.Extensions.Http;
utilizando Microsoft.AspNetCore.Http;
utilizando Microsoft.Extensions.Logging;
utilizando Newtonsoft.Json;

espacio de nombres Company.Function

{
    clase estática pública HttpTriggerCSharp
    {
        [FunctionName ("HttpTriggerCSharp")]
        public static async Task < IActionResult > Ejecutar (
            [HttpTrigger (AuthorizationLevel.Anonymous,
"get", "post", Route = "devices /
{id: int?} ")] HttpRequest req,
            ¿En t? identificación,
            ILogger log)
        {

```

```
    log.LogInformation ("La función de activación  
HTTP de C # procesó una solicitud");  
  
        // Accedemos al parámetro en la dirección  
agregando un parámetro de función // con el mismo nombre  
  
        log.LogInformation ($ "Solicitando información  
para el dispositivo {id}");  
  
  
        // Si habilita la autenticación / autorización  
en el nivel de la aplicación de función,  
  
        //información  
  
        // sobre el usuario autenticado se proporciona  
automáticamente en el  
  
        // HttpContext  
  
        ClaimsPrincipal identities =  
req.HttpContext.User;  
  
        string username = identities.Identity?.Name;  
  
  
        log.LogInformation ($ "Solicitud realizada por  
el usuario {nombre de usuario}");  
  
  
        string name = req.Query ["nombre"];  
  
  
        string requestBody = espera nuevo StreamReader  
(req.Body) .ReadToEndAsync ();  
  
        datos dinámicos = JsonConvert.DeserializeObject  
(requestBody);  
  
        nombre = nombre ?? datos?.nombre;
```

```

    // Personalizamos el enlace de salida
    nombre de retorno! = nulo

    ? (ActionResult) new JsonResult (nuevo
{mensaje = $ "Hola, {nombre}",
username = username, device = id})

    : new BadRequestObjectResult ("Por favor,
pase un nombre en la cadena de consulta o
en el cuerpo de la solicitud ");

}

}

```

El ejemplo del [Listado 1-13](#) muestra los siguientes puntos cuando se trabaja con desencadenadores HTTP:

- Cómo trabajar con autenticación.
- Cómo trabajar con el nivel de autorización.
- Cómo personalizar el punto final de la función, utilizando parámetros de ruta.
- Cómo personalizar el enlace de salida.

Los desencadenadores HTTP se le proporcionan automáticamente de forma inmediata con la función runtime. No es necesario instalar un paquete NuGet específico para trabajar con esta extensión. Utilice el atributo de parámetro `HTTPTrigger` para configurar el desencadenador HTTP. Este disparador acepta los siguientes parámetros:

- **AuthLevel** Este parámetro configura la clave de autorización que debe utilizar para acceder a la función. Los valores permitidos son

- **anónimo** No se requiere clave.
- **función** Este es el valor predeterminado. Debe proporcionar una clave específica para la función.
- **admin** Debe proporcionar la clave maestra.
- **Métodos** Puede configurar los métodos HTTP que acepta su función. De forma predeterminada, el tiempo de ejecución de la función acepta todos los métodos HTTP. El listado 1-13 reduce estos métodos HTTP aceptados a GET y POST. No use este parámetro si establece el parámetro WebHookType.
- **Ruta** Puede personalizar la ruta del punto final utilizado para que la función escuche una nueva solicitud. La ruta predeterminada es `https://<your_function_app>.azurewebsites.net / api / <your_function_name>`.
- **WebHookType** Este parámetro solo está disponible para las funciones en tiempo de ejecución de la versión 1.x. No debe utilizar los parámetros Methods y WebHookType juntos. Este parámetro establece el tipo de webhook para un proveedor específico. Los valores permitidos son
  - **genericJson** Este parámetro se utiliza para proveedores no específicos.
  - **github** Este parámetro se usa para interactuar con los webhooks de GitHub.
  - **slack** Este parámetro se utiliza para interactuar con los webhooks de Slack.

Cuando declaras el tipo de variable que usa tu función como entrada del disparador, puedes usar HttpRequest o un tipo personalizado. Si usa un tipo personalizado, el tiempo de ejecución intenta analizar el cuerpo de la solicitud como un objeto JSON para obtener la información necesaria para configurar sus propiedades de tipo personalizado. Si decide usar HttpRequest para el tipo de parámetro de entrada del disparador, obtiene acceso completo al objeto de solicitud.

Cada aplicación de función de Azure que implemente expone automáticamente un grupo de extremos de administración que puede usar para acceder mediante programación a algunos aspectos de su aplicación, como el estado del host. Estos puntos finales se ven como

Haga clic aquí para ver la imagen del código

```
https://<your_function_app_name>.azurewebsites.net /  
admin / host / status
```

De forma predeterminada, estos puntos de conexión están protegidos por un código de acceso o una clave de autenticación que puede administrar desde su aplicación de función en el portal de Azure, como se muestra en la figura 1-12.

Host Keys (All functions)	
NAME	VALUE
_master	<a href="#">Click to show</a>
default	<a href="#">Click to show</a>
<a href="#">Add new host key</a>	

**Figura 1-12** Gestión de claves de host para una aplicación de función

Cuando usa el desencadenador HTTP, cualquier punto final que publique también está protegido por el mismo mecanismo, aunque las claves que usa para proteger esos puntos finales son diferentes. Puede configurar dos tipos de claves de autorización:

- **host** Estas claves son compartidas por todas las funciones implementadas en la aplicación de funciones. Este tipo de clave permite acceder a cualquier función en el host.
- **función** Estas teclas solo protegen la función donde están definidas.

Cuando define una nueva clave, le asigna un nombre. Si tiene dos teclas de un tipo diferente (host y función) con el mismo nombre, la tecla de función tiene prioridad. También hay dos claves predeterminadas, una por tipo de clave, que también puede utilizar para acceder a sus terminales. Estas claves predeterminadas tienen prioridad sobre cualquier otra clave que haya creado. Si necesita acceder a los puntos finales de administración que mencioné anteriormente, debe usar una clave de host particular llamada \_master. También debe utilizar esta clave administrativa cuando establezca el valor de administrador en el parámetro de configuración del desencadenador AuthLevel. Puede proporcionar la clave adecuada cuando realiza una solicitud a su API

mediante el parámetro de código o el encabezado HTTP de la tecla de función x.

La protección de sus puntos finales mediante las claves de autorización no es una práctica recomendada para los entornos de producción. Solo debe usar claves de autorización en entornos de prueba o desarrollo para controlar el acceso a su API. Para un entorno de producción, debe utilizar uno de los siguientes enfoques:

- **Habilitar la función de autorización / autenticación de la aplicación** Esto integra su API con Azure Active Directory u otros proveedores de identidad de terceros para autenticar a los clientes.
- **Use Azure API Management (APIM)** Esto protege la solicitud entrante a su API, como el filtrado por dirección IP o el uso de autenticación basada en certificados.
- **Implemente su función en un entorno de servicio de aplicaciones (ASE)** Los ASE proporcionan entornos de alojamiento dedicados que le permiten configurar una única puerta de enlace de front-end que puede autenticar todas las solicitudes entrantes.

Si decide utilizar cualquiera de los métodos de seguridad anteriores, debe asegurarse de configurar AuthLevel como anónimo. Puede ver esta configuración en el [Listado 1-13](#) en esta línea:

[Haga clic aquí para ver la imagen del código](#)

```
HttpTrigger (AuthorizationLevel.Anonymous...)
```

Cuando habilita la autenticación / autorización de App Service, puede acceder a la información sobre los usuarios de autenticación leyendo los encabezados HTTP especiales establecidos por App Service. Estos encabezados especiales no pueden ser establecidos por recursos externos; solo pueden ser configurados por App Service. Para proyectos ASP.NET, el marco llena automáticamente un objeto ClaimsPrincipal con la información de autenticación. Puede usar ClaimsPrincipal como un parámetro adicional de la firma de su función o desde el código, usando el contexto de la solicitud, como se muestra anteriormente en el [Listado 1-13](#).

[Haga clic aquí para ver la imagen del código](#)

```
ClaimsPrincipal identities = req.HttpContext.User;
```

```
string username = identities.Identity?.Name;
```

Como se describe en esta sección, el tiempo de ejecución de Azure Functions expone su función de forma predeterminada mediante el siguiente esquema de URL:

[Haga clic aquí para ver la imagen del código](#)

```
https://<your_function_app_name>.azurewebsites.net/api/<your_function_name>
```

Puede personalizar el punto final mediante el parámetro route HTTPTrigger. En el [Listado 1-13](#), establece el parámetro de ruta en devices / {id: int?}. Esto significa que su punto final se ve así:

[Haga clic aquí para ver la imagen del código](#)

```
https://<your_function_app_name>.azurewebsites.net/api/devices/{id: int?}
```

Cuando personaliza la ruta para su función, también puede agregar parámetros a la ruta, que son accesibles a su código agregándolos como parámetros de la firma de su función. Puede utilizar cualquier restricción de ruta de API web (consulte <https://www.asp.net/web-api/overview/web-api-routing-and-actions/attribute-routing-in-web-api-2#constraints>) que puede utilizar al definir una ruta mediante Web API 2.

De forma predeterminada, cuando realiza una solicitud a una función que utiliza un activador HTTP, la respuesta es un cuerpo vacío con estos códigos de estado:

[Haga clic aquí para ver la imagen del código](#)

HTTP 200 OK en el caso del tiempo de ejecución de la función 1.x

HTTP 204 Sin contenido en el caso del tiempo de ejecución de la función 2.x

Si necesita personalizar la respuesta de su función, debe configurar un enlace de salida. Puede utilizar cualquiera de los dos tipos de enlaces de salida, utilizando la declaración de retorno o un parámetro de función. [El Listado 1-13](#) muestra cómo configurar el enlace de salida para devolver un objeto JSON con cierta información.

Es importante recordar los límites asociados con la función cuando planea implementar su función en un entorno de producción. Estos límites son

- **Longitud máxima de la solicitud** La solicitud HTTP no debe superar los 100 MB.
- **Longitud máxima de la URL** Su URL personalizada está limitada a 4096 bytes.
- **Tiempo de espera de ejecución** Su función debe devolver un valor en menos de 230 segundos. Su función puede tardar más en ejecutarse, pero si no devuelve nada antes de ese tiempo, la puerta de enlace expirará con un error HTTP 502. Si su función necesita más tiempo para ejecutarse, debe usar un patrón asíncrono y devolver un extremo de ping para permitir que la persona que llama pregunte por el estado de ejecución de su función.

#### *¿Necesita más revisión? Propiedades del host*

También puede realizar algunos ajustes en el host donde se ejecuta su función utilizando el archivo host.json. Visite el siguiente artículo para revisar todas las propiedades disponibles en el archivo host.json en [https://docs.microsoft.com/en-us/azure/azure-functions/functions-bindings-http-webhook#trigger---hostjson -propiedades](https://docs.microsoft.com/en-us/azure/azure-functions/functions-bindings-http-webhook#trigger---hostjson-propiedades) .

También puede revisar cuáles son los límites asociados con las diferentes versiones de framework y planes de hosting revisando el artículo en <https://docs.microsoft.com/en-us/azure/azure-functions/functions-scale> .



#### *Sugerencia para el examen*

En versiones anteriores de Azure Functions, la información de autenticación solo estaba disponible para proyectos ASP.NET que usaban la clase ClaimsPrincipal. Ahora puede acceder a esa información leyendo los encabezados HTTP especiales establecidos por App Service. Para obtener una lista completa de los encabezados de autenticación, consulte el artículo <https://docs.microsoft.com/en-us/azure/app-service/app-service-authentication-how-to#access-user-claims> .

#### *Implementar funciones duraderas de Azure*

Una característica fundamental de las funciones de Azure es que no tienen estado. Esta característica significa que el tiempo de ejecución de la función no mantiene el estado de los objetos que crea durante la

ejecución de la función si el proceso del host o la VM donde se ejecuta la función se recicla o reinicia.

Azure Durable Functions es una extensión de Azure Functions que proporciona capacidades de flujo de trabajo con estado en un entorno sin servidor. Estas capacidades de flujo de trabajo con estado le permiten

- **Encadenar llamadas a funciones juntas** Este encadenamiento significa que una función puede llamar a otras funciones, lo que mantiene el estado entre llamadas. Estas llamadas pueden ser sincrónicas o asíncronas.
- **Definir el flujo de trabajo por código** No es necesario crear definiciones de flujo de trabajo JSON ni utilizar herramientas externas.
- **Asegúrese de que el estado del flujo de trabajo sea siempre coherente** Cuando una función o actividad en un flujo de trabajo necesita esperar a otras funciones o actividades, el motor del flujo de trabajo crea automáticamente puntos de control para guardar el estado de la actividad.

La principal ventaja de usar Azure Durable Functions es que facilita la implementación de requisitos complejos de coordinación con estado en escenarios sin servidor. Aunque Durable Azure Functions es una extensión de Azure Functions, en el momento de redactar este artículo, no admite todos los idiomas admitidos por Azure Functions. Se admiten los siguientes idiomas:

- **C #** Se admiten tanto las bibliotecas de clases precompiladas como el script C #.
- **F #** Se admiten las bibliotecas de clases precompiladas de F # y el script F #. El script F # solo está disponible para el tiempo de ejecución de Azure Functions 1.x.
- **JavaScript** Solo se admite para el tiempo de ejecución de la versión 2.x del tiempo de ejecución de Azure Functions. Se requiere la versión 1.7.0 o posterior o Azure Durable Functions.

Las funciones duraderas se facturan con las mismas reglas que se aplican a las funciones de Azure. Es decir, solo se le cobra por el tiempo que se ejecutan sus funciones.

Trabajar con funciones duraderas significa que debe lidiar con diferentes tipos de funciones. Cada tipo de función juega un papel diferente en la ejecución del flujo de trabajo. Estos roles son

- **Actividad** Estas son las funciones que hacen el trabajo real. Una actividad es un trabajo que necesita que realice su flujo de trabajo. Por ejemplo, es posible que necesite su código para enviar un documento a un revisor de contenido antes de que otra actividad pueda publicar el documento, o necesita crear una orden de envío para enviar productos a un cliente.
- **Orquestador** Cualquier flujo de trabajo ejecuta funciones de actividad en un orden particular. Las funciones de orquestador definen las acciones que ejecuta un flujo de trabajo. Estas acciones pueden ser funciones de actividad, temporizadores o espera de eventos externos o suborquestaciones. Cada instancia de una función de orquestador tiene un identificador de instancia. Puede generar este identificador manualmente o dejar el marco de función duradera para generarlo dinámicamente.
- **Cliente** Este es el punto de entrada de un flujo de trabajo. Los disparadores como HTTP, la cola o los disparadores de eventos crean instancias de una función de cliente. Las funciones de cliente crean instancias de funciones de orquestador enviando un disparador de orquestador.

De la misma manera que Azure Functions usa desencadenadores y enlaces para enviar y recibir información de funciones, debe usar desencadenadores y enlaces para configurar la comunicación entre los diferentes tipos de funciones duraderas. Las funciones duraderas agregan dos nuevos activadores para controlar la ejecución de las funciones de orquestación y actividad:

- **Activador de orquestación** Estos le permiten trabajar con funciones de orquestación creando nuevas instancias de la función o reanudando instancias que están esperando una tarea. La característica más importante de estos desencadenantes es que son de un solo subprocesso. Cuando usa desencadenadores de orquestación, debe asegurarse de que su código no realice llamadas asíncronas, aparte de esperar tareas de función duradera, u operaciones de E / S. Esto asegura que la función de orquestación

se concentre en llamar a las funciones de actividad en el orden correcto y esperar los eventos o funciones correctos.

- **Activador de actividad** Este es el tipo de activador que debe utilizar al escribir sus funciones de actividad. Estos activadores permiten la comunicación entre las funciones de orquestación y las funciones de actividad. Son multiproceso y no tienen restricciones relacionadas con el subproceso o las operaciones de E / S.

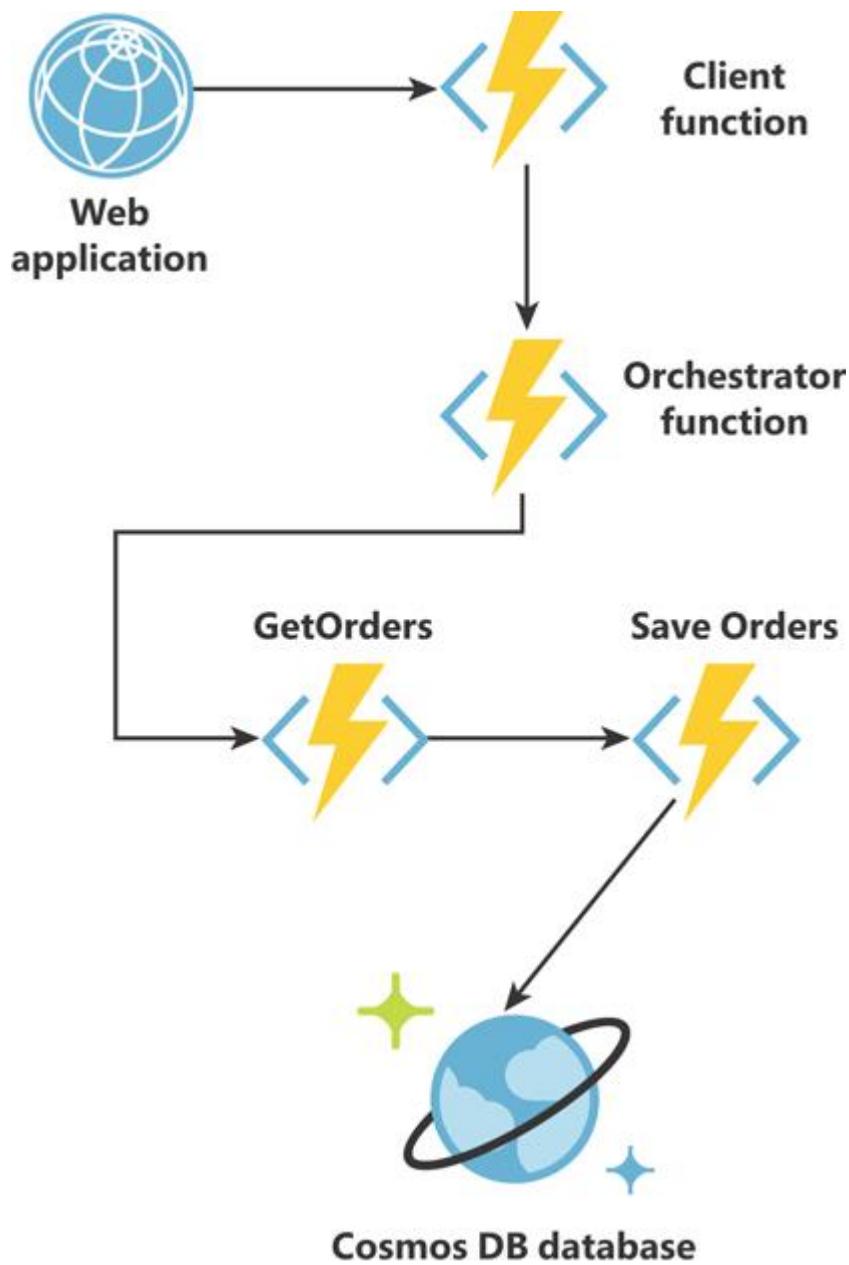
El siguiente ejemplo muestra cómo los diferentes tipos de funciones y disparadores trabajan juntos para procesar y guardar un orden hipotético generado desde una aplicación externa y guardado en una base de datos de Cosmos DB. Aunque el ejemplo es bastante simple, muestra cómo interactúan las diferentes funciones. [La Figura 1-13](#) muestra un diagrama del flujo de trabajo implementado en las funciones que se muestran en los [listados 1-14 a 1-21](#). Para ejecutar este ejemplo, debe cumplir con los siguientes requisitos:

- Una suscripción a Azure.
- Una cuenta de Azure Storage. La función de orquestación necesita una cuenta de almacenamiento de Azure para guardar el estado de cada instancia de función duradera durante la ejecución del flujo de trabajo.
- Una base de datos de Azure Cosmos DB.
- Instale las siguientes dependencias usando este comando:

[Haga clic aquí para ver la imagen del código](#)

```
func extensions install -p < nombre_paquete > -v < versión_paquete >
```

- Cosmos DB:
  - Nombre del paquete: Microsoft.Azure.WebJobs.Extensions.CosmosDB
  - Versión: 3.0.3
- Extensión de funciones duraderas:
  - Nombre del paquete: Microsoft.Azure.WebJobs.Extensions.DurableTask
  - Versión: 1.8.0



**Figura 1-13** Flujo de trabajo de funciones duraderas

Puede ejecutar este ejemplo utilizando su entorno de desarrollo integrado (IDE) favorito. Visual Studio y Visual Studio Code ofrecen varias herramientas que hacen que trabajar con proyectos de Azure sea más cómodo. Utilice los siguientes pasos para configurar su código de Visual Studio y crear las funciones duraderas:

1. Abra su código de Visual Studio.
2. Haga clic en el ícono de Extensiones en el lado izquierdo de la ventana.

3. En el panel Extensiones, en el cuadro de texto Buscar extensiones en Marketplace, escriba **Azure Functions** .
4. En la lista de resultados, en la extensión de Azure Functions, haga clic en el botón Instalar. Dependiendo de su versión de Visual Studio Code, es posible que deba reiniciar Visual Studio Code.
5. Haga clic en el icono de Azure en el lado izquierdo de la ventana de Visual Studio Code.
6. En la sección Funciones, haga clic en Iniciar sesión en Azure para iniciar sesión en Azure.
7. En la sección Funciones, haga clic en el icono del rayo, que crea una nueva función de Azure.
8. En el cuadro de diálogo Crear nuevo proyecto, seleccione JavaScript.
9. En el cuadro de diálogo Seleccionar una plantilla para la primera función de su proyecto, seleccione Desencadenador HTTP.
10. Para la opción Proporcionar un nombre de función, escriba **HTTPTriggerDurable** . Esto crea la primera función que necesita para este ejemplo.
11. Seleccione Anónimo para el Nivel de autorización.
12. Seleccione Abrir en la ventana actual para abrir el proyecto que acaba de crear.

Repita los pasos del 5 al 12 para todas las funciones duraderas que necesita para este ejemplo. Es importante guardar todas las funciones que necesita en la misma carpeta.

Los listados 1-14 y 1-15 muestran el código JavaScript y el archivo de configuración JSON que necesita para crear la función de cliente que llama a la función de orquestación.

**Listado 1-14** Código de función de cliente de Azure Durable Functions

Haga clic aquí para ver la imagen del código

---

```
// NodeJS. HTTPTriggerDurable / index.js

const df = require ("funciones-duraderas");
```

```

module.exports = función asíncrona (contexto, req) {

    context.log ('Ejemplo de funciones duraderas de
JavaScript');

    cliente constante = df.getClient (contexto);

    const instanceId = await client.startNew
(req.params.functionName, undefined,
req.body);

    context.log ('Orquestación iniciada con ID =' +
{instanceId} '.');
}

return client.createCheckStatusResponse
(context.bindingData.req, instanceId);

};

```

**Listado 1-15** Funciones duraderas: archivo de configuración JSON de la función del cliente

[Haga clic aquí para ver la imagen del código](#)

---

```
{
    "inhabilitado": falso,
    "vinculaciones": [
        {
            "authLevel": "anónimo",
            "tipo": "httpTrigger",
            "dirección": "in",
            "nombre": "req",
            "ruta": "orquestadores / {functionName}",
            "función": "función"
        }
    ],
    "funciones": [
        {
            "name": "función"
        }
    ]
}
```

```

    "métodos": [
        "obtener",
        "correo"
    ],
},
{
    "tipo": "http",
    "direction": "out",
    "nombre": "$ retorno"
},
{
    "nombre": "contexto",
    "tipo": "orchestrationClient",
    "dirección": "en"
}
]
}

```

Los listados 1-16 y 1-17 muestran el código JavaScript y el archivo de configuración JSON que necesita para crear la función de orquestación que invoca, en el orden correcto, todas las demás funciones de actividad. Esta función también devuelve a la función del cliente los resultados de la ejecución de las diferentes funciones de actividad.

**Listado 1-16** Código de función de Azure Durable Functions Orchestrator

Haga clic aquí para ver la imagen del código

---

```
// NodeJS. OrchestratorFunction / index.js
```

```

const df = require ("funciones-duraderas");

module.exports = df.orchestrator (función * (contexto) {

    context.log ("Iniciando flujo de trabajo: ejemplo de
cadena");

    orden constante = rendimiento context.df.callActivity
("GetOrder");

    const SavedOrder = rendimiento context.df.callActivity
("SaveOrder", orden);

    return SavedOrder;

}) ;

```

**Listado 1-17** Funciones duraderas: archivo de configuración JSON de la función del orquestador

[Haga clic aquí para ver la imagen del código](#)

---

```
{
    "inhabilitado": falso,
    "vinculaciones": [
        {
            "type": "orchestrationTrigger",
            "direction": "in",
            "nombre": "contexto"
        }
    ]
}
```

}

Los listados 1-18 y 1-19 muestran el código JavaScript y el archivo de configuración JSON que necesita para crear la función de actividad Obtener orden. En este ejemplo, esta función se encarga de construir la información que se utiliza en la función Guardar Orden. En un escenario más complejo, esta función podría obtener información del carrito de compras del usuario desde un sistema de comercio electrónico o cualquier otra fuente potencial.

**Listado 1-18** Código de función de actividad de Azure Durable Functions

Haga clic aquí para ver la imagen del código

---

```
// NodeJS. GetOrder / index.js

module.exports = función asíncrona (contexto) {

    // Crea una orden simulada para probar

    var order = {

        "id": Math.floor (Math.random () * 1000),

        "nombre": "Cliente",

        "fecha": nueva fecha (). toJSON ()

    }

    context.log (orden);

    orden de devolución;

};
```

**Listado 1-19** Archivo de configuración JSON de la función de actividad de Azure Durable Functions

Haga clic aquí para ver la imagen del código

---

```
{

    "inhabilitado": falso,

    "vinculaciones": [
```

```

{
  "type": "activityTrigger",
  "direction": "in",
  "nombre nombre"
}
]
}

```

Los listados 1-20 y 1-21 muestran el código JavaScript y el archivo de configuración JSON que necesita para crear la función de actividad que guarda el pedido en una base de datos de Cosmos DB. En un escenario mucho más complejo, podría utilizar esta función para insertar el pedido en su sistema ERP o enviarlo a otra función de actividad que podría realizar un análisis o procesamiento adicional.

#### **Listado 1-20** Código de función de actividad de Azure Durable Functions

Haga clic aquí para ver la imagen del código

---

```

// NodeJS. SaveOrder / index.js

module.exports = función asincrona (contexto) {

  // Guarda el objeto de pedido recibido de otras
  actividades en un documento de CosmosDB

  context.bindings.orderDocument = JSON.stringify ({
    "id": '$ {context.bindings.order.id}',
    "customerName": context.bindings.order.name,
    "orderDate": context.bindings.order.date,
    "cosmosDate": nueva fecha () . toJSON ()
  }) ;
}
```

```
    context.done ();  
};
```

**Listado 1-21** Archivo de configuración JSON de la función de actividad de Azure Durable Functions

[Haga clic aquí para ver la imagen del código](#)

---

```
{  
  
    "inhabilitado": falso,  
  
    "vinculaciones": [  
  
        {  
  
            "type": "activityTrigger",  
  
            "direction": "in",  
  
            "nombre": "orden"  
  
        }  
  
,  
  
        {  
  
            "name": "orderDocument",  
  
            "tipo": "cosmosDB",  
  
            "databaseName": "ERP_Database",  
  
            "collectionName": "Pedidos",  
  
            "createIfNotExists": verdadero,  
  
            "connectionStringSetting": "CosmosDBStorage",  
  
            "direction": "out"  
  
        }  
  
    ]  
}
```

```
}
```

El punto de entrada en cualquier flujo de trabajo implementado con Durable Functions es siempre una función del cliente. Esta función utiliza el cliente de orquestación para llamar a la función de orquestador. [El listado 1-15](#) muestra cómo configurar el enlace de salida.

[Haga clic aquí para ver la imagen del código](#)

```
{
    "nombre": "contexto",
    "tipo": "orchestrationClient",
    "dirección": "en"
}
```

Cuando utiliza JavaScript para programar su función de cliente, el enlace de salida del cliente de orquestador no se expone directamente mediante el valor del atributo de nombre establecido en el archivo de configuración function.json. En este caso, necesita extraer el cliente real de la variable de contexto usando la función getClient () declarada en el paquete de funciones duraderas, como se muestra en el [Listado 1-14](#).

[Haga clic aquí para ver la imagen del código](#)

```
cliente constante = df.getClient (contexto);
```

Una vez que tenga la referencia correcta al enlace de salida del cliente del orquestador, puede usar el método startNew () para crear una nueva instancia de la función del orquestador. Los parámetros de este método son

- **Nombre de la función del orquestador** En el ejemplo, obtiene este nombre de la solicitud HTTP, utilizando el parámetro URL functionName, como se mostró anteriormente en los [listados 1-14 y 1-15](#).
- **InstanceId** Establece el Id asignado a la nueva instancia de la función de orquestación. Si no proporciona un valor a este parámetro, el método crea un Id aleatorio. En general, debe utilizar la identificación aleatoria generada automáticamente.

- **Entrada** Aquí es donde coloca los datos que su función de orquestación pueda necesitar. Debe utilizar datos serializables JSON para este parámetro.

Una vez que haya creado la instancia de la función de orquestación y haya guardado el *Id* asociado con la instancia, la función del cliente devuelve una estructura de datos con varios puntos finales HTTP útiles. Puede utilizar estos puntos finales para revisar el estado de la ejecución del flujo de trabajo, o finalizar el flujo de trabajo o enviar eventos externos al flujo de trabajo durante la ejecución. A continuación, se muestra un ejemplo de los puntos finales de gestión del flujo de trabajo para la ejecución del ejemplo en un entorno informático local:

#### **Nota Salida de consola**

Para la vibración del espacio, se han recortado algunas líneas. Su salida debe mostrar líneas más largas, incluida la identificación de la instancia y otros códigos.

[Haga clic aquí para ver la imagen del código](#)

```
{
  "id": "789e7eb945a04ab78e74e9216870af28",
  "statusQueryGetUri": "http://localhost:7071/runtime/webhooks/durabletask/instancias...",
  "sendEventPostUri": "http://localhost:7071/runtime/webhooks/durabletask/instancias...",
  "terminatePostUri": "http://localhost:7071/runtime/webhooks/durabletask/instancias...",
  "rewindPostUri": "http://localhost:7071/runtime/webhooks/durabletask/instancias...",
  "purgeHistoryDeleteUri": "http://localhost:7071/runtime/webhooks/durabletask/
```

```
instancias."
```

```
}
```

Este ejemplo usa una función de Azure basada en un desencadenador HTTP, pero su función de cliente no se limita a usar este desencadenador. Puede usar cualquiera de los desencadenadores disponibles en el marco de funciones de Azure.

Una vez que haya creado la instancia de la función de orquestador, esta función llama a las funciones de actividad en el orden definido en el código, como se mostró anteriormente en el [Listado 1-16](#).

[Haga clic aquí para ver la imagen del código](#)

```
orden constante = rendimiento context.df.callActivity  
("GetOrder");  
  
const SavedOrder = rendimiento context.df.callActivity  
("SaveOrder", orden);
```

La función de orquestador usa un disparador de orquestación para obtener la información que envía la función de cliente cuando crea la instancia. El desencadenador de orquestación crea las instancias de las diferentes funciones de actividad mediante el método callActivity () del paquete de funciones duraderas. Este método toma dos parámetros:

- **Nombre de la función de actividad**
- **Entrada** Pone aquí cualquier dato serializable JSON que deseé enviar a la función de actividad.

En el ejemplo, ejecuta la función de actividad GetOrder, que se mostró anteriormente en el [Listado 1-18](#), para obtener el objeto de la orden que usa como parámetro de entrada para la siguiente función de actividad SaveOrder, que se mostró anteriormente en el [Listado 1-20](#), para guardar la información en la base de datos de Cosmos DB configurada en el [Listado 1-21](#).

Puede probar este ejemplo en su equipo local ejecutando las funciones que se revisaron en esta sección, de la misma manera que prueba cualquier otra función de Azure. Una vez que tenga su función ejecutándose, puede probarla usando curl o cartero. Debe realizar una solicitud HTTP GET o POST a esta URL: *http://localhost: 7071 / api / orchestrators / OrchestratorFunction* .

Observe que el parámetro `functionName` de la URL coincide con el nombre de la función del orquestador. Su función de cliente le permite llamar a diferentes funciones de orquestación simplemente proporcionando el nombre correcto de la función de orquestación.

Puede utilizar diferentes patrones al programar la función de orquestación. Estos patrones muestran cómo las funciones de orquestación y actividad interactúan entre sí:

- **Encadenamiento** Las funciones de actividad se ejecutan en un orden específico, donde la salida de una función de actividad es la entrada de la siguiente. Este es el patrón que usó en su ejemplo.
- **Fan out / fan in** Su función de orquestación ejecuta múltiples funciones de actividad en paralelo. El resultado de estas funciones de actividad paralelas es procesado y agregado por una función de actividad de agregación final.
- **API HTTP asíncronas** Este patrón coordina el estado de las operaciones de larga duración con clientes externos.
- **Monitor** Este patrón le permite crear tareas recurrentes usando intervalos de tiempo flexibles.
- **Interacción humana** Utilice este patrón cuando necesite ejecutar funciones de actividad basadas en eventos que una persona puede desencadenar. Un ejemplo de este tipo de patrón es el flujo de trabajo de aprobación de documentos, donde la publicación de un documento depende de la aprobación de una persona.

#### *¿Necesita más revisión? Patrones de función duraderos*

Puede obtener más información sobre los patrones de función duradera revisando el artículo Patrones y conceptos en Microsoft Docs en <https://docs.microsoft.com/en-us/azure/azure-functions/durable/durable-functions-concepts> .



#### *Sugerencia para el examen*

Cuando trabaje con Azure Durable Functions, recuerde que puede pasar información entre las diferentes funciones del flujo de trabajo mediante el mecanismo de enlace.

## RESUMEN DEL CAPÍTULO

---

- Azure proporciona servicios informáticos para implementar su propia infraestructura virtualizada directamente en la nube. También puede implementar arquitecturas híbridas para conectar su infraestructura local con sus recursos IaaS.
- Azure Resource Manager es el servicio en Azure que administra los diferentes recursos que puedes implementar en la nube. Puede definir los recursos y sus dependencias mediante un archivo basado en JSON llamado plantilla ARM.
- Una imagen de contenedor es un paquete de software en el que almacena su código y cualquier biblioteca o dependencia para ejecutar su aplicación en un entorno altamente portátil.
- Cuando crea una nueva instancia de una imagen de contenedor, cada una de estas instancias se denomina "contenedor".
- Puede almacenar las imágenes de su contenedor en una tienda centralizada llamada registro.
- Azure Container Registry es un registro administrado, que se basa en la especificación de código abierto de Docker Registry 2.0.
- Puede ejecutar sus contenedores en varios servicios de Azure, como Azure Managed Kubernetes Service, Azure Container Instance, Azure Batch, Azure App Service o Azure Container Service.
- Azure le proporciona los servicios necesarios para implementar soluciones sin servidor, lo que le permite centrarse en el código y olvidarse de la infraestructura.
- Azure App Services es la base de la oferta sin servidor. Además de los servicios de aplicaciones, puede implementar aplicaciones web, aplicaciones de back-end móviles, API REST o Azure Functions y Azure Durable Functions.
- Cuando trabaja con App Services, solo se le cobra cuando se ejecuta el código.
- Los servicios de aplicaciones se ejecutan sobre los planes de servicios de aplicaciones.

- Un plan de servicio de aplicaciones proporciona los recursos y las máquinas virtuales necesarios para ejecutar su código de servicios de aplicaciones.
- Puede ejecutar más de un App Service además de un solo App Service Plan.
- Al solucionar problemas de su aplicación App Service, puede usar varios tipos para el registro de diagnóstico: registro y diagnóstico del servidor web, error detallado, solicitudes fallidas, diagnóstico de la aplicación y diagnóstico de implementación.
- El registro de diagnóstico se almacena localmente en la VM, donde se ejecuta la instancia de su aplicación.
- El escalado horizontal o el escalado de entrada a salida es el proceso de agregar o eliminar instancias de una aplicación.
- El escalado vertical o el escalado ascendente y descendente es el proceso de agregar o eliminar recursos a la misma máquina virtual que aloja su aplicación.
- Scale In / Out no tiene ningún efecto sobre la disponibilidad de la aplicación.
- El escalado vertical afecta la disponibilidad de la aplicación porque la aplicación debe implementarse en una máquina virtual con la nueva asignación de recursos.
- Puede agregar y quitar recursos a sus aplicaciones mediante el uso de reglas de autoescala.
- Puede aplicar la escala automática solo a algunos tipos de recursos de Azure.
- La escala automática depende de los conjuntos de escalado de máquinas virtuales de Azure.
- Su aplicación debe conocer los cambios en la asignación de recursos.
- Azure Functions es la evolución de WebJobs.
- Las funciones de Azure usan desencadenadores y enlaces para crear instancias de funciones de Azure y enviar o recibir datos hacia o desde servicios externos, como el almacenamiento en cola o el centro de eventos.

- Hay tres versiones de Azure Functions. La versión 1.0 solo admite entornos .NET Framework y Windows. La versión 2.0 y posteriores son compatibles con los entornos .NET Core y Windows y Linux.
- Cuando trabaja con desencadenadores y enlaces, debe instalar el paquete NuGet adecuado para la extensión de función que contiene ese desencadenador o enlace.
- El tiempo de ejecución de Azure Function ya incluye extensiones para temporizadores y desencadenadores HTTP. No es necesario instalar paquetes específicos para usar estos enlaces de desencadenadores.
- Los activadores que crean instancias de funciones pueden basarse en operaciones de datos, temporizadores o webhooks.
- Azure Durable Functions es la evolución de Azure Functions que le permite crear flujos de trabajo en los que se conserva el estado de las instancias en caso de reinicio de la máquina virtual o reaparición del proceso del host de funciones.
- Las funciones de orquestación definen la actividad y el orden de ejecución de las funciones que realizan el trabajo.
- Las funciones de actividad contienen el código que realiza la acción que necesita para un paso en el flujo de trabajo, como enviar un correo electrónico, guardar un documento o insertar información en una base de datos.
- Las funciones de cliente crean la instancia de la función de orquestación mediante un cliente de orquestación.
- Azure Function Apps proporciona los recursos necesarios para ejecutar Azure Functions y Durable Functions.

## EXPERIMENTO MENTAL

---

En este experimento mental, puede demostrar sus habilidades y conocimientos sobre los temas tratados en este capítulo. Puede encontrar las respuestas a este experimento mental en la siguiente sección.

Estás desarrollando una aplicación para realizar la integración entre varios sistemas. Uno de los sistemas es una aplicación heredada que genera algunos informes en un formato de archivo específico. Estos

informes de archivos se cargan en una cuenta de Azure Storage. Su aplicación lee la información de estos informes de archivos e inserta la información en diferentes sistemas de destino. Responda las siguientes preguntas relacionadas con el escenario descrito:

1. Antes de que su solicitud pueda insertar información sobre el destino, la información debe ser aprobada. Este flujo de trabajo de aprobación debe comenzar cuando se agrega un nuevo archivo de informe a la cuenta de almacenamiento de Azure. ¿Qué servicio de Azure se adapta mejor a sus necesidades?
2. Su aplicación tiene problemas de rendimiento. Los problemas de rendimiento solo ocurren durante algunos días del mes. Debe asegurarse de que su aplicación no sufra problemas de rendimiento durante los picos de uso. ¿Cómo puedes lograrlo?

## RESPUESTAS DEL EXPERIMENTO MENTAL

---

Esta sección contiene las soluciones del experimento mental.

1. Como la información debe aprobarse antes de que pueda insertarse en los sistemas de destino, debe usar Azure Durable Functions. Al implementar un patrón de interacción humana, puede esperar a que se valide la información antes de insertarla en el sistema de destino correcto. También puede usar desencadenadores de Azure Blob Storage para iniciar el flujo de trabajo. Como debe esperar la confirmación humana, debe usar Azure Durable Function en lugar de Azure Functions.
2. Puede implementar Azure Durable Functions en Azure App Service Plans. Comenzando con el nivel de precios estándar, puede configurar las reglas de Autoscale para su plan de Azure App Service. Con las reglas de autoescala, puede agregar o quitar recursos al plan de servicio de aplicaciones según sus necesidades. En este escenario, puede agregar más recursos según el consumo de CPU o durante días específicos. Debido a que no se ha descrito ningún patrón específico en el escenario, primero debe estudiar el patrón de uso antes de configurar las reglas de autoescala adecuadas

# Capítulo 2. Desarrollar para almacenamiento de Azure

Todas las aplicaciones funcionan con información o datos. Las aplicaciones crean, transforman, modelan u operan con esa información. Independientemente del tipo o volumen de los datos que utilice su aplicación, tarde o temprano, deberá guardarlos de forma persistente para poder utilizarlos más tarde.

Almacenar datos no es una tarea sencilla y diseñar sistemas de almacenamiento para ese propósito es aún más complicado. Quizás su aplicación necesite manejar terabytes de información, o puede trabajar con una aplicación a la que se debe acceder desde diferentes países, y necesita minimizar el tiempo requerido para acceder a ella. Además, la rentabilidad es un requisito en cualquier proyecto. En general, muchos requisitos dificultan el diseño y el mantenimiento de los sistemas de almacenamiento.

Microsoft Azure ofrece diferentes soluciones de almacenamiento en la nube para satisfacer los requisitos de almacenamiento de sus aplicaciones. Azure ofrece soluciones para hacer que su almacenamiento sea rentable y minimizar la latencia.

## Habilidades cubiertas en este capítulo:

- [Habilidad 2.1: Desarrollar soluciones que utilicen almacenamiento Cosmos DB](#)
- [Habilidad 2.2: Desarrollar soluciones que usen Blob Storage](#)

## HABILIDAD 2.1: DESARROLLAR SOLUCIONES QUE UTILICEN ALMACENAMIENTO COSMOS DB

---

Cosmos DB es un servicio de almacenamiento premium que Azure proporciona para satisfacer su necesidad de un servicio de base de datos distribuido globalmente, de baja latencia, altamente receptivo y siempre en línea. Cosmos DB se ha diseñado teniendo en cuenta la escalabilidad y el rendimiento. Una de las diferencias más significativas entre Cosmos DB y otros servicios de almacenamiento ofrecidos por Azure es la facilidad

con la que puede escalar su solución Cosmos DB en todo el mundo con solo hacer clic en un botón y agregar una nueva región a su base de datos.

Otra característica esencial que debe considerar al evaluar este tipo de servicio de almacenamiento es cómo puede acceder a este servicio desde su código y qué tan difícil sería migrar su código existente a una solución de almacenamiento basada en Cosmos DB. La buena noticia es que Cosmos DB ofrece diferentes API para acceder al servicio. La mejor API para usted depende del tipo de datos que desee almacenar en su base de datos de Cosmos DB. Usted almacena sus datos utilizando enfoques de valor-clave, familia de columnas, documentos o gráficos. Cada una de las diferentes API que ofrece Cosmos DB le permite almacenar sus datos con diferentes esquemas. Actualmente, puede acceder a Cosmos DB mediante las API de SQL, Cassandra, Table, Gremlin y MongoDB.

### **Esta habilidad cubre cómo**

- Seleccione la API adecuada para su solución
- Implementar esquemas de particiones
- Interactuar con los datos usando el SDK apropiado
- Establecer el nivel de coherencia adecuado para las operaciones
- Crear contenedores de Cosmos DB
- Implemente la programación del lado del servidor, incluidos los procedimientos almacenados, los desencadenantes y las notificaciones de cambios de fuentes

#### *Seleccione la API adecuada para su solución*

Cuando esté planificando cómo almacenar la información que su aplicación necesita para funcionar, debe considerar la estructura que necesita usar para almacenar esa información. Es posible que algunas partes de su aplicación necesiten almacenar información utilizando una estructura de valor-clave. Por el contrario, otros pueden necesitar una estructura sin esquema más flexible en la que necesite guardar la información en documentos. Quizás una característica fundamental de su aplicación es que necesita almacenar la relación entre entidades y necesita utilizar una estructura gráfica para almacenar sus datos.

Cosmos DB ofrece una variedad de API para almacenar y acceder a sus datos, según los requisitos que tenga su aplicación:

- **SQL** Esta es la API principal y predeterminada para acceder a sus datos en su cuenta de Cosmos DB. Esta API central le permite consultar objetos JSON utilizando la sintaxis SQL, lo que significa que no necesita aprender otro lenguaje de consulta. Bajo el capó, la API de SQL utiliza el modelo de programación de JavaScript para la evaluación de expresiones, invocaciones de funciones y sistema de escritura. Utilice esta API cuando necesite utilizar una estructura de datos basada en documentos.
- **Table** Puede pensar en Table API como la evolución del servicio Azure Table Storage. Esta API se beneficia de las características de alto rendimiento, baja latencia y alta escalabilidad de Cosmos DB. Puede migrar desde su servicio Azure Table Storage actual sin modificar el código en su aplicación. Otra diferencia fundamental entre Table API para Cosmos DB y Azure Table Storage es que puede definir sus propios índices en sus tablas. De la misma forma que puedes hacer con el servicio Table Storage, Table API te permite almacenar información en tu cuenta de Cosmos DB utilizando una estructura de datos basada en documentos.
- **Cassandra** Cosmos DB implementa el protocolo de conexión para la base de datos de Apache Cassandra en las opciones para almacenar y acceder a datos en la base de datos de Cosmos DB. Esto le permite olvidarse de las operaciones y las tareas de gestión del rendimiento relacionadas con la gestión de las bases de datos de Cassandra. En la mayoría de las situaciones, puede migrar su aplicación desde su base de datos de Cassandra actual a Cosmos DB usando la API de Cassandra simplemente cambiando la cadena de conexión. La API Cassandra de Azure Cosmos DB es compatible con el protocolo de cable CQLv4. Cassandra es una base de datos basada en columnas que almacena información mediante un enfoque de valor clave.
- **MongoDB** Puede acceder a su cuenta de Cosmos DB utilizando la API de MongoDB. Esta base de datos NoSQL le permite almacenar la información de su aplicación en una estructura basada en documentos. Cosmos DB implementa el protocolo de cable compatible con MongoDB 3.2. Esto significa que cualquier

controlador de cliente MongoDB 3.2 que implemente y comprenda esta definición de protocolo puede conectarse sin problemas con su base de datos de Cosmos DB utilizando la API de MongoDB.

- **Gremlin** Basado en el lenguaje transversal de gráficos Apache TinkerPop o Gremlin, esta API le permite almacenar información en Cosmos DB utilizando una estructura gráfica. Esto significa que en lugar de almacenar solo entidades, almacena
  - **Vértices** Puede pensar en un vértice como una entidad en otras estructuras de información. En una estructura gráfica típica, un vértice podría ser una persona, un dispositivo o un evento.
  - **Aristas** Son las relaciones entre vértices. Una persona puede conocer a otra persona, una persona puede poseer un tipo de dispositivo o una persona puede asistir a un evento.
  - **Propiedades** Estos son cada uno de los atributos que puede asignar a un vértice o una arista.

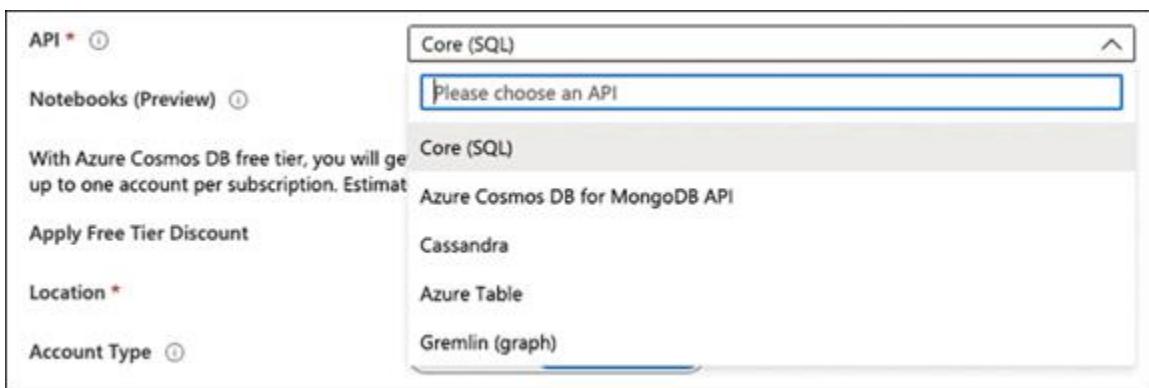
Tenga en cuenta que no puede combinar estas API en una sola cuenta de Cosmos DB. Debe definir la API que desea utilizar para acceder a su cuenta de Cosmos DB cuando la crea. Una vez que haya creado la cuenta, no podrá cambiar la API para acceder a ella.

Azure ofrece SDK para trabajar con las diferentes API que puede usar para conectarse a Cosmos DB. Los lenguajes admitidos son .NET, Java, Node.js y Python. Dependiendo de la API que desee usar para trabajar con Cosmos DB, también puede usar otros lenguajes como Xamarin, Golang o PHP. En esta sección, puede revisar un ejemplo de cada API y aprender a crear, leer, actualizar y eliminar datos utilizando las diferentes API.

Antes de comenzar con los ejemplos, debe crear una cuenta de Cosmos DB para almacenar sus datos. El siguiente procedimiento muestra cómo crear una cuenta gratuita de Cosmos DB con la API de SQL. Puede usar este mismo procedimiento para crear cuentas con las otras API que hemos revisado en esta habilidad:

1. Inicie sesión en Azure Portal (<http://portal.azure.com>).
2. En la esquina superior izquierda de Azure Portal, haga clic en el ícono de menú representado por tres barras horizontales y luego haga clic en Crear un recurso.

3. En el panel Nuevo, en la columna Azure Marketplace, haga clic en Bases de datos. En la columna Destacados, haga clic en Azure Cosmos DB.
4. En la hoja Crear cuenta de Azure Cosmos DB, en el menú desplegable Grupo de recursos, haga clic en el vínculo Crear nuevo debajo del menú desplegable. En el cuadro de diálogo emergente, escriba un nombre para el nuevo grupo de recursos. Alternativamente, puede seleccionar un grupo de recursos existente en el menú desplegable.
5. En la sección Detalles de la instancia, escriba un Nombre de cuenta.
6. En el menú desplegable de API, asegúrese de haber seleccionado la opción Core (SQL), como se muestra en la [Figura 2-1](#).



**Figura 2-1** Selección de una API de Cosmos DB

7. Asegúrese de que el interruptor Notebooks esté desactivado.
8. Asegúrese de que el interruptor Aplicar descuento de nivel gratuito esté configurado en Aplicar.
9. En el menú desplegable Ubicación, seleccione la región más adecuada para usted. Si usa App Services o máquinas virtuales, debe seleccionar la misma región en la que implementó esos servicios.
10. En el Tipo de cuenta, establezca el valor No producción.
11. Deje la redundancia geográfica y la escritura en varias regiones deshabilitadas.
12. En la esquina inferior izquierda de la hoja Crear cuenta de Azure Cosmos DB, haga clic en el botón Revisar + Crear.

13. En la esquina inferior izquierda de la pestaña Revisar + Crear, haga clic en el botón Crear para iniciar la implementación de su cuenta de Cosmos DB.

#### **Nota Emulador de Azure Cosmos DB**

Puede usar el emulador de Azure Cosmos DB durante la etapa de desarrollo de su aplicación. Debe tener en cuenta que existen algunas limitaciones a la hora de trabajar con el emulador en lugar de una cuenta real de Cosmos DB. El emulador solo es compatible con plataformas Windows o Docker para Windows. Puede revisar todas las características del emulador Cosmos DB en <https://docs.microsoft.com/en-us/azure/cosmos-db/local-emulator>.



#### *Sugerencia para el examen*

Puede utilizar diferentes API para acceder a su base de datos de Cosmos DB. Cada API ofrece una función diferente según la forma en que necesite representar sus datos. Recuerde que no puede cambiar la API una vez que haya creado su base de datos Cosmos DB.

#### *Implementar esquemas de particiones*

Cuando guarda datos en su cuenta de Cosmos DB, independientemente de la API que decida usar para acceder a sus datos, Azure coloca los datos en diferentes servidores para adaptarse al rendimiento y el rendimiento que necesita de un servicio de almacenamiento premium como Cosmos DB. Los servicios de almacenamiento utilizan particiones para distribuir los datos. Cosmos DB divide sus datos en partes más pequeñas llamadas particiones que se colocan en el servidor de almacenamiento. Hay dos tipos diferentes de particiones cuando se trabaja con Cosmos DB:

- **Lógico** Puede dividir un contenedor de Cosmos DB en partes más pequeñas según sus criterios. Cada una de estas piezas más pequeñas es una partición lógica. Todos los elementos almacenados en una partición lógica comparten la misma clave de partición.
- **Físicas** Estas particiones son un grupo de réplicas de sus datos que se almacenan físicamente en los servidores. Azure administra automáticamente este grupo de réplicas o conjuntos de réplicas. Una partición física puede contener una o más particiones lógicas.

### *¿Necesita más revisión? Partición física*

El único control que tiene sobre cómo se distribuyen los datos en las particiones físicas es configurar las claves de partición. Si desea revisar cómo se relacionan las particiones lógicas y las particiones físicas entre sí, consulte el siguiente artículo: <https://docs.microsoft.com/en-us/azure/cosmos-db/partition-data#physical-partitions>.

De forma predeterminada, cualquier partición lógica tiene un límite de 20 GB para almacenar datos. Cuando configura una nueva colección, debe decidir si desea que su colección se almacene en una única partición lógica y mantenerla por debajo del límite de 20 GB o permitir que supere ese límite y abarque diferentes particiones lógicas. Si necesita que su contenedor se divida en varias particiones, Cosmos DB necesita alguna forma de saber cómo distribuir sus datos entre las diferentes particiones lógicas. Aquí es donde entra en juego la clave de partición. Tenga en cuenta que esta clave de partición es inmutable, lo que significa que no puede cambiar la propiedad que desea usar como clave de partición una vez que la haya seleccionado.

La elección de la clave de partición correcta es fundamental para lograr el mejor rendimiento. La razón por la que elegir la clave de partición adecuada es tan importante es porque Azure crea una partición lógica para cada valor distinto de su clave de partición. El Listado 2-1 muestra un ejemplo de un documento JSON.

**Listado 2-1** Ejemplo de documento JSON

---

```
{  
    "id": "1",  
    "firstName": "Santiago",  
    "lastName": "Fernández",  
    "ciudad": "Sevilla",  
    "país": "España"  
}
```

Dependiendo de sus datos, las propiedades de la ciudad o el país serían la elección correcta para la clave de partición. Puede encontrar en sus datos que algunos documentos tienen el mismo valor para la propiedad del país,

por lo que se almacenan juntos en la misma partición lógica. El uso de la propiedad id como clave de partición significa que termina con una partición lógica con un solo documento en cada partición. Esta configuración puede ser beneficiosa cuando su aplicación normalmente realiza cargas de trabajo de lectura y utiliza técnicas de paralelización para obtener los datos.

Por otro lado, si selecciona una clave de partición con solo unos pocos valores posibles, puede terminar con particiones "activas". Una partición "activa" es una partición que recibe la mayoría de las solicitudes cuando trabaja con sus datos. La principal implicación de estas particiones "activas" es que, por lo general, alcanzan el límite de rendimiento de la partición, lo que significa que debe aprovisionar más rendimiento. Otro posible inconveniente es que puede alcanzar el límite de 20 GB para una sola partición lógica. Debido a que una partición lógica es el ámbito para transacciones de múltiples documentos eficientes, seleccionar una clave de partición con algunos valores posibles le permite ejecutar transacciones en muchos documentos dentro de la misma partición.

Utilice las siguientes pautas al seleccionar su clave de partición:

- El límite de almacenamiento para una única partición lógica es de 20 GB. Si prevé que sus datos requerirán más espacio para cada valor de la partición, debe seleccionar otra clave de partición.
- Las solicitudes a una sola partición lógica no pueden exceder el límite de rendimiento para esa partición. Si sus solicitudes alcanzan ese límite, se regulan para evitar exceder el límite. Si alcanza este límite con frecuencia, debe seleccionar otra clave de partición porque es muy probable que tenga una partición "activa". El límite de rendimiento mínimo es diferente de las bases de datos a los contenedores. El rendimiento mínimo de las bases de datos es de 100 unidades de solicitud por segundo (RU / s). El rendimiento mínimo de los contenedores es de 400 RU / s.
- Elija claves de partición con una amplia gama de valores y patrones de acceso que puedan distribuir las solicitudes de manera uniforme entre las particiones lógicas. Esto le permite lograr el equilibrio adecuado entre poder ejecutar transacciones entre documentos y escalabilidad. El uso de claves de partición basadas en marcas de tiempo suele ser una mala elección para una clave de partición.

- Revise sus requisitos de carga de trabajo. La clave de partición que elija debería permitir que su aplicación funcione bien en cargas de trabajo de lectura y escritura.
- Los parámetros que suele utilizar en sus solicitudes y consultas de filtrado son buenos candidatos para una clave de partición.

### *¿Necesita más revisión? Fraccionamiento*

Puede revisar más información sobre cómo funciona la partición revisando el siguiente artículo: <https://docs.microsoft.com/en-us/azure/cosmos-db/partitioning-overview>

Puede haber situaciones en las que ninguna de las propiedades de sus elementos sea apropiada para las claves de partición. En esas situaciones, puede crear claves de partición sintéticas. Una partición sintética key es un compuesto clave de dos propiedades concatenadas. En nuestro ejemplo de documento anterior que se muestra en el [Listado 2-1](#), creó una nueva propiedad denominada partitionKey que contiene una cadena que concatena los valores de ciudad y país. Para el documento de ejemplo, el valor de la partición clave debe ser *Sevilla-España*.



### *Sugerencia para el examen*

Recuerde que sus datos se distribuyen entre las diferentes particiones lógicas mediante el uso de la clave de partición. Por esta razón, una vez que haya elegido una clave de partición, no podrá cambiarla.

### *Interactuar con los datos usando el SDK apropiado*

Cosmos DB le permite acceder a los datos utilizando diferentes tipos de API. Una vez que tenga lista su cuenta de Cosmos DB, puede comenzar a crear sus bases de datos y contenedores para trabajar con datos. Recuerde que una vez que elija la API para su cuenta de Cosmos DB, no podrá cambiarla.

El siguiente ejemplo muestra cómo crear una aplicación de consola con .NET Core. El primer ejemplo usa la API SQL de Cosmos DB para crear, actualizar y eliminar algunos elementos en la cuenta de Cosmos DB:

1. Abra Visual Studio Code y cree un directorio para almacenar el proyecto de ejemplo.

2. Abra la Terminal, cambie al directorio del proyecto y escriba el siguiente comando:

```
nueva consola dotnet
```

3. Instale el paquete NuGet para interactuar con su cuenta de Cosmos DB mediante la API de SQL. Escriba el siguiente comando en la Terminal:

[Haga clic aquí para ver la imagen del código](#)

```
dotnet agregar paquete Microsoft.Azure.Cosmos
```

4. Cambie el contenido del archivo Program.cs utilizando el contenido proporcionado en el [Listado 2-2](#). Debe cambiar el espacio de nombres de acuerdo con el nombre de su proyecto.

5. Inicie sesión en Azure Portal (<http://portal.azure.com>).

6. En el cuadro de búsqueda en la parte superior de Azure Portal, escriba el nombre de su cuenta de Cosmos DB y haga clic en el nombre de la cuenta.

7. En la hoja Cuenta de Cosmos DB, en la sección Configuración, haga clic en Claves.

8. En el panel Claves, copie los valores de URI y Claves primarias de la pestaña Claves de lectura y escritura. Debe proporcionar estos valores a EndpointUri y Key Constants en el código que se muestra en el [Listado 2-2](#). (Las partes más importantes del código se muestran en negrita).

#### **Listado 2-2 Ejemplo de API SQL de Cosmos DB**

[Haga clic aquí para ver la imagen del código](#)

---

```
// C # .NET Core. Program.cs

using System.Collections.Immutable;

utilizando System.Xml.Linq;

utilizando System.Diagnostics;

utilizando System.Runtime.CompilerServices;

usando el sistema;
```

```
utilizando System.Linq;
utilizando Microsoft.Azure.Cosmos;
usando System.Threading.Tasks;
usando ch2_1_3_SQL.Model;
usando System.Net;

espacio de nombres ch2_1_3_SQL

{
    programa de clase
    {
        private const string EndpointUri = "<PONGA AQUÍ SU
URL DE PUNTO FINAL>";
        private const string Key = "<PONGA AQUÍ SU CLAVE
COSMOS DB>";
        cliente privado de CosmosClient;
        base de datos privada de la base de datos;
        contenedor contenedor privado;

static void Main (cadena [] argumentos)
{
    intentar
{
```

```
        Demostración del programa = nuevo programa
    () ;

        demo.StartDemo (). Wait ();

    }

    captura (CosmosException ce)

    {

        Excepción baseException =
ce.GetBaseException ();

        System.Console.WriteLine (Se produjo el
error $ "{ce.StatusCode}:

                {ce.Message}, Mensaje:
{baseException.Message} ");

    }

    catch (Excepción ex)

    {

        Excepción baseException =
ex.GetBaseException ();

        System.Console.WriteLine ($ "Ocurrió un
error: {ex.Message}, Mensaje:
{baseException.Message} ");

    }

}
```

```
Tarea asíncrona privada StartDemo ()

{
```

```
Console.WriteLine ("¡Iniciando la demostración  
de la API de SQL de Cosmos DB!");  
  
// Crea una nueva base de datos de demostración  
  
string databaseName = "demoDB_" + Guid.NewGuid  
().ToString ().  
Subcadena (0, 5);  
  
this.SendMessageToConsoleAndWait ($ "Creando la  
base de datos {databaseName} ...");  
  
this.client = new CosmosClient (EndpointUri,  
Key);  
  
this.database = espera a  
this.client.CreateDatabaseIfNotExistsAsync  
(nombre de la base de datos);  
  
// Cree una nueva colección de demostración  
dentro de la base de datos de demostración.  
  
// Esto crea una colección con un rendimiento  
reservado. Puedes personalizar  
las opciones usando un objeto ContainerProperties  
  
// Esta operación tiene implicaciones de  
precios.  
  
string containerName = "colección_" +  
Guid.NewGuid ().ToString () .
```

```
Subcadena (0, 5);

    this.SendMessageToConsoleAndWait ($ "Creación de
demonstración de colección

{containerName} ... ");

        this.container = await
this.database.CreateContainerIfNotExistsAsync

(containerName, "/ LastName");

// Crea algunos documentos en la colección

Persona person1 = nueva persona

{

    Id = "Persona.1",

    FirstName = "Santiago",

    LastName = "Fernandez",

    Dispositivos = nuevo dispositivo []

    {

        nuevo dispositivo {OperatingSystem =
"iOS", CameraMegaPixels = 7,

        Ram = 16, Uso = "Personal"},

        nuevo dispositivo {OperatingSystem =
"Android", CameraMegaPixels = 12,
```

```
Ram = 64, Uso = "Trabajo"}  
},  
Sexo = "Masculino",  
Dirección = nueva dirección  
{  
Ciudad = "Sevilla",  
Country = "España",  
PostalCode = "28973",  
Calle = "Diagonal",  
Estado = "Andalucía"  
},  
IsRegistered = verdadero  
};
```

```
esperar esto.CreateDocumentIfNotExistsAsync  
(databaseName, containerName,  
personal);
```

```
Persona person2 = nueva persona  
{  
Id = "Persona.2",  
FirstName = "Agatha",  
LastName = "Smith",
```

```
Dispositivos = nuevo dispositivo []
{

    nuevo dispositivo {OperatingSystem =
"iOS", CameraMegaPixels = 12,
                    Ram = 32, Uso = "Trabajo"},

    nuevo dispositivo {OperatingSystem =
"Windows", CameraMegaPixels = 12,
                    Ram = 64, Uso = "Personal"}

} ,
Sexo = "Mujer",
Dirección = nueva dirección
{
Ciudad = "Laguna Beach",
País = "Estados Unidos",
PostalCode = "12345",
Calle = "Principal",
Estado = "CA"

},
IsRegistered = verdadero
};

esperar esto.CreateDocumentIfNotExistsAsync
(databaseName, containerName,
```

```
persona2);

        // Realiza algunas consultas a la colección

        this.SendMessageToConsoleAndWait ($ "Obtener
documentos de la colección

{containerName} ... ");

        // Encuentra documentos usando LINQ

        IQueryables <Person> queryablePeople =
this.container.GetItemLinqQueryable
<Person> (true)

        .Where (p => p.Gender == "Male");

        System.Console.WriteLine ("Ejecutando consulta
LINQ para encontrar hombres ...");

        foreach (Person foundPerson en queryablePeople)

        {

            System.Console.WriteLine ($ "\tPerson:
{foundPerson}");

        }

        // Encuentra documentos usando SQL

        var sqlQuery = "SELECCIONAR * DE Persona DONDE
Person.Gender = 'Mujer'";
```

```
        QueryDefinition queryDefinition = new
QueryDefinition (sqlQuery);

        FeedIterator <Person> peopleResultSetIterator =
this.container.GetItemQuery Iterator

<Person> (queryDefinition) ;

System.Console.WriteLine ("Ejecución de una
consulta SQL para encontrar mujeres ...");

while (peopleResultSetIterator.hasMoreResults)

{

    FeedResponse <Person> currentResultSet =
await peopleResultSetIterator.

ReadNextAsync ();

    foreach (Person foundPerson en
currentResultSet )

    {

        System.Console.WriteLine ($ "\tPerson:
{foundPerson} ");

    }

}

Console.WriteLine ("Presione cualquier tecla
para continuar ...");

Console.ReadKey ();

// Actualizar documentos en una colección
```

```
        this.SendMessageToConsoleAndWait ($
    "Actualización de documentos en la colección

{containerName} ... ");

        person2.FirstName = "Mateo";

        person2.Gender = "Hombre";


aguardar this.container.UpsertItemAsync
(person2) ;

        this.SendMessageToConsoleAndWait ($ "Documento
modificado {persona2}" );




// Eliminar un solo documento de la colección

        this.SendMessageToConsoleAndWait ($ "Eliminando
documentos de la colección

{containerName} ... ");


PartitionKey partitionKey = nueva PartitionKey
(person1.LastName) ;

espere this.container.DeleteItemAsync <Persona>
(person1.Id, partitionKey) ;

        this.SendMessageToConsoleAndWait ($ "Documento
eliminado {personal1}" );


// Eliminar la base de datos de demostración
creada y todos sus elementos secundarios

        this.SendMessageToConsoleAndWait ("Limiando su
cuenta de Cosmos DB ...");
```

```

    espera a this.database.DeleteAsync () ;

}

Private void SendMessageToConsoleAndWait (mensaje de
cadena)

{
    Console.WriteLine (mensaje);

    Console.WriteLine ("Presione cualquier tecla
para continuar ...");

    Console.ReadKey ();
}

Tarea asíncrona privada
CreateDocumentIfNotExistsAsync (base de datos de cadenas,
colección de cadenas, persona persona)

{
    intentar

    {

        esperar esto? .container.ReadItemAsync
<Person> (person.Id,
new PartitionKey (person.LastName));

this.SendMessageToConsoleAndWait ($ "El
documento {person.Id} ya existe en la colección
{colección}");

}

captura (CosmosException dce)

```

```

    {

        si (dce.StatusCode ==
HttpStatusCode.NotFound)

        {

            esperar esto? .container.CreateItemAsync
<Person> (persona,

nueva PartitionKey (person.LastName)) ;

            this.SendMessageToConsoleAndWait ($
"Nuevo documento creado

{person.Id} en la colección {colección} ");

        }

    }

}
}

```

Cuando trabaja con la API de SQL, el SDK de Azure Cosmos DB le proporciona las clases adecuadas para trabajar con los diferentes elementos de la cuenta. En el ejemplo que se muestra en el [Listado 2-2](#), debe crear un objeto CosmosClient antes de poder acceder a su cuenta de Azure Cosmos DB. El SDK de Azure Cosmos DB también le proporciona las clases Base de datos y Contenedor para trabajar con estos elementos. Cuando necesite crear una base de datos o un contenedor, puede usar CreateDatabaseIfNotExistsAsync o CreateContainerIfNotExistsAsync, respectivamente. Estos métodos IfNotExists comprueban automáticamente para determinar si el contenedor o la base de datos existe en su cuenta de Cosmos DB; si no existen, el método crea automáticamente el contenedor o la base de datos. Cuando crea un nuevo contenedor en su base de datos, observe que

en este ejemplo, ha proporcionado la PartitionKey utilizando la sobrecarga del constructor adecuada.

Sin embargo, cuando es necesario crear un nuevo documento en la base de datos, que no tiene disponibles de este tipo de IfNotExist s método. En esta situación, tiene dos opciones:

1. Utilice el método UpsertItemAsync, que crea un nuevo documento si el documento no existe o actualiza un documento existente.
2. Implemente su propia versión del método IfNotExist s , por lo que debe verificar si el documento ya existe en el contenedor. Si el documento no existe, cree el documento real, como se muestra en el siguiente fragmento del Listado 2-2 . (El código en negrita muestra los métodos que debe utilizar para crear un documento).

Haga clic aquí para ver la imagen del código

```
intentar

{

    esperar esto? .container.ReadItemAsync <Person> (person.Id, new
PartitionKey

(person.LastName)) ;



    this.SendMessageToConsoleAndWait ($ "El documento {person.Id}
ya existe en

colección {colección} ");

}

captura (CosmosException dce)

{

    si (dce.StatusCode == HttpStatusCode.NotFound)

{
```

```

esperar esto? .container.CreateItemAsync <Person> (persona,
nueva PartitionKey (person.LastName)) ;

this.SendMessageToConsoleAndWait ($ "Creó un nuevo
documento {person.Id} en
colección {colección} ");

}
}

```

Cuando crea el documento con el método CreateItemAsync, observe que puede proporcionar el valor de la clave de partición mediante el siguiente fragmento de código new PartitionKey (person.LastName). Si no proporciona el valor de la clave de partición, el valor correcto se infiere del documento que está intentando insertar en la base de datos.

Debe hacer esta verificación porque obtiene una CosmosException con StatusCode 409 (Conflict) si intenta crear un documento con el mismo Id de un documento ya existente en la colección. De manera similar, obtiene una CosmosException con StatusCode 404 (No encontrado) si intenta eliminar un documento que no existe en el contenedor usando el método DeleteItemAsync o si intenta reemplazar un documento que no existe en el contenedor usando el Método ReplaceItemAsync. Tenga en cuenta que estos dos métodos también aceptan un parámetro de clave de partición.

Cuando crea un documento, debe proporcionar una propiedad Id de tipo cadena a su documento. Esta propiedad necesita identificar su documento dentro de la colección de forma única. Si no proporciona esta propiedad, Cosmos DB la agrega automáticamente al documento mediante una cadena GUID.

Como puede ver en el código de ejemplo en el [Listado 2-2](#), puede consultar sus documentos usando oraciones LINQ o SQL. En este ejemplo, he usado una consulta SQL bastante simple para obtener documentos que representan a una persona con el género masculino. Sin embargo, puede construir oraciones más complejas como una consulta que devuelve todas las personas que viven en un país específico, usando la expresión WHERE

Address.Country = 'Spain', o personas que tienen un dispositivo Android usando WHERE ARRAY\_CONTAINS (Person.Devices, {'OperatingSystem': 'Android'}, verdadero) expresión.

#### *¿Necesita más revisión? Consultas SQL con Cosmos DB*

Puede revisar todas las capacidades y características del lenguaje SQL que implementa Cosmos DB al revisar este artículo:

- **Referencia del lenguaje SQL para Azure Cosmos DB** <https://docs.microsoft.com/en-us/azure/cosmos-db/sql-api-query-reference>

Una vez que haya modificado el archivo Program.cs, debe crear algunas clases adicionales que utilice en el programa principal para administrar documentos. Puede encontrar estas nuevas clases en los listados 2-3 a 2-5.

1. En la ventana Código de Visual Studio, cree una nueva carpeta denominada **Modelo** en la carpeta del proyecto.
2. Cree un nuevo archivo de clase C # en la carpeta Modelo y **asígnele el nombre Person.cs**.
3. Reemplace el contenido del archivo Person.cs con el contenido del Listado 2-3. Cambie el espacio de nombres según sea necesario para su proyecto.
4. Cree un nuevo archivo de clase C # en la carpeta Modelo y **asígnele el nombre Device.cs**.
5. Reemplace el contenido del archivo Device.cs con el contenido del Listado 2-4. Cambie el espacio de nombres según sea necesario para su proyecto.
6. Cree un nuevo archivo de clase C # en la carpeta Modelo y **asígnele el nombre Address.cs**.
7. Reemplace el contenido del archivo Address.cs con el contenido del Listado 2-5. Cambie el espacio de nombres según sea necesario para su proyecto.
8. En este punto, puede ejecutar el proyecto presionando F5 en la ventana Código de Visual Studio. Verifique cómo su código está creando y modificando las diferentes bases de datos, colecciones de documentos y documentos en su cuenta de Cosmos DB. Puede revisar los cambios en su cuenta de Cosmos DB con la herramienta Data Explorer en su cuenta de Cosmos DB en Azure Portal.

**Listado 2-3 Ejemplo de API SQL de Cosmos DB: Person.cs**

Haga clic aquí para ver la imagen del código

---

```
// C # .NET Core.

utilizando Newtonsoft.Json;

espacio de nombres ch2_1_3_SQL.Model

{

    Persona de clase pública

    {

        [JsonProperty (PropertyName = "id")]

        Id de cadena pública {get; colocar; }

        cadena pública FirstName {get; colocar; }

        cadena pública LastName {get; colocar; }

        Dispositivo público [] Dispositivos {get; colocar; }

        Dirección pública Dirección {get; colocar; }

        cadena pública Género {get; colocar; }

        public bool IsRegistered {get; colocar; }

        cadena de anulación pública ToString ()

        {

            return JsonConvert.SerializeObject (esto);

        }

    }

}
```

**Listado 2-4 Ejemplo de API SQL de Cosmos DB: Device.cs**

Haga clic aquí para ver la imagen del código

---

```
// C # .NET Core.

espacio de nombres ch2_1_3_SQL.Model

{

    Dispositivo de clase pública

    {

        public int Ram {get; colocar; }

        OperatingSystem de cadena pública {get; colocar; }

        public int CameraMegaPixels {obtener; colocar; }

        Uso de cadenas públicas {get; colocar; }

    }

}
```

**Listado 2-5** Ejemplo de API SQL de Cosmos DB: Address.cs

Haga clic aquí para ver la imagen del código

---

```
// C # .NET Core.

espacio de nombres ch2_1_3_SQL.Model

{

    Dirección de clase pública

    {

        cadena pública Ciudad {get; colocar; }

        Estado de cadena pública {get; colocar; }

        cadena pública PostalCode {get; colocar; }

        cadena pública País {get; colocar; }

        cadena pública Street {get; colocar; }

    }

}
```

```
}
```

```
}
```

En este punto, puede presionar F5 en su ventana de código de Visual Studio para ejecutar el código. El código se detiene en cada paso para que pueda ver el resultado de la operación directamente en Azure Portal. Siga los siguientes pasos para ver las modificaciones en su cuenta de Cosmos DB:

1. Inicie sesión en Azure Portal (<http://portal.azure.com>).
2. En el cuadro de búsqueda en la parte superior de Azure Portal, escriba el nombre de su cuenta de Cosmos DB y haga clic en el nombre de la cuenta.
3. En la hoja Cuenta de Cosmos DB, haga clic en Explorador de datos.
4. En la hoja Data Explorer, en el lado izquierdo del panel, debajo de la etiqueta SQL API, debería poder ver la lista de bases de datos creadas en su cuenta de Cosmos DB.

Trabajar con la API de MongoDB para Cosmos DB es tan fácil como trabajar con cualquier otra biblioteca de Mongo DB. Solo necesita usar la cadena de conexión que puede encontrar en el panel Cadena de conexión en la sección Configuración en su cuenta de Azure Cosmos DB.

El siguiente ejemplo muestra cómo usar Cosmos DB en su proyecto MongoDB. Para este ejemplo, utilizará MERN (MongoDB, Express, React y Node), que es un marco de trabajo de pila completa para trabajar con MongoDB y NodeJS. Además, debe cumplir con los siguientes requisitos:

- Debe tener la última versión de NodeJS instalada en su computadora.
- Debe tener una cuenta de Azure Cosmos DB configurada para usar la API de MongoDB. Recuerde que puede usar el mismo procedimiento que se usó anteriormente para crear un Cosmos DB con la API de SQL para crear una cuenta de Azure Cosmos DB con la API de MongoDB. Solo necesita seleccionar la API correcta cuando crea su cuenta de Cosmos DB.
- Necesita una de las cadenas de conexión que puede encontrar en el panel Cadena de conexión en su cuenta de Azure Cosmos DB

en Azure Portal. Debe copiar una de estas cadenas de conexión porque debe usarla más adelante en el código.

Utilice los siguientes pasos para conectar un proyecto MERN con Cosmos DB mediante la API de MongoDB:

1. Crea una nueva carpeta para tu proyecto.
2. Abra la terminal y ejecute los siguientes comandos:

[Haga clic aquí para ver la imagen del código](#)

```
clon de git https://github.com/Hashnode/mern-starter.git  
cd mern-starter  
npm install
```

3. Abra su editor preferido y abra la carpeta mern-starter. No cierre la ventana de terminal que abrió antes.
4. En la carpeta mern-starter, en la subcarpeta del servidor, abra el archivo config.js y reemplace el contenido del archivo con el siguiente código:

[Haga clic aquí para ver la imagen del código](#)

```
const config = {  
  
    mongoURL: process.env.MONGO_URL ||  
'<YOUR_COSMOSDB_CONNECTION_STRING>',  
  
    puerto: process.env.PORT || 8000,  
  
};  
  
exportar configuración predeterminada;
```

5. En la ventana de terminal, ejecute el comando npm start. Este comando inicia el proyecto NodeJS y crea un servidor Node que escucha en el puerto 8000.
6. Abra un navegador web y navegue hasta *http://localhost:8000*. Esto abre el proyecto web MERN.
7. Abra una nueva ventana del explorador, navegue hasta Azure Portal y abra el explorador de Data Explorer en su cuenta de Azure Cosmos DB.

8. En el proyecto MERN, cree, modifique o elimine algunas publicaciones. Revise cómo se crea, modifica y elimina el documento de su cuenta de Cosmos DB.

#### *¿Necesita más revisión? Ejemplos de Gremlin y Cassandra*

Como puede ver en los ejemplos anteriores, la integración de su código existente con Cosmos DB no requiere demasiado esfuerzo ni muchos cambios en su código. En aras de la brevedad, decidimos omitir los ejemplos de cómo conectar sus aplicaciones Cassandra o Gremlin con Cosmos DB. Puede aprender a realizar estas integraciones revisando los siguientes artículos:

- **Inicio rápido: compile una aplicación .NET Framework o Core con la cuenta de la API de Azure Cosmos DB Gremlin** <https://docs.microsoft.com/en-us/azure/cosmos-db/create-graph-dotnet>
- **Inicio rápido: cree una aplicación Cassandra con .NET SDK y Azure Cosmos DB** <https://docs.microsoft.com/en-us/azure/cosmos-db/create-cassandra-dotnet>

#### *Establecer el nivel de coherencia adecuado para las operaciones*

Uno de los principales beneficios que ofrece Cosmos DB es la capacidad de tener sus datos distribuidos por todo el mundo con baja latencia al acceder a los datos. Esto significa que puede configurar Cosmos DB para replicar sus datos entre cualquiera de las regiones de Azure disponibles mientras logra una latencia mínima cuando su aplicación accede a los datos de la región más cercana. Si necesita replicar sus datos en una región adicional, solo necesita agregarlos a la lista de regiones en las que sus datos deberían estar disponibles.

Esta replicación en las diferentes regiones tiene un inconveniente: la coherencia de sus datos. Para evitar la corrupción, sus datos deben ser consistentes entre todas las copias de su base de datos. Afortunadamente, el protocolo Cosmos DB ofrece cinco niveles de replicación de coherencia. Pasando de la coherencia al rendimiento, puede seleccionar cómo se comporta el protocolo de replicación al copiar sus datos entre todas las réplicas que están configuradas en todo el mundo. Estos niveles de coherencia son independientes de la región, lo que significa que la región que inició la operación de lectura o escritura o la cantidad de regiones asociadas con su cuenta de Cosmos DB no importa, incluso si configuró una sola región para su cuenta. Este nivel de coherencia se configura en el nivel de Cosmos DB y se aplica a todas las 92 bases de

datos, colecciones y documentos almacenados en el mismo cuenta. Puede elegir entre los niveles de coherencia que se muestran en la Figura 2-2. Utilice el siguiente procedimiento para seleccionar el nivel de coherencia:

1. Inicie sesión en Azure Portal (<http://portal.azure.com>).
2. En el cuadro de búsqueda en la parte superior de Azure Portal, escriba el nombre de su cuenta de Cosmos DB y haga clic en el nombre de la cuenta.
3. En la hoja de su cuenta de Cosmos DB, haga clic en Consistencia predeterminada en la sección Configuración.
4. En la hoja Consistencia predeterminada, seleccione el nivel de consistencia deseado. Sus opciones son Fuerte, Vigencia limitada, Sesión, Prefijo consistente y Eventual.
5. Haga clic en el icono Guardar en la esquina superior izquierda de la hoja Consistencia predeterminada.



**Figura 2-2** Selección del nivel de consistencia

- **Fuerte** Se garantiza que las operaciones de lectura devolverán la versión confirmada más reciente de un elemento; es decir, el usuario siempre lee la última escritura confirmada. Este nivel de consistencia es el único que ofrece una garantía de linealización. Esta garantía tiene un precio. Tiene una mayor latencia debido al tiempo necesario para escribir confirmaciones de operaciones y la disponibilidad puede verse afectada durante fallas.
- **Vigencia limitada** Se garantiza que las lecturas sean coherentes dentro de un retraso preconfigurado. Este retraso puede consistir en varias de las versiones más recientes (K) o en un intervalo de tiempo (T). Esto significa que si realiza operaciones de escritura, la lectura de estas operaciones ocurre en el mismo orden pero con un retraso máximo de K versiones de los datos escritos o T segundos desde que escribió los datos en la base de datos. Para

las operaciones de lectura que ocurren dentro de una región que acepta escrituras, el nivel de consistencia es idéntico al nivel de consistencia Fuerte. Este nivel también se conoce como "garantía de linealización retardada".

- **Sesión** Dirigida a una sesión de cliente, este nivel de coherencia ofrece el mejor equilibrio entre un nivel de coherencia fuerte y el rendimiento proporcionado por el nivel de coherencia eventual. Se adapta mejor a las aplicaciones en las que se producen operaciones de escritura en el contexto de una sesión de usuario.
- **Prefijo consistente** Este nivel garantiza que siempre lea los datos en el mismo orden en que los escribió, pero no hay garantía de que pueda leer todos los datos. Esto significa que si escribe "A, B, C", puede leer "A", "A, B" o "A, B, C" pero nunca "A, C" o "B, A, C".
- **Eventual** No hay garantía para el orden en que lee los datos. En ausencia de una operación de escritura, las réplicas eventualmente convergen. Este nivel de coherencia ofrece un mejor rendimiento a costa de la complejidad de la programación. Utilice este nivel de coherencia si el orden de los datos no es esencial para su aplicación.

#### *Tenga en cuenta las compensaciones de consistencia, disponibilidad y rendimiento*

Cada nivel de coherencia que se muestra en esta sección tiene sus implicaciones en términos de coherencia de datos, disponibilidad de datos y rendimiento de la aplicación. Puede revisar las implicaciones de elegir cada uno de los niveles de coherencia revisando el siguiente artículo: <https://docs.microsoft.com/en-us/azure/cosmos-db/consistency-levels-tradeoffs>.

La mejor elección de nivel de coherencia depende de su aplicación y de la API que desee utilizar para almacenar datos. Como puede ver en los diferentes niveles de consistencia, los requisitos de su aplicación con respecto a la consistencia de lectura de datos versus la disponibilidad, latencia y rendimiento son factores críticos que debe considerar al hacer su selección.

Debe considerar los siguientes puntos cuando use SQL o Table API para su cuenta de Cosmos DB:

- La opción recomendada para la mayoría de las aplicaciones es el nivel de coherencia de la sesión.

- Si está considerando el nivel de coherencia fuerte, le recomendamos que utilice el nivel de coherencia de obsolescencia enlazada porque proporciona una garantía de linealización con un retraso configurable.
- Si está considerando el nivel de consistencia eventual, le recomendamos que utilice el nivel de consistencia de prefijo consistente porque proporciona niveles comparables de disponibilidad y latencia con la ventaja de órdenes de lectura garantizadas.
- Evalúe cuidadosamente los niveles de consistencia fuerte y eventual porque son las opciones más extremas. En la mayoría de las situaciones, otros niveles de coherencia pueden proporcionar un mejor equilibrio entre el rendimiento, la latencia y la coherencia de los datos.

#### *¿Necesita más revisión? Compensación de niveles de consistencia*

Cada nivel de consistencia tiene un precio. Puede revisar las implicaciones de elegir cada nivel de coherencia leyendo el artículo "Compensaciones de coherencia, disponibilidad y rendimiento" en <https://docs.microsoft.com/en-us/azure/cosmos-db/consistency-levels-tradeoffs> .

Cuando utiliza las API de Cassandra o MongoDB, Cosmos DB asigna los niveles de coherencia ofrecidos por Cassandra y MongoDB al nivel de coherencia ofrecido por Cosmos DB. La razón para hacer esto es porque cuando usa estas API, ni Cassandra ni MongoDB ofrecen un nivel de consistencia bien definido. En cambio, Cassandra proporciona niveles de coherencia de lectura o escritura que se asignan al nivel de coherencia de Cosmos DB de las siguientes formas:

- **Nivel de coherencia de escritura de Cassandra** Este nivel se asigna al nivel de coherencia de cuenta de Cosmos DB predeterminado.
- **Nivel de coherencia de lectura de Cassandra** Cosmos DB asigna dinámicamente el nivel de coherencia especificado por el cliente del controlador de Cassandra a uno de los niveles de coherencia de Cosmos DB.

Por otro lado, MongoDB le permite configurar los siguientes niveles de coherencia: Preocupación de escritura, Preocupación de lectura y Directiva maestra. De manera similar al mapeo de los niveles de consistencia de Cassandra, los niveles de consistencia de Cosmos DB se

asignan a los niveles de consistencia de MongoDB de las siguientes maneras:

- **Nivel de coherencia de preocupación de escritura de MongoDB** Este nivel se asigna al nivel de coherencia de cuenta de Cosmos DB predeterminado.
- **Nivel de coherencia de preocupación de lectura de MongoDB** Cosmos DB asigna dinámicamente el nivel de coherencia especificado por el cliente del controlador MongoDB a uno de los niveles de coherencia de Cosmos DB.
- **Configuración de una región maestra** Puede configurar una región como “maestra” de MongoDB configurando la región como la primera región de escritura.

*¿Necesita más revisión? Asignaciones de niveles de coherencia de Cassandra y Mongodb*

Puede revisar cómo se asignan los diferentes niveles de coherencia entre Cassandra y MongoDB y los niveles de coherencia de Cosmos DB en el artículo "Niveles de coherencia y API de Azure Cosmos DB" en <https://docs.microsoft.com/en-us/azure/cosmos-db/niveles-de-consistencia-en-apis> .



### *Sugerencia para el examen*

El nivel de coherencia afecta la latencia y la disponibilidad de los datos. En términos generales, debe evitar los niveles más extremos, ya que tienen un impacto más significativo en su programa que debe evaluarse cuidadosamente. Si no está seguro de qué nivel de consistencia debe usar, debe usar el nivel de sesión, ya que este es el nivel mejor equilibrado.

### *Crear contenedores de Cosmos DB*

Cuando trabaja con Cosmos DB, tiene varias capas en la jerarquía de entidades administradas por la cuenta de Cosmos DB. La primera capa es la cuenta de Azure Cosmos DB, donde elige la API que desea usar para acceder a sus datos. Recuerde que esta API tiene implicaciones sobre cómo se almacenan los datos en las bases de datos.

La segunda capa de la jerarquía es la base de datos. Puede crear tantas bases de datos como necesite en su cuenta de Cosmos DB. Las bases de datos son una forma de agrupar contenedores, y puede pensar en bases de datos como en espacios de nombres. En este nivel, puede configurar el

rendimiento asociado a los contenedores incluidos en la base de datos. Dependiendo de la API que esté utilizando, la base de datos tiene un nombre diferente:

- Base de datos **API SQL** .
- Espacio de claves de la **API de Cassandra** .
- Base de datos **API de MongoDB** .
- Base de datos **API de Gremlin** .
- **API de tabla** Este concepto no se aplica a la API de tabla, aunque bajo el capó cuando crea su primera tabla, Cosmos DB crea una base de datos predeterminada para usted.

Un contenedor en una cuenta de Azure Cosmos DB es la unidad de escalabilidad para el rendimiento y el almacenamiento. Cuando crea un nuevo contenedor, debe configurar la clave de partición para establecer la forma en que los elementos que se almacenarán en el contenedor se distribuirán entre las diferentes particiones lógicas y físicas. Como revisamos en la sección “Implementar esquemas de particionamiento” anteriormente en este capítulo, el rendimiento se distribuye entre las particiones lógicas definidas por la clave de partición.

Cuando crea un nuevo contenedor, puede decidir si el rendimiento del contenedor es uno de los dos modos siguientes:

- **Dedicado** Todo el rendimiento se aprovisiona para un contenedor. En este modo, Azure realiza una reserva de recursos para el contenedor que está respaldado por SLA.
- **Compartido** El rendimiento se comparte entre todos los contenedores configurados en la base de datos, excepto aquellos contenedores que se han configurado como modo de rendimiento dedicado. El rendimiento compartido se configura en el nivel de la base de datos.

Cuando crea un contenedor de Cosmos DB, hay un conjunto de propiedades que puede configurar. Estas propiedades afectan diferentes aspectos del contenedor o la forma en que se almacenan o administran los artículos. La siguiente lista muestra las propiedades de un contenedor que se pueden configurar. Tenga en cuenta que no todas las propiedades están disponibles para todas las API:

- **IndexingPolicy** Cuando agrega un elemento a un contenedor, de forma predeterminada, todas las propiedades del elemento se indexan automáticamente. No importa si todos los elementos de la colección comparten el mismo esquema o si cada elemento tiene su propio esquema. Esta propiedad le permite configurar cómo indexar los elementos en el contenedor. Puede configurar diferentes tipos de índices e incluir o excluir algunas propiedades de los índices.
- **TimeToLive (TTL)** Puede configurar su contenedor para eliminar elementos después de un período de tiempo automáticamente. TimeToLive se expresa en segundos. Puede configurar el valor TTL a nivel de contenedor o artículo. Si configura el TTL a nivel del contenedor, todos los elementos del contenedor tienen el mismo TTL, excepto si configura un TTL para un elemento específico. Un valor de -1 en el TTL significa que el artículo no caduca. Si establece un valor TTL para un artículo en el que su contenedor no tiene configurado un valor TTL, el TTL a nivel del artículo no tiene ningún efecto.
- **ChangeFeedPolicy** Puede leer los cambios realizados en un elemento en un contenedor. El feed de cambios le proporciona los valores originales y modificados de un artículo. Debido a que los cambios persisten, puede procesarlos de forma asincrónica. Puede utilizar esta función para activar notificaciones o llamar a las API cuando se inserta un nuevo elemento o se modifica un elemento existente.
- **UniqueKeyPolicy** Puede configurar qué propiedad del elemento se utiliza como clave única. Con claves únicas, se asegura de que no puede insertar dos elementos con el mismo valor para el mismo elemento. Tenga en cuenta que la unicidad tiene como ámbito la partición lógica. Por ejemplo, si su elemento tiene las propiedades correo electrónico, nombre, apellido y empresa, y define correo electrónico como clave única y empresa como clave de partición, no puede insertar un elemento con los mismos valores de correo electrónico y empresa. También puede crear claves únicas compuestas, como correo electrónico y nombre. Una vez que haya creado una clave única, no podrá cambiarla. Solo puede definir la clave única durante el proceso de creación del contenedor.

## *Observe las propiedades de los contenedores*

Las propiedades disponibles para los contenedores dependen de la API que configuró para su cuenta de Azure Cosmos DB. Para obtener una lista completa de las propiedades disponibles para cada API, consulte el artículo en <https://docs.microsoft.com/en-us/azure/cosmos-db/databases-containers-items#azure-cosmos-containers> .

Utilice el siguiente procedimiento para crear una nueva colección en su cuenta de Cosmos DB. Este procedimiento podría ser ligeramente diferente según la API que utilice para su cuenta de Cosmos DB. En este procedimiento, utiliza una cuenta de Cosmos DB configurada con la API de SQL:

1. Inicie sesión en Azure Portal (<http://portal.azure.com> ).
2. En el cuadro de búsqueda en la parte superior de Azure Portal, escriba el nombre de su cuenta de Cosmos DB y haga clic en el nombre de la cuenta.
3. En la hoja de su cuenta de Cosmos DB, haga clic en Explorador de datos.
4. En la hoja Explorador de datos, haga clic en el ícono Nuevo contenedor en la esquina superior izquierda de la hoja.
5. En el panel Agregar contenedor, que se muestra en la [Figura 2-3](#), proporcione un nombre para la nueva base de datos. Si desea agregar un contenedor a una base de datos existente, puede seleccionar la base de datos haciendo clic en el botón de opción Usar existente.
6. Asegúrese de que la verificación de rendimiento de la base de datos de aprovisionamiento esté seleccionada. Con esta opción, está configurando este contenedor como un contenedor de rendimiento compartido. Si desea crear un contenedor de rendimiento dedicado, desmarque esta opción.
7. Deje el valor de rendimiento establecido en 400. Este es el valor para el rendimiento de la base de datos si se marca la opción anterior. De lo contrario, este valor representa el rendimiento dedicado reservado para el contenedor.
8. En el cuadro de texto Id. Del contenedor, escriba un nombre para el contenedor.
9. En el cuadro de texto Clave de partición, escriba una clave de partición, comenzando con el carácter de barra.

10. Si desea crear una clave única para este contenedor, haga clic en el botón Agregar clave única.

11. Haga clic en el botón Aceptar en la parte inferior del panel.

The screenshot shows the 'Create container' dialog box. It includes fields for 'Database id' (set to 'Create new'), 'Container id' (set to 'e.g., Container1'), 'Indexing' (set to 'Automatic'), and 'Partition key' (set to '/address/zipCode'). There are also options for throughput provisioning ('Provision database throughput') and a note about estimated monthly costs (\$0.032/hourly, \$0.77/daily, \$23.04/monthly).

\* Database id ⓘ  
Create new Use existing  
Type a new database id

Provision database throughput ⓘ

\* Throughput (400 - 100.000 RU/s) ⓘ  
Autopilot (preview) Manual  
400

Estimated spend (USD): **\$0.032 hourly / \$0.77 daily / \$23.04 monthly** (1 region, 400RU/s, \$0.00008/RU)

\* Container id ⓘ  
e.g., Container1

\* Indexing  
Automatic Off  
All properties in your documents will be indexed by default for flexible and efficient queries. [Learn more](#)

\* Partition key ⓘ  
e.g., /address/zipCode  
 My partition key is larger than 100 bytes

Unique keys ⓘ  
+ Add unique key

**Figura 2-3** Creando una nueva colección

*¿Necesita más revisión? Feed de tiempo de vida, índices y cambios*

Puede revisar los detalles de cómo configurar el tiempo de vida, las políticas de índice y la fuente de cambios leyendo los siguientes artículos:

- **Configure Time to Live en Azure Cosmos**  
DB <https://docs.microsoft.com/en-us/azure/cosmos-db/how-to-time-to-live>
- **Restricciones clave únicas en Azure Cosmos**  
DB <https://docs.microsoft.com/en-us/azure/cosmos-db/unique-keys>

- Cambiar patrones de diseño de fuentes en Azure Cosmos DB <https://docs.microsoft.com/en-us/azure/cosmos-db/change-feed-design-patterns>



### *Sugerencia para el examen*

Debe planificar cuidadosamente cómo crear un nuevo contenedor en Azure Cosmos DB. Puede establecer algunas de las propiedades que puede configurar solo durante el proceso de creación. Una vez que haya creado el contenedor, si necesita modificar esas propiedades, debe crear un nuevo contenedor con los valores necesarios y migrar los datos al nuevo contenedor.

*Implemente la programación del lado del servidor, incluidos los procedimientos almacenados, los desencadenantes y las notificaciones de cambios de fuentes*

Cuando trabaja con la API de Cosmos DB, Azure le permite escribir sus desencadenadores, procedimientos almacenados y funciones definidas por el usuario. Puede escribir estos procedimientos y funciones utilizando JavaScript. Antes de poder llamar a un procedimiento almacenado, desencadenador o función definida por el usuario, debe registrarlos. Puede usar Azure Portal, la API de consulta integrada en lenguaje JavaScript en Cosmos DB o el SDK de cliente de la API SQL de Cosmos DB para crear y llamar a sus procedimientos almacenados, desencadenadores y funciones definidas por el usuario.

Cualquier procedimiento almacenado, desencadenador o función definida por el usuario que escriba se registra en un contenedor. Eso significa que debe registrar el procedimiento almacenado en cada contenedor donde desea ejecutar su procedimiento almacenado. También debe tener en cuenta que los procedimientos almacenados y los desencadenadores tienen como ámbito las particiones. Cualquier elemento con un valor de clave de partición diferente de la clave de partición del elemento que activó el desencadenador o el procedimiento almacenado no es visible.

Cuando escribe un procedimiento almacenado, un desencadenador o una función definida por el usuario, debe crear una referencia al contexto de ejecución. Este contexto le da acceso a las solicitudes que activaron el procedimiento almacenado o el activador y le permite trabajar con las respuestas y los elementos que desea insertar en la base de datos. En

términos generales, el contexto le da acceso a todas las operaciones que puede realizar en la base de datos de Azure Cosmos DB. El siguiente procedimiento muestra cómo crear un procedimiento almacenado en una cuenta de la API de SQL de Azure Cosmos DB:

1. Abra Azure Portal (<https://portal.azure.com>).
2. En el cuadro de texto Buscar en el área superior del portal, escriba el nombre de su cuenta de Cosmos DB. Recuerde que debe ser una cuenta de SQL API Cosmos DB.
3. En su cuenta de la API de SQL de Cosmos DB, haga clic en Explorador de datos.
4. Haga clic en una base de datos existente. Si no tiene ninguna base de datos, cree una nueva para realizar pruebas.
5. Haga clic en un contenedor existente o puede crear un contenedor de prueba siguiendo el procedimiento que revisamos en una sección anterior.
6. Haga clic en el botón Nuevo procedimiento almacenado. Este botón crea un nuevo procedimiento almacenado de muestra que puede utilizar como plantilla para sus procedimientos almacenados.
7. En el cuadro de texto Id. De procedimiento almacenado, proporcione un nombre para el procedimiento almacenado.
8. Reemplace el contenido de la pestaña Nuevo procedimiento almacenado con el contenido del [Listado 2-6](#).

**Listado 2-6** Procedimiento almacenado de la API SQL de Cosmos DB

[Haga clic aquí para ver la imagen del código](#)

---

```
// JavaScript

function createNewDocument (docToCreate) {

    var context = getContext ();
    var contenedor = context.getCollection ();
    var respuesta = context.getResponse ();
```

```

        console.log (docToCreate);

        var aceptado = container.createDocument
        (container.getSelfLink (),

            docToCreate,
           
            function (err, docCreated) {
               
                if (err) throw new Error ('Error al crear un
                nuevo documento:' + mensaje de error);
               
                response.setBody (docCreated);
            } );
        }

        si (! aceptado) volver;
    }

```

9. Haga clic en el botón Guardar.
10. Haga clic en el botón Ejecutar. Este botón abre la hoja Parámetros de entrada.
11. En la hoja Input Parameters, en la sección Partition Key Value, cambie el tipo de Custom a String.
12. En la sección Valor de la clave de partición, en el cuadro de texto Valor, escriba un valor para la clave de partición. Recuerde que esta clave de partición es la que ha definido para el contenedor donde está creando este procedimiento almacenado.
13. En el menú desplegable Tipo en la sección Ingresar parámetros de entrada, asegúrese de que el valor Cadena esté seleccionado.
14. En el cuadro de texto Param, escriba el documento en formato JSON que desea insertar. En aras de la simplicidad, utilice una cadena con una estructura similar a {"id": "12345", "clave": "valor"}.
15. Haga clic en el botón Ejecutar en la parte inferior del panel Parámetros de entrada.

16. En el árbol de navegación del Explorador de datos, haga clic en la hoja Elementos debajo del contenedor donde está creando el procedimiento almacenado.

17. Asegúrese de que el nuevo documento se haya insertado correctamente en su contenedor.

#### **Nota Error de solicitud incorrecta**

Si obtiene un error BadRequest cuando ejecuta el ejemplo anterior, revise los valores de los parámetros de entrada. Recuerde que no puede insertar un documento en una partición diferente a la que seleccionó en el Valor de la clave de partición. Por ejemplo, si su clave de partición es el campo "ciudad" y el valor que proporciona es "Sevilla", debe incluir este valor en la sección Ingresar parámetros de entrada. Para este ejemplo, su documento debe tener un aspecto similar a {"país": "España", "ciudad": "Sevilla"}.

Aunque el ejemplo anterior es bastante sencillo, hay algunos puntos interesantes que conviene repasar. Uno de los puntos esenciales que debe tener en cuenta al programar sus procedimientos almacenados, funciones definidas por el usuario o disparador, es el hecho de que los parámetros de entrada siempre tienen el tipo de cadena. Esto significa que si necesita pasar un objeto al procedimiento almacenado, debe convertir el objeto en una cadena y luego volver a convertirlo en un objeto JSON mediante el método JSON.parse () .

Como puede ver, usamos el método global getContext () para obtener una referencia al contexto. Ese contexto nos da acceso a las funciones de la cuenta de Cosmos DB. Luego obtuvimos una referencia al contenedor actual usando el método getContainer () en el contexto. También usamos el método getResponse () del contexto para enviar información al cliente.

Debido a que vamos a crear un nuevo documento en el contenedor, necesitamos usar el método createDocument () en el contenedor. Este método requiere un enlace al contenedor donde vamos a insertar el documento y el documento en sí. Debido a que los métodos requieren un documento JSON, si el valor del parámetro de entrada no es una cadena JSON válida, obtendrá un error de análisis JSON aquí. También proporcionamos una función anónima opcional para gestionar cualquier error que pueda surgir durante la creación del documento. Si no proporciona una función de devolución de llamada, cualquier error se lanza como una excepción.

La creación de un disparador es bastante similar a la creación de un procedimiento almacenado. Los conceptos son equivalentes, pero debe considerar cuándo debe ejecutar la acción de su disparador. Si necesita

realizar una operación antes de que el elemento se inserte en el contenedor, debe utilizar un **pre-activador**. Si necesita realizar una acción después de que el elemento se haya insertado correctamente en el contenedor, debe utilizar un **activador posterior**.

Los pre-activadores no pueden tener parámetros de entrada. Debido a que el elemento no está realmente en la base de datos, debe trabajar con la solicitud que activó el disparador. Esta solicitud contiene la información necesaria para insertar el nuevo artículo en la colección. Puede obtener una referencia a la solicitud utilizando el método `getRequest()` del objeto de contexto. Una vez que haya realizado sus modificaciones en el elemento original, puede enviar el elemento modificado a la base de datos mediante el método `request.setBody()`.

#### *¿Necesita más revisión? Más muestras*

Aunque el ejemplo que revisamos en esta sección puede parecer simplista, cubre algunos puntos importantes que debe tener en cuenta al programar sus elementos del lado del servidor. Los siguientes artículos proporcionan ejemplos más detallados de cómo crear y registrar procedimientos almacenados, funciones definidas por el usuario o desencadenadores mediante JavaScript o C #:

- **Cómo escribir procedimientos almacenados, desencadenadores y funciones definidas por el usuario en Azure Cosmos DB** <https://docs.microsoft.com/en-us/azure/cosmos-db/how-to-write-stored-procedures-triggers-udfs>
- **Cómo escribir procedimientos almacenados y desencadenadores en Azure Cosmos DB mediante la API de consulta de JavaScript** <https://docs.microsoft.com/en-us/azure/cosmos-db/how-to-write-javascript-query-api>
- **Cómo registrar y usar procedimientos almacenados, desencadenadores y funciones definidas por el usuario en Azure Cosmos DB** <https://docs.microsoft.com/en-us/azure/cosmos-db/how-to-use-stored-procedures-triggers-udfs>

#### *¿Necesita más revisión? Cambiar feeds*

Cada cambio, inserción o eliminación que se realiza en un elemento de una colección se registra y almacena automáticamente en la fuente de cambios. Puede usar estas operaciones como desencadenantes de Azure Functions que le permiten enviar notificaciones a otros servicios o realizar cualquier otra acción que considere. Puede revisar los detalles de cómo integrar Azure Functions con la fuente de cambios de Cosmos DB revisando los siguientes artículos:

- **Cómo configurar la directiva de conexión que usa el desencadenador de Azure Functions para Cosmos DB** <https://docs.microsoft.com/en-us/azure/cosmos-db/how-to-configure-cosmos-db-trigger-connection-policy>

- Cambiar fuente en Azure Cosmos DB <https://docs.microsoft.com/en-us/azure/cosmos-db/change-feed>

## HABILIDAD 2.2: DESARROLLAR SOLUCIONES QUE USEN BLOB STORAGE

---

Almacenar información en bases de datos SQL o NoSQL es una excelente manera de guardar esa información cuando necesita guardar documentos sin esquema o si necesita garantizar la integridad de los datos. El inconveniente de estos servicios es que son relativamente caros para almacenar datos que no tienen tales requisitos.

Azure Blob Storage le permite almacenar información que no se ajusta a las características del almacenamiento SQL y NoSQL en la nube. Esta información puede ser imágenes, videos, documentos de oficina o más. Azure Blob Storage aún ofrece características de alta disponibilidad que lo convierten en un servicio ideal para almacenar una gran cantidad de datos, pero a un precio más bajo en comparación con las otras soluciones de almacenamiento de datos que revisamos anteriormente en este capítulo.

### Esta habilidad cubre cómo

- Mover elementos en Blob Storage entre cuentas de almacenamiento o contenedores
- Establecer y recuperar propiedades y metadatos
- Interactuar con los datos usando el SDK apropiado
- Implementar el archivo y la retención de datos
- Implementar almacenamiento en caliente, frío y de archivo

### *Mover elementos en Blob Storage entre cuentas de almacenamiento o contenedores*

Cuando trabaja con Azure Blob Storage, puede haber situaciones en las que necesite mover blobs de una cuenta de almacenamiento a otra o entre contenedores. Para situaciones particulares, existen varias herramientas que puede utilizar para realizar estas tareas:

- **Azure Storage Explorer** es una herramienta gráfica que le permite administrar las diferentes operaciones con los servicios de

almacenamiento como Azure Storage, Azure Data Lake Storage, Azure Cosmos DB y discos virtuales.

- **AzCopy** es una herramienta de línea de comandos para realizar operaciones de copia masiva entre diferentes orígenes y cuentas de Azure Storage.
- **Python** Con el paquete azure-storage-blob, puede administrar su cuenta de Azure Storage con Python.
- **SSIS** El paquete de características del servicio de integración de SQL Server para Azure le permite transferir datos entre sus orígenes de datos locales y su cuenta de almacenamiento de Azure.

Una de las cosas que todas estas opciones tienen en común es que no ofrecen la operación de movimiento como una opción. Si necesita mover blobs o contenedores entre diferentes ubicaciones, debe realizar una operación de copia y luego eliminar el blob o contenedor de origen una vez que la operación de copia finalice correctamente.

El siguiente ejemplo muestra cómo mover un blob llamado testing.zip entre dos contenedores diferentes en diferentes cuentas de Azure Storage mediante el Explorador de Azure Storage. Para este ejemplo, debe crear dos cuentas de almacenamiento de Azure con dos contenedores diferentes. Luego, cargue un blob en una de las cuentas de almacenamiento.

1. Abra el Explorador de Azure Storage. Puede descargarlo desde <https://azure.microsoft.com/en-us/features/storage-explorer/>.
2. En el Explorador de Azure Storage, en el lado izquierdo de la ventana, haga clic en el botón Administrar cuentas.
3. En la sección Administración de cuentas, haga clic en el enlace Agregar una cuenta.
4. En la ventana Conectar a Azure Storage, que se muestra en la Figura 2-4, asegúrese de que la opción Agregar una cuenta de Azure esté seleccionada y, en el menú desplegable de Azure Environment, la opción de Azure esté seleccionada.
5. Haga clic en Siguiente.
6. Inicie sesión en su suscripción de Azure.

7. Una vez que haya iniciado sesión en su suscripción de Azure, su cuenta de Azure debería aparecer en la Administración de cuentas.
8. Haga clic en Aplicar. Esto cambiaría automáticamente a la sección Explorador.
9. En la sección Explorador, en el árbol de navegación del lado izquierdo de la ventana del Explorador de Azure Storage, navegue hasta su cuenta de almacenamiento de origen.
10. Expanda la hoja que representa su cuenta de almacenamiento de origen.



**Figura 2-4** Creación de una nueva colección

11. Expanda la hoja Blob Containers debajo de la hoja de la cuenta de almacenamiento de origen.
12. Haga clic en la hoja que representa el contenedor con el blob que desea mover. Esta acción abre una nueva pestaña en la ventana del Explorador de Azure Storage con el nombre de su contenedor.
13. Haga clic en el blob que desea mover.
14. Haga clic en el botón Copiar en la barra de menú en la parte superior de la pestaña del contenedor.
15. Navegue hasta el contenedor de destino y haga clic en la hoja que representa el contenedor de destino.
16. En la pestaña del contenedor de destino, haga clic en el botón Pegar en la barra de menú.

17. Asegúrese de que en la pestaña Actividades en la parte inferior de la ventana del Explorador de Azure Storage aparezca un mensaje similar al que se muestra en la [Figura 2-5](#).



**Figura 2-5** Creación de una nueva colección

18. Haga clic en la pestaña del contenedor de origen.
19. Seleccione el blob que desea mover.
20. Haga clic en el botón Eliminar en la barra de menú en la pestaña del contenedor.
21. En el cuadro de diálogo Explorador de almacenamiento de Microsoft Azure: Eliminar, haga clic en Eliminar.

El procedimiento para realizar la misma acción de movimiento con otras herramientas como Python o AzCopy es similar al que se muestra en el ejemplo anterior. Primero debe copiar el blob en el contenedor de destino y, una vez que la copia finalice correctamente, puede quitar el blob original.

La herramienta AzCopy es ideal para realizar escenarios de copia incremental o copiar una cuenta completa en otra cuenta. Puede usar el siguiente comando para copiar elementos de blob entre contenedores en diferentes cuentas de almacenamiento:

[Haga clic aquí para ver la imagen del código](#)

```
copia de azcopy <URL_Source_Item> <Source_SASToken>
<URL_Target_Container> <Target_SASToken>
```

#### *¿Necesita más revisión? Mover datos con Python y Ssis*

Puede revisar los detalles de cómo mover blob o contenedores con Python o SSIS consultando los siguientes artículos:

- **Inicio rápido: administre blobs con Python v12**  
**SDK** <https://docs.microsoft.com/en-us/azure/storage/blobs/storage-quickstart-blobs-python>
- **Mueva datos hacia o desde Azure Blob Storage mediante conectores SSIS** <https://docs.microsoft.com/en-us/azure/machine-learning/team-data-science-process/move-data-to-azure-blob-using -chassis>



### *Sugerencia para el examen*

Cuando realiza operaciones de copiar o mover en contenedores o blobs, no está limitado a la misma cuenta de almacenamiento. Puede copiar blobs y contenedores entre cuentas de almacenamiento en diferentes regiones o incluso suscripciones, siempre que tenga suficientes privilegios para acceder a ambas cuentas.

### *Establecer y recuperar propiedades y metadatos*

Cuando trabaja con los servicios de Azure Storage, puede trabajar con información adicional asignada a sus blobs. Esta información adicional se almacena en forma de propiedades del sistema y metadatos definidos por el usuario:

- **Propiedades del sistema** Esta es la información que los servicios de almacenamiento agregan automáticamente a cada recurso de almacenamiento. Puede modificar algunas de estas propiedades del sistema, mientras que otras son de solo lectura. Algunas de estas propiedades del sistema se corresponden con algunos encabezados HTTP. No necesita preocuparse por mantener estas propiedades del sistema porque las bibliotecas cliente de Azure Storage realizan automáticamente las modificaciones necesarias.
- **Metadatos definidos por el usuario** Puede asignar pares clave-valor a un recurso de Azure Storage. Estos metadatos son para sus propios fines y no afectan el comportamiento del servicio Azure Storage. Debe ocuparse de actualizar el valor de estos metadatos de acuerdo con sus necesidades.

Cuando trabaje con metadatos de blob, puede usar el SDK apropiado de su idioma preferido, o puede usar el comando az storage blob metadata de la CLI de Azure. El siguiente ejemplo muestra cómo trabajar con propiedades y metadatos utilizando .NET SDK:

1. Abra Visual Studio Code y cree una carpeta para su proyecto.
2. En la ventana de código de Visual Studio, abra una nueva terminal.

3. Utilice el siguiente comando para crear un nuevo proyecto de consola:

```
nueva consola dotnet
```

4. Utilice el siguiente comando para instalar paquetes NuGet:

Haga clic aquí para ver la imagen del código

```
dotnet agregar paquete <NuGet_package_name>
```

5. Instale los siguientes paquetes de NuGet:

1. Microsoft.Azure.Storage.Blob
2. Microsoft.Azure.Storage.Common
3. Microsoft.Extensions.Configuration
4. Microsoft.Extensions.Configuration.Binder
5. Microsoft.Extensions.Configuration.Json
6. En la carpeta del proyecto, cree un nuevo archivo JSON y **asígnele el nombre AppSettings.json**. Copie el contenido del Listado 2-7 al archivo JSON y reemplace el valor de las variables con los valores de sus cuentas de almacenamiento.
7. Cree un archivo de clase C # y **asígnele el nombre AppSettings.cs**.
8. Reemplace el contenido del archivo AppSettings.cs con el contenido del Listado 2-8. Cambie el nombre del espacio de nombres para que coincida con el nombre de su proyecto.
9. Cree un archivo de clase C # y **asígnele el nombre Common.cs**.
10. Reemplace el contenido del archivo Common.cs con el contenido del Listado 2-9.
11. Cambie el nombre del espacio de nombres para que coincida con el nombre de su proyecto.
12. Reemplace el contenido del archivo Program.cs con el contenido del Listado 2-10. Cambie el nombre del espacio de nombres para que coincida con el nombre de su proyecto.
13. Edite su archivo de proyecto .csproj y agregue el siguiente código dentro de la sección ItemGroup:

[Haga clic aquí para ver la imagen del código](#)

```
<Ninguna Actualización = "AppSettings.json">

    <CopyToOutputDirectory> PreserveNewest </CopyToOutputDirectory>

</Ninguno>
```

14. En este punto, puede establecer algunos puntos de interrupción en el archivo Program.cs para ver, paso a paso, cómo el código mueve los elementos de blob entre los diferentes contenedores y cuentas de almacenamiento.

15. En la ventana de Visual Studio, presione F5 para compilar y ejecutar su código.

**Listado 2-7 Archivo de configuración AppSettings.json**

[Haga clic aquí para ver la imagen del código](#)

---

```
{
    "SASToken": 
"<SASToken_from_your_first_storage_account>",

    "AccountName": 
"<nombre_de_su_primera_cuenta_de_almacenamiento>",

    "ContainerName": "<source_container_name>"

}
```

**Listado 2-8 AppSettings.cs Clase C #**

[Haga clic aquí para ver la imagen del código](#)

---

```
// C # .NET Core
```

```
utilizando Microsoft.Extensions.Configuration;
```

```
espacio de nombres ch2_2_2
```

```

{

    AppSettings de clase pública

    {

        cadena pública SASToken {get; colocar; }

        Cadena pública AccountName {get; colocar; }

        cadena pública ContainerName {get; colocar; }

        AppSettings estáticos públicos LoadAppSettings ()

        {

            IConfigurationRoot configRoot = new
ConfigurationBuilder ()

                .AddJsonFile ("AppSettings.json", false)

                .Construir();

            AppSettings appSettings = configRoot.Get
<AppSettings> ();

            return appSettings;

        }

    }

}

```

**Listado 2-9** Common.cs C # class

[Haga clic aquí para ver la imagen del código](#)

---

```
// C # .NET Core
```

```
usando el sistema;
```

```
utilizando Microsoft.Azure.Storage;
utilizando Microsoft.Azure.Storage.Auth;
utilizando Microsoft.Azure.Storage.Blob;

espacio de nombres ch2_2_2

{

    clase pública común

    {

        CloudBlobClient estático público
CreateBlobClientStorageFromSAS (cadena SASToken,
string accountName)

    {

        CloudStorageAccount storageAccount;
        CloudBlobClient blobClient;
        intentar

        {

            bool useHttps = true;
            StorageCredentials storageCredentials =
nuevas StorageCredentials (SASToken);
            storageAccount = nueva CloudStorageAccount
(storageCredentials,
accountName, null, useHttps);

            blobClient =
storageAccount.CreateCloudBlobClient ();
        }
    }
}
```

```

        }

        catch (System.Exception)

        {

            lanzar;

        }

        return blobClient;

    }

}

```

El Listado 2-10 muestra cómo crear un nuevo contenedor y obtener una lista de algunas propiedades del sistema asignadas automáticamente al contenedor cuando lo crea.

**Listado 2-10** Program.cs Clase C #

Haga clic aquí para ver la imagen del código

---

```

// C # .NET Core

// Obtener propiedades del sistema de un recurso de
almacenamiento

usando el sistema;

usando System.Threading.Tasks;

utilizando Microsoft.Azure.Storage.Blob;

espacio de nombres ch2_2_2

{

```

```
programa de clase

{

    static void Main (cadena [] argumentos)

    {

        Console.WriteLine ("¡Obteniendo la demostración
de propiedades del sistema!");



        AppSettings appSettings =
AppSettings.LoadAppSettings ();



        // Cree un CloudBlobClient para trabajar con la
cuenta de almacenamiento

        CloudBlobClient blobClient =
Common.CreateBlobClientStorageFromSAS

            (appSettings.SASToken,
appSettings.AccountName);





        // Obtenga una referencia de contenedor para el
nuevo contenedor.

        Contenedor CloudBlobContainer =
blobClient.GetContainerReference

            (appSettings.ContainerName);





        // Crea el contenedor si aún no existe

        container.CreateIfNotExists ();
    }
}
```

```

        // Necesitas recuperar las propiedades del
        contenedor antes de obtener sus valores

        container.FetchAttributes();

        Console.WriteLine($"Propiedades del contenedor
{container.StorageUri.

PrimaryUri.ToString()})";

        System.Console.WriteLine($"ETag:
{contenedor.Properties.ETag}");

        System.Console.WriteLine($"LastModifiedUTC:
{contenedor.Properties.

LastModified.ToString()})";

        System.Console.WriteLine($"Estado del
arrendamiento: {contenedor.Properties.LeaseStatus.

Encadenar()})";

        System.Console.WriteLine();

    }

}

}

```

Como puede ver en el código anterior en el [Listado 2-10](#), debe usar el método FetchAttributes () o FetchAttributesAsync () antes de poder leer las propiedades del contenedor, almacenadas en la propiedad Propiedades de los objetos CloudBlobContainer o CloudBlockBlob. Si obtiene valores nulos para las propiedades del sistema, asegúrese de haber llamado al método FetchAttributes () antes de acceder a la propiedad del sistema.

Trabajar con metadatos definidos por el usuario es bastante similar a trabajar con propiedades del sistema. La principal diferencia es que puede agregar sus pares de claves personalizados al recurso de almacenamiento. Estos metadatos definidos por el usuario se almacenan en la propiedad Metadata del recurso de almacenamiento. [El Listado 2-](#)

11 amplía el ejemplo del Listado 2-10 y muestra cómo configurar y leer metadatos definidos por el usuario en el contenedor que creó en el Listado 2-10. Copie el contenido del Listado 2-11 e inserte el código en el archivo Program.cs después del último System.Console.WriteLine () .

**Listado 2-11** Configuración de metadatos definidos por el usuario

Haga clic aquí para ver la imagen del código

---

```
// C # .NET Core

// Agrega algunos metadatos al contenedor que creamos antes

    container.Metadata.Add ("departamento",
" Técnico");

    container.Metadata [ "category" ] = "Base de
conocimientos";

    container.Metadata.Add ( "docType",
"pdfDocuments");



// Guarde los metadatos de los contenedores en
Azure

    container.SetMetadata ();


// Lista de metadatos recién agregados.
Necesitamos buscar todos los atributos antes de ser

// capaz de leer si no, podríamos obtener
valores nulos o extraños

    container.FetchAttributes (


System.Console.WriteLine ( "Metadatos del
contenedor:" );
```

```

foreach (elemento var en contenedor.Metadatos)
{
    System.Console.WriteLine ($ "\tKey: {item.Key}
\t");
    System.Console.WriteLine ($ "\tValue:
{item.Value} ");
}

```

Puede encontrar una lista completa de propiedades del sistema en la referencia del cliente Microsoft.Azure.Storage.Blob .NET en <https://docs.microsoft.com/en-us/dotnet/api/microsoft.azure.storage.blob.blobcontainerproperties>. Las clases BlobContainerProperties y BlobProperties son responsables de almacenar las propiedades del sistema para los recursos de almacenamiento en una cuenta de Blob Storage.

También puede ver y editar las propiedades del sistema y los metadatos definidos por el usuario mediante Azure Portal, las secciones Propiedades y Metadatos en la sección Configuración de su contenedor, o haciendo clic en los puntos suspensivos junto al elemento del blob y seleccionando la opción Propiedades del blob en el menú contextual.

### *Interactuar con los datos usando el SDK apropiado*

Aunque el uso de cualquiera de las opciones que revisamos al principio de la habilidad puede ser apropiado para algunas situaciones, es posible que deba obtener un control más detallado de los elementos que necesita para mover entre contenedores o incluso Cuentas de almacenamiento.

Microsoft proporciona varios SDK para trabajar con datos en sus cuentas de almacenamiento. Puede encontrar SDK para los principales lenguajes de programación compatibles con Microsoft, como .NET, Java, Python, JavaScript (Node.js o navegador), Go, PHP o Ruby.

El siguiente ejemplo escrito en .NET Core muestra cómo mover un elemento de blob entre dos contenedores en la misma cuenta de almacenamiento y cómo mover un elemento de blob entre dos contenedores en diferentes cuentas de almacenamiento. Antes de poder ejecutar este ejemplo, debe crear dos cuentas de almacenamiento con dos

contenedores de blobs. En aras de la simplicidad, debe crear los dos contenedores con el mismo nombre en las dos cuentas de almacenamiento diferentes. Además, debe cargar dos archivos de control como elementos de blob en uno de los contenedores en una cuenta de almacenamiento:

1. Abra Visual Studio Code y cree una carpeta para su proyecto.
2. En la ventana de código de Visual Studio, abra una nueva terminal.
3. Utilice el siguiente comando para crear un nuevo proyecto de consola:

```
nueva consola dotnet
```

4. Utilice el siguiente comando para instalar paquetes NuGet:

[Haga clic aquí para ver la imagen del código](#)

```
dotnet agregar paquete <NuGet_package_name>
```

5. Instale los siguientes paquetes de NuGet:

1. Azure.Storage.Blobs
2. Azure.Almacenamiento.Común
3. Microsoft.Extensions.Configuration
4. Microsoft.Extensions.Configuration.Binder
5. Microsoft.Extensions.Configuration.Json
6. En la carpeta del proyecto, cree un nuevo archivo JSON y **asígnele el nombre AppSettings.json**. Copie el contenido del Listado 2-12 al archivo JSON.
7. Cree un archivo de clase C # y **asígnele el nombre AppSettings.cs**.
8. Reemplace el contenido del archivo AppSettings.cs con el contenido del Listado 2-13. Cambie el nombre del espacio de nombres para que coincida con el nombre de su proyecto.
9. Cree un archivo de clase C # y **asígnele el nombre Common.cs**.
10. Reemplace el contenido del archivo Common.cs con el contenido del Listado 2-14.

11. Cambie el nombre del espacio de nombres para que coincida con el nombre de su proyecto.

12. Reemplace el contenido del archivo Program.cs con el contenido del [Listado 2-15](#). Cambie el nombre del espacio de nombres para que coincida con el nombre de su proyecto.

13. Edite su archivo de proyecto .csproj y agregue el siguiente código dentro de la sección ItemGroup:

[Haga clic aquí para ver la imagen del código](#)

```
<Ninguna Actualización = "AppSettings.json">

<CopyToOutputDirectory> PreserveNewest </CopyToOutputDirectory>

</Ninguno>
```

14. En este punto, puede establecer algunos puntos de interrupción en el archivo Program.cs para ver, paso a paso, cómo el código mueve los elementos de blob entre los diferentes contenedores y cuentas de almacenamiento.

15. En la ventana de Visual Studio, presione F5 para compilar y ejecutar su código. Puede usar Azure Portal o la aplicación de escritorio Microsoft Azure Storage Explorer para revisar cómo cambian sus ubicaciones los elementos de blob.

**Listado 2-12** Archivo de configuración AppSettings.json

[Haga clic aquí para ver la imagen del código](#)

---

```
{

    "SourceSASConnectionString": 
"<SASConnectionString_from_your_first_storage_"

cuenta> ",

    "SourceAccountName": 
"<nombre_de_su_primera_cuenta_de_almacenamiento>",

    "SourceContainerName": "<source_container_name>",

    "DestinationSASConnectionString": 
"<SASConnectionString_from_your_second_storage_"
```

```
        cuenta> ",  
        "DestinationAccountName":  
    "<nombre_de_su_segunda_cuenta_de_almacenamiento>",  
        "DestinationContainerName":  
    "<destination_container_name>"  
}
```

**Listado 2-13 AppSettings.cs Clase C #**

[Haga clic aquí para ver la imagen del código](#)

---

```
// C # .NET Core  
  
utilizando Microsoft.Extensions.Configuration;  
  
  
espacio de nombres ch2_2_3  
  
{  
  
    AppSettings de clase pública  
  
    {  
  
        cadena pública SourceSASConnectionString {get;  
colocar; }  
  
        cadena pública SourceAccountName {get; colocar; }  
  
        cadena pública SourceContainerName {get; colocar; }  
  
        Cadena pública DestinationsSASConnectionString {get;  
colocar; }  
  
        cadena pública DestinationAccountName {get; colocar;  
    }  
  
        cadena pública DestinationContainerName {get;  
colocar; }
```

```

        AppSettings estáticos públicos LoadAppSettings ()

    {

        IConfigurationRoot configRoot = new
ConfigurationBuilder ()

        .AddJsonFile ("AppSettings.json", false)

        .Construir();

        AppSettings appSettings = configRoot.Get
<AppSettings> ();

        return appSettings;

    }

}

```

**Listado 2-14 Clase Common.cs C #**

[Haga clic aquí para ver la imagen del código](#)

---

```

// C # .NET Core

utilizando Azure.Storage.Blobs;

espacio de nombres ch2_2_3

{

    clase pública común

    {

        public static BlobServiceClient
CreateBlobClientStorageFromSAS (string

SASConnectionString)

```

```

    {

        BlobServiceClient blobClient;

        intentar

        {

            blobClient = nuevo BlobServiceClient
(SASConnectionString);

        }

        catch (System.Exception)

        {

            lanzar;

        }

        return blobClient;

    }

}

```

En el Listado 2-15 , las partes del código que son importantes para el proceso de trabajo con el servicio Azure Blob Storage se muestran en negrita.

#### **Listado 2-15** Program.cs Clase C #

[Haga clic aquí para ver la imagen del código](#)

---

```
// C # .NET Core

usando System.Threading.Tasks;

usando el sistema;
```

```
utilizando Azure.Storage.Blobs;  
utilizando Azure.Storage.Blobs.Models;  
  
espacio de nombres ch2_2_3  
{  
    programa de clase  
    {  
        static void Main (cadena [] argumentos)  
        {  
            Console.WriteLine ("¡Copiar elementos entre  
contenedores de demostración!");  
  
            Task.Run (async () => aguardar  
StartContainersDemo ()). Wait ();  
  
            Console.WriteLine ("¡Mover elementos entre  
cuentas de almacenamiento, demostración!");  
  
            Task.Run (async () => await StartAccountDemo  
(())). Wait ();  
        }  
  
        Tarea asíncrona estática pública StartContainersDemo  
()  
        {  
            string sourceBlobFileName = "Testing.zip";  
  
            AppSettings appSettings =  
AppSettings.LoadAppSettings ();
```

```
// Obtenga un cliente en la nube para la cuenta  
de almacenamiento de origen  
  
    BlobServiceClient sourceClient =  
Common.CreateBlobClientStorageFromSAS  
  
(appSettings.SourceSASConnectionString);  
  
  
// Obtenga una referencia para cada contenedor  
  
    var sourceContainerReference =  
sourceClient.GetBlobContainerClient  
  
(appSettings.SourceContainerName);  
  
    var destinationContainerReference =  
sourceClient.GetBlobContainer  
  
Client (appSettings.DestinationContainerName);  
  
  
// Obtener una referencia para el blob de origen  
  
    var sourceBlobReference =  
sourceContainerReference.GetBlobClient  
  
(sourceBlobFileName);  
  
    var destinationBlobReference =  
destinationContainerReference.  
  
GetBlobClient (sourceBlobFileName);  
  
  
// Copie el blob del contenedor de origen al  
contenedor de destino  
  
    aguardar  
destinationBlobReference.StartCopyFromUriAsync (sourceBlob  
Reference.Uri);
```

```
    }

    Tarea asincrona estatica publica StartAccountDemo ()

    {

        string sourceBlobFileName = "Testing.zip";

        AppSettings appSettings =
AppSettings.LoadAppSettings ();



        // Obtenga un cliente en la nube para la cuenta
        de almacenamiento de origen

        BlobServiceClient sourceClient =
Common.CreateBlobClientStorageFromSAS

(appSettings.SourceSASConnectionString);

        // Obtenga un cliente en la nube para la cuenta
        de almacenamiento de destino

        BlobServiceClient destinationClient =
Common.CreateBlobClientStorage

FromSAS (appSettings.DestinationSASConnectionString);





        // Obtenga una referencia para cada contenedor

        var sourceContainerReference =
sourceClient.GetBlobContainerClient

(appSettings.SourceContainerName);

        var destinationContainerReference =
destinationClient.GetBlobContainer

Client (appSettings.DestinationContainerName);
```

```

    // Obtener una referencia para el blob de origen

        var sourceBlobReference =
sourceContainerReference.GetBlobClient

(sourceBlobFileName);

        var destinationBlobReference =
destinationContainerReference.

GetBlobClient (sourceBlobFileName);

    // Mueve el blob del contenedor de origen al
contenedor de destino

        aguardar
destinationBlobReference.StartCopyFromUriAsync (sourceBlob

Reference.Uri);

        aguardar sourceBlobReference.DeleteAsync ();

    }

}

}

```

En este ejemplo, realizó dos operaciones diferentes: una copia entre contenedores en la misma cuenta de almacenamiento y un movimiento entre contenedores en diferentes cuentas de almacenamiento. Como puede ver en el código mostrado anteriormente en el [Listado 2-15](#), el procedimiento de alto nivel para mover elementos blob entre contenedores es

1. Cree una instancia de BlobServiceClient para cada cuenta de almacenamiento que esté involucrada en el movimiento de elementos de blob.
2. Crea una referencia para cada contenedor. Si necesita mover un elemento de blob entre contenedores en una cuenta de

almacenamiento diferente, debe usar el objeto `BlobServiceClient` que representa cada cuenta de almacenamiento.

3. Cree una referencia para cada elemento de blob. Necesita una referencia al elemento de blob de origen porque este es el elemento que va a mover. Utilice la referencia del elemento de blob de destino para realizar la operación de copia real.

4. Una vez que haya terminado con la copia, puede eliminar el elemento de blob de origen mediante el método `DeleteAsync()`.

Aunque este código es bastante sencillo, tiene un problema crítico que puede resolver en las siguientes secciones. Si alguien más modifica el elemento del blob de origen mientras la operación de escritura está pendiente, la operación de copia falla con un código de estado HTTP 412. Vamos a solucionar esto más adelante en esta sección.

Puede observar que el código de este ejemplo utiliza un SDK diferente del código de las secciones anteriores. La razón de esto es que, en el momento de escribir este artículo, existen dos versiones diferentes del SDK de .NET para trabajar con Azure Storage:

- **Microsoft.Azure.Storage.Blob** Esta es la versión 11 del SDK. Puede configurar permisos en los blobs usando esta versión.
- **Azure.Storage.Blobs** Esta es la versión 12 del SDK. Esta versión simplifica la forma de trabajar con Azure Storage Blobs, pero no ofrece el conjunto completo de características que tiene la versión 11. No puede establecer permisos con esta versión.

#### *¿Necesita más revisión? Copia de blob entre cuentas*

Puede revisar los detalles de cómo funciona la copia asincrónica entre cuentas de almacenamiento leyendo este artículo de MSDN, "Introducción al blob de copia asincrónica entre cuentas"

en <https://blogs.msdn.microsoft.com/windowsazurestorage/2012/06/12/introducing-asynchronous-cross-account-copy-blob/>.



#### *Sugerencia para el examen*

Cuando necesite mover un blob a cualquier destino, contenedor o cuenta de almacenamiento, recuerde que primero debe realizar una operación de copia y luego eliminar el blob de origen. No existe tal método de movimiento en la clase `CloudBlockBlob`.

Cuando trabaja con el servicio Blob Storage, en el que varios usuarios o procesos pueden acceder simultáneamente a la misma cuenta de almacenamiento, puede enfrentar un problema cuando dos usuarios o procesos intentan acceder al mismo blob. Azure proporciona un mecanismo de arrendamiento para resolver este tipo de situación. Una concesión es un bloque corto que el servicio de blob establece en un elemento de blob o contenedor para otorgar acceso exclusivo a ese elemento. Cuando adquiere una concesión a un blob, obtiene acceso exclusivo de escritura y eliminación a ese blob. Si adquiere un arrendamiento en un contenedor, obtiene acceso de eliminación exclusivo al contenedor.

Cuando adquiere una concesión para un elemento de almacenamiento, debe incluir el ID de concesión activa en cada operación de escritura que desee realizar en el blob con la concesión. Puedes elegir la duración por el tiempo de arrendamiento cuando lo solicite. Esta duración puede durar de 15 a 60 segundos o una eternidad. Cada arrendamiento puede estar en uno de los siguientes cinco estados:

- **Disponible** El contrato de arrendamiento está desbloqueado y puede adquirir un nuevo contrato de arrendamiento.
- **Arrendado** Se concede un arrendamiento al recurso y el arrendamiento está bloqueado. Puede adquirir un nuevo contrato de arrendamiento si usa la misma identificación que obtuvo cuando creó el contrato de arrendamiento. También puede liberar, cambiar, renovar o romper el contrato de arrendamiento cuando se encuentra en este estado.
- **Caducado** La duración configurada para el arrendamiento ha caducado. Cuando tiene un contrato de arrendamiento en este estado, puede adquirir, renovar, liberar o romper el contrato de arrendamiento.
- **Ruptura** Has incumplido el contrato de arrendamiento, pero aún está bloqueado hasta que expire el período de interrupción. En este estado, puede liberar o romper el contrato de arrendamiento.
- **Roto** El período de descanso ha expirado y el contrato de arrendamiento se ha roto. En este estado, puede adquirir, liberar y romper un contrato de arrendamiento. Debe romper un arrendamiento cuando el proceso que adquirió el arrendamiento finaliza repentinamente, como cuando problemas de conectividad

de red o cualquier otra condición hace que el arrendamiento no se libere correctamente. En estas situaciones, puede terminar con una concesión huérfana y no puede escribir ni eliminar el blob con la concesión huérfana. En esta situación, la única solución es romper el contrato de arrendamiento. También es posible que desee romper un contrato de arrendamiento cuando necesite forzar la liberación del contrato manualmente.

Puede usar Azure Portal para administrar el estado de concesión de un contenedor o elemento de blob, o lo usa mediante programación con el SDK de cliente de Azure Blob Storage. En el ejemplo que se muestra en los [listados 2-12 a 2-15](#), en el que revisamos cómo copiar y mover elementos entre contenedores o cuentas de almacenamiento, vimos que si algún otro proceso o usuario modifica el blob mientras nuestro proceso está copiando los datos, obtenemos un error. Puede evitar esa situación adquiriendo una concesión para el blob que desea mover. [El Listado 2-16](#) muestra en negrita la modificación que necesita agregar al código en el [Listado 2-15](#) para que pueda adquirir una concesión para el elemento blob.

#### **Listado 2-16** Modificación de Program.cs

[Haga clic aquí para ver la imagen del código](#)

---

```
// C # .NET Core

// Agregue líneas en negrita al método StartContainersDemo
en el Listado 2-15

// Agrega la siguiente instrucción using al principio del
archivo:

// usando Azure.Storage.Blobs.Specialized;

Tarea asíncrona estática pública StartContainersDemo ()

{

    string sourceBlobFileName = "Testing.zip";

    AppSettings appSettings =
AppSettings.LoadAppSettings ();
```

```
// Obtenga un cliente en la nube para la cuenta  
de almacenamiento de origen  
  
BlobServiceClient sourceClient =  
Common.CreateBlobClientStorageFromSAS  
(appSettings.SourceSASConnectionString);  
  
  
// Obtenga una referencia para cada contenedor  
  
var sourceContainerReference =  
sourceClient.GetBlobContainerClient  
(appSettings.SourceContainerName);  
  
var destinationContainerReference =  
sourceClient.GetBlobContainerClient  
(appSettings.DestinationContainerName);  
  
  
// Obtener una referencia para el blob de origen  
  
var sourceBlobReference =  
sourceContainerReference.GetBlobClient  
(sourceBlobFileName);  
  
var destinationBlobReference =  
destinationContainerReference.GetBlobClient (br/>  
sourceBlobFileName);  
  
  
// Obtenga el estado de concesión del blob de  
origen  
  
BlobProperties sourceBlobProperties = await  
sourceBlobReference.  
GetPropertiesAsync ();
```

```
System.Console.WriteLine ($ "Estado del
arrendamiento: {sourceBlobProperties.
LeaseStatus} "+
$" \ tstate:
{sourceBlobProperties.LeaseState} "+
$" \ tduration:
{sourceBlobProperties.LeaseDuration} ");

// Adquirir un contrato de arrendamiento
infinito. Si desea establecer una duración para el
arrendamiento

// use

// TimeSpan.FromSeconds(seconds). Recuerde que
los segundos deben ser un valor

// entre 15 y 60.

// Necesitamos guardar el ID de concesión
generado automáticamente por Azure para el lanzamiento

// la concesión más tarde.

string leaseID = Guid.NewGuid (). ToString ();

BlobLeaseClient sourceLease =
sourceBlobReference.

GetBlobLeaseClient (leaseID);

sourceLease.Acquire (nuevo TimeSpan (-1));

sourceBlobProperties = aguardar
sourceBlobReference.GetPropertiesAsync ();
```

```

        System.Console.WriteLine ($ "Estado del
arrendamiento: {sourceBlobProperties.

LeaseStatus} "+

                            $" \ tstate:
{sourceBlobProperties.LeaseState} "+

                            $" \ tduration:
{sourceBlobProperties.LeaseDuration} ") ;



// Copie el blob del contenedor de origen al
contenedor de destino

aguardar
destinoBlobReference.StartCopyFromUriAsync

(sourceBlobReference.Uri);





// Liberar el arrendamiento adquirido
previamente

sourceLease.Release () ;





}

```

Como puede ver en el ejemplo anterior, necesita obtener una referencia a un BlobLeaseClient. Este objeto le permite adquirir nuevas concesiones invocando el método Acquire (). En este ejemplo, creamos una concesión infinita porque usamos un TimeSpan con el valor -1. Una vez que hayamos copiado el blob, debemos liberar la concesión mediante el método Release () del BlobLeaseClient.

#### *¿Necesita más revisión? Arrendamiento de blobs y contenedores*

Puede revisar los detalles de cómo funciona el arrendamiento para blobs y contenedores consultando los siguientes artículos:

- **Arrendamiento Blob** <https://docs.microsoft.com/en-us/rest/api/storageservices/lease-blob>

- **Contenedor de arrendamiento** <https://docs.microsoft.com/en-us/rest/api/storageservices/lease-container>



### *Sugerencia para el examen*

El arrendamiento es el mecanismo que debe usar para asegurarse de que ningún otro usuario o proceso pueda acceder a un blob mientras trabaja con él. Puede crear arrendamientos cronometrados o infinitos. Recuerde que necesita liberar arrendamientos infinitos manualmente.

### *Implementar el archivo y la retención de datos*

Cuando trabaja con datos, los requisitos para acceder a los datos cambian durante la vida útil de los datos. Por lo general, se accede a los datos que se han colocado recientemente en su sistema de almacenamiento con más frecuencia y requieren un acceso más rápido que los datos más antiguos. Si está usando el mismo tipo de almacenamiento para todos sus datos, eso significa que está usando almacenamiento para datos a los que rara vez se accede. Si su almacenamiento se basa en un disco SSD o en cualquier otra tecnología que proporcione los niveles de rendimiento adecuados, esto significa que puede estar desperdiciando un almacenamiento costoso para los datos a los que rara vez se accede. Una solución a esta situación es trasladar los datos a los que se accede con menos frecuencia a un sistema de almacenamiento más económico. El inconveniente de esta solución es que necesita implementar un sistema para rastrear la última vez que se accedió a los datos y moverlos al sistema de almacenamiento correcto.

Azure Blob Storage le brinda la capacidad de establecer diferentes niveles de acceso a sus datos. Estos diferentes niveles de acceso, o niveles, proporcionan diferentes niveles de rendimiento al acceder a los datos. Cada nivel de acceso diferente tiene un precio diferente. Los siguientes son los niveles de acceso disponibles:

- **Caliente** Utilice este nivel para los datos a los que necesita acceder con más frecuencia. Este es el nivel predeterminado que usa cuando crea una nueva cuenta de almacenamiento.
- **Fresco** Puede utilizar este nivel para los datos a los que se accede con menos frecuencia y que se almacenan durante al menos 30 días.

- **Archivo** Utilice este nivel para almacenar datos a los que rara vez se accede y que se almacenan durante al menos 180 días. Este nivel de acceso solo está disponible en el nivel de blob. No puede configurar una cuenta de almacenamiento con este nivel de acceso.

Los diferentes niveles de acceso tienen las siguientes implicaciones de precio y rendimiento:

- Cool tier proporciona una disponibilidad ligeramente menor, reflejada en el acuerdo de nivel de servicio (SLA) debido a los menores costos de almacenamiento; sin embargo, tiene mayores costos de acceso.
- Los niveles calientes y fríos tienen características similares en términos de tiempo de acceso y rendimiento.
- El almacenamiento de archivos es un almacenamiento fuera de línea. Tiene las tasas de costo de almacenamiento más bajas pero tiene costos de acceso más altos.
- Cuanto menores sean los costos de almacenamiento, mayores serán los costos de acceso.
- Puede utilizar el almacenamiento en niveles solo en cuentas de almacenamiento de uso general v2 (GPv2).
- Si desea utilizar el almacenamiento en niveles con una cuenta de almacenamiento de uso general v1 (GPv1), debe convertir a una cuenta de almacenamiento GPv2.

Moverse entre los diferentes niveles de acceso es un proceso transparente para el usuario, pero tiene algunas implicaciones en términos de precios. En general, cuando se pasa de un nivel más cálido a un nivel más frío (de caliente a frío o de caliente a archivado), se le cobran las operaciones de escritura en el nivel de destino. Cuando pasa de un nivel más frío a uno más cálido, del archivo al frío o del frío al caliente, se le cobran las operaciones de lectura desde el nivel de origen. Otra cosa esencial a tener en cuenta es cómo se mueven los datos cuando cambia su nivel de datos de archivo a cualquier otro nivel de acceso. Debido a que los datos en el nivel de archivo se guardan en el almacenamiento fuera de línea, cuando se mueven los datos fuera del nivel de acceso, el servicio de almacenamiento necesita mover los datos nuevamente al almacenamiento en línea. Este proceso se conoce como rehidratación de gotas y puede tardar hasta 15 horas.

Si no configura manualmente el nivel de acceso para un blob, hereda el acceso de su contenedor o cuenta de almacenamiento. Aunque puede cambiar el nivel de acceso manualmente mediante Azure Portal, este proceso crea una sobrecarga administrativa que también podría provocar errores humanos. En lugar de supervisar manualmente los diferentes criterios para mover un blob de un nivel a otro, puede implementar políticas que realicen ese movimiento según los criterios que defina. Utilice estas políticas para definir la gestión del ciclo de vida de sus datos. Puede crear estas políticas de administración del ciclo de vida mediante Azure Portal, Azure PowerShell, la CLI de Azure o la API REST.

Una política de administración del ciclo de vida es un documento JSON en el que define varias reglas que desea aplicar a los diferentes contenedores o tipos de blobs. Cada regla consta de un conjunto de filtros y un conjunto de acciones.

- **Conjunto de filtros** El **conjunto de filtros** limita las acciones a solo un grupo de elementos que coinciden con los criterios de filtrado.
- **Conjunto de acciones** Utilice este conjunto para definir las acciones que se realizan en los elementos que coinciden con el filtro.

El siguiente procedimiento para agregar una nueva directiva mediante Azure Portal:

1. Inicie sesión en Azure Portal (<http://portal.azure.com>).
2. En el cuadro de búsqueda en la parte superior de Azure Portal, escriba el nombre de su cuenta de almacenamiento.
3. En la sección del servicio Blob, haga clic en Administración del ciclo de vida.
4. Copie el contenido del Listado 2-17 y péguelo en el panel Gestión del ciclo de vida.
5. Haga clic en el botón Guardar en la esquina superior izquierda del panel.

**Listado 2-17** Definición de la política de gestión del ciclo de vida

Haga clic aquí para ver la imagen del código

---

```
"reglas": [  
  {  
    "habilitado": verdadero,  
    "nombre": "regla1",  
    "tipo": "Ciclo de vida",  
    "definición": {  
      "acciones": {  
        "baseBlob": {  
          "tierToCool": {  
            "daysAfterModificationGreaterThan": 30  
          },  
          "tierToArchive": {  
            "daysAfterModificationGreaterThan": 90  
          },  
          "Eliminar": {  
            "daysAfterModificationGreaterThan": 2555  
          }  
        },  
        "instantánea": {  
          "Eliminar": {  
            "daysAfterCreationGreaterThan":  
90
```

```

        }

    }

    },
    "filtros": {
        "blobTypes": [
            "blockBlob"
        ],
        "prefixMatch": [
            "contenedor-a"
        ]
    }
}
]
}
}

```

La política anterior se aplica a todos los blobs en el contenedor denominado container-a, como se indica en prefixMatch en la sección de filtros. En las secciones de acciones, puede ver lo siguiente:

- Los blobs que no se modifican en 30 días o más se mueven al nivel cool.
- Los blobs que no se modifican en 90 días o más se mueven al nivel de archivo.
- Los blobs que no se modifican en 2.555 días o más se eliminan de la cuenta de almacenamiento.

También se eliminan las instantáneas que tienen más de 90 días. El motor de gestión del ciclo de vida procesa las políticas cada 24 horas. Esto significa que es posible que no vea los cambios reflejados en su cuenta de

almacenamiento hasta varias horas después de haber realizado los cambios.

**¿Necesita más revisión? Niveles de acceso al almacenamiento y políticas de gestión del ciclo de vida**

Puede ampliar sus conocimientos sobre los niveles de acceso al almacenamiento y la gestión del ciclo de vida revisando los siguientes artículos de Microsoft Docs:

- **Azure Blob Storage: niveles de acceso activo, frío y de archivo** <https://docs.microsoft.com/en-us/azure/storage/blobs/storage-blob-storage-tiers>
- **Administrar el ciclo de vida de Azure Blob Storage** <https://docs.microsoft.com/en-us/azure/storage/blobs/storage-lifecycle-management-concepts>

*Implementar almacenamiento en caliente, frío y de archivo*

En la sección anterior, revisamos cómo configurar políticas de archivado y retención en sus cuentas de almacenamiento. En esta sección, vamos a revisar cómo trabajar con los diferentes niveles de rendimiento usando el SDK de Azure Storage.

El siguiente ejemplo muestra cómo cambiar un blob entre los diferentes niveles de acceso:

1. Abra Visual Studio Code y cree una carpeta para su proyecto.
2. En la ventana de código de Visual Studio, abra una nueva terminal.
3. Utilice el siguiente comando para crear un nuevo proyecto de consola:

```
nueva consola dotnet
```

4. Utilice el siguiente comando para instalar paquetes NuGet:  
[Haga clic aquí para ver la imagen del código](#)

```
dotnet agregar paquete <NuGet_package_name>
```

5. Instale los siguientes paquetes de NuGet:
  1. Azure.Storage.Blobs
  2. Azure.Almacenamiento.Común
  3. Microsoft.Extensions.Configuration

4. Microsoft.Extensions.Configuration.Binder
  5. Microsoft.Extensions.Configuration.Json
6. En la carpeta del proyecto, cree un nuevo archivo JSON y **asígnele el nombre AppSettings.json**. Copie el contenido del Listado 2-18 al archivo JSON.
  7. Cree un archivo de clase C # y **asígnele el nombre AppSettings.cs**.
  8. Reemplace el contenido del archivo AppSettings.cs con el contenido del Listado 2-19. Cambie el nombre del espacio de nombres para que coincida con el nombre de su proyecto.
  9. Cree un archivo de clase C # y **asígnele el nombre Common.cs**.
  10. Reemplace el contenido del archivo Common.cs con el contenido del Listado 2-20.
  11. Cambie el nombre del espacio de nombres para que coincida con el nombre de su proyecto.
  12. Reemplace el contenido del archivo Program.cs con el contenido del Listado 2-21. Cambie el nombre del espacio de nombres para que coincida con el nombre de su proyecto.
  13. Edite su archivo de proyecto .csproj y agregue el siguiente código dentro de la sección ItemGroup:

Haga clic aquí para ver la imagen del código

```
<Ninguna Actualización = "AppSettings.json">  
  
<CopyToOutputDirectory> PreserveNewest </CopyToOutputDirectory>  
  
</Ninguno>
```
14. En este punto, puede establecer algunos puntos de interrupción en el archivo Program.cs para ver, paso a paso, cómo el código mueve los elementos de blob entre los diferentes contenedores y cuentas de almacenamiento.
  15. En la ventana de Visual Studio, presione F5 para compilar y ejecutar su código. Puede usar Azure Portal o la aplicación de escritorio Microsoft Azure Storage Explorer para revisar cómo cambian sus ubicaciones los elementos de blob.

**Listado 2-18** Archivo de configuración AppSettings.json

Haga clic aquí para ver la imagen del código

---

```
{  
  
    "SASConnectionString":  
    "<SASConnectionString_from_your_first_storage_account>",  
  
    "AccountName":  
    "<nombre_de_su_primera_cuenta_de_almacenamiento>",  
  
    "ContainerName": "<source_container_name>"  
  
}
```

**Listado 2-19** AppSettings.cs Clase C #

Haga clic aquí para ver la imagen del código

---

```
// C # .NET Core  
  
utilizando Microsoft.Extensions.Configuration;  
  
  
espacio de nombres ch2_2_6  
  
{  
  
    AppSettings de clase pública  
  
    {  
  
        cadena pública SASConnectionString {obtener;  
colocar; }  
  
        Cadena pública AccountName {get; colocar; }  
  
        cadena pública ContainerName {get; colocar; }  
  
  
        AppSettings estáticos públicos LoadAppSettings ()  
  
        {
```

```

        IConfigurationRoot configRoot = new
ConfigurationBuilder ()

        .AddJsonFile ("AppSettings.json", false)

        .Construir();

        AppSettings appSettings = configRoot.Get
<AppSettings> ();

        return appSettings;

    }

}

```

**Listado 2-20 Clase Common.cs C #**

[Haga clic aquí para ver la imagen del código](#)

---

```

// C # .NET Core

utilizando Azure.Storage.Blobs;

espacio de nombres ch2_2_6

{

clase pública común

{



public static BlobServiceClient
CreateBlobClientStorageFromSAS (string
SASConnectionString)

{

```

```

        intentar

    {
        blobClient = nuevo BlobServiceClient
(SASConnectionString);

    }

    catch (System.Exception)

    {

        lanzar;

    }

    return blobClient;

}

}

```

En el Listado 2-21, las partes del código que son importantes para el proceso de trabajo con los diferentes Niveles de acceso se muestran en negrita.

#### **Listado 2-21** Program.cs Clase C #

Haga clic aquí para ver la imagen del código

---

```

// C # .NET Core

usando System.Threading.Tasks;

usando el sistema;

utilizando Azure.Storage.Blobs;

utilizando Azure.Storage.Blobs.Models;

utilizando Azure.Storage.Blobs.Specialized;

```

```
espacio de nombres ch2_2_6

{

    programa de clase

    {

        static void Main (cadena [] argumentos)

        {

            Console.WriteLine ("Mover blobs entre niveles de
acceso");

            Task.Run (async () => aguardar
StartContainersDemo ()). Wait ();

        }

    }

    Tarea asíncrona estática pública StartContainersDemo
()

{

    string BlobFileName = "Testing.zip";

    AppSettings appSettings =
AppSettings.LoadAppSettings ();



        // Obtenga un cliente en la nube para la cuenta
de almacenamiento

        BlobServiceClient blobClient =
Common.CreateBlobClientStorageFromSAS

(appSettings.SASConnectionString);





        // Obtenga una referencia para cada contenedor
```

```
        var containerReference =
blobClient.GetBlobContainerClient (appSettings.
ContainerName);

        // Obtenga una referencia para el blob

        var blobReference =
containerReference.GetBlobClient (BlobFileName);

        // Obtener el nivel de acceso actual

        BlobProperties blobProperties = await
blobReference.GetPropertiesAsync ();

        System.Console.WriteLine ($ "Nivel de acceso:
{blobProperties.AccessTier} \t" +
                            $ "Inferido:
{blobProperties.AccessTierInferred} \t" +
                            $ "Fecha del último cambio de nivel
de acceso: {blobProperties.
AccessTierChangedOn} ");

        // Cambiar el nivel de acceso a Cool

blobReference.SetAccessTier (AccessTier.Cool);

        // Obtener el nivel de acceso actual

        blobProperties = await
blobReference.GetPropertiesAsync ();

        System.Console.WriteLine ($ "Nivel de acceso:
{blobProperties.AccessTier} \t" +
```

```
    $ "Inferido:  
{blobProperties.AccessTierInferred} \ t" +  
  
        $ "Fecha del último cambio de nivel  
de acceso: {blobProperties.  
AccessTierChangedOn} " ;  
  
  
        // Cambiar el nivel de acceso a archivo  
  
        blobReference.SetAccessTier  
(AccessTier.Archive) ;  
  
  
        // Obtener el nivel de acceso actual  
  
        blobProperties = espera  
blobReference.GetPropertiesAsync () ;  
  
        System.Console.WriteLine ($ "Nivel de acceso:  
{blobProperties.AccessTier} \ t" +  
  
        $ "Inferido:  
{blobProperties.AccessTierInferred} \ t" +  
  
        $ "Fecha del último cambio de nivel  
de acceso: {blobProperties.  
AccessTierChangedOn} " ;  
  
  
        // Cambiar el nivel de acceso a Hot  
  
        blobReference.SetAccessTier (AccessTier.Hot) ;  
  
  
        // Obtener el nivel de acceso actual  
  
        blobProperties = espera  
blobReference.GetPropertiesAsync () ;
```

```

        System.Console.WriteLine($" Nivel de acceso:
{blobProperties.AccessTier} \t" +
                            $" Inferido:
{blobProperties.AccessTierInferred} \t" +
                            $" Fecha del último cambio de nivel
de acceso: {blobProperties.

AccessTierChangedOn} \t "+

                            $" Estado del archivo: {
blobProperties.ArchiveStatus }");

    }

}

```

Como puede ver, puede cambiar a los diferentes niveles de acceso mediante el método `SetAccessTier()` del objeto `BlobClient` que representa el blob con el que está trabajando. También debe usar la propiedad de metadatos `AccessTier` del blob para obtener, que es el nivel de acceso actual donde se almacena el blob. Debe prestar especial atención a la propiedad `ArchiveStatus`. Si intenta cambiar el nivel de acceso de un blob que se está rehidratando desde el nivel de archivo, obtendrá una excepción. La propiedad `AccessTierInferred` también es esencial, ya que indica si el nivel de acceso actual se hereda del contenedor o está configurado en el blob.



### *Sugerencia para el examen*

No debe intentar cambiar el nivel de acceso de un blob almacenado en una cuenta de almacenamiento de Azure Gen1. Solo Azure Storage Account Gen2 permite trabajar con niveles de acceso. Si intenta usar el método `SetAccessTier()` con un blob en una cuenta de almacenamiento de Azure Gen1, obtendrá una excepción.



### *Sugerencia para el examen*

Puede pasar de caliente a frío y viceversa para acceder a niveles sin necesidad de esperar la rehidratación. El proceso de rehidratación solo ocurre cuando mueve un blob del archivo a cualquier otro nivel de acceso.

## RESUMEN DEL CAPÍTULO

---

- Cosmos DB es un servicio de almacenamiento premium que brinda acceso de baja latencia a los datos distribuidos en todo el mundo.
- La propiedad del sistema PartitionKey define la partición donde se almacena la entidad.
- La elección de la PartitionKey correcta es fundamental para lograr el nivel de rendimiento adecuado.
- Puede acceder a Cosmos DB mediante diferentes API: SQL, Table, Gremlin (Graph), MongoDB y Cassandra.
- Puede crear sus índices personalizados en Cosmos DB.
- Puede elegir la propiedad que se utiliza como clave de partición.
- Debe evitar seleccionar claves de partición que creen demasiadas o muy pocas particiones lógicas.
- Una partición lógica tiene un límite de 20 GB de almacenamiento.
- Los niveles de coherencia definen cómo se replican los datos entre las diferentes regiones de una cuenta de Cosmos DB.
- Hay cinco niveles de coherencia: fuerte, obsolescencia limitada, sesión, prefijo coherente y eventual.
- El nivel de consistencia fuerte proporciona un nivel más alto de consistencia pero también tiene una latencia más alta.
- El nivel de coherencia eventual proporciona una latencia más baja y una coherencia de datos más baja.
- Puede mover elementos de blob entre contenedores en la misma cuenta de almacenamiento o contenedores en diferentes cuentas de almacenamiento.

- El servicio Azure Blob Storage ofrece tres niveles de acceso diferentes con diferentes precios para el almacenamiento y el acceso a los datos.
- Puede mover los datos a los que se accede con menos frecuencia a niveles de acceso frío o de archivo para ahorrar dinero.
- Puede gestionar automáticamente el movimiento entre niveles de acceso mediante la implementación de políticas de gestión del ciclo de vida.

## EXPERIMENTO MENTAL

---

En este experimento mental, puede demostrar sus habilidades y conocimientos sobre los temas cubiertos en este capítulo. Puede encontrar las respuestas a este experimento mental en la siguiente sección.

Está desarrollando una aplicación web que necesita trabajar con información con una estructura que puede cambiar durante la vida útil del proceso de desarrollo. Necesita consultar esta información utilizando diferentes criterios. Debe asegurarse de que su aplicación devuelva los resultados de las consultas lo más rápido posible. Su aplicación necesita obtener información de un sistema externo. El sistema externo carga información en una cuenta de Azure Blob Storage Gen1.

Con esta información en mente, responda las siguientes preguntas:

1. Durante las fases de prueba, se da cuenta de que la clave de partición está creando "puntos calientes". ¿Qué debes hacer para solucionar la situación?
2. La información proporcionada por el sistema externo debe almacenarse durante varios años por motivos legales. Una vez que la información es procesada por su aplicación, la información ya no es necesaria. Debe proporcionar una solución segura y rentable.

## RESPUESTAS DEL EXPERIMENTO MENTAL

---

Esta sección contiene las soluciones al experimento mental.

1. Aparece una "zona activa" en un contenedor de Cosmos DB cuando elige una clave de partición que almacena la mayoría de los elementos en la misma partición lógica. Puede resolver el "punto caliente" cambiando la

clave de partición y eligiendo una clave de partición que distribuya los elementos de manera uniforme en las diferentes particiones lógicas. Desafortunadamente, no puede modificar una clave de partición una vez que haya creado el contenedor. En este escenario, debe crear un nuevo contenedor con la nueva clave de partición y luego migrar todos los datos al nuevo contenedor utilizando la herramienta AzCopy.

2. Dado que necesita conservar los datos durante varios años, debe utilizar los niveles de acceso a la cuenta de almacenamiento. Debido a que está utilizando una cuenta de almacenamiento Gen1, no puede usar niveles de acceso en esa cuenta de almacenamiento. Necesita actualizar su cuenta de Blob Storage a Gen2. Una vez que tenga su cuenta de almacenamiento Gen2, puede configurar una política de gestión del ciclo de vida para pasar automáticamente al nivel de archivo para aquellos archivos a los que no se ha accedido durante algún tiempo

# Capítulo 3. Implementar la seguridad de Azure

Independientemente de la aplicación, la mayoría de ellos tienen un requisito estándar: proteger la información que administra. Con respecto a la seguridad, debe pensar en las cinco dimensiones de la conciencia de seguridad de la información: integridad, disponibilidad, confidencialidad, autorización y no repudio. Cada una de estas dimensiones es útil para evaluar los diferentes riesgos y las contramedidas que necesita implementar para mitigar los riesgos asociados.

Implementar el mecanismo de seguridad apropiado en su aplicación puede ser tedioso y potencialmente propenso a errores. Azure ofrece varios mecanismos para agregar medidas de seguridad a sus aplicaciones, controlar los diferentes aspectos de seguridad para acceder a sus datos y controlar los servicios que dependen de sus aplicaciones.

## **Habilidades cubiertas en este capítulo:**

- [Habilidad 3.1: Implementar la autenticación y autorización de usuarios](#)
- [Habilidad 3.2: Implementar soluciones seguras en la nube](#)

## **HABILIDAD 3.1: IMPLEMENTAR LA AUTENTICACIÓN Y AUTORIZACIÓN DE USUARIOS**

---

Cuando un usuario desea acceder a su aplicación, el usuario debe demostrar que es la persona que dice ser. La autenticación es la acción que realiza el usuario para probar su identidad. El usuario demuestra su identidad utilizando información que solo el usuario conoce. Un sistema de autenticación debe abordar cómo proteger esa información para que solo el usuario apropiado pueda acceder a ella, mientras que nadie más, ni siquiera el sistema de autorización, puede acceder a ella. Una solución para este problema es permitir que el usuario acceda a sus datos utilizando dos mecanismos diferentes para probar su identidad:

información que solo el usuario conoce y que muestra algo, un token, que solo el usuario tiene. Este enfoque se conoce como *autenticación multifactor*.

Azure proporciona un mecanismo seguro para integrar la autenticación en sus aplicaciones. Puede utilizar sistemas de autenticación de factor único o multifactorial sin preocuparse por los intrincados detalles de la implementación de este tipo de sistema.

Autenticar a los usuarios antes de que puedan acceder a su aplicación es solo una parte de la ecuación. Una vez que sus usuarios han sido autenticados, debe decidir si algún usuario puede acceder a cualquier parte de su aplicación o si algunas partes de su aplicación están restringidas. La autorización controla qué acciones o secciones puede realizar el usuario una vez que ha sido autorizado.

### **Esta habilidad cubre cómo**

- Implementar la autenticación OAuth2
- Cree e implemente firmas de acceso compartido
- Registre aplicaciones y use Azure Active Directory para autenticar usuarios
- Controlar el acceso a los recursos mediante el control de acceso basado en roles (RBAC)

### *Implementar la autenticación OAuth2*

El proceso de autenticación requiere que el usuario proporcione evidencia de que el usuario es la persona que dice ser. En el mundo real, puede encontrar varios ejemplos de autenticación; por ejemplo, cada vez que muestra su licencia de conducir a un oficial de policía, en realidad se está autenticando contra el oficial de policía. En el mundo digital, esta autenticación ocurre al proporcionar cierta información que solo usted conoce, como una palabra secreta (una contraseña), un certificado digital o cualquier tipo de token que solo usted posea.

Tiene una variedad de opciones para implementar dicho mecanismo de autenticación en su aplicación. Cada implementación tiene sus pros y sus contras, y el mecanismo de autenticación adecuado depende del nivel de seguridad que requiera para su aplicación.

La forma más básica de autenticar a un usuario es la autenticación basada en formularios. Cuando usa este mecanismo, necesita programar un formulario web que le pida al usuario un nombre de usuario y una contraseña. Una vez que el usuario envía el formulario, la información en el formulario se compara con los valores almacenados en su sistema de almacenamiento. Este sistema de almacenamiento puede ser una base de datos relacional, una base de datos NoSQL o incluso un archivo simple con diferentes formatos almacenados en un servidor. Si la información proporcionada por el usuario coincide con la información almacenada en su sistema, la aplicación envía una cookie al navegador del usuario. Esta cookie almacena una clave o algún tipo de identificación para autenticar solicitudes posteriores para acceder a su aplicación sin pedirle repetidamente al usuario su nombre de usuario y contraseña.

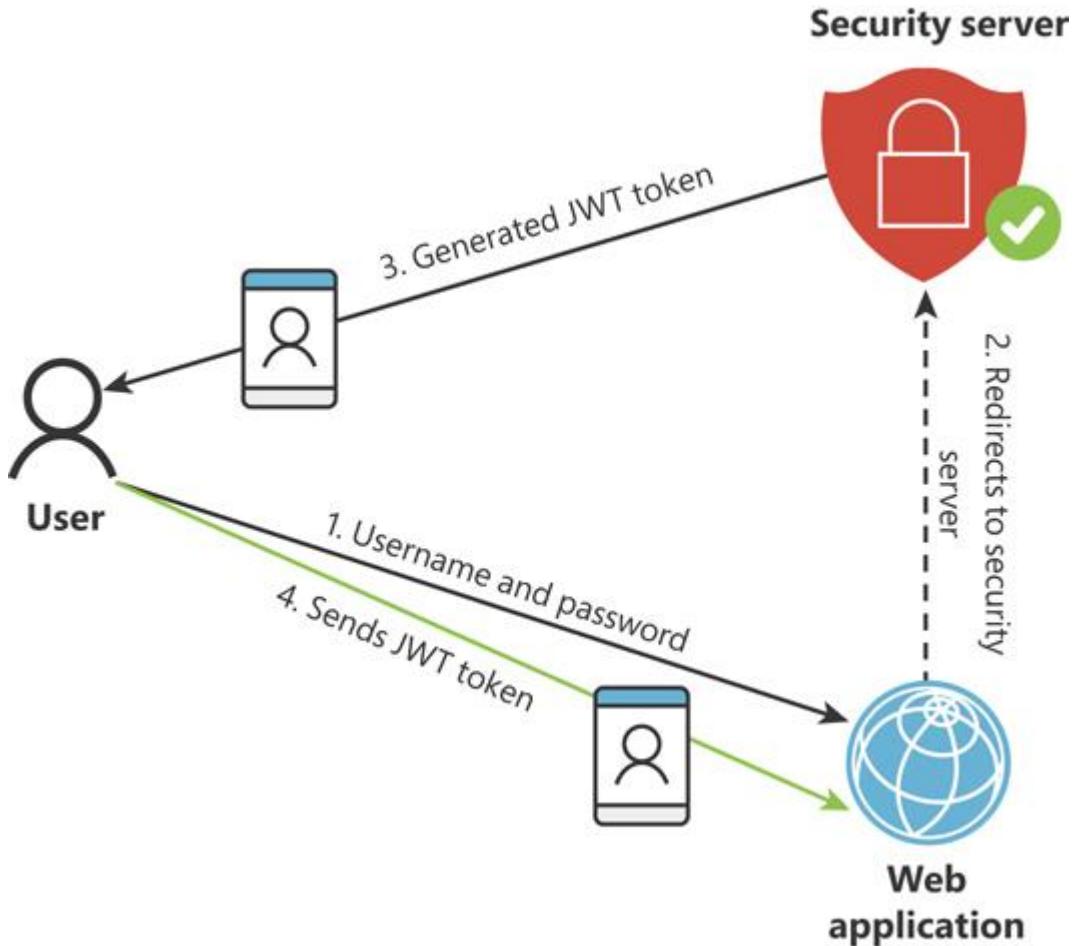
Uno de los inconvenientes más importantes del uso de la autenticación basada en formularios es la dependencia del mecanismo de autenticación de las cookies. Otro inconveniente es que tiene estado, lo que requiere que su servidor mantenga una sesión de autenticación para rastrear la actividad entre el servidor y el cliente. Esta dependencia de las cookies y la gestión de la sesión de autenticación hace que sea más difícil escalar las soluciones mediante la autenticación basada en formularios. Un punto adicional a considerar es que las cookies no funcionan bien (o es un desafío trabajar con ellas) en aplicaciones móviles. Afortunadamente, existen alternativas a la autenticación basada en formularios que son más adecuadas para los requisitos que tienen escenarios móviles o de IoT; Además, existen alternativas que pueden mejorar la escalabilidad de su aplicación web.

La autenticación basada en tokens es el mecanismo de autenticación más extendido para entornos y escenarios que requieren una alta escalabilidad o no admiten el uso de cookies. La autenticación basada en token consiste en un token firmado que su aplicación usa para autenticar solicitudes y otorgar acceso a los recursos en su aplicación. El token no contiene el nombre de usuario y la contraseña de su usuario. En cambio, el token almacena cierta información sobre el usuario autenticado que su servidor puede usar para otorgar acceso a los recursos de su aplicación.

Cuando utiliza la autenticación basada en token, sigue un flujo de trabajo similar al que se muestra en la Figura 3-1 :

1. Un usuario no autenticado se conecta a su aplicación web.

2. Su aplicación web redirige al usuario a la página de inicio de sesión. Esta página de inicio de sesión puede ser proporcionada por su aplicación web que actúa como un servidor de seguridad o por un servidor de seguridad externo.
3. El servidor de seguridad valida la información proporcionada por el usuario, por lo general, el nombre de usuario y la contraseña, y genera un token JWT.
4. El servidor de seguridad envía el token JWT al usuario. El navegador o la aplicación móvil que el usuario utilizó para conectarse a su aplicación es responsable de almacenar este token JWT para reutilizarlo en las siguientes solicitudes.
5. Los navegadores o la aplicación móvil proporcionan el token JWT a su aplicación web en cada solicitud siguiente.



**Figura 3-1** Flujo de trabajo básico de la autenticación basada en token

Hay varias implementaciones de tokens, pero la más extendida es JSON Web Token o JWT. Un token JWT consta de

- **Encabezado** El encabezado contiene el nombre del algoritmo utilizado para firmar el token.
- **Cuerpo o carga útil** El cuerpo o la carga útil contiene información diferente sobre el token y el propósito de este token. El cuerpo contiene varios campos estándar o reclamos que se definen en el RFC 7519 y cualquier otro campo personalizado que pueda necesitar para su aplicación.
- **Firma criptográfica** Esta es una cadena que valida su token y asegura que el token no haya sido dañado o manipulado incorrectamente.

Una de las principales ventajas de utilizar la autenticación basada en tokens es que puede delegar el proceso de gestión de la identidad de los usuarios a servidores de seguridad externos. Gracias a esta delegación, puede abstraerse de la implementación de administrar y almacenar tokens JWT y nombres de usuario y contraseñas. Eso es lo que hace cuando desea permitir que sus usuarios accedan a su aplicación mediante el uso de sus cuentas de Facebook, Google o Twitter. Su aplicación confía en los procesos de identificación y autenticación realizados por estos servidores de seguridad externos o administradores de identidad, y usted otorga acceso a su aplicación en función de la información almacenada en el token JWT proporcionado por el servidor de seguridad. Aún necesita almacenar cierta información sobre el usuario. Todavía,

Microsoft proporciona el marco de identidad para trabajar con la autenticación. Puede utilizar Identity Framework para agregar autenticación basada en token a su aplicación. Como se mencionó anteriormente, puede implementar su propia autenticación basada en token o utilizar un proveedor de autenticación externo que realice la verificación del inicio de sesión y la contraseña del usuario. El siguiente ejemplo muestra cómo crear una aplicación web simple con la autenticación de Google habilitada. Puede utilizar un procedimiento similar para habilitar otro inicio de sesión social en su aplicación, como cuentas de Facebook, Twitter o Microsoft.

1. Abra Visual Studio 2019 en su computadora.
2. En la ventana de bienvenida de Visual Studio 2019, en la columna Introducción, haga clic en Crear un nuevo proyecto.

3. En la ventana Crear un proyecto nuevo, en el menú desplegable Todos los idiomas, seleccione C #.
4. En el cuadro de texto Buscar plantillas, escriba **asp.net**.
5. En la lista de resultados, haga clic en Aplicación web ASP.NET (.NET Framework).
6. Haga clic en el botón Siguiente en la esquina inferior derecha de la ventana.
7. En Configure Your New Project, escriba un nombre de proyecto, una ubicación y un nombre de solución para su proyecto.
8. Haga clic en el botón Crear en la esquina inferior derecha de la ventana.
9. En la ventana Crear una nueva aplicación web ASP.NET, seleccione la plantilla MVC de la lista de plantillas en el medio del lado izquierdo de la ventana. MVC es para Model-View-Controller.
10. En el lado derecho de la ventana Crear una nueva aplicación web ASP.NET, en la sección Autenticación, haga clic en el enlace Cambiar.
11. En la ventana Cambiar autenticación, seleccione Cuentas de usuario individuales de las opciones disponibles en la columna de la izquierda.
12. Haga clic en el botón Aceptar para cerrar la ventana Cambiar autenticación.
13. Haga clic en el botón Crear en la esquina inferior derecha de la ventana.

En este punto, ha creado una aplicación web ASP.NET básica configurada para usar la autenticación basada en formularios. Ahora, puedes comprobar que funciona la autenticación básica en esta aplicación. En este punto, no está utilizando la autenticación OAuth2:

1. Presione F5 para ejecutar el proyecto.
2. En la esquina superior derecha del navegador web de su aplicación, haga clic en Registrarse.
3. En el formulario de registro, ingrese una dirección de correo electrónico y una contraseña.

4. Haga clic en el botón Registrarse en la parte inferior del formulario.

Una vez que se haya registrado, iniciará sesión automáticamente y podrá cerrar la sesión y volver a iniciarla para asegurarse de que todo funcione correctamente.

Ahora, use los siguientes pasos para modificar la aplicación web recién creada para agregar autenticación OAuth2:

1. En el Explorador de soluciones, haga clic en el nombre de su proyecto y presione F4.
2. En la sección Servidor de desarrollo, asegúrese de que el valor de la configuración SSL habilitado esté establecido en Verdadero.
3. Copie la URL de SSL debajo de la configuración de SSL habilitado y cierre la ventana Propiedades.
4. En el Explorador de soluciones, haga clic con el botón derecho en el nombre del proyecto y haga clic en Propiedades en la parte inferior del menú contextual. Esto abre el archivo csproj de su proyecto en una nueva pestaña.
5. En la pestaña del archivo csproj de su proyecto, seleccione la pestaña Web y pegue la URL de SSL en el cuadro de texto URL del proyecto en la sección Servidores.
6. Abra el archivo HomeController.cs y agregue el `RequireHttps` atributo a la `HomeController` clase:

Haga clic aquí para ver la imagen del código

```
[RequireHttps]

HomeController de clase pública: Controlador

{

    Índice de resultado de acción público ()
```

7. Cree un proyecto de Google para integrar su aplicación web con la plataforma de autenticación de Google. Necesita una cuenta de Google para estos pasos:

1. Inicie sesión en su cuenta de Google.

2. Navegue a <https://developers.google.com/identity/sign-in/web/devconsole-project>.
3. Haga clic en Configurar un proyecto.
4. En el cuadro de diálogo Configurar un proyecto para el inicio de sesión de Google, seleccione Crear un proyecto nuevo en el menú desplegable.
5. Ingrese un nombre para su proyecto y haga clic en Siguiente en la esquina inferior izquierda del cuadro de diálogo.
6. En el cuadro de diálogo Configure Your OAuth Client, escriba el nombre de su aplicación web. Este nombre se muestra en la ventana de consentimiento que aparece al usuario durante el inicio de sesión.
7. Haga clic en Siguiente en la esquina inferior izquierda del cuadro de diálogo.
8. En el cuadro de diálogo Configure su cliente OAuth, seleccione Servidor web en el campo ¿Desde dónde está llamando? Menú desplegable.
9. En el cuadro de texto URI de redireccionamiento autorizado, use la URL SSL que copió en el paso 4 y cree un URI de redireccionamiento con esta estructura:

```
<YOUR_URL_SSL> / signin-google
```

Utilice el siguiente ejemplo como referencia:

[Haga clic aquí para ver la imagen del código](#)

```
https://localhost:44395 / signin-google
```

10. Haga clic en Crear en la esquina inferior izquierda del cuadro de diálogo.
11. ¡En el Estás listo! cuadro de diálogo, haga clic en el botón Descargar configuración del cliente. Alternativamente, puede copiar los campos ID de cliente y Secreto de cliente. Necesitará estos valores en un paso posterior.
12. Haga clic en el enlace de la Consola API en la parte inferior del cuadro de diálogo.

13. En el lado izquierdo de Google Console, haga clic en Biblioteca.
  14. En el cuadro de texto Buscar API y servicios, escriba **Google+**.
  15. En la lista de resultados, haga clic en API de Google+.
  16. En la ventana de la API de Google+, haga clic en el botón Habilitar.
8. En el archivo App\_Start / Startup.Auth.cs, en el `ConfigureAuth` método, descomente las siguientes líneas:

Haga clic aquí para ver la imagen del código

```
app.UseGoogleAuthentication (nuevo
    GoogleOAuth2AuthenticationOptions ())
{
    ClientId = "",
    ClientSecret = ""
});
```

9. Utilice los valores de Id. De cliente y Secreto de cliente que copió en el paso 7k y asigne el valor a la variable correspondiente en el código anterior. (Tenga en cuenta que estos elementos deben colocarse entre comillas).
10. Presione F5 para ejecutar la aplicación.
11. Haga clic en Iniciar sesión en la esquina superior izquierda de la aplicación web.
12. En la opción Usar otro servicio para iniciar sesión en el lado izquierdo de la página, haga clic en el botón Google.
13. Inicie sesión con su cuenta de Google.
14. Haga clic en el botón Registrarse. Una vez que haya iniciado sesión con Google, se le redirigirá a la aplicación web. Debido a que su cuenta de Google no existe en la base de datos de su aplicación, obtiene un formulario de registro.

15. Está registrado y ha iniciado sesión en su aplicación con una cuenta de Google.

Una vez que haya iniciado sesión en la aplicación con su cuenta de Google, puede revisar la base de datos y buscar la nueva información de inicio de sesión. Puede ver las conexiones de la base de datos en el Explorador de objetos de SQL Server. Puede abrir el Explorador de objetos de SQL Server desde el menú Ver. Puede ver en la tabla AspNetUsers que hay una nueva entrada para el inicio de sesión de su nueva cuenta de Google, pero el campo PasswordHash está vacío. También puede revisar la tabla AspNetUserLogins. Esta tabla contiene todos los inicios de sesión de proveedores de autorización externos, como Google, que se han registrado en su aplicación.

El ejemplo anterior revisó cómo puede usar servidores de identidad de terceros, como Google, para autenticar usuarios en su aplicación usando el protocolo OAuth2. Ahora, vamos a revisar cómo crear su servidor OAuth2. Por lo general, agrega este tipo de autenticación cuando necesita estas aplicaciones de terceros para acceder a algunos servicios o recursos de su código. Puede otorgar acceso a estas aplicaciones creando un token que autentique la aplicación de terceros y otorgue acceso a partes específicas de su servicio HTTP.

El protocolo OAuth aborda la necesidad de asegurar el acceso a los recursos y la información en su aplicación mediante el proceso de un tercero. Sin OAuth, si desea otorgar acceso a una aplicación externa a los recursos de su aplicación, debe usar un nombre de usuario y una contraseña. Si la aplicación de terceros se ve comprometida, el nombre de usuario y la contraseña también se verán comprometidos y sus recursos estarán expuestos. El protocolo OAuth define cuatro roles diferentes:

- **Propietario del recurso** Esta es la persona o entidad que puede otorgar acceso a los recursos. Si el propietario del recurso es una persona, también se le puede denominar usuario.
- **Servidor de recursos** Este es el servidor que aloja los recursos que desea compartir. Este servidor debe poder aceptar y responder a los códigos de acceso utilizados para acceder al recurso.
- **Cliente** Esta es la aplicación de terceros que necesita acceder al recurso. El cliente realiza las solicitudes necesarias al servidor de recursos en nombre del propietario del recurso. El término

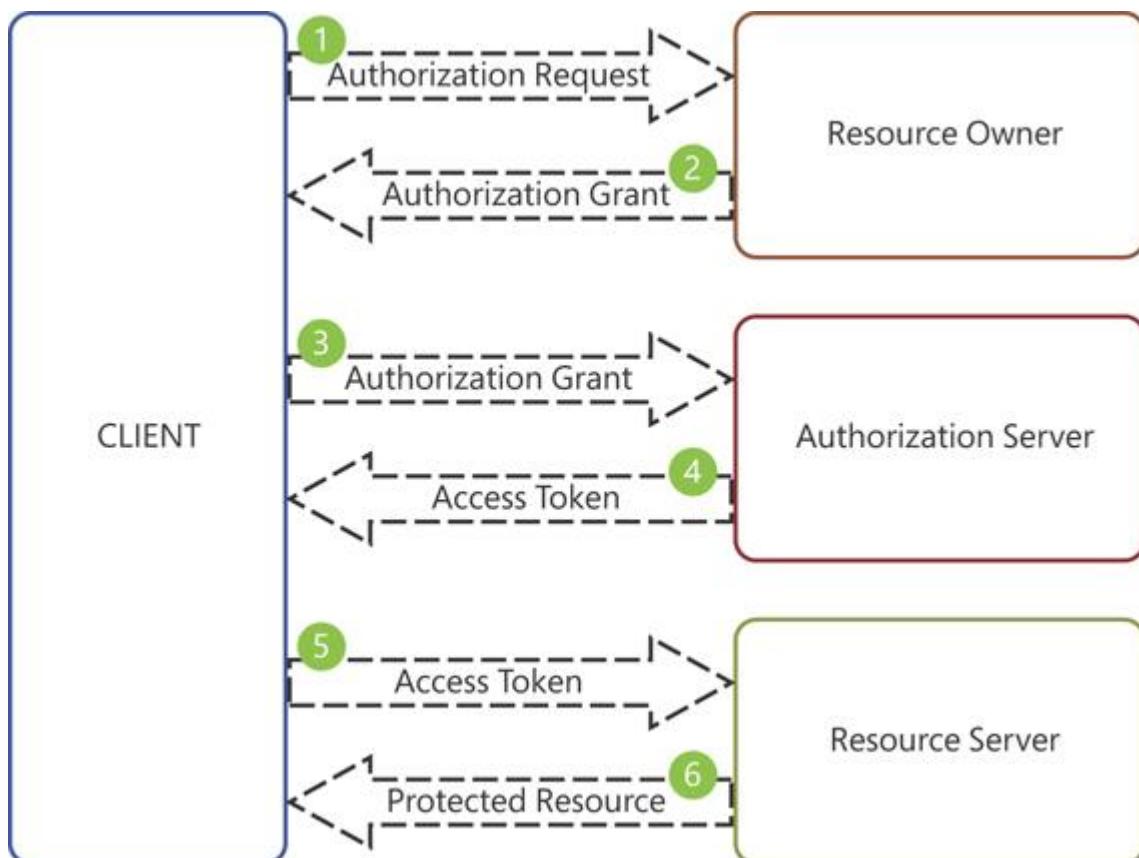
"cliente" no implica necesariamente una implementación específica, como un servidor, un escritorio o cualquier otro tipo de dispositivo.

- **Servidor de autorización** Este es el servidor que emite el token de acceso al cliente para acceder a los recursos. El cliente debe estar autenticado antes de que pueda obtener el token correcto.

La figura 3-2 muestra el flujo de autenticación básico para OAuth.

Como puede ver en la Figura 3-2 , el proceso de adquirir un token para acceder a un recurso protegido consta de los siguientes pasos:

- **Solicitud de autenticación** El cliente solicita acceso al recurso protegido. El propietario del recurso, en función de los privilegios del cliente, concede acceso al cliente para acceder al recurso. La autenticación del cliente puede ser realizada directamente por el propietario del recurso o preferiblemente por el servidor de autenticación.



**Figura 3-2** Flujo de autenticación básica de OAuth

- **Concesión de autenticación** Cuando el propietario del recurso otorga al cliente acceso al recurso, el cliente envía una concesión de autenticación, que es un código o credencial que representa el permiso para acceder al recurso, que ha sido otorgado por el propietario del recurso. El cliente utiliza esta credencial de concesión de autenticación para solicitar un token de acceso al servidor de autorización. Hay cuatro mecanismos diferentes para manejar esta autenticación:

- **Código de autorización** El cliente indica al propietario del recurso que solicite autenticación al servidor de autenticación. Una vez que el propietario del recurso está autenticado, el servidor de autenticación crea un código de autorización que el propietario del recurso envía al cliente. El cliente utiliza este código de autorización como concesión para solicitar el token de acceso.
- **Implícito** Con este flujo de concesión de autenticación, el servidor de autenticación no autentica al cliente. En cambio, el cliente obtiene el token de acceso sin necesidad de autenticarse en el servidor de recursos mediante una concesión de autenticación. Este flujo implícito es un flujo de código de autorización simplificado. Para mejorar la seguridad en este flujo, el servidor de recursos utiliza el URI de redireccionamiento proporcionado por el cliente.
- **Credenciales del propietario del recurso** En lugar de usar un código de autorización o autenticación implícita, el cliente usa las credenciales del propietario del recurso para autenticarse en el servidor de recursos. Este tipo de concesión de autenticación debe usarse solo cuando existe un alto nivel de confianza entre el cliente y el propietario del recurso.
- **Credenciales del cliente** El cliente proporciona sus credenciales para acceder al recurso. Esta concesión de autenticación es útil para escenarios en los que el cliente necesita acceso a recursos que están protegidos por el mismo servidor de autorización que el cliente y están bajo el control del cliente. Este tipo de concesión de autenticación también es útil si el servidor de recursos y el cliente acordaron la misma autorización para los recursos y el cliente.

- **Token de acceso** El cliente solicita un token de acceso del servidor de autorización que le permite acceder al recurso en el servidor de recursos. El cliente envía este token de acceso al servidor de recursos con cada solicitud para acceder al recurso. Este token de acceso tiene fecha de vencimiento. Una vez que el token de acceso caduca, el token no es válido y el cliente debe solicitar otro token de acceso. Para facilitar el proceso de renovación del token de acceso, el servidor de autenticación proporciona dos tokens diferentes: el token de acceso real y un token de actualización. El cliente usa el token de actualización cuando necesita renovar un token de acceso caducado.
- **Recurso protegido** Este es el recurso al que el cliente desea acceder. El servidor de recursos protege el recurso. El cliente debe enviar el token de acceso al servidor de recursos cada vez que necesite acceder al recurso.

#### *¿Necesita más revisión? El marco de autorización de Oauth 2.0*

Puede obtener más información sobre los detalles de cómo funciona el marco de autorización de OAuth 2.0 revisando el RFC 6749 oficial en <https://tools.ietf.org/html/rfc6749>.

El siguiente ejemplo muestra cómo implementar la autenticación OAuth 2.0 en su aplicación Web API. En este ejemplo, creará un servidor de autorización, un servidor de recursos y un cliente que puede solicitar un token de acceso antes de acceder al recurso. En aras de la legibilidad, hemos dividido los pasos para implementar este ejemplo en diferentes partes. Los siguientes pasos muestran cómo crear el servidor de autorización:

1. Abra Visual Studio 2019.
2. Haga clic en Archivo> Nuevo> Proyecto.
3. En la ventana Crear un proyecto nuevo, en el menú desplegable Todos los idiomas, seleccione C #.
4. En el cuadro de texto Buscar plantillas, escriba **asp.net**.
5. En la lista de resultados, haga clic en Aplicación web ASP.NET (.NET Framework).
6. Haga clic en el botón Siguiente en la esquina inferior derecha de la ventana.

7. En la ventana Configure su nuevo proyecto, escriba un nombre de proyecto, una ubicación y un nombre de solución para su proyecto.

8. Haga clic en el botón Crear en la esquina inferior derecha de la ventana.

9. En la ventana Crear una nueva aplicación web ASP.NET, haga clic en la plantilla MVC.

10. Haga clic en el enlace Cambiar en la sección Autenticación en el lado derecho de la ventana.

11. En la ventana Cambiar autenticación, haga clic en la opción Cuentas de usuario individuales.

12. Haga clic en el botón Aceptar en la ventana Cambiar autenticación.

13. Haga clic en el botón Crear en la ventana Crear una nueva aplicación web ASP.NET.

14. En Visual Studio, abra el archivo en App\_Start> Startup.Auth.cs y agregue la siguiente línea al principio del archivo:

[Haga clic aquí para ver la imagen del código](#)

```
utilizando Microsoft.Owin.Security.OAuth;
```

15. Agregue el código que se muestra en el Listado 3-1 al archivo Startup.Auth.cs. Debe agregar este código al `ConfigureAuth()` método, después de la línea

[Haga clic aquí para ver la imagen del código](#)

```
app.UseTwoFactorRememberBrowserCookie (DefaultAuthenticationTypes.  
TwoFactorRememberBrowserCookie);
```

16. Asegúrese de que existan las siguientes declaraciones using en el archivo Startup.Auth.cs para evitar errores de compilación:

1. using System;
2. using Microsoft.AspNet.Identity;
3. using Microsoft.AspNet.Identity.Owin;
4. using Owin;
5. using Microsoft.Owin;
6. using Microsoft.Owin.Security.Cookies;

```
7.    using Microsoft.Owin.Security.OAuth;
8.    using Microsoft.Owin.Security.Infrastructure;
9.    using AuthorizationServer.Constants;
10.   using System.Threading.Tasks;
11.   using System.Collections.Concurrent;
12.   using System.Security.Claims;
13.   using System.Security.Principal;
14.   using System.Linq;
15.   using <your_project's_name>.Models;
```

**Listado 3-1** Adición del servidor de autorización OAuth

Haga clic aquí para ver la imagen del código

---

```
// C#. ASP.NET.

// Configurar el servidor de autorización

    app.UseOAuthAuthorizationServer (nuevo
OAuthAuthorizationServerOptions

    {

        AuthorizeEndpointPath = nueva PathString
(Paths.AuthorizePath),

        TokenEndpointPath = nueva PathString
(Paths.TokenPath),

        ApplicationCanDisplayErrors = verdadero,

#if DEPURAR

        AllowInsecureHttp = true,

#terminara si

        Proveedor = nuevo
OAuthAuthorizationServerProvider

    {

        OnValidateClientRedirectUri =
ValidateClientRedirectUri,
```

```
        OnValidateClientAuthentication =
ValidateClientAuthentication,
        OnGrantResourceOwnerCredentials =
GrantResourceOwnerCredentials,
        OnGrantClientCredentials =
GrantClientCredentials
    },
}

// El proveedor del código de autorización es el objeto
encargado de crear y recibir
// el código de autorización.

AuthorizationCodeProvider = nuevo
AuthenticationTokenProvider

{
    OnCreate = CreateAuthenticationCode,
    OnReceive = ReceiveAuthenticationCode,
},
}

// El proveedor del token de actualización está a
cargo de crear y recibir la actualización
// token.

RefreshTokenProvider = nuevo
AuthenticationTokenProvider

{
    OnCreate = CreateRefreshToken,
    OnReceive = ReceiveRefreshToken,
}
}
```

```

} ) ;

// Protege los recursos en este servidor.

app.UseOAuthBearerAuthentication (new
OAuthBearerAuthenticationOptions

{
} );

```

Este código configura el servidor de autenticación OAuth mediante el `UseOAuthAuthorizationServer()` método. Este método acepta un `OAuthAuthorizationServerOptions` objeto para configurar varios puntos finales útiles:

16. **AuthorizeEndpointPath** El punto final autorizado es la ruta en el servidor de autorización a la que la aplicación cliente redirige al usuario-agente para obtener el consentimiento del usuario o del propietario del recurso para acceder al recurso. Con este consentimiento, la aplicación cliente puede solicitar un token de acceso.

17. **TokenEndpointPath** Esta es la ruta en el servidor de autorización que el cliente usa para obtener un token de acceso. Si el cliente está configurado con un secreto de cliente, el cliente debe proporcionar este secreto de cliente en la solicitud para obtener un nuevo token.

18. **AllowInsecureHttp** Esta configuración permite al cliente realizar solicitudes a los puntos finales autorizados y token mediante el uso de HTTP URI en lugar de HTTPS URI.

19. **Proveedor** Su aplicación de servidor de autorización debe proporcionar los métodos delegados necesarios para procesar los diferentes eventos que surgen durante el flujo de autorización de OAuth. Puede hacer esto implementando la interfaz `OAuthAuthorizationServerProvider` o usando la implementación predeterminada proporcionada por el `OAuthAuthorizationServerProvider` objeto. En este ejemplo, utiliza el `OAuthAuthorizationServerProvider` objeto y proporciona cuatro funciones de delegado para los diferentes eventos. [Los](#)

listados 3-2 a 3-5 muestran los diferentes métodos de delegado que usa para los eventos administrados por este proveedor.

20. **AuthorizationCodeProvider** Cuando el servidor de autorización autentica al cliente, el servidor necesita enviar un código de autorización al servidor. Este proveedor gestiona los eventos que surgen durante la gestión del código de autenticación. Los listados 3-6 y 3-7 muestran los métodos de delegado que administran los eventos de creación o recepción de un código.

21. **RefreshTokenProvider** Este objeto controla los eventos que suceden cuando el cliente solicita una actualización de un token de acceso. Los listados 3-8 y 3-9 muestran los métodos de delegado que controlan los eventos de creación y recepción de una solicitud de actualización de un token de acceso.

17. Agregue el contenido de Listings 3-2 a 3-9 al archivo Startup.Auth.cs. Agregue estos métodos a la clase Startup. La implementación de estos delegados no es adecuada para entornos de producción. Por ejemplo, la validación del URI de redireccionamiento del cliente y la autenticación de los clientes se basan en un valor codificado almacenado en la clase Cliente. En un escenario del mundo real, debería tener estas entidades almacenadas en una base de datos. En este ejemplo, la creación del token de acceso, que se muestra en el Listado 3-4, se almacena en un diccionario en memoria. En un escenario del mundo real, debe guardar en una base de datos los tokens de acceso que otorga a los clientes.

**Listado 3-2** Delegado OnValidateClientRedirectUri

Haga clic aquí para ver la imagen del código

---

// C#. ASP.NET.

```
Tarea privada ValidateClientRedirectUri (contexto  
OAuthValidateClientRedirectUriContext)
```

```
{
```

```

        if (context.ClientId == Clients.Client1.Id)

        {

            context.Validated
(Clientes.Cliente1.RedirectUrl);

        }

        else if (context.ClientId ==
Clients.Client2.Id)

        {

            context.Validated
(Clientes.Cliente2.RedirectUrl);

        }

        return Task.FromResult (0);

    }

}

```

**Listado 3-3 Delegado de OnValidateClientAuthentication**

Haga clic aquí para ver la imagen del código

---

// C#. ASP.NET.

```

Tarea privada ValidateClientAuthentication
(OAuthValidateClientAuthenticationContext

contexto)

{

    string clientId;

    string clientSecret;

    if (context.TryGetBasicCredentials (out
clientId, out clientSecret) ||
        context.TryGetFormCredentials (out
clientId, out clientSecret))

```

```

    {

        if (clientId == Clients.Client1.Id &&
clientSecret == Clients.Client1.

            Secreto)

        {

            context.Validated ();

        }

        else if (clientId == Clients.Client2.Id
&& clientSecret == Clients.

            Cliente2.Secreto)

        {

            context.Validated ();

        }

    }

    return Task.FromResult (0);

}

```

**Listado 3-4** Delegado de OnGrantResourceOwnerCredentials

Haga clic aquí para ver la imagen del código

---

// C#. ASP.NET.

```

Tarea privada OnGrantResourceOwnerCredentials
(OAuthGrantResourceOwnerCredentialsContext

contexto)

{

```

```

        Identidad ClaimsIdentity = nueva
ClaimsIdentity (nueva GenericIdentity (context.

        Nombre de usuario,
OAuthDefaults.AuthenticationType), context.Scope.Select
(x =>

        new Claim ("urn: oauth: scope", x)));

        context.Validated (identidad);

return Task.FromResult (0);

}

```

**Listado 3-5** Delegado de OnGrantClientCredentials

[Haga clic aquí para ver la imagen del código](#)

---

```

// C#. ASP.NET.

Tarea privada GrantClientCredentials (contexto
OAuthGrantClientCredentialsContext)

{

        var identity = new ClaimsIdentity (new
GenericIdentity (context.ClientId,

        OAuthDefaults.AuthenticationType),
context.Scope.Select (x =>

        new Claim ("urn: oauth: scope", x)));

        context.Validated (identidad);

return Task.FromResult (0);

```

```
}
```

**Listado 3-6** Código de autorización para el delegado de OnCreate

Haga clic aquí para ver la imagen del código

---

```
// C#. ASP.NET.
```

```
Private void CreateAuthenticationCode (contexto  
AuthenticationTokenCreateContext)  
  
{  
  
    context.SetToken (Guid.NewGuid () .  
ToString ("n") + Guid.NewGuid () .  
  
ToString ("n"));  
  
    AuthenticationCodes [context.Token] =  
context.SerializeTicket ();  
  
}
```

**Listado 3-7** Código de autorización para el delegado de OnReceive

Haga clic aquí para ver la imagen del código

---

```
// C#. ASP.NET.
```

```
Private void ReceiveAuthenticationCode (contexto  
AuthenticationTokenReceiveContext)  
  
{  
  
    valor de cadena;  
  
    if (_authenticationCodes.TryRemove  
(context.Token, out value))  
  
    {  
  
        context.DeserializeTicket (valor);  
  
    }  
  
}
```

**Listado 3-8** Token de actualización para el delegado de OnCreate

Haga clic aquí para ver la imagen del código

---

// C#. ASP.NET.

```
Private void CreateRefreshToken (contexto  
AuthenticationTokenCreateContext)  
  
{  
  
    context.SetToken (context.SerializeTicket  
());  
  
}
```

**Listado 3-9** Token de actualización para el delegado de OnReceive

Haga clic aquí para ver la imagen del código

---

// C#. ASP.NET.

```
Private void ReceiveRefreshToken (contexto  
AuthenticationTokenReceiveContext)  
  
{  
  
    context.DeserializeTicket (context.Token);  
  
}
```

18. Agregue la siguiente propiedad privada a la clase Startup en el archivo Startup.Auth.cs:

Haga clic aquí para ver la imagen del código

```
private readonly ConcurrentDictionary <cadena, cadena>  
_authenticationCodes =  
  
    new ConcurrentDictionary <cadena, cadena>  
(StringComparer.OrdinalIgnoreCase);
```

19. En la ventana del Explorador de soluciones, agregue una nueva carpeta a su proyecto llamada Constantes.

20. En la ventana del Explorador de soluciones, haga clic con el botón derecho en la carpeta Constantes y haga clic en Agregar> Nuevo elemento.
21. En la ventana Nuevo elemento, en el control de árbol en el lado izquierdo de la ventana, haga clic en Instalado> Visual C #> Código.
22. Haga clic en la plantilla denominada Clase.
23. En la parte inferior de la ventana Agregar nuevo elemento, escriba **Clients.cs** en el cuadro de texto Nombre.
24. Haga clic en el botón Agregar en la esquina inferior derecha de la ventana.
25. Reemplace el contenido del archivo Clients.cs con el contenido del Listado 3-10. Cambie el espacio de nombres para que coincida con el nombre de su proyecto.

**Listado 3-10** Clients.cs

Haga clic aquí para ver la imagen del código

---

// C#. ASP.NET.

```
espacio de nombres <YOUR_PROJECT'S_NAME>.Constants
{
    Clientes de clase pública
    {
        cliente estático público de solo lectura Client1
        = nuevo cliente

        {
            Id = "123456",
            Secreto = "abcdef",
            RedirectUrl =
            Paths.AuthorizeCodeCallBackPath
```

```

    } ;

        cliente estático público de solo lectura Client2
= nuevo cliente

    {

        Id = "78901",
        Secret = "aasdadasdef",
        RedirectUrl =
Paths.ImplicitGrantCallBackPath

    } ;

}

cliente de clase pública

{

    Id de cadena pública {get; colocar; }

    cadena pública Secret {get; colocar; }

    cadena pública RedirectUrl {get; colocar; }

}

```

26. En la ventana del Explorador de soluciones, haga clic en el nombre de su proyecto y presione F4.
27. En la ventana de propiedades de su proyecto, asegúrese de que el valor de SSL habilitado esté establecido en Verdadero.
28. Copie el valor de la configuración de URL SSL.

29. Haga clic con el botón derecho en el nombre del proyecto y haga clic en el elemento del menú Propiedades en la parte inferior del menú contextual.
30. En la pestaña de propiedades del proyecto en Visual Studio, haga clic en el elemento Web en el lado izquierdo de la ventana.
31. En la sección Servidores, pegue el valor de URL de SSL que copió en el paso 28 en el cuadro de texto URL del proyecto.
32. Agregue una nueva clase C # vacía a la carpeta Constants y **asígnelle el nombre Paths.cs**. Puede repetir los pasos del 20 al 24 para crear una nueva clase de C #.
33. Reemplace el contenido del archivo Paths.cs con el código que se muestra en el [Listado 3-11](#).
34. Pegue el valor de la URL SSL que copió en el paso 28 en las siguientes constantes:
  0. **AuthorizationServerBaseAddress**
  1. **ResourceServerBaseAddress**
  2. **ImplicitGrantCallBackPath** Asegúrese de no eliminar la parte URI. Esta constante debería verse como<SSL URL>/Home/SignIn.
  3. **AuthorizeCodeCallBackPath** Asegúrese de no eliminar la parte URI. Esta constante debería verse como<SSL URL>/Manage.

#### **Listado 3-11** Paths.cs

[Haga clic aquí para ver la imagen del código](#)

---

```
// C#. ASP.NET.  
  
espacio de nombres <YOUR_PROJECT'S_NAME>.Constants  
  
{  
  
    rutas de clases públicas  
  
    {  
  
        public const string  
AuthorizationServerBaseAddress = "https://localhost:  
44317";
```

```

        public const string ResourceServerBaseAddress =
"https://localhost: 44317";

        cadena const pública ImplicitGrantCallBackPath =
"https://localhost: 44317 / Inicio / Iniciar
sesión";

        public const string AuthorizeCodeCallBackPath =
"https://localhost: 44317 / Manage";

        public const string AuthorizePath = "/ OAuth /
Authorize";

        public const string TokenPath = "/ OAuth /
Token";

        public const string LoginPath = "/ Cuenta /
Inicio de sesión";

        public const string LogoutPath = "/ Cuenta /
Cerrar sesión";

        public const string MePath = "/ api / Me";

    }

}

```

En este punto, debe crear el controlador de API que administra las solicitudes al punto final Authorize y Token. Cuando configuró el servidor de autenticación, usó el siguiente fragmento de código para configurar los puntos finales que el servidor usa para atender las solicitudes de OAuth:

[Haga clic aquí para ver la imagen del código](#)

```

app.UseOAuthAuthorizationServer (nuevo
OAuthAuthorizationServerOptions

{
    AuthorizeEndpointPath = nueva PathString
(Paths.AuthorizePath),

```

```
    TokenEndpointPath = nueva PathString  
(Paths.TokenPath),
```

Si revisa el valor de los parámetros `AuthorizePath` y `TokenPath` en su `Paths` clase, puede ver que sus valores son `/OAuth/Authorize` y `/OAuth/Token`, respectivamente. Ahora, debe crear el controlador que gestiona las solicitudes a estos puntos finales.

35. En la ventana del Explorador de soluciones, haga clic con el botón derecho en las carpetas Controladores de su proyecto y luego elija Agregar> Controlador.
36. En la ventana Add Scaffold, elija MVC 5 Controller - Empty.
37. Haga clic en el botón Agregar.
38. En la ventana Agregar controlador, escriba **OAuthController**.
39. Abra el archivo `OAuthController.cs` y reemplace el contenido del archivo con el código que se muestra en el [Listado 3-12](#).

**Listado 3-12** OAuthController.cs

[Haga clic aquí para ver la imagen del código](#)

---

```
// C#. ASP.NET.  
  
utilizando System.Security.Claims;  
  
usando System.Web;  
  
usando System.Web.Mvc;  
  
  
espacio de nombres <your_project's_name>.Controllers  
{  
  
    clase pública OAuthController: Controlador  
    {  
  
        // OBTENER: OAuth / Authorize
```

```

    Public ActionResult Authorize ()
    {
        si (Response.StatusCode! = 200)
        {
            return View ("AuthorizeError");
        }

        var autenticación =
HttpContext.GetOwinContext ().Autenticación;

        var ticket =
authentication.AuthenticateAsync ("ApplicationCookie").Resultado;

        var identidad = ticket! = null?
ticket.Identity: nulo;

        si (identidad == nulo)
        {
            authentication.Challenge
("ApplicationCookie");

            return new HttpUnauthorizedResult ();
        }

        var alcances = (Request.QueryString.Get
("alcance") ?? "") .Split ('');

        if (Request.HttpMethod == "POST")
        {

```

```
        if (! string.IsNullOrEmpty
(Request.Form.Get ("submit.Grant")))

        {

            identidad = nueva ClaimsIdentity
(identity.Cclaims, "Bearer", identity.

NameClaimType,
identity.RoleClaimType);

        foreach (alcance var en alcances)

        {

            identity.AddClaim (new Claim
("urn: oauth: scope", scope));

        }

        autenticación.SignIn (identidad);

    }

    if (! string.IsNullOrEmpty
(Request.Form.Get ("submit.Login")))

    {

        authentication.SignOut
("ApplicationCookie");

        authentication.Challenge
("ApplicationCookie");

        return new HttpUnauthorizedResult
();

    }

}
```

```
        volver Ver ();
    }
}
```

40. En el Explorador de soluciones, haga clic con el botón derecho en Vistas> OAuth y luego seleccione Agregar> Ver.

41. En la ventana Agregar vista, en el campo Nombre de vista, escriba **Autorizar**.

42. Haga clic en Agregar.

43. Reemplace el contenido del archivo Authorize.cshtml con el código que se muestra en el Listado 3-13:

**Listado 3-13** Authorize.cshtml

Haga clic aquí para ver la imagen del código

---

```
// C#. ASP.NET.
```

```
@ {
```

```
    ViewBag.Title = "Autorizar";
```

```
}
```

```
@using System.Security.Claims
```

```
@usando System.Web
```

```
@ {
```

```
    var autenticación = Context.GetOwinContext ().Autenticación;
```

```
    var ticket = authentication.AuthenticateAsync ("ApplicationCookie"). Resultado;
```

```
    var identidad = ticket! = null? ticket.Identity:nulo;
```

```

    var alcances = (Request.QueryString.Get ("alcance")
?? "") .Split ('');

}

<!DOCTYPE html>

<html xmlns = "http://www.w3.org/1999/xhtml">

<cabeza>

<title> @ ViewBag.Title </title>

</head>

<cuerpo>

<h1> Servidor de autorización </h1>

<h2> Autorizar OAuth2 </h2>

<método de formulario = "POST">

<p> Hola, @ identity.Name </p>

<p> Una aplicación de terceros desea hacer lo
siguiente en su nombre: </p>

<ul>

@foreach (alcance var en alcances)

{

    <li> @ alcance </li>

}

</ul>

<p>

<input type = "submit" name = "submit.Grant"
value = "Grant" />

```

```

        <input type = "submit" name = "submit.Login"
value = "Iniciar sesión como un usuario diferente" />

    </p>

</form>

</body>

</html>

```

44. Agregue otra vista vacía llamada **AuthorizeError**.

45. Reemplace el contenido del archivo AuthorizeError.cshtml con el código que se muestra en el [Listado 3-14](#):

**Listado 3-14** AuthorizeError.cshtml

[Haga clic aquí para ver la imagen del código](#)

---

```

// C#. ASP.NET.

@ {
    ViewBag.Title = "AuthorizeError";

}

@usando el sistema

@using System.Security.Claims

@usando System.Web

@usando Microsoft.Owin

@ {

    IOwinContext owinContext = Context.GetOwinContext
();

    var error = owinContext.Get <cadena>
("oauth.Error");

    var errorDescription = owinContext.Get <cadena>
("oauth.ErrorDescription");

```

```

        var errorUri = owinContext.Get <cadena>
        ("oauth.ErrorUri");

    }

<!DOCTYPE html>

<html xmlns = "http://www.w3.org/1999/xhtml">

<cabeza>

    <title> @ ViewBag.Title </title>

</head>

<cuerpo>

    <h1> Katana.Sandbox.WebServer </h1>

    <h2> Error de autorización de OAuth2 </h2>

    <p> Error: @error </p>

    <p> @errorDescription </p>

</body>

</html>

```

Este ejemplo solo proporciona una implementación para el punto final Authorize en aras de la simplicidad. Un usuario autorizado en su aplicación necesita otorgar acceso a los recursos en su aplicación explícitamente. Cuando el usuario otorga esos privilegios, la aplicación crea automáticamente un token OAuth en memoria que puede usar para realizar una solicitud a los recursos protegidos. En un escenario del mundo real, este proceso debe separarse en dos puntos finales diferentes: Autorizar y Token. Debe utilizar el punto final Token para crear o actualizar el token de acceso emitido por el servidor de autorización.

Ahora que ha creado y configurado su servidor de autorización, puede crear el servidor de recursos. En este ejemplo, creará el servidor de recursos en la misma aplicación en la que implementó el servidor de autorización. En un escenario del mundo real, puede usar la misma

aplicación o puede usar una aplicación diferente implementada por un servidor diferente o Azure App Service.

1. En la ventana del Explorador de soluciones, haga clic con el botón derecho en la carpeta Controladores en su proyecto y haga clic en Agregar> Controlador.
2. En la ventana Add New Scaffolded Item, seleccione Web API 2 Controller - Empty template.
3. Haga clic en el botón Agregar.
4. En la ventana Agregar controlador, escriba **MeController** y haga clic en el botón Agregar.
5. Reemplace el contenido del archivo MeController.cs con el código que se muestra en el [Listado 3-15](#). Este controlador es bastante simple y solo devuelve la información almacenada en el token que proporcionas al servidor de recursos cuando intentas acceder al recurso.

**Listado 3-15** MeController.cs

[Haga clic aquí para ver la imagen del código](#)

---

```
// C#. ASP.NET.

usando System.Collections.Generic;

utilizando System.Linq;

utilizando System.Security.Claims;

usando System.Web.Http;

espacio de nombres <your_project's_name>.Controllers

{

    [Autorizar]

    clase pública MeController: ApiController

    {
```

```

// OBTENER api / <controller>
public IEnumerable <objeto> Get ()
{
    var identity = User.Identity como
ClaimsIdentity;

    devolver identidad.Claims.Select (c => new
{
    Tipo = c. Tipo,
    Valor = c. Valor
}) ;
}

}

```

6. En la ventana del Explorador de soluciones, en la carpeta App\_Start, cambie el nombre del archivo WebApiConfig.cs a Startup.WebApi.cs.
7. En la ventana de Visual Studio, haga clic en Herramientas> Administrador de paquetes NuGet> Administrar paquetes NuGet para la solución.
8. En la pestaña Administrador de paquetes NuGet, haga clic en Examinar.
9. Escriba **Microsoft asp.net web api owin** y presione Entrar.
10. Haga clic en el paquete Microsoft.AspNet.WebApi.Owin.
11. En el lado derecho de la pestaña NuGet Manager, haga clic en la casilla de verificación junto a su proyecto.
12. Haga clic en el botón Instalar.
13. En la ventana Vista previa de cambios, haga clic en Aceptar.
14. En Aceptación de licencia, haga clic en el botón Acepto.

15. Abra el archivo Startup.WebApi.cs y cambie el contenido del archivo con el contenido que se muestra en el Listado 3-16.

**Listado 3-16** Startup.WebApi.cs

Haga clic aquí para ver la imagen del código

---

```
// C#. ASP.NET.

utilizando Microsoft.Owin.Security.OAuth;
usando Owin;
usando System.Web.Http;

espacio de nombres <your_project's_name>

{
    Inicio de clase parcial pública
    {
        public void ConfigureWebApi (aplicación
IAppBuilder)
        {
            var config = new HttpConfiguration ();
            // Configuración y servicios de API web
            // Configure la API web para utilizar solo
            // la autenticación de token de portador.
            config.SuppressDefaultHostAuthentication ();
            config.Filters.Add (nuevo
HostAuthenticationFilter (OAuthDefaults
.Tipo de autenticación));
        }
    }
}
```

```

    // Rutas de API web

    config.MapHttpAttributeRoutes () ;




    config.Routes.MapHttpRoute (
        nombre: "DefaultApi",
        routeTemplate: "api / {controller} /
{id}",
        valores predeterminados: nuevo {id =
RouteParameter.Optional}

) ;



app.UseWebApi (config);

}

}

```

**16.** Abra el archivo Startup.cs y agregue la siguiente línea al final del método Configuration ():

```
ConfigureWebApi (aplicación);
```

Una vez que haya implementado el servidor de recursos en su aplicación, debería poder realizar solicitudes al servidor de autorización para obtener acceso al recurso publicado por el servidor de recursos. Como vio en el flujo de trabajo de OAuth, debe ser autenticado por el servidor de autorización antes de poder obtener un token de acceso. Esto significa que debe iniciar sesión en la aplicación antes de poder realizar solicitudes al punto final / OAuth / Authorize.

Ahora puede crear su aplicación cliente que realiza solicitudes al servidor de autorización y al servidor de recursos. Esa aplicación cliente puede ser la misma aplicación que usó para implementar los servidores de

autorización y recursos. Va a modificar la plantilla MVC predeterminada para realizar solicitudes a los servidores de autorización y recursos.

1. En la ventana de Visual Studio, haga clic en Herramientas> Administrador de paquetes NuGet> Administrar paquetes NuGet para la solución.
2. En la pestaña Administrador de paquetes NuGet, haga clic en Examinar.
3. Escriba **DotNetOpenAuth.OAuth2.Client** y presione Entrar.
4. Haga clic en el paquete DotNetOpenAuth.OAuth2.Client. Este paquete NuGet facilita la interacción con los servidores OAuth.
5. En el lado derecho de la pestaña NuGet Manager, haga clic en la casilla de verificación junto a su proyecto.
6. Haga clic en el botón Instalar.
7. En la ventana Vista previa de cambios, haga clic en Aceptar.
8. Abra el archivo ManageController.cs.
9. Agregue las siguientes instrucciones using al archivo ManageController.cs:
  1. **usando el sistema;**
  2. **utilizando System.Linq;**
  3. **usando System.Threading.Tasks;**
  4. **usando System.Web;**
  5. **usando System.Web.Mvc;**
  6. **utilizando Microsoft.AspNet.Identity;**
  7. **utilizando Microsoft.AspNet.Identity.Owin;**
  8. **utilizando Microsoft.Owin.Security;**
  9. **utilizando AuthorizationServer.Models;**
  10. **utilizando AuthorizationServer.Constants;**
  11. **utilizando DotNetOpenAuth.OAuth2;**
  12. **usando System.Net.Http;**
10. Reemplace el `Index()` método con el código que se muestra en el [Listado 3-17](#).

**Listado 3-17** Método de índice en ManageController.cs

Haga clic aquí para ver la imagen del código

---

// C#. ASP.NET.

```
Public async Task <ActionResult> Índice (mensaje  
ManageMessageId?)  
  
{  
  
    ViewBag.StatusMessage =  
  
        mensaje ==  
        ManageMessageId.ChangePasswordSuccess? "Tu contraseña ha  
        sido  
  
        cambió."  
  
        : mensaje ==  
        ManageMessageId.SetPasswordSuccess? "Tu contraseña ha  
        sido  
  
        colocar."  
  
        : mensaje ==  
        ManageMessageId.SetTwoFactorSuccess? "Tu doble factor  
  
        se ha establecido el proveedor  
        de autenticación ".  
  
        : mensaje == ManageMessageId.Error? "Se ha  
        producido un error."  
  
        : mensaje ==  
        ManageMessageId.AddPhoneSuccess? "Tu número de teléfono  
        era  
  
        adicional."  
  
        : mensaje ==  
        ManageMessageId.RemovePhoneSuccess? "Tu número de  
        teléfono era  
  
        remoto."
```

```
: "";  
  
    var userId = User.Identity.GetUserId ();  
  
    var model = nuevo IndexViewModel  
  
    {  
  
        HasPassword = HasPassword (),  
  
        PhoneNumber = espera  
        UserManager.GetPhoneNumberAsync (userId),  
  
        TwoFactor = espera  
        UserManager.GetTwoFactorEnabledAsync (userId),  
  
        Inicios de sesión = espera  
        UserManager.GetLoginsAsync (userId),  
  
        BrowserRemedered = espera  
        AuthenticationManager.  
  
        TwoFactorBrowserRemederedAsync (userId)  
  
    };  
  
    ViewBag.AccessToken = Request.Form  
    ["AccessToken"] ?? "";  
  
    ViewBag.RefreshToken = Request.Form  
    ["RefreshToken"] ?? "";  
  
    ViewBag.Action = "";  
  
    ViewBag.ApiResponse = "";  
  
    InitializeWebServerClient ();
```

```
        var accessToken = Request.Form  
        ["AccessToken"];  
  
        if (string.IsNullOrEmpty (accessToken))  
  
        {  
  
            var AuthorizationState =  
            _webServerClient.ProcessUserAuthorization (  
                Pedido);  
  
            si (estado de autorización! = nulo)  
  
            {  
  
                ViewBag.AccessToken =  
                AuthorizationState.AccessToken;  
  
                ViewBag.RefreshToken =  
                AuthorizationState.RefreshToken;  
  
                ViewBag.Action = Request.Path;  
  
            }  
  
        }  
  
        if (! string.IsNullOrEmpty (Request.Form.Get  
        ("submit.Authorize")))  
  
        {  
  
            var userAuthorization =  
            _webServerClient.PrepareRequestUserAuthorization (  
                nuevo [] {"biografía", "notas"});  
  
            userAuthorization.Send (HttpContext);  
        }  
    }  
}
```

```
        Response.End () ;

    }

    else if (! string.IsNullOrEmpty
(Request.Form.Get ("submit.Refresh")))

    {

        var state = new AuthorizationState

        {

            AccessToken = Request.Form
["AccessToken"] ,

            RefreshToken = Request.Form
["RefreshToken"]

        } ;

        if
(_webServerClient.RefreshAuthorization (estado))

        {

            ViewBag.AccessToken =
state.AccessToken;

            ViewBag.RefreshToken =
state.RefreshToken;

        }

    }

    else if (! string.IsNullOrEmpty
(Request.Form.Get ("submit.CallApi")))

    {

        var resourceServerUri = new Uri
(Paths.ResourceServerBaseAddress);
```

```

        var cliente = nuevo HttpClient
(_webServerClient.CreateAuthorizingHandler

                    (accessToken));

        var body = client.GetStringAsync (nuevo
Uri (resourceServerUri,

                    Paths.MePath)). Resultado;

ViewBag.ApiResponse = cuerpo;

}

volver Vista (modelo);

}

```

11. Agregue la siguiente propiedad a la clase ManageController:  
[Haga clic aquí para ver la imagen del código](#)

```
privado WebServerClient _webServerClient;
```

12. Agregue el siguiente método auxiliar a la clase  
ManageController:

[Haga clic aquí para ver la imagen del código](#)

```

vacío privado InitializeWebServerClient ()

{

    var AuthorizationServerUri = new Uri
(Paths.AuthorizationServerBaseAddress);

    var AuthorizationServer = new AuthorizationServerDescription

    {

        AuthorizationEndpoint = nuevo Uri (AuthorizationServerUri,
Paths.AuthorizePath),

        TokenEndpoint = nuevo Uri (AuthorizationServerUri,
Paths.TokenPath)

```

```
};

_webServerClient = new WebServerClient (AuthorizationServer,
Clients.Client1.Id,
Clientes.Cliente1.Secreto);

}
```

13. En el `Application_Start()` método del archivo Global.asax.cs, agregue la siguiente línea:

[Haga clic aquí para ver la imagen del código](#)

```
AntiForgeryConfig.SuppressXFrameOptionsHeader = true;
```

14. Agregue la siguiente instrucción `using` al archivo Global.asax.cs:

```
utilizando System.Web.Helpers;
```

15. En la ventana del Explorador de soluciones, haga clic en Vistas> Administrar> Index.cshtml.

16. Agregue el código que se muestra en el [Listado 3-18](#) después de la sección Autenticación de dos factores en el archivo Index.cshtml.

**Listado 3-18** Sección de concesión del código de autorización

[Haga clic aquí para ver la imagen del código](#)

---

```
// C#. ASP.NET.
```

```
<dt> Cliente de concesión de código de autorización: </dt>
```

```
<dd>
```

```
    <form id = "form1" action = "@ ViewBag.Action"
method = "POST">
```

```
        <div>
```

```
            Token de acceso <br />
```

```
        <input id = "AccessToken" name =
"AccessToken" width = "604" type = "text"
                value = "@
ViewBag.AccessToken" />

        <input id = "Authorize" name =
"submit.Authorize" value = "Authorize"
                type = "enviar" />
<br />
<br />
        Actualizar Token <br />
        <input id = "RefreshToken" name =
"RefreshToken" width = "604"
                type = "text" value = "@
ViewBag.RefreshToken" />
        <input id = "Refresh" name =
"submit.Refresh" value = "Refresh"
                type = "enviar" />
<br />
<br />
        <input id = "CallApi" name =
"submit.CallApi" value = "Acceso protegido
API de recursos "type =" submit "/>
</div>
<div>
    @ ViewBag.ApiResponse
</div>
```

```
</form>  
</dd>
```

En este punto, su aplicación de ejemplo está lista para probar la implementación de los diferentes actores que participan en el flujo de trabajo de OAuth. Los siguientes pasos muestran cómo probar su implementación de OAuth para asegurarse de que funcione correctamente:

1. Abra el proyecto de ejemplo en Visual Studio y presione F5 para ejecutar el proyecto.
2. Debería abrirse una nueva ventana del navegador web con su aplicación web. Haga clic en el enlace Registrarse ubicado en la esquina superior izquierda de la página.
3. En la página de registro, agregue una dirección de correo electrónico y una contraseña y confirme la contraseña. Luego haga clic en el botón Registrarse. Utilizará este usuario para otorgar privilegios al cliente OAuth para realizar solicitudes al punto final / OAuth / Me.
4. Una vez que haya registrado el nuevo usuario, se iniciará sesión automáticamente y se le redirigirá a la página de inicio.
5. En la página de inicio, haga clic en el enlace de correo electrónico de su usuario en la esquina superior izquierda de la página de inicio.
6. En la página Administrar, haga clic en el botón Autorizar, que lo redirige a la página Servidor de autorización.
7. En la página del servidor de autorización, revise la información proporcionada y haga clic en el botón Otorgar. Después de otorgar acceso a la aplicación cliente OAuth, obtiene el token de acceso y actualización que se muestra en la Figura 3-3, que es necesario para realizar solicitudes al servidor de recursos.

<b>Authorization Code</b>	<b>Access Token</b>
<b>Grant Client:</b>	aHdhc49E4vcvY9qtFWUZb! <input type="button" value="Authorize"/>
<b>Refresh Token</b>	
Hcg51F5zu-JzKEJm2jUEDN <input type="button" value="Refresh"/>	
<input type="button" value="Access Protected Resource API"/>	

**Figura 3-3** Token de actualización y acceso de OAuth

8. Haga clic en Access Protected Resource API para realizar una solicitud al punto final / OAuth / Me. Debe obtener toda la información almacenada en la declaración de identidad que utiliza para realizar esta solicitud, incluidas las notas y la biografía de los ámbitos.

#### *¿Necesita más revisión? Servidor de autorización Oauth*

En este ejemplo, revisó cómo implementar la autorización y el servidor de recursos, el cliente y el propietario del recurso en la misma aplicación web. Aunque este es un escenario válido, normalmente encontrará que estos roles se implementan en una aplicación separada. El código que revisamos se basa en el ejemplo explicado en el artículo de Microsoft Docs en <https://docs.microsoft.com/en-us/aspnet/aspnet/overview/owin-and-katana/owin-oauth-20-authorization-servidor>. En ese artículo, puede revisar cómo implementar cada función en aplicaciones independientes.



#### Sugerencia para el examen

Cuando trabaje con la autenticación OAuth2, recuerde que no necesita almacenar la información de nombre de usuario y contraseña en su sistema. Puede delegar esa tarea en servidores de autenticación especializados. Una vez que el usuario se ha autenticado correctamente, el servidor de autenticación envía un token de acceso que puede utilizar para confirmar la identidad del cliente. Este token de acceso debe actualizarse una vez que caduque. OAuth2 puede usar un token de actualización para solicitar un nuevo token de acceso sin volver a pedirle al usuario sus credenciales.

## *Cree e implemente firmas de acceso compartido*

Hasta ahora, todos los mecanismos de protección y control de acceso que revisamos en este apartado tenían que ver con proteger la información gestionada directamente por tu aplicación. Estos mecanismos son buenos si su aplicación administra y presenta la información al usuario. Aún así, no son apropiados para otros servicios que también pueden almacenar información administrada por su aplicación. Si su aplicación usa cuentas de Azure Storage para almacenar algunos informes, imágenes o documentos en una tabla, y desea otorgar acceso a terceros a esa información, ninguno de los mecanismos revisados anteriormente es apropiado para este escenario.

Cuando trabaja con almacenamiento, necesita controlar quién y cuánto tiempo un proceso, persona o aplicación puede acceder a sus datos. Azure Storage le permite controlar este acceso en función de varios niveles de protección:

- **Autorización de clave compartida** Utilice una de las dos claves de acceso configuradas en el nivel de la cuenta de Azure Storage para construir la solicitud correcta para acceder a los recursos de la cuenta de Azure Storage. Debe utilizar el encabezado de autorización para utilizar la clave de acceso en su solicitud. La clave de acceso proporciona acceso a toda la cuenta de Azure Storage y a todos sus contenedores, como blobs, archivos, colas y tablas. Puede considerar que las claves de la cuenta de Azure Storage son como la contraseña raíz de la cuenta de Azure Storage.
- **Firmas de acceso compartido** Utilice las firmas de acceso compartido (SAS) para limitar el acceso a contenedores específicos dentro de la cuenta de almacenamiento. La ventaja de usar SAS es que no es necesario compartir la clave de acceso de la cuenta de Azure Storage. También puede configurar un mayor nivel de granularidad al configurar el acceso a sus datos.

El inconveniente de usar claves de acceso compartido es que si se expone cualquiera de las dos claves de acceso, también se exponen la cuenta de Azure Storage y todos los contenedores y datos de la cuenta de Azure Storage. Las claves de acceso también nos permiten crear o eliminar elementos en la cuenta de Azure Storage.

Las firmas de acceso compartido le brindan un mecanismo para compartir el acceso con clientes o aplicaciones a su cuenta de Azure Storage sin exponer toda la cuenta. Puede configurar cada SAS con un nivel de acceso diferente a cada uno de los siguientes:

- **Servicios** Puede configurar SAS para otorgar acceso solo a los servicios que necesita, como blob, archivo, cola o tabla.
- **Tipos de recursos** Puede configurar el acceso a un servicio, contenedor u objeto. Para el servicio Blob, esto significa que puede configurar el acceso a las llamadas a la API en el nivel de servicio, como los contenedores de listas. Si configura el token SAS en el nivel del contenedor, puede realizar llamadas a la API, como obtener o configurar metadatos del contenedor o crear nuevos blobs. Si decide configurar el acceso a nivel de objeto, puede realizar llamadas a la API, como crear o actualizar blobs en el contenedor.
- **Permisos** Configure la acción o acciones que el usuario puede realizar en los recursos y servicios configurados.
- **Fecha de vencimiento** Puede configurar el período durante el cual el SAS configurado es válido para acceder a los datos.
- **Direcciones IP** Puede configurar una única dirección IP o un rango de direcciones IP a las que se les permite acceder a su almacenamiento.
- **Protocolos** Puede configurar si el acceso a su almacenamiento se realiza mediante protocolos solo HTTPS o HTTP y HTTPS. No puede otorgar acceso al protocolo solo HTTP.

Azure Storage usa los valores de los parámetros anteriores para construir la firma que otorga acceso a su almacenamiento. Puede configurar tres tipos diferentes de SAS:

- **SAS de delegación de usuarios** Este tipo de SAS se aplica solo a Blob Storage. Utiliza una cuenta de usuario de Azure Active Directory para proteger el token SAS
- **Cuenta SAS** Cuenta SAS controla el acceso a toda la Cuenta de almacenamiento. También puede controlar el acceso a las operaciones a nivel de servicio, como obtener estadísticas del servicio o obtener o establecer propiedades del servicio. Debe

utilizar la clave de la cuenta de almacenamiento para proteger este tipo de SAS.

- **Service SAS** Service SAS delega el acceso solo a servicios específicos dentro de la cuenta de almacenamiento. Debe utilizar la clave de la cuenta de almacenamiento para proteger este tipo de SAS.

Independientemente del tipo de SAS, debe construir un token SAS para acceder. Agrega este token SAS a la URL que usa para acceder a su recurso de almacenamiento. Uno de los parámetros de un token SAS es la firma. El servicio Cuenta de almacenamiento de Azure usa esta firma para autorizar el acceso a los recursos de almacenamiento. La forma de crear esta firma depende del tipo de SAS que esté utilizando.

Para SAS de delegación de usuarios, debe usar una clave de delegación de usuarios creada con las credenciales de Azure Active Directory (Azure AD). El usuario utilizado para crear esta clave de delegación debe haber otorgado

el `Microsoft.Storage/storageAccounts/blobServices/generateUserDelegationKey` y `action` permiso de Control de acceso de la base de roles. Revisamos la autorización de control de acceso basada en roles en “[Controlar el acceso a los recursos mediante el uso de controles de acceso basados en roles \(RBAC\)](#)” más adelante en este capítulo.

Para el servicio o la cuenta SAS, debe usar la clave de la cuenta de almacenamiento de Azure para crear la firma que debe incluir en el token SAS. Para construir el URI de SAS para un SAS de cuenta, debe utilizar los parámetros que se muestran en la [Tabla 3-1](#).

**Tabla 3-1** Parámetros de URI SAS de cuenta

Nombre del parámetro	Parámetro URI	Requerido	Descripción
api-version	api-version	NO	Puede configurar la versión de la API de almacenamiento que procesa su solicitud.
SignedVersion	sv	SÍ	Establece la versión del servicio de almacenamiento firmado que se usa para autenticar su solicitud. La versión debería ser 2015-04-05 o posterior.

<b>Nombre del parámetro</b>	<b>Parámetro URI</b>	<b>Requerido</b>	<b>Descripción</b>
SignedServices	ss	SÍ	<p>Establece los servicios a los que concede otorgar acceso a más de un servicio como valores permitidos:</p> <ul style="list-style-type: none"> <li>• <b>Blob</b> Debe usar el valor (b) SAS.</li> <li>• <b>Cola</b> Debe utilizar el valor (c) SAS.</li> <li>• <b>Tabla</b> Debe utilizar el valor (t) SAS.</li> <li>• <b>Archivo</b> Debe utilizar el valor (f) SAS.</li> </ul>
SignedResourceTypes	srt	SÍ	<p>Establece el tipo de recurso al que concede configurar más de un tipo de recurso combinando más de uno de los valores posibles:</p> <ul style="list-style-type: none"> <li>• <b>Servicio</b> Debe utilizar los valores (s) SAS.</li> <li>• <b>Contenedor</b> Debe utilizar el valor (c) SAS.</li> <li>• <b>Objeto</b> Debe utilizar el valor (o) SAS.</li> </ul>
SignedPermission	sp	SÍ	<p>Configura los permisos que otorga a los usuarios y servicios configurados en parámetros autorizados. Aunque todos los permisos se aplican a todos los usuarios y servicios. La siguiente lista solo muestra los permisos que se aplican al servicio Blob:</p> <ul style="list-style-type: none"> <li>• <b>Leer</b> Debe utilizar el valor (r) SAS.</li> <li>• <b>Escribir</b> Debe utilizar el valor (w) SAS.</li> <li>• <b>Eliminar</b> Debe utilizar el valor (d) SAS.</li> <li>• <b>Lista</b> Debe utilizar el valor (l) SAS.</li> <li>• <b>Agregar</b> Debe usar el valor (a) SAS.</li> <li>• <b>Crear</b> Debe utilizar el valor (c) SAS.</li> </ul>

<b>Nombre del parámetro</b>	<b>Parámetro URI</b>	<b>Requerido</b>	<b>Descripción</b>
			Si establece un permiso que es significativo para el servicio o tipo de recurso que no establece en los parámetros anteriores, el permiso se ignora silenciosamente.
SignedStart	st	NO	Establece la fecha y la hora en las que el token será válido. Debe expresarse en UTC utilizando ISO 8601: <ul style="list-style-type: none"> <li>• AAAA-MM-DD</li> <li>• AAAA-MM-DDThh:mmTZD</li> <li>• AAAA-MM-DDThh:mm:ssTZD</li> </ul>
SignedExpiry	se	SÍ	Establece la fecha y la hora en las que el token dejará de ser válido. Debe expresarse en UTC utilizando ISO 8601.
SignedIP	sip	NO	Establece la IP o el rango de direcciones de origen que el servicio de almacenamiento acepta solo si se utilizan rangos de IP, los límites se incluyen.
SignedProtocol	spr	NO	Establece el protocolo permitido para solicitudes. Los valores correctos son <ul style="list-style-type: none"> <li>• Solo HTTPS (https)</li> <li>• HTTP y HTTPS (https, http)</li> </ul>
Signature	sig	SÍ	Esta es una cadena calculada con HMAC-SHA256 que se codifica con Base64 que la API usa para firmar la solicitud. Calcula la firma en función de los parámetros que proporcionó en el URI de SAS. Esta firma es válida para procesar su solicitud.

Utilice el siguiente procedimiento para construir y probar su propio token SAS de cuenta:

1. Inicie sesión en el portal de gestión (<http://portal.azure.com>).

2. En el cuadro de búsqueda en la parte superior de Azure Portal, escriba el nombre de su cuenta de almacenamiento.
3. En la hoja Cuenta de almacenamiento, haga clic en Firma de acceso compartido en la sección Configuración.
4. En el panel Firma de acceso compartido, anule la selección de las casillas de verificación Archivo, Tabla y Cola en Servicios permitidos, como se muestra en la [Figura 3-4](#). Deje seleccionada la casilla de verificación Blob.



**Figura 3-4** Configuración de la política SAS de la cuenta

5. Asegúrese de que todas las opciones en Tipos de recursos permitidos y Permisos permitidos estén marcadas, como se muestra en la [Figura 3-4](#).
6. En la sección Fecha / hora de inicio y vencimiento, establezca una fecha para la fecha y hora de inicio y finalización durante las cuales la cuenta de almacenamiento de Azure acepta solicitudes con este token.
7. Asegúrese de que las direcciones IP permitidas no tengan ningún valor en el cuadro de texto y que solo HTTPS esté seleccionado en la sección Protocolos permitidos.
8. En el menú desplegable Clave de firma, asegúrese de haber seleccionado el valor Clave1.
9. Haga clic en el botón Generar SAS y cadena de conexión en la parte inferior del panel.
10. Copie la URL SAS de Blob Service. Ahora puede probar su token SAS con una herramienta como Postman, curl, un navegador web o Microsoft Azure Storage Explorer.

11. Abra el Explorador de almacenamiento de Microsoft Azure. Si no tiene esta herramienta instalada, puede descargarla desde <https://azure.microsoft.com/en-us/features/storage-explorer/>.
12. En la ventana del Explorador de Microsoft Azure Storage, en el lado izquierdo de la ventana, haga clic en el botón con un icono de enchufe. Este botón abre el cuadro de diálogo Conectar.
13. En el cuadro de diálogo Conectar, seleccione la opción Usar un URI de firma de acceso compartido (SAS).
14. Haga clic en el botón Siguiente en la parte inferior del cuadro de diálogo.
15. En Adjuntar con SAS URI, escriba un nombre para su conexión en el cuadro de texto Nombre para mostrar.
16. En el cuadro de texto URI, pegue la URL que copió en el paso 10.
17. Haga clic en el botón Siguiente.
18. Haga clic en el botón Conectar.

Una vez que se crea la conexión, debería poder ver su servicio Blob Storage y crear nuevos contenedores o blobs dentro de los contenedores.

Si necesita restringir el acceso a sus recursos y limitarlo solo a tablas o entidades, puede crear una SAS de servicio. Este tipo de token SAS es bastante similar a un SAS de cuenta; debe crear un URI que anexar a la URL que usa para solicitar su servicio Blob Storage. La cuenta y el servicio SAS comparten la mayoría de los parámetros de URI, aunque algunos parámetros son específicos del servicio y debe tenerlos en cuenta al crear su token de servicio SAS. La Tabla 3-2 muestra los parámetros que debe establecer para crear un SAS de Blob Service. Otros servicios de Azure Storage requieren diferentes parámetros.

**Tabla 3-2** Parámetros de URI SAS del servicio BLOB

Nombre del parámetro	Parámetro URI	Requerido	Descripción
SignedVersion	sv	SÍ	Establece la versión del servicio de almacenamiento que se usa para autenticar su solicitud. Esta debe ser 2015-04-05 o posterior.

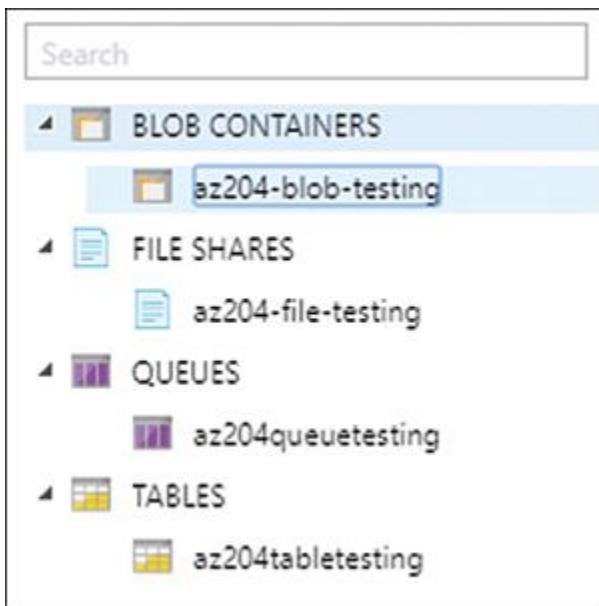
<b>Nombre del parámetro</b>	<b>Parámetro URI</b>	<b>Requerido</b>	<b>Descripción</b>
SignedResource	sr	SÍ	Establece el tipo de recurso compartido: <ul style="list-style-type: none"><li>• <b>Blob</b> Debe usar el valor (b) en</li><li>• <b>Contenedor</b> Debe utilizar el valor de SAS.</li></ul>
SignedPermission	sp	SÍ	Configura los permisos que otorga al recurso compartido. Debe omitir este parámetro si usa una Política de acceso almacenado.
SignedStart	st	NO	Establece la fecha y la hora en las que el token será válido. Debe expresarse en UTC utilizando el formato ISO 8601: <ul style="list-style-type: none"><li>• AAAA-MM-DD</li><li>• AAAA-MM-DDThh:mmTZD</li><li>• AAAA-MM-DDThh:mm:ssTZD</li></ul> Si usa una versión de API 2012-02-12 o posterior, la diferencia entre signedstart y signedexpiry no debe ser mayor a una hora a menos que esté usando un contenedor.
SignedExpiry	se	SÍ	Establece la fecha y la hora en las que el token dejará de ser válido. Debe expresarse en UTC utilizando el formato ISO 8601. Debe omitir este parámetro si decide utilizar una política de acceso almacenado.
SignedIP	sip	NO	Establece la IP o el rango de direcciones IP que el servicio de almacenamiento acepta solicitudes. Si las IP utilizan rangos de IP, los límites se incluyen. Debe omitir este parámetro si decide utilizar una política de acceso almacenado.
SignedProtocol	spr	NO	Establece el protocolo permitido para solicitudes. Los valores válidos son <ul style="list-style-type: none"><li>• Solo HTTPS (https)</li></ul>

<b>Nombre del parámetro</b>	<b>Parámetro URI</b>	<b>Requerido</b>	<b>Descripción</b>
			<ul style="list-style-type: none"> <li>• HTTP y HTTPS (https, http)</li> </ul>
SignedIdentifier	Si	NO	Relaciona el URI de SAS que está construyendo la política de acceso almacenado en su cuenta de almacenamiento. El uso de políticas de acceso proporciona un mayor nivel de seguridad.
Signature	sig	SÍ	Esta es una cadena computada HMAC-SHA256 usando Base64 que la API usa para autenticar la solicitud. Calcula la firma en función de los datos que proporcionó en el URI de SAS. Esta firma debe ser enviada con la solicitud para que la API la procese.
Cache-Control	rscc	NO	Requiere la versión (sv) establecida en 2015-02-28 o posterior para el servicio Blob y 2015-02-28 o posterior para el servicio de archivos.
Content-Disposition	rscd	NO	Requiere la versión (sv) establecida en 2015-02-28 o posterior para el servicio Blob y 2015-02-28 o posterior para el servicio de archivos.
Content-Encoding	rsce	NO	Requiere la versión (sv) establecida en 2015-02-28 o posterior para el servicio Blob y 2015-02-28 o posterior para el servicio de archivos.
Content-Language	rscl	NO	Requiere la versión (sv) establecida en 2015-02-28 o posterior para el servicio Blob y 2015-02-28 o posterior para el servicio de archivos.
Content-Type	rsct	NO	Requiere la versión (sv) establecida en 2015-02-28 o posterior para el servicio Blob y 2015-02-28 o posterior para el servicio de archivos.

El siguiente ejemplo muestra cómo crear una firma de acceso compartido para un contenedor de blobs. Este token SAS otorga acceso al contenedor de blobs y a todos los blobs almacenados dentro del contenedor de blobs.

blobs. Para este ejemplo, necesita una cuenta de almacenamiento de Azure con un contenedor de blobs configurado con el nivel de acceso privado:

1. Abra Azure Portal (<https://portal.azure.com>).
2. En el cuadro de texto de búsqueda en la parte superior de Azure Portal, escriba el nombre de su cuenta de Azure Storage.
3. En la lista Resultados, haga clic en el nombre de su cuenta de Azure Storage.
4. En la hoja de su cuenta de Azure Storage, haga clic en StorageExplorer (vista previa) en el menú de navegación en el lado izquierdo de la hoja.
5. En el panel Explorador de Storage (vista previa) que se muestra en la [Figura 3-5](#), expanda el nodo Blob Containers y haga clic con el botón derecho en el contenedor, al que debe otorgar acceso.



**Figura 3-5** Servicios de almacenamiento en el Explorador de almacenamiento (vista previa)

6. En el menú contextual sobre su contenedor de blobs, haga clic en Obtener firma de acceso compartido.
7. En el panel de Firma de acceso compartido que se muestra en la [Figura 3-6](#), configure la Hora de inicio, la Hora de vencimiento y los Permisos que desea otorgar al token SAS.



**Figura 3-6** Creación de una firma de acceso compartido

8. Haga clic en el botón Crear en la parte inferior del panel.
9. En el panel Firma de acceso compartido, copie la URL del SAS recién generado. Puede compartir esta URL de SAS con cualquier tercero que necesite acceder a este blob específico.

Puede usar estos mismos pasos para crear un token SAS para un solo blob en un contenedor. Simplemente navegue usando el Explorador de Storage hasta el blob que desea compartir, haga clic con el botón derecho en el blob y haga clic en Obtener firma de acceso compartido en el menú contextual.

Como puede imaginar, un inconveniente de utilizar este enfoque es que cualquier persona que tenga acceso a la URL de SAS puede acceder a la información protegida por ese SAS. Puede mejorar la seguridad de los tokens SAS creando una Política de acceso almacenado y adjuntando la política al token SAS. Políticas de acceso almacenadas le permite definir políticas de acceso que están asociadas y almacenadas con la tabla que desea proteger. Cuando define una Política de acceso almacenado, proporciona un identificador a la política. Luego, usa este identificador cuando construye el token Service SAS. Debe incluir este identificador cuando construya la firma que autentica el token y es parte del propio SAS.

La ventaja de utilizar una Política de acceso almacenado es que usted define y controla la validez y el vencimiento de la política sin necesidad de modificar el token de SAS del servicio. El uso de una Política de acceso almacenado también mejora la seguridad al ocultar los detalles de la Política de acceso al usuario, ya que solo proporciona el nombre de la Política de acceso almacenado. Puede asociar hasta cinco políticas de acceso almacenadas diferentes.

**¿Necesita más revisión? Trabajar con políticas de acceso almacenadas**

Trabajar con políticas de acceso almacenado es similar a trabajar con políticas de acceso ad-hoc. Puede revisar cómo trabajar con las políticas de acceso almacenado revisando los siguientes artículos:

- <https://docs.microsoft.com/en-us/rest/api/storageservices/define-stored-access-policy>
- <https://docs.microsoft.com/en-us/azure/storage/common/storage-stored-access-policy-define-dotnet>

El siguiente ejemplo muestra cómo crear un token SAS de delegación de usuarios mediante una aplicación de consola .NET Core:

1. Abra Visual Studio Code y cree una carpeta para su proyecto.
2. En la ventana de código de Visual Studio, abra una nueva terminal.
3. Utilice el siguiente comando para crear un nuevo proyecto de consola:

```
nueva consola dotnet
```

4. Utilice el siguiente comando para instalar paquetes NuGet:

[Haga clic aquí para ver la imagen del código](#)

```
dotnet agregar paquete <NuGet_package_name>
```

5. Instale los siguientes paquetes de NuGet:
  1. Azure.Storage.Blobs
  2. Azure.Identity
6. Abra el archivo Program.cs y reemplace el contenido con el código que se muestra en el [Listado 3-19](#).

**Listado 3-19** Program.cs

Haga clic aquí para ver la imagen del código

---

```
// C#. ASP.NET.

using System;
using Azure.Storage.Blobs;
using Azure.Storage.Blobs.Models;
using Azure.Storage.Sas;
using Azure;
using Azure.Identity;

namespace ch3_1_2
{
    class Program
    {
        static void Main (string[] arguments)
        {
            string storageAccount = "az204testing";

            DateTimeOffset startTimeKey =
                DateTimeOffset.UtcNow;

            DateTimeOffset endTimeKey =
                DateTimeOffset.UtcNow.AddDays (7);

            DateTimeOffset startTimeSAS = startTimeKey;
            DateTimeOffset endTimeSAS = startTimeSAS.AddDays
(1);
```

```
        Uri blobEndpointUri = new Uri ($ "https: //  
{storageAccount} .blob.core.windows.  
                neto");  
  
        var defaultCredentials = new  
DefaultAzureCredential (verdadero);  
  
        BlobServiceClient blobClient = nuevo  
BlobServiceClient (blobEndpointUri,  
                defaultCredentials);  
  
        //Consigue la llave. Vamos a utilizar esta clave  
para crear el SAS.  
  
        UserDelegationKey clave =  
blobClient.GetUserDelegationKey (startTimeKey,  
                endTimeKey);  
  
        System.Console.WriteLine ($ "La clave de usuario  
comienza en: {key.SignedStartsOn}");  
        System.Console.WriteLine ($ "La clave de usuario  
expira el: {key.SignedExpiresOn}");  
        System.Console.WriteLine ($ "Servicio de clave  
de usuario: {key.SignedService}");  
        System.Console.WriteLine ($ "Versión de clave de  
usuario: {key.SignedVersion}");
```

```

    // Necesitamos usar BlobSasBuilder para crear el
    SAS

        BlobSasBuilder blobSasBuilder = new
    BlobSasBuilder ()

    {

        StartsOn = startTimeSAS,
        ExpiresOn = endTimeSAS
    };

    // Establecemos los permisos Crear, Listar,
    Agregar, Leer y Escribir

    blobSasBuilder.SetPermissions ("clarw");

    string sasToken =
blobSasBuilder.ToSasQueryParameters

    (clave, storageAccount) .ToString ();

    System.Console.WriteLine ($ "Token SAS:
{sasToken}");
```

}

}

}

### **Nota Authorizationpermissionmismatch**

Si obtiene una excepción mientras ejecuta el código en el [Listado 3-19](#), y el mensaje de excepción es algo similar a "Esta solicitud no está autorizada para realizar esta operación con este permiso" con el código de error AuthorizationPermissionMismatch, no se preocupe; no hay nada de malo en tu código. Esta excepción ocurre porque el usuario que está utilizando para ejecutar este código no tiene suficientes privilegios para obtener una

clave de delegación de usuario. Puede resolver este problema otorgando los permisos correctos a su usuario. Vamos a revisar cómo otorgar privilegios mediante el uso del control de acceso basado en roles en una sección posterior.

Como puede ver en el fragmento de código en negrita del [Listado 3-19](#), usa el  `GetUserDelegationKey()` método para obtener una clave de delegación de usuario para su cuenta de Azure Storage. El usuario que está utilizando para obtener esta clave debe tener asignado

el `Microsoft.Storage/storageAccounts/blobServices/generateUserDelegationKey/action` permiso; de lo contrario, obtendrá una excepción.

Una vez que tenga su clave de delegación de usuario, use la `BlobSasBuilder` clase para crear un objeto que construya el token SAS por usted. Con la instancia de la `BlobSasBuilder` clase, puede configurar los permisos que necesita para acceder al contenedor. En este caso, utilice el `SetPermission()` método con el parámetro `clarw` que coincide con los permisos que se muestran en la [Tabla 3-1](#). En este ejemplo, debido a que no establecimos ningún nombre de contenedor, obtenemos un token SAS para la cuenta de Azure Blob Storage.

Usando el `ToSasQueryParameters()` método de la `BlobSasBuilder` clase, obtienes el token SAS real. Debe proporcionar la clave de delegación de usuario que obtuvo anteriormente para este método para obtener el token SAS.

Una vez que obtenga su token SAS, puede usarlo para acceder a su cuenta de Azure Storage. El código del [Listado 3-20](#) muestra cómo interactuar con su cuenta de Azure Storage mediante el token SAS que creó en el [Listado 3-19](#). Si desea probar este código, simplemente reemplace el contenido del archivo Program.cs que creó en el ejemplo anterior por el contenido del [Listado 3-20](#). Antes de reemplazar su código, debe agregar el paquete System.IO NuGet ejecutando el siguiente comando en la ventana de terminal de Visual Studio Code:

```
dotnet agregar paquete System.IO
```

#### **Listado 3-20 extensión Program.cs**

[Haga clic aquí para ver la imagen del código](#)

---

```
// C#. ASP.NET.
```

```
usando el sistema;
```

```
utilizando Azure.Storage.Blobs;
```

```
utilizando Azure.Storage.Blobs.Models;
utilizando Azure.Storage.Sas;
usando Azure;
utilizando Azure.Identity;
usando System.IO;

espacio de nombres ch3_1_2

{
    programa de clase

    {
        static void Main (cadena [] argumentos)

        {
            string storageAccount = "az204testing";
            string containerName = "az204-blob-testing";
            string blobName =
System.IO.Path.GetRandomFileName ();

            DateTimeOffset startTimeKey =
DateTimeOffset.UtcNow;

            DateTimeOffset endTimeKey =
DateTimeOffset.UtcNow.AddDays (7);

            DateTimeOffset startTimeSAS = startTimeKey;
            DateTimeOffset endTimeSAS =
startTimeSAS.AddYears (1);
```

```
        Uri blobEndpointUri = new Uri ($ "https: //  
{storageAccount} .blob.core.  
                                windows.net ");  
  
        var defaultCredentials = new  
DefaultAzureCredential (verdadero);  
  
        BlobServiceClient blobClient = nuevo  
BlobServiceClient (blobEndpointUri,  
defaultCredentials);  
  
        //Consigue la llave. Vamos a utilizar esta clave  
para crear el SAS.  
  
        UserDelegationKey clave =  
blobClient.GetUserDelegationKey (startTimeKey,  
endTimeKey);  
  
        Console.WriteLine ($ "La clave de usuario  
comienza en: {key.SignedStartsOn}");  
  
        Console.WriteLine ($ "La clave de usuario expira  
el: {key.SignedExpiresOn}");  
  
        Console.WriteLine ($ "Servicio de clave de  
usuario: {key.SignedService}");  
  
        Console.WriteLine ($ "Versión de clave de  
usuario: {key.SignedVersion}");
```

```
// Necesitamos usar BlobSasBuilder para crear el
SAS

    BlobSasBuilder blobSasBuilder = nuevo
    BlobSasBuilder ()

    {

        BlobContainerName = containerName,
        BlobName = blobName,
        Recurso = "b",
        StartsOn = startTimeSAS,
        ExpiresOn = endTimeSAS,
        Protocolo =
        Azure.Storage.Sas.SasProtocol.Https

    } ;

// Establecemos los permisos Crear, Listar,
Aregar, Leer y Escribir

    blobSasBuilder.SetPermissions
    (BlobSasPermissions.All);

    string sasToken =
blobSasBuilder.ToSasQueryParameters

    (clave, cuenta de almacenamiento) .ToString ();

Console.WriteLine ($ "Token SAS: {sasToken}");

// Construimos el URI completo para acceder a la
cuenta de almacenamiento de Azure
```

```
UriBuilder blobUri = new UriBuilder ()  
{  
    Scheme = "https",  
    Host = ${storageAccount}  
.blob.core.windows.net",  
    Path = ${containerName} / {blobName}",  
    Query = sasToken  
};  
  
// Creamos un archivo de texto aleatorio  
usando (System.IO.StreamWriter sw =  
System.IO.File.CreateText (blobName))  
{  
    sw.Write ("Este es un blob de prueba para  
cargar usando SAS delegado por el usuario  
tokens ");  
}  
  
BlobClient testingBlob = new BlobClient  
(blobUri.Uri);  
testingBlob.Upload (blobName);  
  
// Ahora volvemos a descargar el blob e  
imprimimos el contenido.
```

```

        Console.WriteLine ($ "Leer contenido del blob de
prueba {blobName} ");

        Console.WriteLine ();



        BlobDownloadInfo downloadInfo =
testingBlob.Download ();


        usando (StreamReader sr = new StreamReader
(downloadInfo.Content, true))

{

    línea de cuerda;

    while ((línea = sr.ReadLine ()) != null)

    {

        Console.WriteLine (línea);

    }

}

Console.WriteLine ();


        Console.WriteLine ("Terminado de leer el
contenido del blob de prueba");





}
}

```

Hemos puesto las partes esenciales en negrita en el [Listado 3-20](#). Cuando necesite usar el token SAS para trabajar con cuentas de Azure Storage,

debe construir el token SAS correcto para el elemento con el que está trabajando. Esto significa que, si va a trabajar con un contenedor, debe crear un token SAS para ese contenedor y obtener una referencia al contenedor mediante BlobContainerClient. Una vez que tenga la referencia al contenedor, puede seguir trabajando con otros elementos secundarios sin necesidad de crear un nuevo token SAS para cada elemento dentro del contenedor.

En el ejemplo del [Listado 3-20](#), creamos un archivo de texto aleatorio con algo de contenido que cargamos en el contenedor y luego lo descargamos nuevamente. Luego creamos un token SAS para cargar el archivo de texto aleatorio. Observe que creamos el token SAS que apunta a un blob que ni siquiera existe. Una vez que tenemos el token SAS correcto, con los permisos correctos, creamos un `BlobClient` objeto usando el URI que apunta a la ubicación final en la cuenta de Azure Blob Storage dentro del contenedor. Usamos el token SAS como parámetro de consulta del URI. Una vez que tenemos nuestro `BlobClient` objeto que representa el blob, podemos realizar todas las operaciones necesarias sin necesidad de crear un nuevo token SAS para el mismo blob, siempre que el token no haya expirado.

#### *¿Necesita más revisión? Firmas de acceso compartido*

Si desea leer más sobre cómo trabajar con firmas de acceso compartido, no solo con el servicio Azure Blob Storage, sino con otros servicios de Azure Storage, como tablas, cola o archivos, puede revisar el artículo en <https://docs.microsoft.com/en-us/rest/api/storageservices/delegate-access-with-shared-access-signature>.



#### *Sugerencia para el examen*

Si planea trabajar con SAS de delegación de usuarios, debe tener en cuenta que este tipo de SAS solo está disponible para Azure Blob Storage y Azure Data Lake Storage Gen2. No puede utilizar ninguna de las políticas de acceso almacenado cuando trabaja con SAS de delegación de usuarios.

*Registre aplicaciones y use Azure Active Directory para autenticar usuarios*

Puede asegurar el acceso a la información administrada por su aplicación utilizando varios mecanismos, como autenticación basada en formularios,

autenticación SSL, autenticación de Windows o autenticación OAuth2, entre otros. Cada uno de estos mecanismos tiene ventajas y desventajas.

La sección "Implementar la autenticación OAuth2" anteriormente en este capítulo revisó cómo usar la autenticación OAuth2 con una aplicación web básica. Cuando revisamos los conceptos de OAuth2 en esa sección, vio que en el flujo de autenticación de OAuth2, hay un servidor de seguridad que se encarga de proporcionar los mecanismos de seguridad para autenticar a los usuarios. Una vez que el servidor de seguridad se autentica, el servidor emite un token que su aplicación puede validar y usar para autenticar la solicitud del cliente de su aplicación. Al trabajar con el servidor de seguridad, puede utilizar su propia implementación de un servidor OAuth2, o puede confiar en servicios de seguridad de terceros, como Facebook, Google o LinkedIn, entre otros.

Microsoft también ofrece la posibilidad de utilizar sus servicios para la autenticación OAuth2. Microsoft proporciona autenticación OAuth2 a través de su servicio de identidad Azure Active Directory. En su capa más básica, este es un servicio gratuito que puede usar si desea que los usuarios de su aplicación puedan iniciar sesión usando cuentas de Microsoft Outlook.com para cuentas personales o las cuentas de Azure Active Directory para cuentas profesionales.

Antes de que su aplicación pueda usar el servicio Azure Active Directory para autenticar a los usuarios de su aplicación, debe registrar la aplicación en su inquilino de Azure Active Directory. Al registrar su solicitud, hay algunos puntos que debe considerar antes de proceder con el registro:

- **Tipos de cuenta admitidos** Debe considerar si los usuarios de su aplicación
  - **Solo usuarios de su organización** Cualquier persona que tenga una cuenta de usuario en su inquilino de Azure Active Directory podría usar su aplicación.
  - **Usuarios de cualquier organización** Utilice esta opción cuando desee que cualquier usuario con una cuenta de Azure Active Directory profesional o educativa pueda iniciar sesión en su aplicación.
  - **Usuarios de cualquier organización o cuentas de Microsoft** Utilice esta opción si desea que sus usuarios inicien sesión en su aplicación utilizando cuentas

profesionales, educativas o cualquiera de las cuentas de Microsoft disponibles gratuitamente.

- **Plataforma** La autenticación OAuth2 no se limita a las aplicaciones web. También puede utilizar este tipo de autenticación con plataformas móviles, como iOS o Android, o plataformas de escritorio, como macOS, Console, IoT, Windows o UWP.

El siguiente procedimiento muestra cómo registrar una aplicación web en Azure Active Directory:

1. Abra el portal de Azure (<https://portal.azure.com>),
2. En el cuadro de texto Buscar recursos, servicios y documentos en la parte superior media de Azure Portal, escriba **Azure Active Directory**.
3. En la lista de resultados, en la sección Servicios, haga clic en Azure Active Directory.
4. En la página de Azure Active Directory, en la sección Administrar, haga clic en Registros de aplicaciones.
5. En la hoja Registros de aplicaciones, haga clic en el botón Nuevo registro en la esquina superior izquierda del panel.
6. En la hoja Registrar una aplicación, escriba el nombre de su aplicación en el cuadro de texto Nombre.
7. En el control de opción Tipos de cuenta admitidos, seleccione Cuentas en este directorio organizativo únicamente.
8. Haga clic en el botón Registrar en la esquina inferior izquierda de la hoja.

El procedimiento anterior muestra cómo realizar un registro de aplicación simple. Ahora necesita configurar el registro de su aplicación de acuerdo con las necesidades de su aplicación. Uno de los conjuntos de opciones más importantes que debe configurar correctamente es la configuración de autenticación, que se muestra en la [Figura 3-7](#). Utilice la configuración de autenticación para administrar las opciones de autenticación para su aplicación. En este caso, configure las URL de redireccionamiento que utiliza Azure Active Directory para autenticar las solicitudes de su aplicación. Si la URL de redireccionamiento proporcionada por su aplicación no coincide con ninguna de las URL configuradas en esta sección, la autenticación falla.

Platform configurations

Depending on the platform or device this application is targeting, additional configuration may be required such as redirect URIs, specific authentication settings, or fields specific to the platform.

[+ Add a platform](#)

Supported account types

Who can use this application or access this API?

Accounts in this organizational directory only (only - Single tenant)

Accounts in any organizational directory (Any Azure AD directory - Multitenant)

### Figura 3-7 Configuración de autenticación

Los otros dos conjuntos críticos de configuraciones que debe considerar son Certificados y secretos y Permisos de API. Certificados y secretos le permite administrar los Certificados y los secretos que su aplicación necesita utilizar para proporcionar la identidad de la aplicación cuando solicita un token. Utilice los permisos de API para configurar el permiso necesario para llamar a otras API, ya sea de Microsoft, su organización u otras API de terceros. El siguiente ejemplo muestra cómo crear una aplicación web simple que usa la autenticación de Azure Active Directory. Aunque puede registrar la aplicación para este ejemplo directamente desde el asistente en Visual Studio 2019, preferimos mostrarle cómo realizar el registro de una aplicación directamente desde el portal de Azure. En este ejemplo, utilizará la aplicación que registró en el procedimiento. Si no siguió ese procedimiento, debe revisarlo y seguirlo antes de poder continuar con el siguiente ejemplo:

1. Abra Visual Studio 2019.
2. En la ventana de bienvenida de Visual Studio 2019, en la columna Introducción, haga clic en Crear un nuevo proyecto.
3. En la ventana Crear un proyecto nuevo, en el menú desplegable Todos los idiomas, seleccione C #.
4. En el cuadro de texto Buscar plantillas, escriba **asp.net**.
5. En la lista de resultados, haga clic en Aplicación web ASP.NET (.NET Framework).
6. Haga clic en el botón Siguiente en la esquina inferior derecha de la ventana.

7. En la ventana Configure su nuevo proyecto, escriba un nombre de proyecto, una ubicación y un nombre de solución para su proyecto.
8. Haga clic en el botón Crear en la esquina inferior derecha de la ventana.
9. En la ventana Crear una nueva aplicación web ASP.NET, seleccione la plantilla MVC en la lista de plantillas en el medio del lado izquierdo de la ventana. MVC es para Model-View-Controller.
10. En el lado derecho de la ventana Crear una nueva aplicación web ASP.NET, en la sección Autenticación, asegúrese de que Autenticación esté configurada en Sin autenticación.
11. Haga clic en el botón Crear en la esquina inferior derecha de la ventana.
12. Abra Azure Portal (<https://portal.azure.com>) y navegue hasta la aplicación que registró en el ejemplo anterior.
13. En la hoja Información general de su aplicación en Azure Portal, copie el valor del parámetro Id. De aplicación (cliente). Necesita este valor para un paso posterior.
14. En la sección Administrar en el lado izquierdo de la hoja de la aplicación, haga clic en Certificados y secretos.
15. En la hoja Certificados y secretos, en el área Secretos del cliente, haga clic en el botón Nuevo secreto del cliente.
16. Escriba una descripción en el cuadro de texto de este secreto de cliente.
17. Haga clic en el botón Agregar.
18. En el área Secretos del cliente, en la lista de secretos del cliente, copie el valor del secreto del cliente que acaba de crear. Necesita este valor en un paso posterior.
19. En la ventana del Explorador de soluciones en la ventana de Visual Studio 2019, haga clic con el botón derecho en el nodo Servicios conectados.
20. Haga clic en Agregar servicio conectado en el menú contextual.

21. En las ventanas de Connected Services, haga clic en Autenticación con Azure Active Directory.
22. En la ventana Configurar la autenticación de Azure AD, en la sección Introducción, haga clic en el botón Siguiente en la parte inferior derecha de la ventana.
23. En la sección Inicio de sesión único, en el menú desplegable Dominio, escriba el nombre de su inquilino. Puede encontrar esta información en Azure Portal, en la hoja Información general de su inquilino de Azure Active Directory.
24. En el área Ajustes de configuración, seleccione Usar ajustes de una aplicación de Azure AD existente para configurar su proyecto.
25. En el cuadro de texto ID de cliente, pegue el valor que copió en el paso 13.
26. Deje el URI de redireccionamiento en blanco.
27. Haga clic en el botón Siguiente en la parte inferior derecha de la ventana.
28. En la sección Acceso al directorio, marque la opción Leer datos del directorio.
29. En el cuadro de texto Secreto del cliente, copie el secreto del cliente que creó en el paso 18.
30. Haga clic en el botón Finalizar.
31. En la ventana Autenticación de Azure AD, espere a que el asistente agregue todo el código necesario a su aplicación.
32. En el Explorador de soluciones, haga clic en el nombre de su proyecto y presione F4.
33. En la ventana Propiedades, copie el valor de la configuración de URL de SSL. Necesita este valor en un paso posterior.
34. Abra Azure Portal y navegue hasta su aplicación registrada en su inquilino de Azure Active Directory.
35. En la hoja de la aplicación registrada, en la sección Administrar, haga clic en Autenticación.
36. En la hoja Autenticación, en el área Configuraciones de plataforma, haga clic en el botón Agregar una plataforma.

37. En el panel Configurar plataformas, en la sección Aplicaciones web, haga clic en Web.
38. En el cuadro de texto Redirigir URI, copie el valor de la configuración de URL SSL que copió en el paso 33.
39. En la sección Subvención implícita, marque Tokens de identificación.
40. Haga clic en el botón Configurar.
41. En su ventana de Visual Studio 2019, presione F5 para ejecutar su proyecto.

Una vez que ejecute la aplicación web, debería aparecer una página de inicio de sesión genérica de Microsoft en su navegador. Debe proporcionar una cuenta de usuario válida de su inquilino para iniciar sesión. En este punto, su aplicación usa Azure Active Directory para autenticar usuarios. Además, también puede leer información de su inquilino.

Cuando ejecuta la autenticación con el asistente de Azure Active Directory, hay algunos cambios en el código que debe comprender. El asistente agrega algunas propiedades a la `appSettings` sección en el archivo `web.config`. Estas propiedades representan la configuración relevante necesaria para conectarse a su inquilino de Azure Active Directory:

- **ida: ClientId** Este es el ID que representa su aplicación registrada en su inquilino de Azure Active Directory.
- **ida: AADInstance** Proporciona la instancia que utilizará para la autenticación. En la mayoría de las situaciones, utiliza instancias públicas en general. Solo necesita cambiar este valor si su inquilino está alojado en instancias aisladas como Gobierno, China o Alemania.
- **ida: Dominio** Este es su inquilino de Azure Active Directory, donde registró su aplicación.
- **ida: TenantId** Este es el ID que representa al inquilino donde registró su aplicación.
- **ida: PostLogoutRedirectUri** Esta es la URL a la que Microsoft redirige al usuario una vez que el proceso de autenticación finaliza correctamente. Este valor debe coincidir con el valor configurado en el registro de su aplicación en Azure Portal.

- **ida: ClientSecret** El secreto del cliente es similar a la contraseña que usa su aplicación para autenticarse en el servicio Azure Active Directory antes de que pueda interactuar con las API protegidas por el servicio de identidad.

Al igual que con cualquier otro sistema de autorización en C #, debe agregar el `[Authorized]` atributo a cualquier recurso que desee proteger. En este caso, el asistente agrega este atributo a cualquier controlador existente en su aplicación. Finalmente, el fragmento de código más crítico es el que se usa para configurar la autenticación OpenID / OAuth2. [El Listado 3-21](#) muestra el fragmento de código agregado a su archivo Startup.Auth.cs para conectar su aplicación con Azure Active Directory.

**Listado 3-21** Extensión Program.cs

[Haga clic aquí para ver la imagen del código](#)

---

```
// C#. ASP.NET.

app.UseOpenIdConnectAuthentication (
    nuevas OpenIdConnectAuthenticationOptions
    {
        ClientId = clientId,
        Autoridad = Autoridad,
        PostLogoutRedirectUri =
        postLogoutRedirectUri,
        Notificaciones = nuevo
        OpenIdConnectAuthenticationNotifications ()
        {
            // Si hay un código en la respuesta
            de OpenID Connect, canjee //
            para obtener un token de acceso y un
            token de actualización, y almacenar esos
```

```

        // fuera.

        AuthorizationCodeReceived =
(contexto) =>

    {

        var code = context.Code;

        ClientCredential credential =
new ClientCredential

(clientId, appKey);

        string signedInUserID =
context.AuthenticationTicket

.Identidad.FindFirst (ClaimTypes

.NameIdentifier) .Value;

        AuthenticationContext
authContext =
new AuthenticationContext

(Autoridad,

new ADALTokenCache

(firmadoInUserID));

        return
authContext.AcquireTokenByAuthorizationCodeAsync (
código, nuevo Uri
(HttpContext.Current.Request.Url

.GetLeftPart
(UriPartial.Path)), credencial,
graphResourceId);

    }

```

```
}
```

```
} ) ;
```

### ***¿Necesita más revisión? Autenticación con el asistente de Azure AD***

Si desea leer más sobre los cambios realizados por el asistente de autenticación con Azure AD, consulte el artículo en <https://docs.microsoft.com/en-us/azure/active-directory/develop/vs-active-directory-dotnet-que-paso>.



### ***Sugerencia para el examen***

Cuando registra una nueva aplicación en su inquilino de Azure Active Directory, debe considerar cuál será su usuario de destino. Si necesita que cualquier usuario de cualquier organización de Azure Active Directory pueda iniciar sesión en su aplicación, debe configurar una aplicación multiusuario. En esos escenarios de múltiples inquilinos, el registro y la administración de la aplicación siempre se realiza en su inquilino y no en ningún otro inquilino externo.

### ***Controlar el acceso a los recursos mediante el uso de controles de acceso basados en roles (RBAC)***

Cuando trabaja con su suscripción de Azure, hay situaciones en las que debe otorgar acceso a otros usuarios. Estos usuarios pueden necesitar acceso solo a recursos específicos dentro de su suscripción, como un grupo de recursos específico o una base de datos SQL de Azure. Puede lograr este nivel de granularidad mediante el uso de controles de acceso basados en roles (RBAC).

Construido sobre Azure Resource Manager, RBAC proporciona un control de acceso detallado a los diferentes recursos en una suscripción de Azure. Al trabajar con RBAC, debe tener en cuenta los siguientes conceptos:

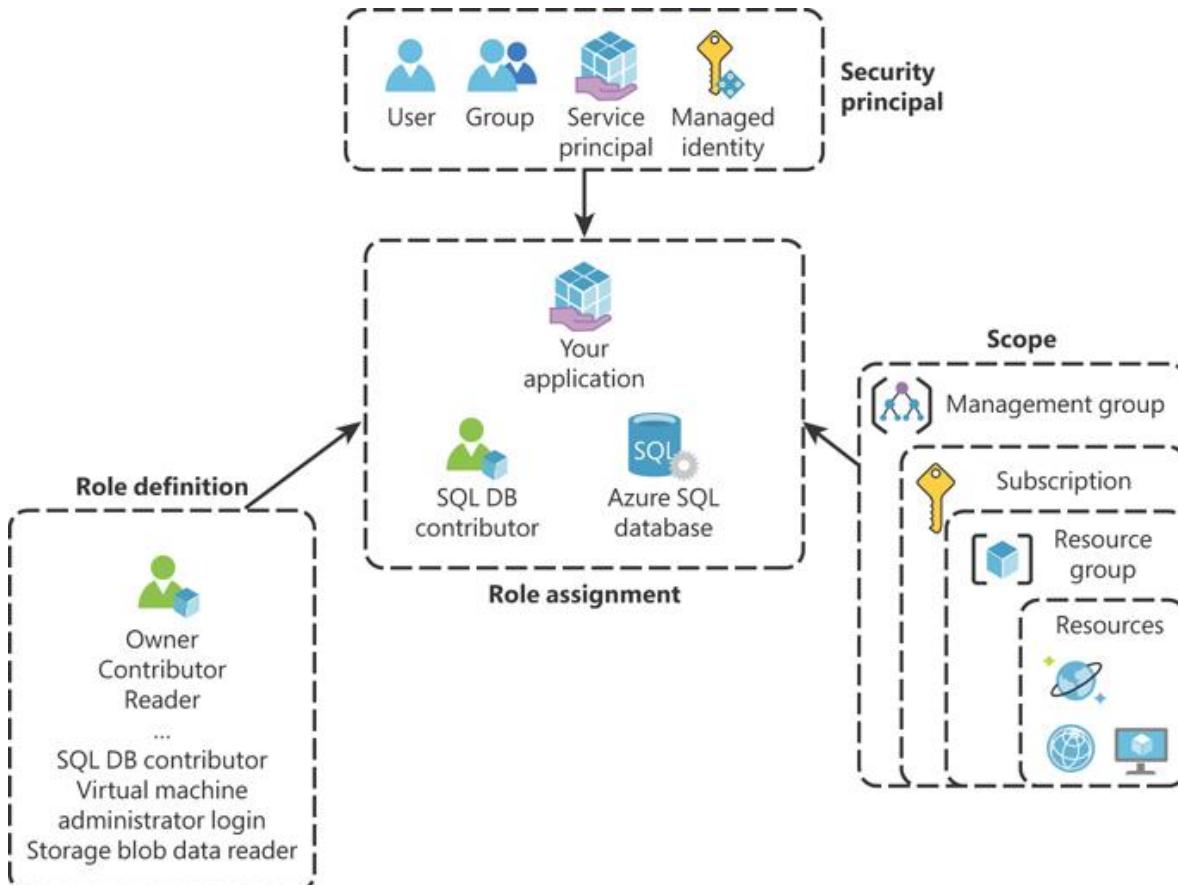
- **Principal de seguridad** Esta es la entidad que solicita permiso para realizar una acción. Una entidad de seguridad puede ser una de las siguientes:
  - **Usuario** Es una persona que tiene un perfil en un inquilino de Azure Active Directory. No está limitado a su

propio inquilino. También puede asignar un rol a los usuarios de otros inquilinos.

- **Grupo** Este es un conjunto de usuarios.
- **Principal de servicio** Esto es como un usuario para una aplicación. Una entidad de servicio representa una aplicación dentro del inquilino.
- **Identidad administrada** Este tipo de identidad representa aplicaciones en la nube que necesitan acceder a recursos en su inquilino de Azure. Azure administra automáticamente este tipo de identidad.
- **Permiso** Esta es la acción que puede realizar con un recurso. Un ejemplo de una acción sería solicitar una clave de delegación de usuario para crear un token SAS. Otro ejemplo de acción es enumerar el contenido de un contenedor. No puede asignar directamente un rol a una entidad de seguridad. Siempre debe utilizar una función o una definición de función.
- **Definición de rol** Por lo general, conocida como solo rol para abreviar, una definición de rol es una colección de permisos. Asignas un rol a una entidad de seguridad. Hay muchos roles predefinidos en Azure que puede usar para administrar el acceso a los recursos. Hay cuatro roles fundamentales:
  - **Propietario** Otorga acceso completo a todos los recursos del alcance.
  - **Las subvenciones para colaboradores** modifican el acceso a todos los recursos del alcance. Puede realizar todas las operaciones de modificación, incluida la eliminación, con los recursos del alcance. No puede otorgar roles a otros directores de seguridad.
  - **Reader** Grants acceso de lectura a todos los recursos del alcance.
  - **Administrador de acceso de usuario** Útil solo para administrar el acceso de los usuarios a los recursos de Azure.

Además de estos cuatro roles integrados fundamentales, hay roles específicos para cada servicio, como Colaborador de máquina virtual o Lector de cuentas de Cosmos DB.

- **Alcance** Este es el grupo de recursos donde asigna el rol. Puede establecer un rol en cuatro niveles diferentes: grupo de administración (un grupo de suscripciones), suscripción, grupo de recursos y recurso. Estos cuatro niveles están organizados en una relación padre-hijo donde el grupo de administración es el nivel más alto y el recurso es el más bajo. Cuando asigna un rol a un nivel, esos permisos son heredados por los niveles inferiores. Eso significa que si otorga el rol de propietario a un usuario en el nivel de suscripción, ese usuario tiene los privilegios de propietario en todos los grupos de recursos y recursos en esa suscripción.
- **Asignación de roles** Esta es la unión entre las diferentes partes de RBAC. Una asignación de funciones conecta una entidad de seguridad con una función y un ámbito. [La figura 3-8](#) muestra la relación entre los diferentes elementos RBAC.



**Figura 3-8** Control de acceso basado en roles

El siguiente procedimiento muestra cómo otorgar la función Colaborador a un grupo de recursos:

1. Abra Azure Portal (<https://portal.azure.com> ).
2. En el cuadro de texto Buscar recursos, servicios y documentos, escriba el nombre del grupo de recursos al que desea otorgar el rol de Colaborador.
3. En la lista de resultados, haga clic en el nombre del grupo de recursos.
4. En la hoja Grupo de recursos, haga clic en Control de acceso (IAM) en el lado izquierdo del control.
5. En la hoja de Control de acceso (IAM), haga clic en el botón Agregar en la esquina superior izquierda de la hoja.
6. En el menú contextual que aparece debajo del botón Agregar, haga clic en Agregar asignación de rol.
7. En la hoja Agregar asignación de función, en el menú desplegable Función, seleccione Colaborador.
8. Deje el menú desplegable Asignar acceso a establecido en el valor predeterminado.
9. En el cuadro de texto Seleccionar, escriba el nombre del usuario, grupo o entidad de servicio al que desea asignar el rol de Colaborador.
10. En la lista de resultados debajo del cuadro de texto Seleccionar, haga clic en la entidad de seguridad a la que desea asignar el rol.
11. Haga clic en el botón Guardar en la esquina inferior izquierda de la hoja Agregar asignación de funciones.

Realizar asignaciones de roles a otros niveles, como grupos de administración, suscripciones o recursos, es similar al procedimiento anterior. En general, la asociación de roles RBAC se realiza en la sección Control de acceso (IAM) de cada nivel.

#### *¿Necesita más revisión? Roles personalizados de Rbac*

Aunque Azure proporciona una buena cantidad de roles integrados, puede haber situaciones en las que los roles integrados no tengan los privilegios adecuados para sus necesidades. En esas situaciones, puede crear un rol personalizado. Cuando define un rol personalizado, usa el rol de la misma manera que usa un rol integrado. Debe crear una definición de rol para su rol configurando los permisos necesarios para su rol personalizado. Puede encontrar más información sobre cómo crear roles personalizados revisando el artículo en <https://docs.microsoft.com/en-us/azure/role-based-access-control/custom-roles> .



### *Sugerencia para el examen*

Cuando asigne roles de servicio específicos, revise cuidadosamente los permisos otorgados por el rol. En general, otorgar acceso a un recurso no otorga acceso a los datos administrados por ese recurso. Por ejemplo, el colaborador de la cuenta de almacenamiento otorga acceso para administrar las cuentas de almacenamiento, pero no otorga acceso a los datos en sí.

## HABILIDAD 3.2: IMPLEMENTAR SOLUCIONES SEGURAS EN LA NUBE

---

La habilidad anterior revisó cómo proteger el acceso a los datos autenticando y autorizando a los usuarios que intentan acceder a la información administrada en su aplicación. Esta protección es solo una parte de los mecanismos que debe implementar para proteger sus datos. También debe asegurarse de que toda la configuración necesaria para ejecutar su aplicación en los diferentes entornos se gestione de forma segura. La razón para asegurar también esa configuración es que la configuración tiene las contraseñas, certificados y secretos necesarios para acceder a la información administrada por su aplicación.

Cuando cifra sus datos, debe utilizar claves o secretos de cifrado y descifrado para acceder a los datos y protegerlos. Almacenar estos secretos y claves de cifrado es tan importante como cifrar los datos. Perder una clave de cifrado o descifrado es similar a perder las llaves de su casa. Azure Key Vault le permite almacenar de forma segura estas claves de cifrado / descifrado, así como otros secretos o certificados que sus aplicaciones pueden requerir en un almacén de cifrado seguro en Azure. Junto con Managed Identities, los servicios de Azure Key Vault le permiten almacenar de forma segura sus secretos sin necesidad de almacenar una contraseña, certificado o cualquier tipo de credencial para acceder a sus secretos.

### **Esta habilidad cubre cómo**

- Proteja los datos de configuración de la aplicación mediante la configuración de la aplicación y la API de KeyVault

- Administre claves, secretos y certificados mediante la API de KeyVault
- Implementar identidades administradas para el recurso de Azure

*Proteja los datos de configuración de la aplicación mediante la configuración de la aplicación y la API de KeyVault*

La mayoría de las aplicaciones medianas y grandes de hoy en día se basan en arquitecturas distribuidas. Independientemente de si la infraestructura que ejecuta su aplicación está basada en máquinas virtuales, contenedores, computación sin servidor o cualquier otro tipo de computación, necesita compartir la configuración entre los elementos que ejecutan el mismo componente de su aplicación. Por ejemplo, si su aplicación se ejecuta en un clúster de Internet Information Services detrás de un equilibrador de carga, todas las máquinas virtuales que alojan el servicio IIS comparten la misma configuración para ejecutar su aplicación.

Este tipo de escenario es donde Azure App Configuration se convierte en una herramienta útil. Azure App Configuration le permite almacenar toda la configuración necesaria para su aplicación en la nube en un único repositorio. Otros servicios de Azure también le permiten administrar la configuración de sus aplicaciones, pero tienen algunas diferencias cruciales que debe considerar:

- **Configuración de Azure App Service** Como ya sabe, puede crear configuraciones para su Azure App Service. Esta configuración se aplica a la instancia que está configurando. Incluso puede crear diferentes valores de configuración para diferentes ranuras de implementación. Por otro lado, el servicio Azure App Configuration es un servicio de configuración centralizado que le permite compartir la misma configuración entre diferentes instancias de Azure App Service. También debe tener en cuenta que el servicio de configuración de aplicaciones de Azure no se limita a Azure App Service. También puede usarlo con aplicaciones en contenedores o con aplicaciones que se ejecutan dentro de máquinas virtuales.

- **Azure Key Vault** Azure Key Vault le permite almacenar de forma segura contraseñas, secretos y cualquier otra configuración que su aplicación pueda necesitar. El cifrado se realiza mediante cifrado a nivel de hardware, entre otras características interesantes como la rotación de certificados o las políticas de acceso granular. Aunque Azure App Configuration cifra el valor de su configuración, Azure Key Vault aún proporciona niveles más altos de seguridad. Puede usar Azure Key Vault junto con Azure App Configuration creando referencias en Azure App Configuration a elementos almacenados en Azure Key Vault.

Cuando trabaja con la configuración de la aplicación de Azure, debe tratar con dos componentes diferentes: el almacén de configuración de la aplicación y el SDK. El almacén de configuración de la aplicación de Azure es el lugar donde almacena su configuración. Cuando configura su tienda de configuración de aplicaciones de Azure, puede elegir entre dos niveles de precios diferentes: gratuito y estándar. La principal diferencia entre los dos niveles de precios es la cantidad de tiendas que puede crear en una suscripción. En el nivel gratuito, está limitado a una tienda por suscripción, mientras que no tiene tal limitación en el nivel estándar. Otras diferencias entre los dos niveles son el tamaño máximo de la tienda: 10 MB en el nivel gratuito frente a 1 GB en el nivel estándar, o el tamaño del historial de claves: 7 días en el nivel gratuito frente a 30 días en el nivel estándar. Puede cambiar del nivel gratuito al estándar en cualquier momento, pero no puede volver al nivel gratuito desde el nivel estándar. El siguiente procedimiento muestra cómo crear una configuración de aplicación de Azure:

1. Abra Azure Portal (<https://portal.azure.com>).
2. Haga clic en el botón Crear un recurso en la sección Servicios de Azure en la parte superior de Azure Portal.
3. En la hoja Nueva, escriba la **configuración** de la **aplicación** en el cuadro de texto Buscar en el mercado.
4. En la lista de resultados debajo del cuadro de texto, haga clic en Configuración de la aplicación.
5. En la hoja Configuración de la aplicación, haga clic en el botón Crear.
6. En la hoja Configuración de la aplicación, escriba un nombre para el almacén de Configuración de la aplicación en el cuadro de

texto Nombre del recurso. El nombre debe contener solo caracteres ASCII alfanuméricos o el carácter de guión (-) y debe tener entre 5 y 50 caracteres.

7. Seleccione la suscripción donde desea implementar su tienda de configuración de aplicaciones usando el menú desplegable Suscripción.
8. En el menú desplegable Grupo de recursos, seleccione el grupo de recursos donde desea implementar su tienda de configuración de aplicaciones. Alternativamente, puede crear un nuevo grupo de recursos haciendo clic en el enlace Crear nuevo debajo del menú desplegable Grupo de recursos.
9. Seleccione una ubicación en el menú desplegable Ubicación.
10. En el menú desplegable Nivel de precios, seleccione Gratis.
11. Haga clic en el botón Crear en la parte inferior de la hoja.

Una vez que haya creado su tienda de configuración de aplicaciones de Azure, puede crear los pares clave-valor para almacenar su configuración. Antes de crear su primer par clave-valor, debe revisar cómo funcionan las claves dentro de la tienda de configuración de aplicaciones.

Una clave es un identificador asociado con un valor almacenado en la tienda de configuración de aplicaciones. Utiliza la clave para recuperar un valor de la tienda. Las claves distinguen entre mayúsculas y minúsculas, por lo que "appSample204" y "APPSAMPLE204" son claves diferentes. Esto es importante porque algunos lenguajes o marcos no distinguen entre mayúsculas y minúsculas para la configuración, por lo que no debe utilizar la distinción entre mayúsculas y minúsculas para las claves de diferenciación. Al nombrar una tecla, puede usar cualquier carácter Unicode, excepto el asterisco (\*), la coma (,) y la barra inclinada invertida (\). Si necesita incluir alguno de estos caracteres reservados, debe anteponer el carácter de escape de barra diagonal inversa (\). Como práctica recomendada, debería considerar la posibilidad de utilizar un espacio de nombres al nombrar sus claves. Al usar un carácter separador entre los diferentes niveles, puede crear una jerarquía de configuraciones dentro de su tienda. Como el servicio de configuración de la aplicación de Azure no analiza ni analiza sus claves, usted elige el espacio de nombres que mejor se adapte a sus necesidades. Algunos ejemplos de claves que utilizan espacios de nombres son

## Haga clic aquí para ver la imagen del código

AppSample: Desarrollo: DbConnection

AppSample: AUS: WelcomeMessage

También puede agregar un atributo de etiqueta a una clave. De forma predeterminada, el atributo de etiqueta es nulo. Puede usar la etiqueta para hacer que los valores sean diferentes usando la misma clave. Esto es especialmente útil cuando se usa para entornos de implementación: Los siguientes tres ejemplos son claves diferentes porque las etiquetas son diferentes:

## Haga clic aquí para ver la imagen del código

Key = AppSample: DBConnection - Label = Desarrollar

Key = AppSample: DBConnection - Label = Stage

Key = AppSample: DBConnection - Label = Producción

Al crear un nuevo par clave-valor, tiene un límite de 10,000 para el tamaño del par. Este límite se aplica al tamaño de la clave, más el tamaño de la etiqueta opcional, más el tamaño del valor. También debe tener en cuenta que las mismas limitaciones que se aplican a la cadena que usa para la clave son las mismas para el valor. Es decir, puede usar cualquier carácter Unicode para el valor, excepto asterisco (\*), coma (,) y barra inclinada invertida (\). Si necesita incluir alguno de estos caracteres reservados, debe anteponer el carácter de escape de barra diagonal inversa (\).

## *¿Necesita más revisión? Gestión de funciones y configuración dinámica*

Puede aprovechar la configuración de la aplicación de Azure para implementar características más avanzadas como la administración de características o la configuración dinámica. Los siguientes artículos le brindan más información sobre estas funciones:

- **Gestión de funciones** <https://docs.microsoft.com/en-us/azure/azure-app-configuration/quickstart-feature-flag-aspnet-core>
- **Habilite la configuración dinámica** <https://docs.microsoft.com/en-us/azure/azure-app-configuration/enable-dynamic-configuration-aspnet-core>

Una vez que haya revisado los conceptos básicos de la configuración de la aplicación de Azure, puede revisar cómo usar este servicio en su código. El siguiente ejemplo se basa en el código del [Capítulo 2](#) en la sección "[Interactuar con los datos usando el SDK apropiado](#)". En este

ejemplo, va a modificar el código para usar una tienda de contenedor de aplicaciones de Azure en lugar de usar un archivo JSON de AppSettings:

1. Abra Visual Studio Code y cree una carpeta para su proyecto.
2. En la ventana de código de Visual Studio, abra una nueva terminal.
3. Utilice el siguiente comando para crear un nuevo proyecto de consola:

```
nueva consola dotnet
```

4. Utilice el siguiente comando para instalar paquetes NuGet:

[Haga clic aquí para ver la imagen del código](#)

```
dotnet agregar paquete <NuGet_package_name>
```

5. Instale los siguientes paquetes de NuGet:

1. Azure.Storage.Blobs
2. Azure.Almacenamiento.Común
3. Azure.Identity
4. Microsoft.Extensions.Configuration
5. Microsoft.Extensions.Configuration.AzureAppConfiguration

6. Cree un archivo de clase C # y **asígnelle el nombre **AppSettings.cs****.

7. Reemplace el contenido del archivo AppSettings.cs con el contenido del [Listado 3-22](#). Cambie el nombre del espacio de nombres para que coincida con el nombre de su proyecto.

8. Cree un archivo de clase C # y **asígnelle el nombre **Common.cs****.

9. Reemplace el contenido del archivo Common.cs con el contenido del [Listado 3-23](#).

10. Cambie el nombre del espacio de nombres para que coincida con el nombre de su proyecto.

11. Reemplace el contenido del archivo **Program.cs** con el contenido del [Listado 3-24](#). Cambie el nombre del espacio de nombres para que coincida con el nombre de su proyecto.

### Listado 3-22 AppSettings.cs

Haga clic aquí para ver la imagen del código

---

```
// C#. ASP.NET.

usando el sistema;

utilizando Microsoft.Extensions.Configuration;

espacio de nombres ch3_2_1

{

    AppSettings de clase pública

    {

        cadena pública SourceSASConnectionString {get;
colocar; }

        cadena pública SourceAccountName {get; colocar; }

        cadena pública SourceContainerName {get; colocar; }

        Cadena pública DestinationSASConnectionString {get;
colocar; }

        cadena pública DestinationAccountName {get; colocar;
    }

        cadena pública DestinationContainerName {get;
colocar; }

    }

    AppSettings estáticos públicos LoadAppSettings ()

    {

        var builder = new ConfigurationBuilder ();

            builder.AddAzureAppConfiguration
(Environment.GetEnvironmentVariable
```

```
        ("Cadena de conexión"));

var config = constructor.Build ();
AppSettings appSettings = new AppSettings ();
appSettings.SourceSASConnectionString = config
["TestAZ204: StorageAccount:
Fuente:
ConnectionString "];

appSettings.SourceAccountName = config
["TestAZ204: StorageAccount: Fuente:
Nombre de la
cuenta "];

appSettings.SourceContainerName = config
["TestAZ204: StorageAccount:
Fuente:
ContainerName "];

appSettings.DestinationSASConnectionString =
config ["TestAZ204: Almacenamiento

Cuenta: Destino:

Cadena de conexión"];

appSettings.DestinationAccountName = config
["TestAZ204: StorageAccount:
Destino:
AccountName "];

appSettings.DestinationContainerName = config
["TestAZ204: StorageAccount:
```

```
        Destino:  
        ContainerName "];  
  
        return appSettings;  
  
    }  
  
}  
  
}
```

**Listado 3-23** Common.cs

Haga clic aquí para ver la imagen del código

---

```
// C#. ASP.NET.  
  
utilizando Azure.Storage.Blobs;  
  
  
espacio de nombres ch3_2_1  
  
{  
  
    clase pública común  
  
{  
  
        BlobServiceClient estático público  
CreateBlobClientStorageFromSAS (  
  
            cadena SASConnectionString)  
  
{  
  
        BlobServiceClient blobClient;  
  
        intentar  
  
{  
  
            blobClient = nuevo BlobServiceClient  
(SASConnectionString);
```

```

        }

        catch (System.Exception)

        {

            lanzar;

        }

        return blobClient;

    }

}

```

**Listado 3-24** Program.cs

Haga clic aquí para ver la imagen del código

---

```

// C#. ASP.NET.

usando System.Threading.Tasks;

usando el sistema;

utilizando Azure.Storage.Blobs;

espacio de nombres ch3_2_1

{

    programa de clase

    {

        static void Main (cadena [] argumentos)

```

```
        Console.WriteLine ("¡Copiar elementos entre  
contenedores de demostración!");  
  
        Task.Run (async () => aguardar  
StartContainersDemo ()). Wait ();  
  
    }  
  
  
    Tarea asíncrona estática pública StartContainersDemo  
()  
  
{  
  
    string sourceBlobFileName = "Testing.zip";  
  
    AppSettings appSettings =  
AppSettings.LoadAppSettings ();  
  
  
    // Obtenga un cliente en la nube para la cuenta  
de almacenamiento de origen  
  
    BlobServiceClient sourceClient =  
Common.CreateBlobClientStorageFromSAS (  
  
        appSettings.SourceSASConnectionString);  
  
  
    // Obtenga una referencia para cada contenedor  
  
    var sourceContainerReference =  
sourceClient.GetBlobContainerClient (aplicación  
  
        Settings.SourceContainerName);  
  
    var destinationContainerReference =  
sourceClient.GetBlobContainerClient (  
  
        appSettings.DestinationContainerName);
```

```

        // Obtener una referencia para el blob de origen

        var sourceBlobReference =
sourceContainerReference.GetBlobClient (
                                sourceBlobFileName);

        var destinationBlobReference =
destinationContainerReference.

                                GetBlobClient
(sourceBlobFileName);

        // Mueve el blob del contenedor de origen al
contenedor de destino

        aguardar
destinationBlobReference.StartCopyFromUriAsync
(sourceBlobReference

(Uri);

}

}

```

Los listados 3-23 y 3-24 son en su mayoría los mismos archivos que puede encontrar en el ejemplo del Capítulo 2 en la sección "Interactuar con los datos usando el SDK apropiado". El archivo AppSettings.cs que se muestra en el Listado 3-22 contiene toda la magia para trabajar con el servicio de configuración de aplicaciones. Al igual que con cualquier aplicación .NET Core normal, debe crear un ConfigurationBuilder objeto para administrar la configuración de la aplicación. Una vez que obtenga su constructor, utilice el método del objeto constructor para cargar todas las configuraciones almacenadas en la tienda de configuración de aplicaciones. Una vez que haya cargado todas las configuraciones, puede acceder a cada par clave-valor simplemente usando la clave correcta,

como puede ver en el [Listado 3-24](#). El `AddAzureAppConfiguration()` método de extensión para conectarse a la tienda de configuración de aplicaciones. Finalmente, usa el `Build()`

En este punto, si intenta ejecutar este ejemplo, obtendrá algunas excepciones porque no está proporcionando la cadena de conexión necesaria para acceder a su tienda de configuración de aplicaciones de Azure. Tampoco definió ningún par clave-valor en su tienda de configuración de aplicaciones, por lo que incluso si pudiera acceder a la tienda, obtendría valores nulos. Utilice los siguientes pasos para obtener la cadena de conexión necesaria para acceder a la tienda de configuración de aplicaciones y defina cada uno de los pares clave-valor necesarios:

1. Abra el portal de Azure (<https://portal.azure.com> ).
2. En el cuadro de texto Buscar recursos, servicios y documentos en la parte superior de Azure Portal, escriba el nombre de su tienda de configuración de aplicaciones.
3. En la hoja de App Configuration Store, haga clic en Access Keys en la sección Configuración.
4. En la hoja Claves de acceso, copie una de las cadenas de conexión haciendo clic en el ícono azul en el lado derecho junto al cuadro de texto Cadena de conexión.
5. En la ventana de Visual Studio Code, abra una nueva terminal y escriba el siguiente comando. Reemplaza el texto `<your_connection_string>` con el valor que en el paso 4:

[Haga clic aquí para ver la imagen del código](#)

```
setx ConnectionString "<your_connection_string>"
```

6. Reinicie su ventana de Visual Studio Code. Debe realizar este paso para asegurarse de que la variable de entorno que definió en el paso anterior esté disponible para su código.
7. En Azure Portal, en la hoja del almacén de configuración de aplicaciones, haga clic en Explorador de configuración en la sección Operaciones en el lado izquierdo de la hoja.
8. En el Explorador de configuración, haga clic en el botón Crear en la esquina superior izquierda de la hoja.

9. En el panel Crear, que se muestra en la [Figura 3-9](#), escriba el nombre de la clave en el cuadro de texto Clave. Utilice una de las claves que se muestran en el [Listado 3-22](#).

The screenshot shows a 'Create' dialog box with the following fields:

- Key \***: An input field with a red asterisk indicating it is required.
- Value**: An input field.
- Label**: A dropdown menu showing '(No label)'.
- Content type**: An input field.

At the bottom is a light gray 'Apply' button.

**Figura 3-9** Crear un nuevo valor-clave

10. En el cuadro de texto Valor, proporcione el valor de la clave correspondiente. Recuerde que está usando valores del ejemplo en el [Capítulo 2](#) en la sección "[Interactuar con los datos usando el SDK apropiado](#)". Los valores correctos son específicos para su escenario, pero puede usar el [Listado 2-12](#) como referencia.

11. Haga clic en el botón Aplicar.

12. Repita los pasos del 8 al 10 hasta que cree un par clave-valor para cada configuración en el [Listado 3-22](#). Aquí está la lista completa para su referencia:

1. TestAZ204: StorageAccount: Fuente:ConnectionString
2. TestAZ204: StorageAccount: Fuente: AccountName
3. TestAZ204: StorageAccount: Fuente: ContainerName
4. TestAZ204: StorageAccount: Destino:ConnectionString
5. TestAZ204: StorageAccount: Destino: AccountName

6. TestAZ204: StorageAccount: Destino: Nombre del contenedor
13. En la ventana de Visual Studio Code, presione F5 para ejecutar su proyecto. En este punto, su código debería poder conectarse a su tienda de configuración de aplicaciones de Azure y recuperar todas las configuraciones necesarias.

Este ejemplo le muestra los conceptos básicos para trabajar con Azure App Configuration, pero también muestra algunos inconvenientes que debe tener en cuenta para los entornos de producción. En este ejemplo, definió una variable de entorno para almacenar la cadena de conexión para conectarse al Tienda de configuración de aplicaciones. Aunque esta podría ser una configuración válida para entornos de prueba o desarrollo, existen implicaciones de seguridad que debe tener en cuenta para los entornos de producción. Debería considerar el uso de Managed Service Identity para entornos de producción, en lugar de usar cadenas de conexión.

Otra mejora de seguridad que debe considerar es almacenar cadenas de conexión directamente como un valor clave en su tienda de configuración de aplicaciones. Para este tipo de información confidencial, debe almacenarla como un secreto en Azure Key Vault y crear una referencia de Azure Key Vault en su tienda de configuración de aplicaciones que apunte al secreto correcto. En aras de la brevedad, no incluimos el procedimiento de cómo crear referencias de Key Vault, pero puede revisar una referencia completa en el artículo en <https://docs.microsoft.com/en-us/azure/azure-app-configuration/usage-key-vault-references-dotnet-core>.

#### *¿Necesita más revisión? Mejores prácticas*

Puede revisar algunas de las prácticas recomendadas al trabajar con la configuración de la aplicación Azure leyendo el artículo en <https://docs.microsoft.com/en-us/azure/app-configuration/howto-best-practices>.



#### *Sugerencia para el examen*

Cuando defina su par clave-valor, recuerde que está limitado a una longitud máxima de 10,000. Recuerde también que las claves distinguen entre mayúsculas y minúsculas, por lo que "AppSetting" y "appsetting" se tratan como claves diferentes.

## *Administre claves, secretos y certificados mediante la API de KeyVault*

Azure Key Vault es el servicio proporcionado por Microsoft para almacenar de forma segura claves secretas y certificados en un almacén seguro y centralizado. Al utilizar Azure Key Vault, los desarrolladores ya no necesitan almacenar esta información confidencial en sus equipos mientras desarrollan una aplicación. Gracias al control de acceso basado en identidad, solo necesita configurar una política para otorgar acceso al servicio necesario o los principales de usuario al almacén seguro. Otra ventaja es que puede aplicar un control de acceso detallado, permitiendo el acceso a secretos específicos solo a la aplicación o al usuario necesarios.

El siguiente ejemplo muestra cómo usar la API de KeyVault para crear, leer, actualizar o eliminar los diferentes elementos que puede almacenar en Azure Key Vault. Necesita un Azure App Service vacío y un Azure Key Vault configurado en su suscripción de Azure para ejecutar este ejemplo.

1. Abra Azure Portal en <https://portal.azure.com> .
2. En el cuadro de texto de búsqueda en la parte superior de Azure Portal, escriba el nombre de su aplicación web de Azure.
3. Haga clic en el nombre de su aplicación web de Azure en la lista de resultados debajo del cuadro de texto.
4. En la hoja Azure Web App Service, haga clic en el elemento de menú Identidad en la sección Configuración.
5. En el control del interruptor de estado, haga clic en la opción Activado.
6. Clic en Guardar.
7. En el cuadro de diálogo Habilitar identidad administrada asignada por el sistema, haga clic en Sí.
8. Una vez que habilita la identidad administrada asignada por el sistema, obtiene el ID de objeto o principal asociado con su Servicio de aplicaciones de Azure.
9. En el cuadro de texto de búsqueda en la parte superior de Azure Portal, escriba el nombre de su Azure Key Vault. Haga clic en el nombre de su Azure Key Vault en la lista de resultados.

10. En la hoja de Key Vault, haga clic en Políticas de acceso en la sección Configuración en el menú de navegación.
11. En la hoja Políticas de acceso, haga clic en el enlace Agregar política de acceso.
12. En el panel Agregar política de acceso, haga clic en el menú desplegable Configurar desde plantilla y seleccione la opción Administración de claves, secretos y certificados.
13. Haga clic en Seleccionar principal.
14. En el panel Principal, escriba el nombre de su Azure App Service en el cuadro de texto Seleccionar.
15. En la lista de resultados, haga clic en el nombre de su servicio de aplicaciones de Azure.
16. Haga clic en el botón Seleccionar.
17. En el panel Agregar política de acceso, haga clic en el botón Agregar.
18. En la hoja Políticas de acceso, haga clic en el botón Guardar en la esquina superior izquierda de la hoja.
19. Repita los pasos del 10 al 18 y agregue la cuenta de usuario que usa para acceder a su suscripción de Azure. Debe agregar esta política para poder depurar su código con Visual Studio. Debe asegurarse de agregar la directiva para otorgar acceso a la misma cuenta de usuario que usa para acceder a su suscripción de Azure desde Visual Studio.
20. Abra Visual Studio 2019.
21. En la ventana de bienvenida de Visual Studio 2019, en la columna Introducción, haga clic en Crear un nuevo proyecto.
22. En la ventana Crear un proyecto nuevo, en el menú desplegable Todos los idiomas, seleccione C #.
23. En el cuadro de texto Buscar plantillas, escriba **asp.net**.
24. En la lista de resultados, haga clic en Aplicación web ASP.NET (.NET Framework).
25. Haga clic en el botón Siguiente en la parte inferior derecha de la ventana.

26. En Configure Your New Project, escriba un nombre de proyecto, una ubicación y un nombre de solución para su proyecto.
27. Haga clic en el botón Crear en la parte inferior derecha de la ventana.
28. En la ventana Crear una nueva aplicación web ASP.NET, seleccione la plantilla MVC en la lista de plantillas en el medio del lado izquierdo de la ventana. MVC es para Model-View-Controller.
29. En el lado derecho de la ventana Crear una nueva aplicación web ASP.NET, en la sección Autenticación, asegúrese de que Autenticación esté configurada en Sin autenticación.
30. Haga clic en el botón Crear en la esquina inferior derecha de la ventana.
31. En la ventana de Visual Studio, haga clic en Herramientas> Administrador de paquetes NuGet> Administrar paquetes NuGet para la solución.
32. En la pestaña Administrador de paquetes NuGet, haga clic en Examinar.
33. Escriba **Microsoft.Azure.Services.AppAuthentication** y presione Entrar.
34. Haga clic en el paquete Microsoft.Azure.Services.AppAuthentication.
35. En el lado derecho de la pestaña NuGet Manager, haga clic en la casilla de verificación junto a su proyecto.
36. Haga clic en el botón Instalar.
37. En la ventana Vista previa de cambios, haga clic en Aceptar.
38. En la ventana Aceptación de licencia, haga clic en el botón Acepto.
39. Repita los pasos 32 a 38 e instale el paquete Microsoft.Azure.KeyVault.
40. Abra el archivo HomeController.cs en la carpeta Controllers.
41. Reemplace el contenido del `Index()` método con el contenido del Listado 3-25. Es posible que deba agregar los siguientes espacios de nombres al archivo HomeController.cs:

1. Microsoft.Azure.KeyVault
2. Microsoft.Azure.KeyVault.Models
3. Microsoft.Azure.Services.AppAuthentication
4. Sistema.Threading
5. System.Threading.Tasks

**Listado 3-25** Creación, eliminación, actualización y lectura de elementos de Key Vault

Haga clic aquí para ver la imagen del código

---

// C#. ASP.NET.

```
Índice de resultado de acción público ()  
  
{  
  
    string keyVaultName = "<YOUR_VAULT's_NAME>";  
  
    string vaultBaseUrl = $ "https://  
{keyVaultName}.vault.azure.net";  
  
  
  
  
    // Obtenga un token para acceder a Key Vault.  
  
    var azureServiceTokenProvider = new  
AzureServiceTokenProvider ();  
  
  
  
    // Cree un cliente de Key Vault para acceder a  
los elementos de la bóveda;  
  
    var keyVault = new KeyVaultClient (nuevo  
KeyVaultClient.AuthenticationCallback  
  
(azureServiceTokenProvider.KeyVaultTokenCallback));
```

```

    // Gestionar secretos en Key Vault.

    // Crea un nuevo secreto

    string secretName = "secret-az204";

    Task.Run (async () => aguardar
keyVault.SetSecretAsync (vaultBaseURL,
                        secretName,
                        "Este es un valor de prueba secreto")). Wait
();

    var secret = Task.Run (async () => aguardar
keyVault.GetSecretAsync

    ($ "{vaultBaseURL} / secrets / {secretName}"')). GetAwaiter (). GetResult ();

    // Actualizar un secreto existente

    Task.Run (async () => aguardar
keyVault.SetSecretAsync (vaultBaseURL,
                        secretName,
                        "Se actualizó el valor de prueba secreto")). Wait ();

    secret = Task.Run (async () => aguardar
keyVault.GetSecretAsync

    ($ "{vaultBaseURL} / secrets /
{secretName}"')). GetAwaiter (). GetResult ();

    // Eliminar el secreto

    Task.Run (async () => aguardar
keyVault.DeleteSecretAsync (vaultBaseURL,
                            secretName)). Wait ();

```

```
// Administrar certificados en Key Vault

string certName = "cert-az204";

// Crea un nuevo certificado autofirmado

var policy = new CertificatePolicy

{

    IssuerParameters = nuevos IssuerParameters

    {

        Nombre = "Yo",

    },

    KeyProperties = nuevas KeyProperties

    {

        Exportable = verdadero,

        KeySize = 2048,

        KeyType = "RSA"

    },

    SecretProperties = nuevas SecretProperties

    {

        ContentType = "aplicación / x-pkcs12"

    },

    X509CertificateProperties = nuevo

X509CertificateProperties

{

    Asunto = "CN = AZ204KEYVAULTDEMO"
```

```
        }

    } ;

        Task.Run (async () => aguardar
keyVault.CreateCertificateAsync (vaultBaseUrl,
                                certName, policy, new CertificateAttributes
{Enabled = true})). Wait ();

        // Cuando crea un nuevo certificado en Key
Vault, lleva algo de tiempo

        // antes de que esté listo.

        // Agregamos algo de tiempo de espera aquí por
simplicidad.

        Thread.Sleep (10000);

        var certificate = Task.Run (async () => aguardar
keyVault.GetCertificateAsync
                                (vaultBaseUrl, certName)). GetAwaiter () .
GetResult ();

        // Actualiza las propiedades asociadas con el
certificado.

        CertificatePolicy updatePolicy = new
CertificatePolicy

        {

            X509CertificateProperties = nuevo
X509CertificateProperties

            {

                SubjectAlternativeNames = new
SubjectAlternativeNames

                {


```

```
        DnsNames = new []
{"az204.examref.testing"}

    }

};

Task.Run (async () => aguardar
keyVault.UpdateCertificatePolicyAsync (
    vaultBaseUrl, certName,
updatePolicy)). Wait ();

Task.Run (async () => aguardar
keyVault.CreateCertificateAsync (vaultBaseUrl,
    certName)). Wait ();

Thread.Sleep (10000);

certificate = Task.Run (async () => aguardar
keyVault.GetCertificateAsync (
    vaultBaseUrl,
certName)). GetAwaiter
(). GetResult ();

Task.Run (async () => aguardar
keyVault.UpdateCertificateAsync (certificado.
```

```
CertificateIdentifier.Identifier, nulo,  
new  
CertificateAttributes {Enabled =  
    false})). Espera ();  
  
Thread.Sleep (10000);  
  
// Elimina el certificado autofirmado.  
Task.Run (async () => aguardar  
keyVault.DeleteCertificateAsync (vaultBaseURL,  
    certName)). Wait ();  
  
// Administrar claves en el Key Vault  
string keyName = "clave-az204";  
  
NewKeyParameters keyParameters = new  
NewKeyParameters  
{  
    Kty = "EC",  
    CurveName = "SECP256K1",  
    KeyOps = new [] {"firmar", "verificar"}  
};  
  
Task.Run (async () => aguardar  
keyVault.CreateKeyAsync (vaultBaseURL, keyName,  
    keyParameters)). Wait ();
```

```

        var key = Task.Run (async () => aguardar
keyVault.GetKeyAsync (vaultBaseUrl,
                      keyName)).

GetAwaiter (). GetResult ();

// Actualizar claves en Key Vault

Task.Run (async () => aguardar
keyVault.UpdateKeyAsync (vaultBaseUrl, keyName,
                        Atributos clave nuevos y
nulos {Expires = DateTime.UtcNow.
                     AddYears (1)})). Wait ();

key = Task.Run (async () => aguardar
keyVault.GetKeyAsync (vaultBaseUrl,
                      keyName)). GetAwaiter
(). GetResult ();

// Eliminar claves del Key Vault

Task.Run (async () => aguardar
keyVault.DeleteKeyAsync (vaultBaseUrl, keyName)). Esperar ();

volver Ver ();
}

```

En este punto, debería poder ejecutar el ejemplo. Como no realizó ninguna modificación en ninguna vista, no debería poder ver ningún cambio en Azure Key Vault. Para poder ver cómo este código crea, lee,

modifica y elimina los diferentes tipos de elementos en su Azure Key Vault, debe establecer algunos puntos de interrupción:

1. Agregue un punto de interrupción a las siguientes líneas:

[Haga clic aquí para ver la imagen del código](#)

```
string secretName = "secret-az204";  
  
string certName = "cert-az204";  
  
string keyName = "clave-az204";
```

2. Abra Azure Key Vault en Azure Portal, como se muestra en el paso 9 del procedimiento anterior.

3. En la hoja de Azure Key Vault, haga clic en Secretos en la sección Configuración del menú de navegación.

4. En Visual Studio, presione F5 para depurar su proyecto.

5. Cuando llegue al punto de interrupción, presione F10 y vuelva al portal de Azure para ver los resultados. Debe usar el botón Actualizar para ver los cambios en Azure Key Vault.

#### **Nota Acceso prohibido**

Si obtiene un Error de acceso prohibido mientras depura su aplicación en Visual Studio, asegúrese de haber creado una Política de acceso para la cuenta de usuario que ha configurado en Visual Studio para conectarse con su suscripción de Azure. Debe asegurarse de que la política de acceso otorgue todos los privilegios necesarios a los diferentes tipos de objetos en Azure Key Vault. Asegúrese también de que la cuenta que está usando para el desarrollo esté correctamente autenticada en Azure Active Directory. Compruebe su cuenta de desarrollo en Herramientas> Opciones> Autenticación del servicio de Azure. Si hay un enlace Volver a autenticar debajo de su cuenta de desarrollo, haga clic en el enlace para autenticarse nuevamente.

Cuando trabaja con la API de KeyVault, debe crear un `KeyVaultClient` objeto que sea responsable de la comunicación con los servicios de Azure Key Vault. Como se describe en el ejemplo de la sección "Implementar la identidad de servicio administrado (MSI) / autenticación de entidad de servicio", debe obtener un token de acceso para autenticar su entidad de servicio o usuario en Azure Key Vault. El siguiente fragmento de código muestra cómo realizar esta autenticación:

[Haga clic aquí para ver la imagen del código](#)

```
var azureServiceTokenProvider = new  
AzureServiceTokenProvider ();
```

```

var keyVault = new KeyVaultClient (nuevo
KeyVaultClient.AuthenticationCallback (
    azureServiceTokenProvider.KeyVaultTokenCallback));

```

Ahora puede usar la variable keyVault para trabajar con los diferentes tipos de elementos. La API de KeyVault proporciona métodos especializados para cada tipo de elemento. De esta forma, debe usar el `SetSecretAsync()` método para crear un nuevo secreto en Azure Key Vault. El siguiente fragmento de código muestra cómo crear un nuevo secreto:

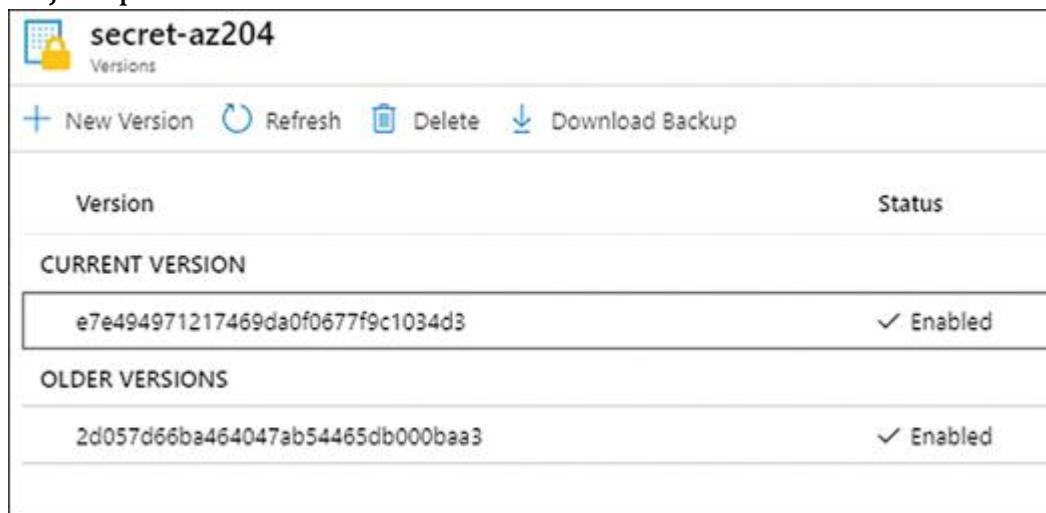
[Haga clic aquí para ver la imagen del código](#)

```

Task.Run (async () => await keyVault.SetSecretAsync
(vaultBaseUrl, secretName, "Este es un
valor de prueba secreto")).Wait ();

```

Si intenta crear un nuevo secreto, clave o certificado con el mismo nombre de un objeto que ya existe en la bóveda, está creando una nueva versión de ese objeto, como se muestra en la [Figura 3-10](#). La única excepción a esta regla es si ha habilitado la eliminación temporal en Azure Key Vault e intenta crear un nuevo secreto con el mismo nombre que un objeto eliminado. En esa situación, obtiene una excepción de colisión. Puede hacer clic en cada versión para revisar las propiedades del objeto para esa versión.



The screenshot shows the Azure Key Vault interface for a secret named 'secret-az204'. At the top, there's a 'Versions' section with buttons for 'New Version', 'Refresh', 'Delete', and 'Download Backup'. Below this is a table with columns 'Version' and 'Status'. The first row, labeled 'CURRENT VERSION', contains the value 'e7e494971217469da0f0677f9c1034d3' and 'Enabled'. The second row, labeled 'OLDER VERSIONS', contains the value '2d057d66ba464047ab54465db000baa3' and 'Enabled'. Both rows have checkmarks in the 'Status' column.

Version	Status
<b>CURRENT VERSION</b>	
e7e494971217469da0f0677f9c1034d3	✓ Enabled
<b>OLDER VERSIONS</b>	
2d057d66ba464047ab54465db000baa3	✓ Enabled

**Figura 3-10** Un objeto secreto con diferentes versiones

La mayoría de los métodos de la API de KeyVault que funcionan con elementos requieren la URL de la bóveda y el nombre del elemento al que desea acceder. En este ejemplo, define una variable con el valor correcto al comienzo del `Index()` método, como se muestra en el siguiente fragmento de código:

[Haga clic aquí para ver la imagen del código](#)

```
string keyVaultName = "<YOUR_VAULT's_NAME>";  
  
string vaultBaseUrl = $ "https:// {keyVaultName}  
.vault.azure.net";
```

Estos métodos suelen estar sobrecargados para aceptar un identificador de objeto en lugar de la URL base de la bóveda y el nombre del objeto. El identificador tiene la siguiente forma:

[Haga clic aquí para ver la imagen del código](#)

```
https:// {keyvault-name} .vault.azure.net / {object-type}  
/ {object-name} / {object-version}
```

Dónde:

- *Keyvault-name* es el nombre del *almacén* de claves donde se almacena el objeto.
- *-Tipo de objeto* es el tipo de objeto que desea trabajar. Este valor puede ser secretos, claves o certificados.
- *Object-name* es el nombre que le da al objeto en la bóveda.
- *Object-version* es la versión del objeto al que desea acceder.

La creación de una clave o certificado utiliza un enfoque ligeramente diferente al que usó para crear un secreto. Las claves y los certificados son objetos más complejos y requieren alguna configuración adicional para crearlos. El siguiente fragmento de código extraído del [Listado 3-25](#) muestra cómo crear un nuevo certificado autofirmado en Azure Key Vault:

[Haga clic aquí para ver la imagen del código](#)

```
// Crea un nuevo certificado autofirmado  
  
var policy = new CertificatePolicy  
  
{
```

```
IssuerParameters = nuevos IssuerParameters
{
    Nombre = "Yo",
},
KeyProperties = nuevas KeyProperties
{
    Exportable = verdadero,
    KeySize = 2048,
    KeyType = "RSA"
},
SecretProperties = nuevas SecretProperties
{
    ContentType = "aplicación / x-pkcs12"
},
X509CertificateProperties = nuevo
X509CertificateProperties
{
    Asunto = "CN = AZ204KEYVAULTDEMO"
}
};

Task.Run (async () => aguardar
keyVault.CreateCertificateAsync (vaultBaseUrl, certName,
política, nuevos atributos de certificado {Enabled =
true})). Wait ();
```

Debe crear un objeto CertificatePolicy antes de poder crear el certificado. Una política de certificado es un objeto que define las propiedades de cómo crear un certificado y cualquier versión nueva asociada con el objeto de certificado. Utilice este objeto de política de certificado como parámetro del `CreateCertificateAsync()` método. Si necesita modificar alguna propiedad de un certificado existente, debe definir una nueva política de certificado, actualizar la política usando el `UpdateCertificatePolicyAsync()` método y crear una nueva versión de certificado usando el `CreateCertificateAsync()` método, como se muestra en el siguiente fragmento de código:

[Haga clic aquí para ver la imagen del código](#)

```
// Actualiza las propiedades asociadas con el certificado.

CertificatePolicy updatePolicy = new CertificatePolicy

{

    X509CertificateProperties = nuevo
X509CertificateProperties

    {

        SubjectAlternativeNames = new
SubjectAlternativeNames

        {

            DnsNames = new [] {"az204.examref.testing"}

        }

    }

};

Task.Run (async () => aguardar
keyVault.UpdateCertificatePolicyAsync (vaultBaseUrl,
certName,

updatePolicy)). Wait ();

Task.Run (async () => aguardar
keyVault.CreateCertificateAsync (vaultBaseUrl, certName))
```

```
. Esperar();
```

Eliminar un objeto del almacén de claves es bastante sencillo; sólo tiene que proporcionar la dirección URL base de la bóveda y el nombre del objeto a

la `DeleteSecretAsync()`, `DeleteCertificateAsync()` o `DeleteKeyAsync()` método. Azure Key Vault también admite operaciones de eliminación temporal en los objetos protegidos o en el propio almacén. Esta opción está activada de forma predeterminada. Cuando elimina temporalmente un objeto o una bóveda, el proveedor de Azure Key Vault los marca automáticamente como eliminados, pero retiene el objeto o la bóveda durante un período predeterminado de 90 días. Esto significa que puede recuperar el objeto eliminado más tarde si es necesario.

#### *¿Necesita más revisión? Más detalles sobre claves, secretos y certificados*

Puede encontrar más información sobre los detalles de los diferentes tipos de objetos que están disponibles en el servicio Azure Key Vault revisando el artículo en <https://docs.microsoft.com/en-us/azure/key-vault/about-keys-secrets-and-certificates>.



#### *Sugerencia para el examen*

El tipo de información que normalmente almacena en Azure Key Vault es información esencial que debe mantenerse en secreto, como contraseñas, cadenas de conexión, claves privadas y cosas por el estilo. Al configurar el acceso a su Key Vault, revise cuidadosamente el nivel de acceso que otorga al principal de seguridad. Como práctica recomendada, siempre debe aplicar el principio de privilegio mínimo. Otorga acceso a los diferentes niveles en un Key Vault mediante la creación de políticas de acceso.

#### *Implementar identidades administradas para recursos de Azure*

Cuando diseña su aplicación, normalmente identifica los diferentes servicios o sistemas de los que depende su aplicación. Por ejemplo, su aplicación puede necesitar conectarse a una base de datos de Azure SQL para almacenar datos o puede necesitar conectarse a Azure Event Hub para leer mensajes de otros servicios. En todas estas situaciones, existe una necesidad común de autenticarse con el servicio antes de poder acceder a él. En el caso de la base de datos SQL de Azure, debe usar una

cadena de conexión; Si necesita conectarse a un centro de eventos de Azure, debe usar una combinación de publicadores de eventos y tokens de firma de acceso compartido (SAS).

El inconveniente de este enfoque es que debe almacenar una credencial de seguridad, un token o una contraseña para poder autenticarse en el servicio al que desea acceder. Esto es un inconveniente porque puede encontrar que esta información se almacena en las computadoras de los desarrolladores o se registra en el control de fuente por error. Puede abordar la mayoría de estas situaciones mediante el uso de Azure Key Vault, pero su código aún debe autenticarse en Azure Key Vault para obtener la información para acceder a los otros servicios.

Afortunadamente, Azure Active Directory (Azure AD) proporciona las identidades administradas para los recursos de Azure (anteriormente conocido como identidad de servicio administrado) que elimina la necesidad de usar credenciales para autenticar su aplicación en cualquier servicio de Azure que admita la autenticación de Azure AD. Esta característica crea automáticamente una identidad administrada que puede usar para autenticarse en cualquier servicio que admita la autenticación de Azure AD sin necesidad de proporcionar ninguna credencial.

Cuando trabaja con identidades administradas, puede trabajar con dos tipos diferentes:

- **Identidades administradas asignadas por el sistema** Son identidades que Azure habilita automáticamente cuando crea una instancia de servicio de Azure, como una máquina virtual (VM) de Azure o un almacén de lago de datos de Azure. Azure crea una identidad asociada con la nueva instancia y la almacena en el inquilino de Azure AD asociado con la suscripción donde creó la instancia de servicio. Si decide eliminar la instancia de servicio, Azure elimina automáticamente la instancia administrada asociada con la instancia de servicio almacenada en el inquilino de Azure AD.
- **Identidades administradas asignadas por el usuario** Puede crear sus identidades administradas en el inquilino de Azure AD asociado con su suscripción de Azure. Puede asociar este tipo de identidad administrada a una o más instancias de servicio. El ciclo de vida de la identidad administrada es independiente de la instancia de servicio. Esto significa que si

elimina la instancia de servicio, la identidad administrada asignada por el usuario permanece en el inquilino de Azure AD. Debe eliminar la identidad administrada manualmente.

Por lo general, usa las identidades administradas asignadas por el sistema cuando su carga de trabajo está contenida dentro del mismo recurso de Azure, o necesita identificar de forma independiente cada una de las instancias de servicio, como máquinas virtuales. Por otro lado, si necesita otorgar acceso a una carga de trabajo que se distribuye en diferentes recursos o necesita preautorizar un recurso como parte de un flujo de aprovisionamiento, debe usar identidades administradas asignadas por el usuario.

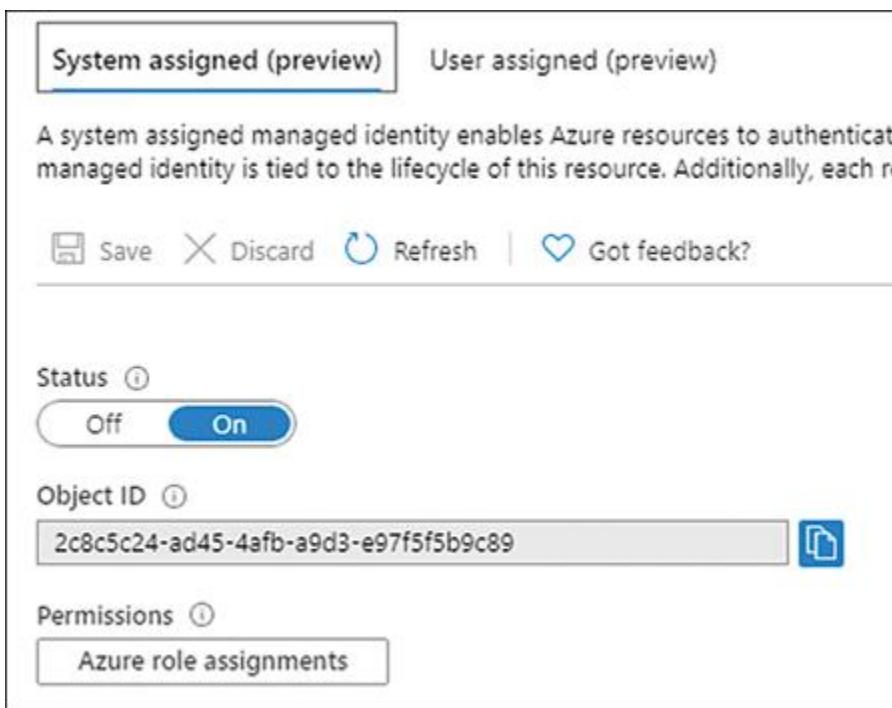
Cuando trabaje con identidades administradas, debe tener en cuenta tres conceptos:

- **Id. De cliente** Este es un identificador único generado por Azure AD. Este ID asocia la aplicación y la entidad de servicio durante su aprovisionamiento inicial.
- **ID principal** Este es el ID del principal de servicio asociado con la identidad administrada. Una entidad de servicio y una identidad administrada están estrechamente vinculadas, pero son objetos diferentes. La entidad de servicio es el objeto que usa para otorgar acceso basado en roles a un recurso de Azure.
- **Servicio de metadatos de instancia de Azure (IMDS)** Cuando usa identidades administradas en una máquina virtual de Azure, puede usar el IMDS para solicitar un token de acceso de OAuth desde su aplicación implementada dentro de la máquina virtual. El IMDS es un punto final REST al que puede acceder desde su VM utilizando una dirección IP no enrutable (169.254.169.254).

El siguiente ejemplo muestra cómo crear una identidad asignada por el sistema en un servicio de aplicaciones de Azure y cómo usar esta identidad administrada desde su código para acceder a Azure Key Vault. Para este ejemplo, debe tener un Azure App Service vacío, un Azure Key Vault y al menos un elemento en Azure Key Vault. También debe tener su Visual Studio conectado a la suscripción de Azure donde ha configurado Azure Key Vault.

1. Abra Azure Portal en <https://portal.azure.com> .

2. En el cuadro de texto de búsqueda en la parte superior de Azure Portal, escriba el nombre de su aplicación web de Azure. Si no tiene una aplicación web de Azure, puede crear una nueva aplicación web de Azure mediante el procedimiento en <https://docs.microsoft.com/en-in/azure/app-service/app-service-web-empezar-dotnet>.
3. En la hoja Azure Web App Service, haga clic en el elemento de menú Identidad en la sección Configuración.
4. En el control del interruptor de estado, haga clic en la opción Activado.
5. Haga clic en el botón Guardar.
6. En el cuadro de diálogo Habilitar identidad administrada asignada por el sistema, haga clic en el botón Sí.
7. Una vez que habilita la identidad administrada asignada por el sistema, obtiene el ID de objeto o principal, como se muestra en la [Figura 3-11](#).



**Figura 3-11** Identidad administrada asignada por el sistema

8. Abra Visual Studio 2019.
9. En la ventana de bienvenida de Visual Studio 2019, en la columna Introducción, haga clic en Crear un nuevo proyecto.

10. En la ventana Crear un proyecto nuevo, en el menú desplegable, menú desplegable Todos los idiomas, seleccione C #.
11. En el cuadro de texto Buscar plantillas, escriba **asp.net**.
12. En la lista de resultados, haga clic en Aplicación web ASP.NET (.NET Framework).
13. Haga clic en el botón Siguiente en la parte inferior derecha de la ventana.
14. En Configure su nuevo proyecto, escriba un nombre de proyecto, una ubicación y un nombre de solución para su proyecto.
15. Haga clic en el botón Crear en la parte inferior derecha de la ventana.
16. En la ventana Crear una nueva aplicación web ASP.NET, seleccione la plantilla MVC en la lista de plantillas en el medio del lado izquierdo de la ventana. MVC es para Model-View-Controller.
17. En el lado derecho de la ventana Crear una nueva aplicación web ASP.NET, en la sección Autenticación, asegúrese de que Autenticación esté configurada en Sin autenticación.
18. Haga clic en el botón Crear en la parte inferior derecha de la ventana.
19. En la ventana de Visual Studio, haga clic en Herramientas> Administrador de paquetes NuGet> Administrar paquetes NuGet para la solución.
20. En la pestaña Administrador de paquetes NuGet, haga clic en Examinar.
21. Escriba **Microsoft.Azure.Services.AppAuthentication** y presione Entrar.
22. Haga clic en el paquete Microsoft.Azure.Services.AppAuthentication.
23. En el lado derecho de la pestaña NuGet Manager, haga clic en la casilla de verificación junto a su proyecto.
24. Haga clic en el botón Instalar.
25. En la ventana Vista previa de cambios, haga clic en Aceptar.

26. En la ventana Aceptación de licencia, haga clic en el botón Acepto.
27. Repita los pasos del 20 al 26 e instale el paquete Microsoft.Azure.KeyVault.
28. Abra el archivo HomeController.cs en la carpeta Controllers.
29. Agregue las siguientes declaraciones al archivo HomeController.cs:

[Haga clic aquí para ver la imagen del código](#)

```
utilizando Microsoft.Azure.KeyVault;  
  
utilizando Microsoft.Azure.Services.AppAuthentication;  
  
usando System.Threading.Tasks;
```

30. Reemplace el contenido del método Index () con el contenido del Listado 3-26. Los fragmentos de código cruciales relacionados con el acceso a Azure Key Vault están resaltados en negrita.

**Listado 3-26** Obteniendo un secreto del almacén de claves

[Haga clic aquí para ver la imagen del código](#)

---

```
// C#. ASP.NET.  
  
string keyVaultName = "<PUT_YOUR_KEY_VAULT_NAME_HERE>";  
  
string secretName = "<PUT_YOUR_SECRET_NAME_HERE>";  
  
  
// Obtenga un token para acceder a Key Vault.  
  
var azureServiceTokenProvider = new  
AzureServiceTokenProvider ();  
  
  
  
// Cree un cliente de Key Vault para acceder a los elementos  
de la bóveda.  
  
var keyVault = new KeyVaultClient (nuevo  
KeyVaultClient.AuthenticationCallback (
```

```

    azureServiceTokenProvider.KeyVaultTokenCallback));
}

var secret = Task.Run (async () => aguardar
keyVault.GetSecretAsync (
    $"https:// {keyVaultName}
.vault.azure.net / secrets / {secretName}"))
    .GetAwaiter ().GetResult ();

ViewBag.KeyVaultName = keyVaultName;
ViewBag.keyName = secretName;
ViewBag.secret = secret.Value;

volver Ver ();

```

Como puede ver, este código es bastante similar al código del [Listado 3-25](#). El motivo es que usó identidades administradas para obtener acceso a Azure Key Vault en el ejemplo del [Listado 3-25](#). Antes de poder acceder a Azure Key Vault, debe obtener un token de OAuth mediante la clase AzureServiceTokenProvider. Luego, puede crear su cliente de Azure Key Vault y obtener cualquier elemento almacenado en el almacén. Cuando cree el cliente de Azure Key Vault, asegúrese de proporcionar KeyVaultTokenCallback. Incluso si obtiene un token de acceso válido, aún debe otorgar acceso a su aplicación Azure App Service en Azure Key Vault.

1. Abra el archivo Vistas> Inicio> Index.cshtml.
2. Agregue el contenido del [Listado 3-27](#) al final del archivo.

**Listado 3-27** Adición de información secreta a la página de inicio

[Haga clic aquí para ver la imagen del código](#)

---

// C#. ASP.NET.

```

<div class = "fila">

    <div class = "col-lg-12">

        <dl class = "dl-horizontal">

            <dt> Nombre de Key Vault: </dt>

            <dd> @ ViewBag.keyVaultName </dd>

            <dt> Nombre de clave: </dt>

            <dd> @ ViewBag.keyName </dd>

            <dt> Clave secreta: </dt>

            <dd> @ ViewBag.secret </dd>

        </dl>

    </div>

</div>

```

En este punto, puede ejecutar su proyecto y ver los resultados. Dependiendo de las políticas de acceso definidas en su almacén de claves de Azure, es posible que su usuario de Azure ya tenga acceso a los secretos almacenados en el almacén de claves. En ese caso, debería poder acceder al secreto almacenado en Azure Key Vault. Si obtiene una excepción al ejecutar la aplicación web, es muy probable que no tenga acceso a Azure Key Vault. Los siguientes pasos muestran cómo otorgar acceso a su aplicación Azure App Service en Azure Key Vault.

1. Abra Azure Portal (<https://portal.azure.com>).
2. Escriba el nombre de Azure Key Vault en el cuadro de texto de búsqueda en la parte superior de Azure Portal. Si aún no tiene un Azure Key Vault y necesita crear uno nuevo, puede utilizar el procedimiento en <https://docs.microsoft.com/en-us/azure/key-vault/quick-create-portal>.
3. En la hoja de Azure Key Vault, haga clic en Directivas de acceso en la sección Configuración.

4. En la hoja Políticas de acceso, haga clic en Agregar nuevo.
5. En la página Agregar política de acceso, seleccione Administración secreta en el menú desplegable Configurar desde plantilla.
6. Haga clic en el control Seleccionar principal.
7. En el panel Principal, escriba el nombre de su Azure App Service en el cuadro de texto Seleccionar. Su Azure App Service debería aparecer en la lista debajo del cuadro de texto.
8. Haga clic en el nombre de su App Service en la lista debajo del cuadro de texto Seleccionar.
9. Haga clic en el botón Seleccionar en la parte inferior del panel.
10. Haga clic en el botón Agregar en la parte inferior de la hoja Agregar política de acceso.
11. Haga clic en el botón Guardar en la parte superior de la hoja Políticas de acceso.
12. En la ventana de Visual Studio, haga clic con el botón derecho en el nombre de su proyecto en la ventana del Explorador de soluciones.
13. En el menú contextual, haga clic en Publicar.
14. En la ventana Elija un destino de publicación, asegúrese de que App Service esté seleccionado en el lado izquierdo de la ventana.
15. En la sección Azure App Service, haga clic en Seleccionar existente.
16. Haga clic en el botón Crear perfil en la esquina inferior derecha de la ventana.
17. En la ventana de App Service, en la vista de árbol en la parte inferior de la ventana, busque su App Service y haga clic en él.
18. Haga clic en el botón Aceptar.

En este punto, Visual Studio comienza a publicar su aplicación web en el Servicio de aplicaciones de Azure seleccionado. Cuando finalice la operación de publicación, debería poder ver su aplicación web mostrando el contenido del secreto almacenado en su Key Vault.



### *Sugerencia para el examen*

Puede configurar dos tipos diferentes de identidades administradas: asignadas por el sistema y asignadas por el usuario. Las identidades administradas asignadas por el sistema están vinculadas a la instancia del servicio. Si elimina la instancia de servicio, la identidad administrada asignada por el sistema también se elimina automáticamente. Puede asignar las mismas identidades administradas asignadas por el usuario a varias instancias de servicio.

## RESUMEN DEL CAPÍTULO

---

- La autenticación es el acto de demostrar que un usuario es quien dice ser.
- Un usuario se autentica proporcionando cierta información que solo el usuario conoce.
- Existen varios mecanismos de autenticación que brindan diferentes niveles de seguridad.
- Algunos de los mecanismos de autenticación se basan en formularios, tokens o certificados.
- El uso de la autenticación basada en formularios requiere que su aplicación almacene las contraseñas de sus usuarios.
- La autenticación basada en formularios requiere HTTPS para que el proceso de autenticación sea más seguro.
- Con la autenticación basada en token, puede delegar la autorización a proveedores de autenticación de terceros.
- Puede agregar inicios de sesión sociales a su aplicación mediante la autenticación basada en tokens.
- La autenticación multifactor es un mecanismo de autenticación que requiere que los usuarios proporcionen más de una pieza de información que solo el usuario conoce.
- Puede implementar fácilmente la autenticación multifactor mediante Azure Active Directory.

- Hay cuatro actores principales en la autenticación OAuth: cliente, servidor de recursos, propietario de recursos y servidor de autenticación.
- El propietario del recurso debe autenticar al cliente antes de enviar la concesión de autorización.
- El token de acceso otorga acceso al recurso alojado en el servidor de recursos.
- La concesión de autorización o el código de autorización otorga al cliente los derechos necesarios para solicitar un token de acceso al servidor de autorización.
- El cliente usa el token de actualización para obtener un nuevo token de acceso cuando caduca sin necesidad de solicitar un nuevo código de autorización.
- El token web JSON es la implementación más extendida de tokens OAuth.
- Las firmas de acceso compartido (SAS) es un mecanismo de autenticación para otorgar acceso a cuentas de almacenamiento de Azure sin compartir claves de cuenta.
- Los tokens de firmas de acceso compartido (SAS) deben estar firmados.
- Hay tres tipos de token SAS: delegación de usuario, cuenta y servicio SAS.
- Los tokens SAS de delegación de usuarios se firman con una clave asignada a un usuario de Azure Active Directory.
- La cuenta y el servicio SAS se firman con la clave de la cuenta de Azure Storage.
- Puede ocultar los detalles de los tokens SAS de la URL utilizando Políticas de acceso almacenadas.
- Los tokens de firma de acceso compartido proporcionan un control de acceso detallado a sus cuentas de almacenamiento de Azure.
- Puede crear un token SAS para los niveles de servicio, contenedor y artículo.

- Debe registrar aplicaciones en Azure Active Directory para poder autenticar a los usuarios mediante su inquilino.
- Hay tres tipos de cuentas admitidas para la autenticación: cuentas solo en el directorio de la organización, cuentas en cualquier directorio de la organización y cuentas de Microsoft.
- Debe proporcionar una URL de retorno para autenticar su aplicación al solicitar la autenticación de usuario.
- Debe configurar un secreto o un certificado cuando su aplicación necesite acceder a información en otras API.
- La autorización de control de acceso basado en roles (RBAC) proporciona un control de acceso detallado a los recursos.
- Una entidad de seguridad es una entidad a la que puede otorgar privilegios.
- Las entidades de seguridad son usuarios, grupos, entidades de servicio e identidades administradas.
- Un permiso es una acción que una entidad de seguridad puede realizar con un recurso.
- Una definición de rol, o rol, es un grupo de permisos.
- Un alcance es un nivel en el que puede asignar un rol.
- Una asociación de roles es una relación entre un principal de seguridad, un rol y un ámbito.
- Hay cuatro ámbitos: grupos de administración, suscripción, grupo de recursos y recursos.
- Puede centralizar la configuración de su aplicación distribuida mediante Azure App Configuration.
- Azure App Configuration almacena la información mediante pares clave-valor.
- Los valores de la configuración de la aplicación de Azure están cifrados.
- Azure Key Vault proporciona mejor seguridad que el servicio de configuración de aplicaciones de Azure.
- El límite de tamaño para una configuración de aplicación de Azure es 10,000, incluida la clave, la etiqueta y el valor.

- Puede crear referencias desde elementos de configuración de la aplicación de Azure a elementos de Azure Key Vault.
- Azure Key Vault le permite almacenar tres tipos de objetos: claves, secretos y certificados.
- Debe usar la autenticación de identidades administradas para acceder a Azure Key Vault.
- Debe definir una política de certificado antes de crear un certificado en Azure Key Vault.
- Si importa un certificado a Azure Key Vault, se crea automáticamente una directiva de certificado predeterminada.

## EXPERIMENTO MENTAL

---

En este experimento mental, demuestre sus habilidades y conocimiento de los temas cubiertos en este capítulo. Puede encontrar respuestas a este experimento mental en la siguiente sección.

Está desarrollando una aplicación web para su empresa. La aplicación se encuentra en las primeras etapas de desarrollo. Esta aplicación es una aplicación interna que será utilizada únicamente por los empleados de la empresa. Su empresa utiliza Office 365 conectado con su dominio de Active Directory. La aplicación necesita usar información de Office 365. Responda las siguientes preguntas sobre la implementación de seguridad de esta aplicación:

1. Los empleados deben poder acceder a la aplicación utilizando el mismo nombre de usuario y contraseña que utilizan para acceder a Office 365. ¿Qué debe hacer?
2. Utiliza Azure App Services para desarrollar la aplicación. Debe asegurarse de que la aplicación web pueda acceder a otros servicios de Azure sin usar credenciales en su código. ¿Qué deberías hacer?
3. Debe asegurarse de que la configuración de su aplicación se almacene en el almacenamiento central. También debe proporcionar la mejor seguridad para la información confidencial, como cadenas de conexión y contraseñas. ¿Qué deberías hacer?

## RESPUESTAS DEL EXPERIMENTO MENTAL

---

Esta sección contiene la solución al experimento mental. Cada respuesta explica por qué la opción de respuesta es correcta.

1. Debe usar la autenticación OAuth con Azure Active Directory (Azure AD). Si desea que su aplicación pueda usar la autenticación OAuth de Azure AD, debe registrar su aplicación en su inquilino de Azure AD. Debido a que su aplicación necesita acceso a la información en Office 365, también debe crear un secreto de cliente antes de poder acceder a la API de Microsoft Graph. Cuando conecta Office 365 con un dominio de Active Directory (AD), los usuarios del dominio de AD pueden autenticarse en Office 365 con el mismo nombre de usuario y contraseña que usan en el dominio de AD. Office 365 usa un inquilino de Azure AD para administrar las identidades de los usuarios en la suscripción. Su organización ya ha configurado la sincronización entre AD y Office 365 y Azure AD. Al usar la autenticación OAuth con Azure AD,
2. Debe utilizar la autenticación de Identidad de servicio administrado (MSI). Con la característica, Azure autentica los servicios en función de una entidad de servicio configurada en una instancia de servicio. Puede usar la autenticación MSI con servicios que admiten la autenticación de Azure AD, como Azure Key Vault o Azure SQL Databases. Debe habilitar una identidad administrada asignada por el sistema o asignada por el usuario en su Azure App Service. Con MSI, Azure SQL Database autentica la identidad asignada a su Azure App Service sin necesidad de que proporcione ninguna contraseña.
3. Debe crear una tienda de configuración de aplicaciones de Azure. Este es el servicio adecuado para almacenar de forma segura los ajustes de configuración de su aplicación en un almacenamiento centralizado. Aunque el almacén de configuración de la aplicación de Azure proporciona almacenamiento seguro al cifrar el valor de los pares clave-valor que representan su configuración, debe usar las referencias de Key Vault en su almacén de configuración de la aplicación de Azure para esa información confidencial que requiere un mayor nivel de seguridad. Azure Key Vault usa cifrado basado en hardware para almacenar claves, secretos y certificados

# Capítulo 4. Supervisar, solucionar problemas y optimizar las soluciones de Azure

Brindar una buena experiencia a tus usuarios es uno de los factores clave para el éxito de tu aplicación. Varios factores afectan la experiencia del usuario, como un buen diseño de interfaz de usuario, facilidad de uso, buen rendimiento y baja tasa de fallas. Puede asegurarse de que su aplicación funcionará bien asignando más recursos a su aplicación, pero si no hay suficientes usuarios usando su aplicación, es posible que esté desperdiando recursos y dinero.

Para asegurarse de que su aplicación funcione correctamente, debe implementar un mecanismo de supervisión que le ayude a obtener información sobre el comportamiento de su aplicación. Esto es especialmente importante durante períodos de uso pico o fallas. Azure proporciona varias herramientas que le ayudan a supervisar, solucionar problemas y mejorar el rendimiento de su aplicación.

## Habilidades cubiertas en este capítulo:

- [Habilidad 4.1: Integrar el almacenamiento en caché y la entrega de contenido dentro de las soluciones](#)
- [Habilidad 4.2: Soluciones de instrumentos para respaldar el monitoreo y el registro](#)

## HABILIDAD 4.1: INTEGRAR EL ALMACENAMIENTO EN CACHÉ Y LA ENTREGA DE CONTENIDO DENTRO DE LAS SOLUCIONES

---

Cualquier aplicación web que implemente ofrece dos tipos de contenido: dinámico y estático.

- El contenido dinámico es el tipo de contenido que cambia según la interacción del usuario. Un ejemplo de contenido dinámico

es un tablero con varios gráficos o una lista de movimientos de usuarios en una aplicación bancaria.

- El contenido estático es el mismo para todos los usuarios de la aplicación. Las imágenes y los archivos PDF son ejemplos de contenido estático (siempre que no se generen dinámicamente) que los usuarios pueden descargar desde su aplicación.

Si los usuarios de su aplicación acceden a ella desde varias ubicaciones en todo el mundo, puede mejorar el rendimiento de la aplicación entregando el contenido desde la ubicación más cercana al usuario. Para el contenido estático, puede mejorar el rendimiento copiando el contenido en diferentes servidores de caché distribuidos por todo el mundo. Con esta técnica, los usuarios pueden recuperar el contenido estático de la ubicación más cercana con menor latencia, lo que mejora el rendimiento de su aplicación.

Para contenido dinámico, puede usar software de caché para almacenar los datos más accedidos. Esto significa que su aplicación devuelve la información de la caché, que es más rápido que reprocesar los datos o obtenerlos del sistema de almacenamiento.

### **Esta habilidad cubre cómo**

- Desarrollar código para implementar CDN en soluciones
- Configure las políticas de caducidad y caché para las cachés de FrontDoor, CDN y Redis
- Almacenar y recuperar datos en Azure Redis Cache

#### *Desarrollar código para implementar CDN en soluciones*

Una red de entrega de contenido (CDN) es un grupo de servidores distribuidos en diferentes ubicaciones en todo el mundo que pueden entregar contenido web a los usuarios. Debido a que la CDN tiene servidores distribuidos en varias ubicaciones, cuando un usuario realiza una solicitud a la CDN, la CDN entrega el contenido desde el servidor más cercano al usuario.

La principal ventaja de usar Azure CDN con su aplicación es que Azure CDN almacena en caché el contenido estático de su aplicación. Cuando un usuario realiza una solicitud a su aplicación, la CDN almacena el contenido estático, como imágenes, documentos y archivos de hojas de estilo. Cuando un segundo usuario de la misma ubicación que el primer

usuario accede a su aplicación, la CDN entrega el contenido almacenado en caché, lo que evita que su servidor web entregue el contenido estático. Puede utilizar soluciones CDN de terceros, como Verizon o Akamai, con Azure CDN.

Para usar Azure CDN con su solución, debe configurar un perfil. Este perfil contiene la lista de puntos finales en su aplicación que se incluirían en la CDN. El perfil también configura el comportamiento de la entrega de contenido y el acceso de cada punto final configurado. Cuando configura un perfil de Azure CDN, debe elegir entre usar la CDN de Microsoft o usar las CDN de Verizon o Akamai.

Puede configurar tantos perfiles como necesite para agrupar sus puntos finales en función de diferentes criterios, como dominio de Internet, aplicación web o cualquier otro criterio. Tenga en cuenta que los niveles de precios de Azure CDN se aplican a nivel de perfil, por lo que puede configurar diferentes perfiles con diferentes características de precios. Al igual que con cualquier solución CDN en el mundo real, necesita una aplicación web para ejecutar los procedimientos y demostraciones a lo largo de esta habilidad. El siguiente procedimiento muestra cómo crear una aplicación web básica en Visual Studio y publicarla en una aplicación web de Azure. Puede usar esta aplicación web de Azure en todos los ejemplos del resto de esta habilidad:

1. Abra Visual Studio 2019 en su computadora.
2. En la ventana de inicio de Visual Studio 2019, en la columna denominada Comenzar, haga clic en el vínculo Continuar sin código en la parte inferior de la columna.
3. Haga clic en el menú Herramientas y elija Obtener herramientas y funciones. Verifique que la sección ASP.NET And Web Development In The Web & Cloud esté marcada.
4. En la ventana de Visual Studio 2019, seleccione Archivo> Nuevo> Proyecto para abrir la ventana Nuevo proyecto.
5. En la ventana Crear un nuevo proyecto, seleccione C # en el menú desplegable debajo del cuadro de texto Buscar plantillas en la parte superior derecha de la ventana.
6. En el menú desplegable Todos los tipos de proyectos, seleccione Web.

7. En la lista de plantillas en el lado derecho de la ventana, seleccione Aplicación web ASP.NET Core.
8. En la ventana Configure Your New Project, complete los siguientes pasos:
  1. Seleccione un nombre para el proyecto.
  2. Ingrese una ruta para la ubicación de la solución.
  3. En el menú desplegable Solución, seleccione Crear una nueva solución.
  4. Ingrese un nombre para la solución.
9. Haga clic en el botón Crear en la esquina inferior derecha de la ventana Configure su nuevo proyecto. Esto abre la ventana Crear una nueva aplicación web ASP.NET Core.
10. En la ventana Crear una nueva aplicación web ASP.NET Core, asegúrese de que los siguientes valores estén seleccionados en los dos menús desplegables en la parte superior de la ventana:
  0. .NET Core
  1. ASP.NET Core 3.1
11. Seleccione Aplicación web en el área Plantillas de proyecto en el centro de la ventana.
12. Desmarque la opción Configurar para HTTPS en la parte inferior derecha de la ventana.
13. Haga clic en el botón Crear en la esquina inferior derecha de la ventana Crear una nueva aplicación web ASP.NET Core.
14. En el lado derecho de la ventana de Visual Studio, en la ventana del Explorador de soluciones, haga clic con el botón derecho en el nombre del proyecto.
15. En el menú contextual, haga clic en Publicar. Esto abre la ventana Elija un destino de publicación.
16. En la ventana Elija un destino de publicación, asegúrese de que App Service esté seleccionado en la lista de destinos disponibles en el lado izquierdo de la ventana.
17. En la sección Azure App Service, en el lado derecho de la ventana, asegúrese de que la opción Crear nueva esté seleccionada.

18. En la esquina inferior derecha de la ventana, haga clic en el botón Crear perfil, que abre la ventana Crear servicio de aplicaciones.
19. En la ventana Crear App Service, agregue una nueva cuenta de Azure. Esta cuenta debe tener suficientes privilegios en la suscripción para crear nuevos grupos de recursos, servicios de aplicaciones y un plan de servicios de aplicaciones.
20. Una vez que haya agregado una cuenta válida, puede configurar los ajustes para publicar su aplicación web.
21. En el cuadro de texto Nombre de la aplicación, ingrese un nombre para el Servicio de la aplicación. De forma predeterminada, este nombre coincide con el nombre que le dio a su proyecto.
22. En el menú desplegable Suscripción, seleccione la suscripción en la que desea crear el Servicio de aplicaciones.
23. En el menú desplegable Grupo de recursos, seleccione el grupo de recursos en el que desea crear App Service y el plan de App Service. Si necesita crear un nuevo grupo de recursos, puede hacerlo haciendo clic en el enlace Nuevo en el lado derecho del menú desplegable.
24. A la derecha del menú desplegable Plan de alojamiento, haga clic en el enlace Nuevo para abrir la ventana Configurar plan de alojamiento.
25. En la ventana Configurar plan de alojamiento, escriba un nombre para el plan de App Service en el cuadro de texto Plan de App Service.
26. Seleccione una región del menú desplegable Ubicación.
27. Seleccione un tamaño de máquina virtual en el menú desplegable Tamaño.
28. Haga clic en el botón Aceptar en la esquina inferior derecha de la ventana. Esto cierra la ventana Configurar plan de alojamiento.
29. En la esquina inferior derecha de la ventana Crear App Service, haga clic en el botón Crear. Esto inicia la creación de los recursos necesarios y la carga del código en App Service.
30. Una vez finalizado el proceso de publicación, Visual Studio abre su navegador web predeterminado con la URL del App Service

recién implementado. Esta URL tendrá la estructura `https://<your_app_service_name>.azurewebsites.net`.

Una vez que haya creado su aplicación web de prueba de Azure, puede usar la URL que obtuvo en el paso 30 en el procedimiento anterior con el resto de los procedimientos de esta habilidad. El siguiente procedimiento muestra cómo crear un perfil de Azure CDN con un punto de conexión para almacenar en caché el contenido de una aplicación web:

1. Abra Azure Portal (<https://portal.azure.com>).
2. Haga clic en el botón Crear un recurso en la sección Servicios de Azure.
3. En la hoja Nuevo, en el cuadro de texto Buscar en el mercado, escriba CDN.
4. En la lista de resultados, haga clic en CDN.
5. En la hoja CDN, haga clic en el botón Crear.
6. En la hoja del perfil CDN, escriba un Nombre para el perfil.
7. Seleccione un grupo de recursos existente en el menú desplegable. Alternativamente, puede crear un nuevo grupo de recursos haciendo clic en el enlace Crear nuevo debajo del menú desplegable Grupo de recursos.
8. En el menú desplegable Nivel de precios, seleccione Microsoft estándar.
9. Haga clic en el botón Crear en la parte inferior de la hoja de perfil CDN.
10. En el cuadro de texto Buscar en la parte superior de Azure Portal, escriba el nombre de su perfil de CDN.
11. En la lista de resultados, haga clic en el nombre de su perfil de CDN.
12. En la hoja de perfil CDN, que se muestra en la [Figura 4-1](#), haga clic en el botón Punto final.

The screenshot shows the Azure portal interface for a CDN profile named 'az204cdn'. On the left, there's a navigation menu with options like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Settings, Properties, Quickstart, and Locks. The main content area displays basic profile details: Resource group (az204-storageaccounts-rg), Status (Active), Subscription (Microsoft Partner Network), and Subscription ID. Below this is a section titled 'Endpoints' which currently shows 'No endpoints are associated with this profile'.

**Figura 4-1** Hoja de perfil CDN

13. En el panel Agregar un punto final, escriba un Nombre para el punto final. Tenga en cuenta que este nombre debe ser único a nivel mundial.
14. En el menú desplegable Tipo de origen, seleccione Aplicación web.
15. En el menú desplegable Origin Hostname, seleccione el nombre de su aplicación web.
16. En el cuadro de texto Ruta de origen, escriba la ruta a la aplicación que necesita incluir en el CDN.
17. Deje el valor del encabezado del host de origen como está. El valor del encabezado del host de origen debe coincidir con el valor del nombre de host de origen.
18. Deje las otras opciones como están.
19. Haga clic en el botón Agregar.

La propagación del contenido a través de la CDN depende del tipo de CDN que configuró. Para Microsoft CDN estándar, la propagación generalmente se completa en 10 minutos. Una vez que se completa la propagación de la CDN, puede acceder a su aplicación web utilizando el punto final que configuró en el procedimiento anterior: `https://<your_endpoint's_name>.azureedge.net`.

Una vez que haya configurado el punto final, puede aplicar algunas opciones avanzadas para ajustar la CDN a sus necesidades:

- **Dominio DNS personalizado** De forma predeterminada, cuando se utiliza la CDN, los usuarios acceden a su aplicación mediante la URL `https://<your_endpoint's_name>.azureedge.net`. Esta URL no sería apropiada para su aplicación. Puede asignar dominios DNS más apropiados al punto de conexión CDN, como `https://app.contoso.com`, que permite a sus usuarios acceder a su aplicación web mediante una URL relacionada con su empresa y su nombre de dominio DNS.
- **Compresión** Puede configurar el punto final CDN para comprimir algunos tipos MIME. La CDN realiza esta compresión sobre la marcha cuando el contenido se entrega desde el cache. Comprimir el contenido le permite entregar archivos más pequeños, mejorando el rendimiento general de la aplicación.
- **Reglas de almacenamiento en caché** Puede controlar cómo se almacena el contenido en la caché estableciendo diferentes reglas para diferentes rutas o tipos de contenido. Al configurar una regla de caché, puede modificar el tiempo de caducidad de la caché, según las condiciones que configure. Las reglas de almacenamiento en caché solo están disponibles para perfiles de Azure CDN Standard de Verizon y Azure CDN Standard de Akamai.
- **Filtrado geográfico** Puede bloquear o permitir el contenido de una aplicación web en países específicos de todo el mundo.
- **Optimización** Puede configurar el CDN para optimizar la entrega de diferentes tipos de contenido. Dependiendo del tipo de perfil, puede optimizar su endpoint para
  - Entrega web general
  - Aceleración dinámica del sitio
  - Transmisión de medios en general
  - Transmisión de medios de video a pedido
  - Descargas de archivos grandes

#### *Tenga en cuenta la aceleración dinámica del sitio*

Aunque Dynamic Site Acceleration es parte de las características proporcionadas por Azure CDN, esta no es estrictamente una solución de caché. Si necesita usar Dynamic Site

Acceleration con los servicios de Microsoft Azure, debe usar Azure Front Door Service en lugar de Azure CDN.

Si necesita crear dinámicamente nuevos perfiles y puntos de conexión de CDN, Microsoft proporciona la biblioteca de CDN de Azure para .NET y la biblioteca de CDN de Azure para Node.js. Con estas bibliotecas, puede automatizar la mayoría de las operaciones revisadas en esta sección.

#### **¿Necesita más revisión? Cómo funciona el almacenamiento en caché**

El almacenamiento en caché de contenido web implica trabajar con encabezados HTTP, establecer los tiempos de vencimiento adecuados o decidir qué archivos deben incluirse en la caché. Puede revisar los detalles de cómo funciona el almacenamiento en caché leyendo el artículo en <https://docs.microsoft.com/en-us/azure/cdn/cdn-how-caching-works>.



#### **Sugerencia para el examen**

Las redes de entrega de contenido (CDN) son adecuadas para almacenar en caché contenido estático que cambia con poca frecuencia. Aunque Azure CDN de Akamai y Azure CDN de Verizon incluyen Dynamic Site Acceleration (DSA), esta característica no es lo mismo que un sistema de caché. No debe confundir la optimización de Azure CDN DSA con la caché de Azure CDN.

#### *Configure las políticas de caducidad y caché para las cachés de FrontDoor, CDN y Redis*

Cuando trabaja con contenido almacenado en caché, necesita controlar la duración o validez de ese contenido. Aunque el contenido estático generalmente tiene una tasa de cambio baja, este tipo de contenido puede cambiar. Por ejemplo, si está almacenando en caché el logotipo de su empresa y el logotipo se cambia, sus usuarios no verán el cambio en la aplicación hasta que el nuevo logotipo se cargue en la caché. En este escenario, simplemente puede purgar o eliminar el logotipo antiguo del caché, y la nueva imagen se cargará en el caché tan pronto como el primer usuario acceda a la aplicación.

Este mecanismo de purga manual de la caché podría ser apropiado para un escenario muy específico. Aún así, en términos generales, debería considerar el uso de un mecanismo automático para tener el contenido más actualizado en su sistema de caché. Cuando agrega contenido a una caché de CDN, el sistema asigna automáticamente un valor de TimeToLive

(TTL) al archivo de contenido en lugar de comparar continuamente el archivo en la caché con el contenido original en el servidor web. El sistema de caché comprueba si el TTL es más bajo que el tiempo actual. Si el TTL es menor que el tiempo actual, la CDN considera que el contenido está actualizado y lo mantiene en la caché. Si el TTL expira, el CDN marca el contenido como obsoleto o inválido. Cuando el siguiente usuario intenta acceder al archivo de contenido no válido, la CDN compara el archivo en caché con el contenido del servidor web. Si ambos archivos coinciden, el CDN actualiza la versión del archivo en caché y hace que el archivo vuelva a ser válido restableciendo el tiempo de vencimiento. Si los archivos en la caché y el servidor web no coinciden, el CDN elimina el archivo de la caché y actualiza el contenido con el archivo de contenido más actualizado en el servidor web.

El contenido almacenado en caché puede volverse inválido si se elimina el contenido de la memoria caché o se alcanza el tiempo de caducidad. Puede configurar el TTL predeterminado asociado con un sitio mediante el encabezado HTTP Cache-Control. El valor de este encabezado se establece de diferentes maneras:

- **Configuración de CDN predeterminada** Si no configura ningún valor para el TTL, Azure CDN configura automáticamente un valor predeterminado de siete días.
- **Reglas de almacenamiento en caché** Puede configurar los valores TTL de forma global o mediante el uso de reglas de coincidencia personalizadas. Las reglas globales de almacenamiento en caché afectan a todo el contenido de la CDN. Las reglas de almacenamiento en caché personalizadas controlan el TTL para diferentes rutas o archivos en su aplicación web. Incluso puede deshabilitar el almacenamiento en caché para algunas partes de su aplicación web.
- **Archivos web.config** Utilice el archivo web.config para establecer la hora de caducidad de la carpeta. Incluso puede configurar archivos web.config para diferentes carpetas configurando diferentes valores TTL. Utilice el siguiente código XML para configurar el TTL:

[Haga clic aquí para ver la imagen del código](#)

```
<configuración>
```

```

<system.webServer>

    <staticContent>

        <clientCache cacheControlMode = "UseMaxAge"
cacheControlMaxAge =

            "3.00: 00: 00" />

    </staticContent>

</system.webServer>

</configuration>

```

- **Mediante programación** Si trabaja con ASP.NET, puede controlar el comportamiento de almacenamiento en caché de CDN estableciendo la `HttpResponse.Cache` propiedad. Puede utilizar el siguiente código para establecer el tiempo de caducidad del contenido en cinco horas:

[Haga clic aquí para ver la imagen del código](#)

```

// Establecer los parámetros de almacenamiento en caché.

Response.Cache.SetExpires (DateTime.Now.AddHours (5));

Response.Cache.SetCacheability (HttpCacheability.Public);

Response.Cache.SetLastModified (DateTime.Now);

```

Use el siguiente procedimiento para crear reglas de almacenamiento en caché en su CDN de Azure. Tenga en cuenta que puede configurar reglas de almacenamiento en caché solo para los perfiles de Azure CDN para Verizon y Azure CDN para Akamai:

1. Abra Azure Portal (<https://portal.azure.com>).
2. Haga clic en el botón Crear un recurso.
3. En la hoja Nuevo, en el cuadro de texto Buscar en el mercado, escriba CDN.
4. En la lista de resultados, haga clic en CDN.
5. En la hoja CDN, haga clic en el botón Crear.
6. En la hoja del perfil CDN, escriba un Nombre para el perfil.

7. Seleccione un grupo de recursos existente en el menú desplegable. Alternativamente, puede crear un nuevo grupo de recursos haciendo clic en el enlace Crear nuevo debajo del menú desplegable Grupo de recursos.
8. En el menú desplegable Nivel de precios, seleccione Estándar Akamai.
9. Marque la casilla de verificación Create A New CDN Endpoint Now.
10. Escriba un nombre para el punto final en el cuadro de texto Nombre del punto final de CDN. Tenga en cuenta que este nombre no puede ser el mismo que el nombre del perfil CDN.
11. En el menú desplegable Tipo de origen, seleccione Aplicación web.
12. En el menú desplegable Origin Hostname, seleccione el nombre de su aplicación web.
13. Haga clic en el botón Crear en la parte inferior de la hoja Perfil de CDN.
14. En el cuadro de texto Buscar en la parte superior de Azure Portal, escriba el nombre de su perfil de CDN.
15. En la lista de resultados, haga clic en el nombre de su perfil de CDN.
16. En el panel Descripción general, en la hoja de perfil CDN, en la lista de Extremos, haga clic en el extremo existente.
17. En la hoja Endpoint, haga clic en Reglas de almacenamiento en caché en la sección Configuración del menú de navegación.
18. En el panel Reglas de almacenamiento en caché, que se muestra en la [Figura 4-2](#), configure el menú desplegable Comportamiento de almacenamiento en caché en Anular en la sección Reglas de almacenamiento en caché globales.

**Global caching rules**

These rules affect the CDN caching behavior for all requests, and can be overridden using Custom Cache Rules below for certain scenarios. Note that the Query string caching behavior setting does not affect files that are not cached by the CDN.

Caching behavior: Override

Cache expiration duration: Days 15, Hours 0, Minutes 0, Seconds 0

Query string caching behavior: Ignore query strings

**Custom caching rules**

Create caching rules based on specific match conditions. These rules override the default settings above, and are evaluated from top to down. This means that rules lower in the list have more specific match conditions than rules above them. Therefore it makes more sense to have more specific rules towards the bottom of the list so they are not overwritten by a general rule below a rule for path '/folder/\*'.

Match Condition	Match Value(s)	Caching Behavior	Days
File extension(s)	png	Override	4
			0

**Figura 4-2 Configuración de reglas de almacenamiento en caché**

19. Establezca la Duración de caducidad de la caché en 15 días.
20. En la lista Reglas personalizadas de almacenamiento en caché, cree una nueva regla personalizada. Establezca el menú desplegable Condición de coincidencia en Extensión (es) de archivo.
21. En el cuadro de texto Coincidir valores, escriba **png**.
22. En el menú desplegable Caching Behavior, seleccione Override.
23. En la columna Días, escriba **4**.
24. En la esquina superior izquierda del panel, haga clic en el botón Guardar.

Cuando trabaja con Azure Cache para Redis, también puede establecer el TTL para los diferentes valores almacenados en la base de datos en memoria. Si no establece un TTL para el par clave / valor, la entrada en la caché no caducará. Cuando crea una nueva entrada en la base de datos en memoria, establece el valor TTL como un parámetro del `StringSet()` método. El siguiente fragmento de código muestra cómo establecer un TTL de 5 horas en un valor de cadena:

[Haga clic aquí para ver la imagen del código](#)

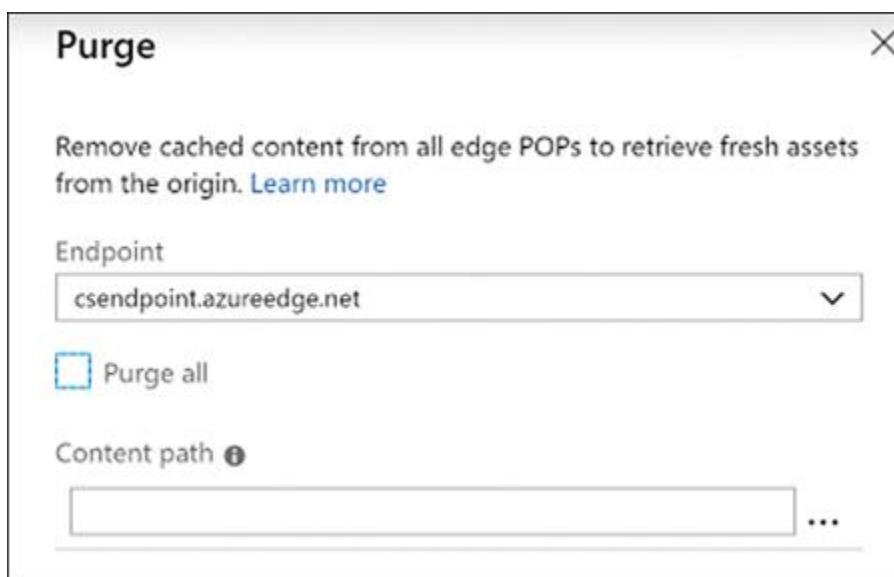
```
_cache.StringSet (clave, Serializar (valor), nuevo TimeSpan  
(5, 0, 0));
```

Además de invalidar el contenido de la caché por el vencimiento del contenido, puede invalidar manualmente el contenido eliminándolo directamente de la CDN o Redis Cache. Puede quitar una clave de la base de datos en memoria de Azure Cache for Redis. Puede utilizar los siguientes métodos:

- **Método KeyDelete ()** Utilice este método para eliminar una única clave de la base de datos. Debe utilizar este método con una instancia de base de datos.
- **Método FlushAllDatabases ()** Utilice este método para quitar todas las claves de todas las bases de datos en Azure Cache para Redis.

Para Azure CDN, puede invalidar parte o todo el contenido del perfil de CDN mediante la opción Purgar disponible en Azure Portal. Use el siguiente procedimiento para purgar contenido de su perfil de Azure CDN:

1. Abra Azure Portal (<https://portal.azure.com>).
2. En el cuadro de texto Buscar en la parte superior de Azure Portal, escriba el nombre de su perfil de CDN.
3. En el panel Descripción general, en la hoja de su perfil CDN, haga clic en el botón Purgar.
4. En el panel Purgar, que se muestra en la [Figura 4-3](#), seleccione el punto final que desea purgar del control del menú desplegable.



### **Figura 4-3 Purga de contenido de la caché**

5. En el cuadro de texto Ruta del contenido, escriba la ruta que desea purgar de la caché. Si desea purgar todo el contenido de la caché, debe marcar la casilla de verificación Purgar todo.

#### **Nota Purgar todo y comodines en Azure CDN para Akamai**

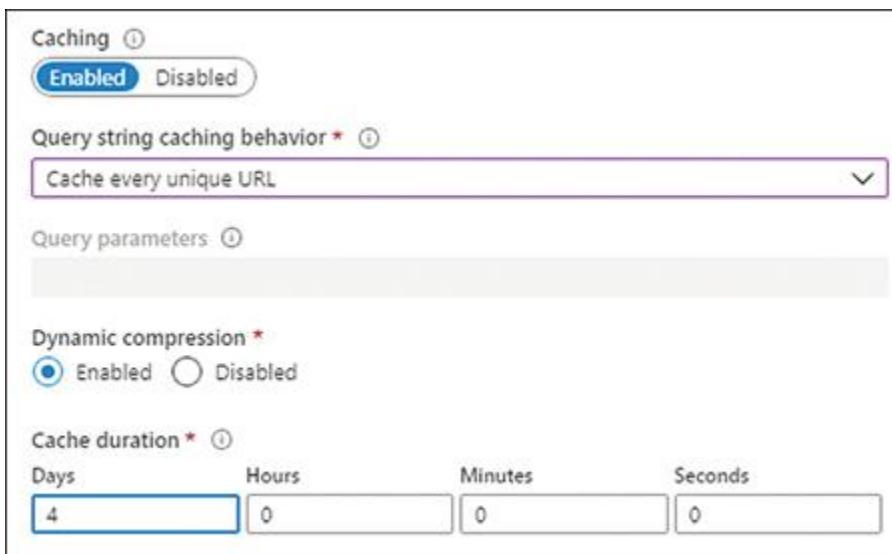
En el momento de redactar este documento, las opciones Purgar todo y Comodín no están disponibles para las CDN de Akamai.

Azure CDN no es el único servicio que ofrece Microsoft para almacenar contenido en caché. El servicio Azure Front Door le permite enrutar el tráfico de manera eficiente a la ubicación más cercana al usuario. Como parte de las características que ofrece el servicio Azure Front Door, también le permite almacenar en caché el contenido al proporcionar una CDN. Al igual que con Azure CDN, puede configurar la caché y el tiempo de caducidad de los elementos en la caché.

La configuración de la caché se realiza a nivel de reglas de enrutamiento. Con el servicio Azure Front Door, puede enrutar el tráfico por diferentes rutas en su URL a diferentes grupos de back-end que hospedan su aplicación. Una regla de enrutamiento define cada una de estas rutas. Con esta estructura en mente, puede configurar el almacenamiento en caché para algunas partes de su aplicación, mientras que otras permanecen sin almacenar en caché. El siguiente procedimiento muestra cómo habilitar el almacenamiento en caché en una regla de enrutamiento. Este procedimiento asume que ya ha implementado Azure Front Door. Debido a que no revisamos cómo trabajar con Azure Front Door anteriormente en este capítulo, puede implementar una demostración de Front Door usando la guía de inicio rápido en <https://docs.microsoft.com/en-us/azure/frontdoor/Quickstart-create-front-door>.

1. Abra Azure Portal (<https://portal.azure.com>).
2. En el cuadro de texto Buscar recursos, servicios y documentos, escriba el nombre de su instancia de Azure Front Door.
3. Haga clic en el nombre de su instancia de Azure Front Door en la lista de resultados.
4. En su hoja Azure Front Door, haga clic en Front Door Designer en la sección Configuración en el menú de navegación en el lado izquierdo de la hoja.

5. En la hoja Front Door Designer, haga clic en una de las reglas de enrutamiento dentro del rectángulo verde con el título Reglas de enrutamiento.
6. En el panel Actualizar regla de enrutamiento, desplácese hacia abajo hasta la parte inferior del panel.
7. Cambie el control del conmutador Caching de Disabled a Enabled.
8. En la configuración de la caché que se muestra en la [Figura 4-4](#), cambie el valor de Duración de la caché de 0 días a 4 días.



**Figura 4-4** Configuración de la caché de Azure Front Door

9. Haga clic en el botón Actualizar.
10. Haga clic en el botón Guardar en la esquina superior izquierda de la hoja Front Door Designer.

También puede controlar la caducidad de la caché de un elemento individual configurando los encabezados de caché adecuados. Los siguientes encabezados HTTP controlan la caché y la caducidad de un elemento en la caché de Azure Front Door:

- **Cache-Control: max-age** Expresado en segundos, este encabezado controla cuánto tiempo es válido el elemento en el caché. Por ejemplo, si establece este valor en 3600, el elemento se puede usar hasta 60 minutos antes de que el servicio Azure Front Door realice una solicitud al grupo de back-end para obtener una versión nueva del elemento.

- **Cache-Control: s-maxage** Expresada en segundos, esta directiva es similar a la anterior pero es significativa solo en entornos CDN. Esta directiva tiene prioridad sobre las directivas max-age y expira.
- **Caduca** Expresada usando una marca de tiempo de fecha HTTP, esta directiva establece la fecha y hora hasta que el elemento es válido en la caché. Las directivas max-age y s-maxage tienen prioridad sobre esta directiva.

Purgar el contenido de la caché es tan simple como en los servicios de Azure CDN. Los siguientes pasos muestran cómo purgar el contenido de Azure Front Door:

1. Abra Azure Portal (<https://portal.azure.com>).
2. En el cuadro de texto Buscar recursos, servicios y documentos, escriba el nombre de su instancia de Azure Front Door.
3. Haga clic en el nombre de su instancia de Azure Front Door en la lista de resultados.
4. En su hoja Azure Front Door, haga clic en Front Door Designer en la sección Configuración en el menú de navegación en el lado izquierdo de la hoja.
5. En la hoja Front Door Designer, haga clic en el botón Purgar en la parte superior izquierda de la hoja.
6. En el panel Purgar, marque la casilla de verificación Purgar todo. Alternativamente, si desea purgar solo una parte del contenido almacenado en caché, escriba la ruta del contenido que desea purgar en el cuadro de texto Ruta del contenido debajo de la casilla de verificación Purgar todo.
7. Haga clic en el botón Purgar en la parte inferior del panel.

#### *¿Necesita más revisión? Almacenamiento en caché de Azure Front Door*

Azure Front Door es un sistema avanzado de enrutamiento y almacenamiento en caché. Este servicio le permite almacenar en caché archivos grandes y comprimir datos sobre la marcha. Puede revisar más detalles sobre cómo funciona el almacenamiento en caché de Azure Front Door leyendo el artículo en <https://docs.microsoft.com/en-us/azure/frontdoor/front-door-caching>.

## *Almacenar y recuperar datos en Azure Redis Cache*

Redis es un sistema de caché de código abierto que le permite trabajar *como en* un almacén de estructura de datos en memoria, caché de base de datos o agente de mensajes. Azure Redis Cache o Azure Cache for Redis es una implementación de Redis administrada por Microsoft. Azure Redis Cache tiene tres capas de precios que le brindan diferentes niveles de características:

- **Básico** Este es el nivel con la menor cantidad de funciones, menor rendimiento y mayor latencia. Debe usar este nivel solo con fines de desarrollo o prueba. No hay ningún acuerdo de nivel de servicio (SLA) asociado con el nivel básico.
- **Estándar** Este nivel ofrece una caché de Redis replicada primaria-secundaria de dos nodos administrada por Microsoft. Este nivel tiene asociado un SLA de alta disponibilidad del 99,9 por ciento.
- **Premium** Este es un clúster de Redis de nivel empresarial administrado por Microsoft. Este nivel ofrece el grupo completo de funciones con el mayor rendimiento y menor latencia. El clúster de Redis también se implementa en hardware más potente. Este nivel tiene un SLA de alta disponibilidad del 99,9 por ciento.

### *Nota Escalado del servicio de caché de Azure Redis*

Puede escalar su servicio de caché de Azure Redis existente a un nivel superior, pero no puede escalar su nivel actual a uno inferior.

Cuando trabaja con Azure Cache para Redis, puede usar diferentes patrones de implementación que resuelven diferentes problemas, según la arquitectura de su aplicación:

- **Cache-Aparte** En la mayoría de las situaciones, su aplicación almacena los datos que administra en una base de datos. Acceder a los datos de una base de datos es una operación relativamente lenta porque depende del tiempo para acceder al sistema de almacenamiento en disco. Una solución sería cargar la base de datos en la memoria, pero este enfoque es costoso; en la mayoría de los casos, la base de datos simplemente no cabe en la memoria disponible. Una solución para mejorar el rendimiento de su aplicación en estos escenarios es almacenar los datos más accesibles en la caché. Cuando el sistema back-end cambia los datos

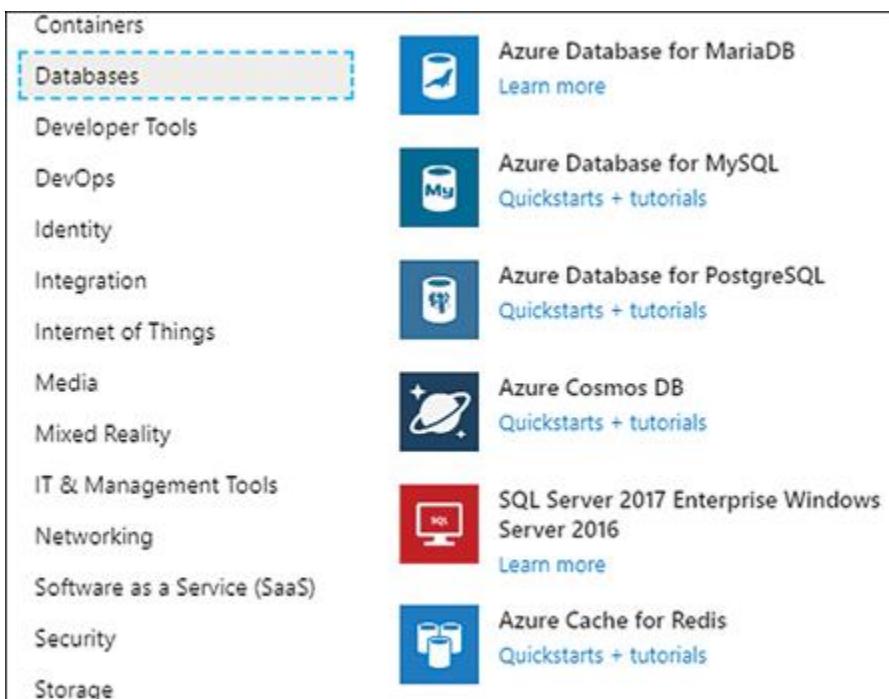
en la base de datos, el mismo sistema también puede actualizar los datos en la caché, lo que hace que el cambio esté disponible para todos los clientes.

- **Almacenamiento en caché de contenido** La mayoría de las aplicaciones web utilizan plantillas de páginas web que utilizan elementos comunes, como encabezados, pies de página, barras de herramientas, menús, hojas de estilo, imágenes, etc. Estos elementos de plantilla son elementos estáticos (o al menos no cambian con frecuencia). El almacenamiento de estos elementos en Azure Cache para Redis evita que sus servidores web proporcionen estos elementos y mejora el tiempo que necesitan para generar contenido dinámico.
- **Almacenamiento en caché de la sesión del usuario** Este patrón es una buena idea si su aplicación necesita registrar demasiada información sobre el historial del usuario o los datos que necesita asociar con las cookies. Almacenar demasiada información en una cookie de sesión perjudica el rendimiento de su aplicación. Puede guardar parte de esa información en su base de datos y almacenar un puntero o índice en la cookie de sesión que apunta a ese usuario a la información en la base de datos. Si usa una base de datos en memoria, como Azure Cache para Redis, en lugar de una base de datos tradicional, su aplicación se beneficia de los tiempos de acceso más rápidos a los datos almacenados en la memoria.
- **Cola de trabajos y mensajes** Puede usar Azure Cache para Redis para implementar una cola distribuida que ejecuta tareas de larga duración que pueden afectar negativamente el rendimiento de su aplicación.
- **Transacciones distribuidas** Una transacción es un grupo de comandos que deben completarse o fallar juntos. Cualquier transacción debe garantizar que los datos estén siempre en un estado estable. Si su aplicación necesita ejecutar transacciones, puede usar Azure Cache for Redis para implementar estas transacciones.

Puede trabajar con Azure Cache para Redis utilizando diferentes lenguajes, como ASP.NET, .NET, .NET Core, Node.js, Java o Python. Antes de poder agregar características de almacenamiento en caché a su código

mediante Azure Redis Cache, debe crear su base de datos de Azure Cache para Redis mediante el siguiente procedimiento:

1. Abra Azure Portal (<https://portal.azure.com>).
2. Haga clic en Crear un recurso en la sección Servicios de Azure.
3. En la hoja Nueva, haga clic en Bases de datos en el menú de navegación en el lado izquierdo de la hoja.
4. En la lista de servicios de base de datos, que se muestra en la Figura 4-5, haga clic en el elemento Azure Cache For Redis.



**Figura 4-5** Creación de un nuevo recurso Azure Cache for Redis

5. En la hoja Nueva caché de Redis, escriba un nombre DNS para su recurso de Redis.
6. Seleccione la suscripción, el grupo de recursos y la ubicación del menú desplegable correspondiente que mejor se adapte a sus necesidades.
7. En el menú desplegable Nivel de precios, seleccione el nivel C0 básico.
8. Haga clic en el botón Crear en la parte inferior de la hoja Nueva caché de Redis.

La implementación de su nueva caché de Azure para Redis tarda unos minutos en completarse. Una vez que se completa la implementación, debe obtener las claves de acceso para su instancia de Azure Cache para Redis. Usa esta información en su código para conectar el servicio Redis en Azure.

Si usa cualquiera de los lenguajes .NET, puede usar el cliente StackExchange.Redis para acceder a su recurso Azure Cache for Redis. También puede utilizar este cliente de Redis para acceder a otras implementaciones de Redis. Al leer o escribir valores en Azure Cache for Redis, debe crear un `ConnectionMultiplexer` objeto. Este objeto crea una conexión a su servidor Redis. La `ConnectionMultiplexer` clase está diseñada para ser reutilizada tanto como sea posible.

Por esta razón, debe almacenar este objeto y reutilizarlo en todo su código, siempre que sea posible reutilizarlo. Crear una conexión es una operación costosa. Por esta razón, no debe crear un `ConnectionMultiplexer` objeto para cada operación de lectura o escritura en la caché de Redis. Una vez que haya creado su `ConnectionMultiplexer` objeto, puede usar cualquiera de las operaciones disponibles en el paquete StackExchange.Redis. A continuación, se muestran las operaciones básicas que puede utilizar con Redis:

- **Usar Redis como base de datos** Obtiene una base de datos de Redis, usando el `GetDatabase()` método, para escribir y leer valores de la base de datos. Utiliza los métodos `StringSet()` o `StringGet()` para escribir y leer.
- **Usar Redis como una cola de mensajería** Obtiene un objeto de suscriptor del cliente Redis, usando el `GetSubscriber()` método. Luego, puede publicar mensajes en una cola, usando el `Publish()` método, y leer mensajes de una cola, usando el `Subscribe()` método. Las colas en Redis se conocen como "canales".

El siguiente procedimiento muestra cómo conectarse a una base de datos de Azure Cache para Redis y leer y escribir datos hacia y desde la base de datos mediante una aplicación ASP.NET:

1. Abra Visual Studio 2019.
2. En la ventana de bienvenida de Visual Studio 2019, en la columna Introducción, haga clic en Crear un nuevo proyecto.
3. En la ventana Crear un proyecto nuevo, en el menú desplegable Todos los idiomas, seleccione C #.

4. En el cuadro de texto Buscar plantillas, escriba **asp.net**.
5. En la lista de resultados, haga clic en Aplicación web ASP.NET (.NET Framework).
6. Haga clic en el botón Siguiente en la parte inferior derecha de la ventana.
7. En Configure Your New Project, escriba un nombre de proyecto, una ubicación y un nombre de solución para su proyecto.
8. Haga clic en el botón Crear en la parte inferior derecha de la ventana.
9. En la ventana Crear una nueva aplicación web ASP.NET, seleccione la plantilla MVC en la lista de plantillas en el lado medio izquierdo de la ventana. MVC es para Model-View-Controller.
10. En el lado derecho de la ventana Crear una nueva aplicación web ASP.NET, en la sección Autenticación, asegúrese de que Autenticación esté configurada en Sin autenticación.
11. Haga clic en el botón Crear en la esquina inferior derecha de la ventana.
12. En la ventana de Visual Studio, seleccione Herramientas> Administrador de paquetes NuGet> Administrar paquetes NuGet para la solución.
13. En la pestaña Administrador de paquetes NuGet, haga clic en Examinar.
14. Escriba **StackExchange.Redis** y presione Entrar.
15. Haga clic en el paquete StackExchange.Redis.
16. En el lado derecho de la pestaña NuGet Manager, haga clic en la casilla de verificación junto a su proyecto.
17. Haga clic en el botón Instalar.
18. En la ventana Vista previa de cambios, haga clic en Aceptar.
19. En la ventana Aceptación de licencia, haga clic en el botón Acepto.
20. Abra Azure Portal (<https://portal.azure.com>).

21. En el cuadro de texto de búsqueda en la parte superior central del portal, escriba el nombre de su caché de Azure para Redis que creó en el ejemplo anterior.
22. Haga clic en Azure Cache for Redis en la lista de resultados.
23. En la hoja Azure Cache for Redis, haga clic en Claves de acceso en la sección Configuración en el menú de navegación en el lado izquierdo de la hoja.
24. En la hoja Claves de acceso, copie el valor de la Cadena de conexión principal (StackExchange.Redis). Necesita este valor en los siguientes pasos.
25. En la ventana de Visual Studio, abra el archivo Web.config.
26. En el `<appSettings>` sección, agregue el siguiente código:

[Haga clic aquí para ver la imagen del código](#)

```
<add key = "CacheConnection" value = "<value_copied_in_step_24>" />
```

#### **Tenga en cuenta las mejores prácticas de seguridad**

En el desarrollo del mundo real, debe evitar poner cadenas de conexión y secretos en archivos que podrían verificarse con el resto de su código. Para evitar esto, puede colocar la `<appSettings>` sección con las claves que contienen los secretos sensibles o las cadenas de conexión en un archivo separado fuera de la carpeta de control del código fuente. Luego agregue el parámetro de archivo a la `<appSettings>` etiqueta que apunta a la `appSettings` ruta del archivo externo . También puede usar Azure App Configuration junto con Azure Key Vault para almacenar sus cadenas de conexión.

27. Abra el archivo HomeController.cs en la carpeta Controllers.
28. Agregue las siguientes declaraciones using al archivo HomeController.cs:

```
29.     usando System.Configuration;  
  
        usando StackExchange.Redis;
```

30. Agregue el código del [Listado 4-1](#) a la `HomeController` clase.

#### **Listado 4-1 Método HomeController.RedisCache**

[Haga clic aquí para ver la imagen del código](#)

---

```
// C#. ASP.NET.
```

```
public ActionResult RedisCache ()  
{  
    ViewBag.Message = "Un ejemplo simple con Azure Cache  
para Redis en ASP.NET.";  
  
    var lazyConnection = new Lazy  
<ConnectionMultiplexer> (() =>  
  
{  
  
    string cacheConnection =  
ConfigurationManager.AppSettings ["CacheConnection"]  
  
.Encadenar();  
  
    return ConnectionMultiplexer.Connect  
(cacheConnection);  
  
});  
  
// Necesita crear un objeto ConnectionMultiplexer  
para acceder a Redis  
  
// caché.  
  
// Entonces puede obtener una instancia de una base  
de datos.  
  
Caché IDatabase = lazyConnection.Value.GetDatabase  
();  
  
  
// Realizar operaciones de caché utilizando el  
objeto de caché ...
```

```
// Ejecuta un comando simple de Redis  
  
ViewBag.command1 = "PING";  
  
ViewBag.command1Result = cache.Execute  
(ViewBag.command1) .ToString ();  
  
  
// Obtención y colocación simple de tipos de datos  
integrales en la caché  
  
ViewBag.command2 = "OBTENER mensaje";  
  
ViewBag.command2Result = cache.StringGet  
("Mensaje") .ToString ();  
  
  
// Escribe un nuevo valor en la base de datos.  
  
ViewBag.command3 = "SET Message \" ¡Hola! ¡La caché  
está funcionando desde ASP.NET! \"";  
  
ViewBag.command3Result = cache.StringSet ("Mensaje",  
"¡Hola! La caché está funcionando  
  
desde ASP.NET! ") .ToString ();  
  
  
// Obtén el mensaje que escribimos en el paso  
anterior  
  
ViewBag.command4 = "OBTENER mensaje";  
  
ViewBag.command4Result = cache.StringGet  
("Mensaje") .ToString ();  
  
  
// Obtener la lista de clientes, útil para ver si la  
lista de conexiones está creciendo ...
```

```

ViewBag.command5 = "LISTA DE CLIENTES";

ViewBag.command5Result = cache.Execute ("CLIENTE",
"LIST"). ToString (). Reemplazar (
"id =", "\ rid =");

lazyConnection.Value.Dispose ();

volver Ver ();
}

}

```

31. En el Explorador de soluciones, haga clic con el botón derecho en Vistas> Carpeta de inicio y seleccione Agregar> Ver en el menú contextual.
32. En la ventana Agregar vista, escriba **RedisCache** para el Nombre de la vista.
33. Haga clic en el botón Agregar.
34. Abra el archivo RedisCache.cshtml.
35. Reemplace el contenido del archivo RedisCache.cshtml con el contenido del [Listado 4-2](#).

**Listado 4-2** Vista de RedisCache

[Haga clic aquí para ver la imagen del código](#)

---

```

// C#. ASP.NET.

@ {

    ViewBag.Title = "Azure Cache for Redis Test";

}

```

```
<h2> @ ViewBag.Title. </h2>

<h3> @ ViewBag.Message </h3>

<br /> <br />

<table border = "1" cellpadding = "10">

    <tr>

        <th> Comando </th>

        <th> Resultado </th>

    </tr>

    <tr>

        <td> @ ViewBag.command1 </td>

        <td><pre>@ViewBag.command1Result</pre> </td>

    </tr>

    <tr>

        <td> @ ViewBag.command2 </td>

        <td><pre>@ViewBag.command2Result</pre> </td>

    </tr>

    <tr>

        <td> @ ViewBag.command3 </td>

        <td><pre>@ViewBag.command3Result</pre> </td>

    </tr>

    <tr>

        <td> @ ViewBag.command4 </td>

        <td><pre>@ViewBag.command4Result</pre> </td>

    </tr>
```

```

</tr>

<tr>

    <td> @ ViewBag.command5 </td>

    <td><pre>@ViewBag.command5Result</pre> </td>

</tr>

</table>

```

36. Presione F5 para ejecutar su proyecto localmente.

37. En el navegador web que ejecuta su proyecto, agregue el URI / Home / RedisCache a la URL. Su resultado debería verse como en la Figura 4-6.

Command	Result
PING	PONG
GET Message	
SET Message "Hello! The cache is working from ASP.NET!"	True
GET Message	Hello! The cache is working from ASP.NET!
CLIENT LIST	id=9774 addr=127.0.0.1:35187 fd=8 name=PORTAL_CONSOLE age=152 id=9853 addr=83.56.0.194:61343 fd=18 name=DEV-CS age=1 idle=0 id=9854 addr=83.56.0.194:61344 fd=14 name=DEV-CS age=1 idle=1

**Figura 4-6** Resultados de ejemplo



### Sugerencia para el examen

Puede usar Azure Cache para Redis para contenido estático y los datos dinámicos más accesibles. Puede utilizarlo para bases de datos en memoria o colas de mensajes mediante un patrón de publicación / suscripción.

### *¿Necesita más revisión? Más detalles sobre Redis*

Puede revisar las características, patrones y transacciones del sistema de caché de Redis leyendo los siguientes artículos:

- <https://stackexchange.github.io/StackExchange.Redis/Basics>
- <https://stackexchange.github.io/StackExchange.Redis/Transactions>
- <https://stackexchange.github.io/StackExchange.Redis/KeysValues>

## HABILIDAD 4.2: SOLUCIONES DE INSTRUMENTOS PARA RESPALDAR EL MONITOREO Y EL REGISTRO

---

Saber cómo se comporta su aplicación durante el funcionamiento habitual es fundamental, especialmente para los entornos de producción. Necesita obtener información sobre la cantidad de usuarios, el consumo de recursos, las transacciones y otras métricas que pueden ayudarlo a solucionar problemas de su aplicación si ocurre un error. Agregar métricas personalizadas a su aplicación también es importante al crear alertas que le advierten cuando su aplicación no se comporta como se esperaba.

Azure proporciona características para monitorear el consumo de recursos asignados a su aplicación. Además, puede monitorear las transacciones y cualquier otra métrica que pueda necesitar, lo que le permite comprender completamente cómo se comporta su aplicación en condiciones que generalmente son difíciles de simular o probar. También puede usar estas métricas para crear reglas de autoescala de manera eficiente para mejorar el rendimiento de su aplicación.

### **Esta habilidad cubre cómo**

- [Configurar la instrumentación en una aplicación o servicio mediante Application Insights](#)
- [Analice los datos de registro y solucione problemas de soluciones mediante Azure Monitor](#)
- [Implementar alertas y pruebas web de Application Insights](#)
- [Implementar código que maneje fallas transitorias](#)

## *Configurar la instrumentación en una aplicación o servicio mediante Application Insights*

Microsoft le brinda la capacidad de monitorear su aplicación mientras se está ejecutando mediante Application Insights. Esta herramienta se integra con su código, lo que le permite monitorear lo que sucede dentro de su código mientras se ejecuta en un entorno de nube, local o híbrido. También puede habilitar Application Insights para aplicaciones que ya están implementadas en Azure sin modificar el código ya implementado.

Al agregar un pequeño paquete de instrumentación, puede medir varios aspectos de su aplicación. Estas medidas, conocidas como telemetría, se envían automáticamente al componente Application Insight implementado en Azure. Según la información enviada desde los flujos de telemetría desde su aplicación al portal de Azure, puede analizar el rendimiento de su aplicación y crear alertas y paneles, que le ayudarán a comprender mejor cómo se comporta su aplicación. Aunque Application Insights debe implementarse en Azure Portal, su aplicación se puede ejecutar en Azure, en otras nubes públicas o en su infraestructura local. Cuando implementa la instrumentación de Application Insights en su aplicación, monitorea los siguientes puntos:

- **Tasas de solicitudes, tiempos de respuesta y tasas de fallas** Puede ver qué páginas solicitan sus usuarios con más frecuencia, distribuidas a lo largo del tiempo. Puede encontrar que sus usuarios tienden a visitar páginas específicas al comienzo del día, mientras que otras páginas son más visitadas al final del día. También puede controlar el tiempo que tarda su servidor en entregar la página solicitada o incluso si hubo fallas al entregar la página. Debe monitorear las tasas de falla y los tiempos de respuesta para asegurarse de que su aplicación funcione correctamente y que sus usuarios tengan una experiencia agradable.
- **Tasas de dependencia, tiempos de respuesta y tasas de falla** Si su aplicación depende de servicios externos (como cuentas de almacenamiento de Azure), servicios de seguridad de Google o Twitter para autenticar a sus usuarios, o cualquier otro servicio externo, puede monitorear el desempeño de estos servicios externos y cómo están afectando su aplicación.

- **Excepciones** La instrumentación realiza un seguimiento de las excepciones generadas por los servidores y navegadores mientras se ejecuta la aplicación. Puede revisar los detalles del seguimiento de la pila para cada excepción a través de Azure Portal. También puede ver estadísticas sobre las excepciones que surgen durante la ejecución de su aplicación.
- **Vistas de página y rendimiento de carga** Medir el rendimiento de la entrega de páginas de su servidor es solo una parte de la ecuación. Con Application Insights, también puede obtener información sobre las visitas a la página y el rendimiento de carga informado desde el lado del navegador.
- **Llamadas AJAX** Esto mide el tiempo que tardan las llamadas AJAX realizadas desde las páginas web de su aplicación. También mide las tasas de falla y el tiempo de respuesta.
- **Recuentos de usuarios y sesiones** Puede realizar un seguimiento del número de usuarios que están conectados a su aplicación. Del mismo modo que el mismo usuario puede iniciar varias sesiones, puede realizar un seguimiento del número de sesiones conectadas a su aplicación. Esto le permite medir claramente el umbral de usuarios simultáneos admitidos por su aplicación.
- **Contadores de rendimiento** Puede obtener información sobre los contadores de rendimiento de la máquina servidor (CPU, memoria y uso de la red) desde la que se ejecuta su código.
- **Diagnósticos de hosts** Los diagnósticos de hosts pueden obtener información de su aplicación si está implementada en un entorno Docker o Azure.
- **Registros de seguimiento de diagnóstico** Los mensajes de **registro de seguimiento** se pueden utilizar para correlacionar los eventos de seguimiento con las solicitudes realizadas a la aplicación por los usuarios.
- **Eventos y métricas personalizados** Aunque la instrumentación lista para usar que ofrece Application Insights ofrece mucha información, algunas métricas son demasiado específicas para su aplicación como para generalizarlas e incluirlas en la telemetría general. Para esos casos, puede crear métricas personalizadas para monitorear su servidor y código de

cliente. Esto le permite monitorear las acciones del usuario, como los pagos del carrito de compras o la puntuación del juego.

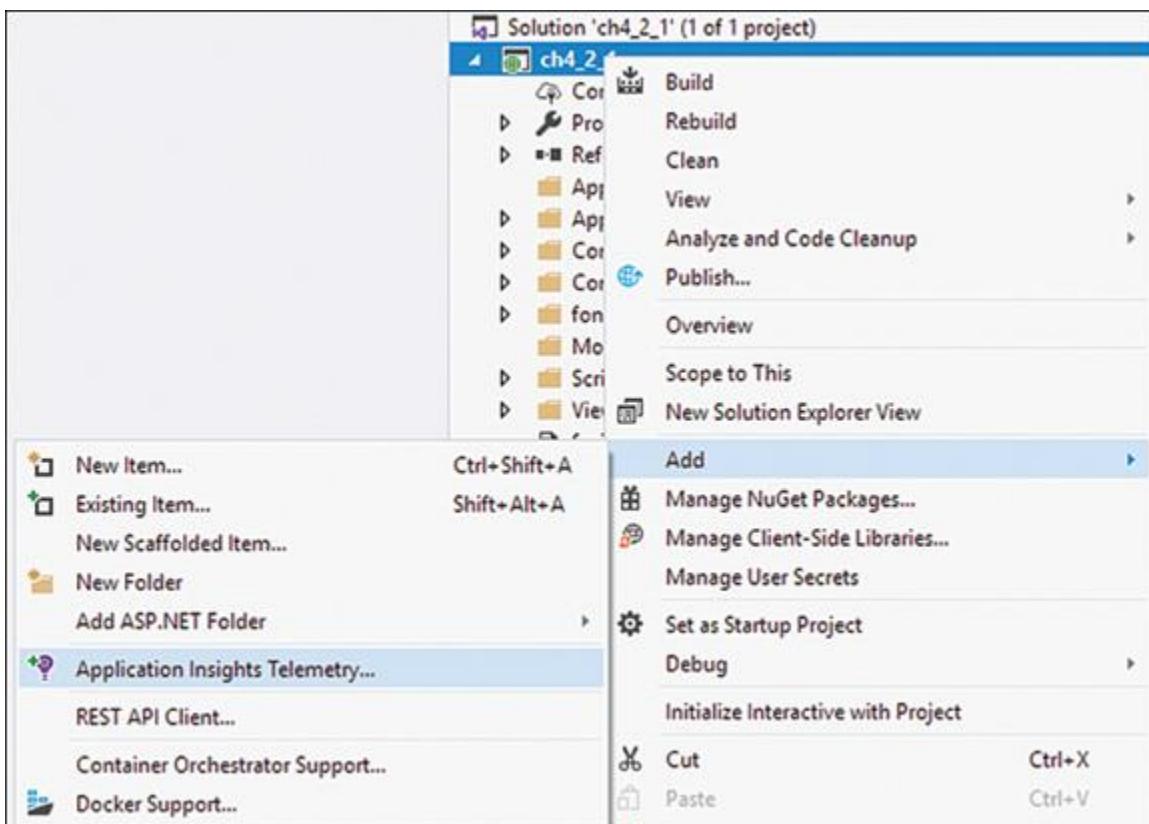
Application Insights no se limita a los lenguajes .NET. Hay bibliotecas de instrumentación disponibles para otros lenguajes, como Java, JavaScript o Node.js. También hay bibliotecas disponibles para otras plataformas como Android o iOS. Puede utilizar el siguiente procedimiento para agregar instrumentación de Application Insight a su aplicación ASP.NET. Para ejecutar este ejemplo, debe cumplir con estos requisitos previos:

- Una suscripción de Azure.
- Visual Studio 2017/2019. Si no tiene Visual Studio, puede descargar la edición Community de forma gratuita desde <https://visualstudio.microsoft.com/free-developer-offers/>.
- Instale las siguientes cargas de trabajo en Visual Studio:
  - ASP.NET y desarrollo web, incluidos los componentes opcionales.
  - Desarrollo de Azure.

En este ejemplo, creará una nueva aplicación MVC a partir de una plantilla y luego agregará la instrumentación de Application Insights. Puede utilizar el mismo procedimiento para agregar instrumentación a cualquiera de sus aplicaciones ASP.NET existentes:

1. Abra Visual Studio 2019.
2. En la ventana de inicio de Visual Studio, haga clic en el botón Crear un nuevo proyecto en la sección Comenzar en el lado derecho de la ventana.
3. En la ventana Crear un nuevo proyecto, en el cuadro de búsqueda, escriba **MVC** .
4. Seleccione la plantilla Aplicación web ASP.NET (.NET Framework).
5. Haga clic en el botón Siguiente en la esquina inferior derecha de la ventana.
6. Escriba un nombre para su proyecto y solución en los cuadros Nombre del proyecto y Nombre de la solución, respectivamente.
7. Seleccione la ubicación donde se almacenará su proyecto.

8. Haga clic en el botón Crear en la esquina inferior derecha de la ventana.
9. En la ventana Crear una nueva aplicación web ASP.NET, seleccione la plantilla MVC.
10. Haga clic en el botón Crear en la esquina inferior derecha de la ventana.
11. En la ventana del Explorador de soluciones, haga clic con el botón derecho en el nombre de su proyecto.
12. En el menú contextual, que se muestra en la Figura 4-7, seleccione Agregar> Telemetría de Application Insights.



**Figura 4-7** Adición de telemetría de Application Insights

13. En la página Configuración de Application Insights, haga clic en el botón Comenzar en la parte inferior de la página.
14. En la página Registre su aplicación con Application Insights, asegúrese de que la Cuenta de Azure y la Suscripción de Azure correctas estén seleccionadas en los menús desplegables.

15. Haga clic en el enlace Configurar ajustes debajo del menú desplegable Recursos.
16. En el cuadro de diálogo Configuración de Application Insights, seleccione el grupo de recursos y la ubicación donde desea crear el nuevo recurso de Application Insight.
17. Haga clic en el botón Registrarse.
18. En la pestaña Configuración de Application Insights, haga clic en el botón Recopilar seguimientos de System.Diagnostics en la parte inferior de la pestaña. Habilitar esta opción le permite enviar un mensaje de registro directamente a Application Insights.

En este punto, Visual Studio comienza a agregar los paquetes y las dependencias necesarios a su proyecto. Visual Studio también configura automáticamente la clave de instrumentación, lo que permite que su aplicación se conecte al recurso de Application Insights creado en Azure. Ahora su proyecto está conectado con la instancia de Application Insights implementada en Azure. Tan pronto como ejecute su proyecto, la instrumentación de Application Insights comienza a enviar información a Azure. Puede revisar esta información en Azure Portal o en Visual Studio. Utilice los siguientes pasos para acceder a Application Insights desde Visual Studio y Azure Portal:

1. Desde la ventana de Visual Studio, en la ventana del Explorador de soluciones, navegue hasta el nombre de su proyecto y seleccione Servicios conectados> Información de la aplicación.
2. Haga clic con el botón derecho en Application Insights.
3. En el menú contextual, haga clic en Buscar telemetría en vivo. La pestaña Búsqueda de Application Insights aparece en Visual Studio.
4. En el Explorador de soluciones, haga clic con el botón derecho en Application Insights para abrir Azure Portal Application Insights desde Visual Studio.
5. En el menú contextual, haga clic en Abrir portal de Application Insights.

Además de las métricas estándar que vienen de fábrica con la instrumentación predeterminada de Application Insights, también puede agregar sus eventos y métricas personalizados a su código. Con eventos y

métricas personalizados, puede analizar y solucionar problemas de lógica y flujos de trabajo que son específicos de su aplicación. El siguiente ejemplo muestra cómo modificar la aplicación MVC que creó en el ejemplo anterior para agregar eventos y métricas personalizados:

1. Abra el proyecto que creó en el ejemplo anterior.
2. Abra el archivo HomeController.cs.
3. Agregue las siguientes declaraciones using al principio del archivo:

[Haga clic aquí para ver la imagen del código](#)

```
using Microsoft.ApplicationInsights;
```

```
using System.Diagnostics;
```

4. Reemplace el contenido de la clase HomeController en el archivo HomeController.cs con el contenido del [Listado 4-3](#).

**Listado 4-3 Clase HomeController**

[Haga clic aquí para ver la imagen del código](#)

---

```
// C#. ASP.NET.
```

```
HomeController de clase pública: Controlador
```

```
{
```

```
    telemetría privada de TelemetryClient;
```

```
    private double indexLoadCounter;
```

```
    Public HomeController ()
```

```
{
```

```
        // Cree un TelemetryClient que pueda usarse
        durante la vida del
```

```
        // Controlador.
```

```
        telemetry = new TelemetryClient ();
```

```
        // Inicialice algunos contadores para las
        // métricas personalizadas.

        // Esta es una métrica falsa solo con fines
        // de demostración.

        indexLoadCounter = new Random ().Next
(1000);

    }
```

```
índice de resultado de acción público ()

{

    // Este ejemplo es trivial ya que
    ApplicationInsights ya registró el

    // carga de la página.

    // Puede usar este ejemplo para rastrear
    diferentes eventos en el

    // solicitud.

    telemetry.TrackEvent ("Cargando la página de
índice");

    // Antes de poder enviar una métrica
    // personalizada, debe utilizar GetMetric

    //método.

    telemetry.GetMetric
("CountOfIndexPageLoads").TrackValue
(indexLoadCounter);
```

```
// Este ejemplo trivial muestra cómo
rastrear excepciones usando Application

//Perspectivas.

// También puede enviar un mensaje de
seguimiento a Application Insights.

intentar

{

    Trace.TraceInformation ("Provocando una
excepción trivial");

    lanzar una nueva System.Exception (@
"Trivial Exception for testing Tracking

        Función de excepción en Application
Insights ");

}

catch (System.Exception ex)

{

    Trace.TraceError ("Capturar y
administrar la excepción trivial");

    telemetry.TrackException (ex);

}

// Debe indicarle al TelemetryClient que
envíe todos los datos en memoria a

// el ApplicationInsights.

telemetry.Flush ();

volver Ver ();
```

```
}
```

```
public ActionResult Acerca de ()
```

```
{
```

```
    ViewBag.Message = "La página de descripción  
de su aplicación.;"
```

```
        // Este ejemplo es trivial ya que  
ApplicationInsights ya registra el
```

```
        // carga de la página.
```

```
        // Puede usar este ejemplo para rastrear  
diferentes eventos en el
```

```
        // solicitud.
```

```
        telemetry.TrackEvent ("Cargando la página  
Acerca de");
```

```
volver Ver ();
```

```
}
```

```
Contacto público ActionResult ()
```

```
{
```

```
    ViewBag.Message = "Su página de contacto";
```

```
        // Este ejemplo es trivial ya que  
ApplicationInsights ya registra la carga
```

```
        // de la página.
```

```

        // Puede usar este ejemplo para rastrear
        diferentes eventos en el

        // solicitud.

        telemetry.TrackEvent ("Cargando la página de
contacto") ;

    volver Ver () ;

}

```

**5.** En el Explorador de soluciones, abra el archivo ApplicationInsights.config.

**6.** En el <Add  
Type="Microsoft.ApplicationInsights.Extensibility.PerfCounterCollect  
or.PerformanceCollectorModule,  
Microsoft.AI.PerfCounterCollector">elemento XML, agregue el  
siguiente elemento XML secundario:

[Haga clic aquí para ver la imagen del código](#)

```

<EnableIISExpressPerformanceCounters> verdadero
</EnableIISExpressPerformanceCounters>

```

### **Constructores de controladores de nota**

En el ejemplo anterior, usamos una propiedad privada en el constructor para crear e inicializar un objeto TelemetryClient. En una aplicación del mundo real, debe utilizar técnicas de inyección de dependencia para inicializar correctamente la clase Controller. Existen varios marcos, como Unity, Autofac o Ninject, que pueden ayudarlo a implementar el patrón de inyección de dependencia en su código.

En este punto, puede presionar F5 y ejecutar su proyecto para ver cómo su aplicación envía información a Application Insights. Si revisa la pestaña de búsqueda de Application Insights, puede ver los mensajes, que se muestran en la Figura 4-8, que su aplicación está enviando a Application Insights.

 5/25/2020 9:54:59 PM - Page View
Home Page - My ASP.NET Application
Page URL: <a href="https://localhost:44315/">https://localhost:44315/</a> Page view duration: 42.357s Browser: Chrome 83.0
 5/25/2020 9:54:53 PM - Exception
! Trivial Exception for testing Tracking Exception feature in Application Insights Exception type: System.Exception Failed method: ch4_2_1.Controllers.HomeController.Index
 5/25/2020 9:54:53 PM - Trace
Capturing and managing the trivial exception Severity level: Error Device type: PC
 5/25/2020 9:54:53 PM - Trace
Raising a trivial exception Severity level: Information Device type: PC
 5/25/2020 9:54:53 PM - Custom Event
Loading the Index page Device type: PC
 5/25/2020 9:54:53 PM - Request
GET Home/Index Request URL: <a href="https://localhost:44315/">https://localhost:44315/</a> Response code: 200 Response time: 4.745s

**Figura 4-8** Mensajes de Application Insights

Envía mensajes a Application Insights utilizando

la `TelemetryClass` clase. Esta clase le proporciona los métodos adecuados para enviar los diferentes tipos de mensajes a Application Insights. Puede enviar eventos personalizados mediante el `TrackEvent()` método. Utiliza este método para rastrear eventos significativos en su aplicación, como cuando el usuario crea un nuevo carrito de compras en una aplicación web de comercio electrónico o el usuario gana un juego en una aplicación móvil.

Si necesita realizar un seguimiento del valor de ciertas variables o propiedades en su código, puede usar la combinación de métodos `GetMetric()` y `TrackValue()`. El `GetMetric()` método recupera una métrica del espacio de nombres `azure.applicationinsight`. Si la métrica no existe en el espacio de nombres, la biblioteca de Application Insights crea automáticamente una nueva. Una vez que tenga una referencia a la métrica correcta, puede usar el `TrackValue()` método para agregar un valor a esa métrica. Puede utilizar estas métricas personalizadas para configurar alertas o reglas de escala automática. Use los siguientes pasos para ver las métricas personalizadas en Azure Portal:

1. Desde la ventana de Visual Studio, en la ventana del Explorador de soluciones, navegue hasta el nombre de su proyecto y seleccione Servicios conectados> Información de la aplicación.
2. Haga clic con el botón derecho en Application Insights.

3. En el menú contextual, haga clic en Abrir portal de Application Insights.
4. En la hoja de Application Insights, haga clic en Métricas en la sección Supervisión del menú de navegación en el lado izquierdo de la hoja.
5. En la hoja Métricas, en la barra de herramientas sobre el gráfico vacío, en el menú desplegable Espacio de nombres de métricas, seleccione azure.applicationinsight.
6. En el menú desplegable Métrica, seleccione CountOfIndexPageLoad. Esta es la métrica personalizada que definió en el ejemplo anterior.
7. En el menú desplegable Agregación, seleccione Recuento. Los valores de su gráfico serán diferentes pero deberían ser similares a los de la [Figura 4-9](#).



**Figura 4-9 Gráfico métrico personalizado**

También puede enviar mensajes de registro a Application Insights mediante la integración entre System.Diagnostics y Application Insights. Cualquier mensaje enviado al sistema de diagnóstico mediante la `Trace` clase aparece en Application Insights como un mensaje de

seguimiento. En esta misma línea, use el `TraceException()` método para enviar el seguimiento de la pila y la excepción a Application Insights. La ventaja de hacer esto es que puede correlacionar fácilmente las excepciones con las operaciones que estaban realizando su código cuando ocurrió la excepción.



### *Sugerencia para el examen*

Recuerde que Application Insights es una solución para monitorear el comportamiento de una aplicación en diferentes plataformas, escrita en diferentes idiomas. Puede utilizar Application Insights con aplicaciones web y aplicaciones nativas o aplicaciones móviles escritas en .NET, Java, JavaScript o Node.js. No es necesario ejecutar su aplicación en Azure. Solo necesita usar Azure para implementar el recurso de Application Insights que usa para analizar la información enviada por su aplicación.

### *¿Necesita más revisión? Creación de eventos y métricas personalizados*

Puede crear métricas y eventos más complejos que el que revisamos aquí. Para operaciones complejas, puede rastrear todas las acciones dentro de una operación para correlacionar correctamente todos los mensajes generados durante la ejecución de la operación. Puede obtener más información sobre cómo crear eventos y métricas personalizados leyendo el artículo en <https://docs.microsoft.com/en-us/azure/azure-monitor/app/api-custom-events-metrics> .

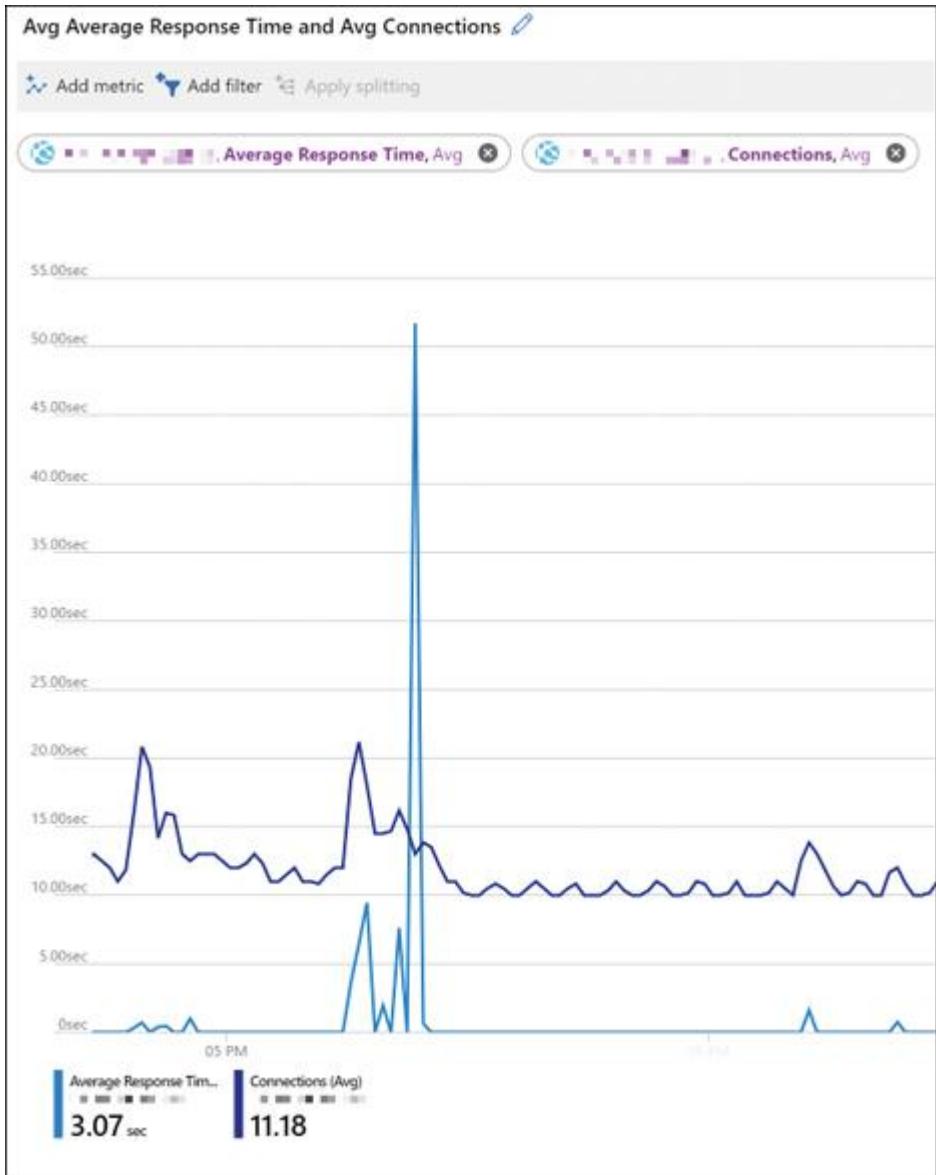
### *Analice los datos de registro y solucione problemas de soluciones mediante Azure Monitor*

Azure Monitor es una herramienta compuesta por varios elementos que lo ayudan a monitorear y comprender mejor el comportamiento de sus soluciones. Application Insights es una herramienta para recopilar información de sus soluciones. Una vez que tenga la información recopilada, puede utilizar las herramientas de análisis para revisar los datos y solucionar problemas de su aplicación. Según la información que necesite analizar, puede utilizar Metric Analytics o Log Analytics.

Puede utilizar Metric Analytics para revisar las métricas estándar y personalizadas enviadas desde su aplicación. Una métrica es un valor numérico que está relacionado con algún aspecto en un momento particular de su solución. El uso de CPU, la memoria libre y el número de solicitudes son ejemplos de métricas; Además, puede crear sus propias métricas personalizadas. Debido a que las métricas son livianas, puede

usarlas para monitorear escenarios casi en tiempo real. Los datos de las métricas se analizan representando los valores de las métricas en un intervalo de tiempo utilizando diferentes tipos de gráficos. Utilice los siguientes pasos para revisar gráficos:

1. Abra Azure Portal (<https://portal.azure.com>).
2. En el cuadro de texto Buscar recursos, servicios y documentos en la parte superior de Azure Portal, escriba monitor.
3. Haga clic en Supervisar en la sección Servicios en la lista de resultados.
4. En la hoja Monitor, haga clic en Métricas en el menú de navegación en el lado izquierdo de la hoja.
5. En la hoja Métricas, el panel Seleccionar un alcance debería aparecer automáticamente.
6. En el panel Seleccionar un alcance, en el árbol de alcance, seleccione la suscripción o los grupos de recursos que contienen el Servicio de aplicaciones de Azure que contiene las métricas que desea agregar al gráfico.
7. En el menú desplegable Tipo de recurso, debajo del árbol de alcance, seleccione solo el tipo de recurso de App Services.
8. En el menú desplegable de App Service, seleccione uno de sus App Services.
9. Haga clic en el botón Aplicar en la parte inferior del panel.
10. En la hoja Métricas, seleccione la métrica Tiempo de respuesta promedio en el menú desplegable Métrica.
11. Haga clic en el botón Agregar métrica en la parte superior del gráfico. Puede agregar varias métricas al mismo gráfico, lo que significa que puede analizar diferentes métricas que están relacionadas entre ellas.
12. Repita el paso 10 para agregar la métrica Conexiones. La Figura 4-10 muestra las métricas agregadas al gráfico.

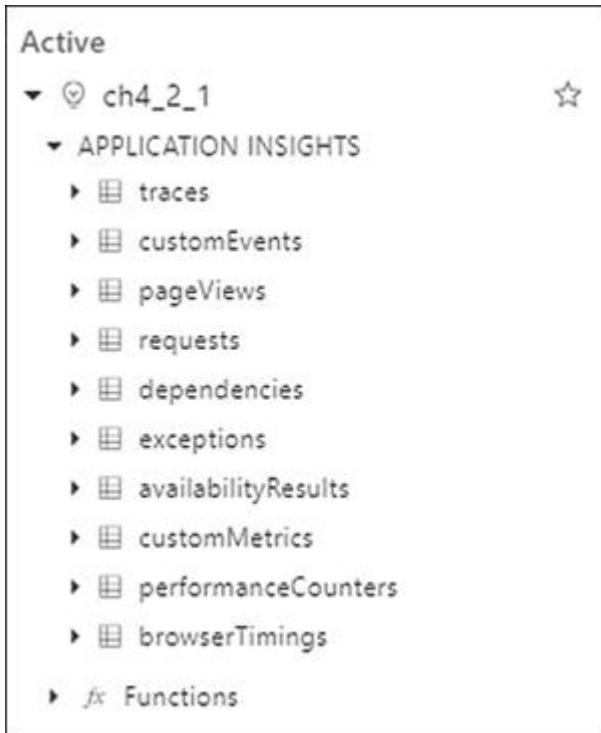


**Figura 4-10** Configuración de métricas para un gráfico

Utiliza Log Analytics para analizar el seguimiento, los registros, los eventos, las excepciones y cualquier otro mensaje enviado desde su aplicación. Los mensajes de registro son más complejos que las métricas porque pueden contener mucha más información que un simple valor numérico. Puede analizar los mensajes de registro mediante consultas para recuperar, consolidar y analizar los datos recopilados. Log Analytics para Azure Monitor usa una versión del lenguaje de consulta de Kusto. Puede crear sus consultas para obtener información de los datos almacenados en Azure Monitor. Para hacerlo, complete los siguientes pasos:

1. Abra Azure Portal (<https://portal.azure.com> ).
2. En el cuadro de texto Buscar recursos, servicios y documentos en la parte superior de Azure Portal, escriba **monitor** .
3. Haga clic en Supervisar en la sección Servicios en la lista de resultados.
4. En la hoja Monitor, haga clic en Registros en el menú de navegación en el lado izquierdo de la hoja.
5. En la hoja Registros, haga clic en el botón Comenzar.
6. En la hoja Registros, el panel Seleccionar un alcance debería aparecer automáticamente.
7. En el panel Seleccionar un alcance, en el árbol de alcance, navegue hasta los recursos que contienen los registros que desea consultar. Haga clic en la casilla de verificación junto al recurso. Puede seleccionar solo recursos del mismo tipo. Para este ejemplo, el tipo de recurso debe ser Application Insights.
8. Haga clic en el botón Aplicar en la parte inferior del panel.
9. En la hoja Registros, escriba **trazos** en el área de texto.
10. Haga clic en el botón Ejecutar.
11. Puede revisar el resultado de su consulta en la sección debajo del área de texto de la consulta.

Esta consulta simple devuelve todos los eventos de error de seguimiento almacenados en su espacio de trabajo de Application Insights. Puede utilizar consultas más complejas para obtener más información sobre su solución. Los campos disponibles para las consultas dependen de los datos cargados en el espacio de trabajo. El esquema de datos gestiona estos campos. [La Figura 4-11](#) muestra el esquema asociado con un lugar de trabajo que almacena datos de Application Insights.



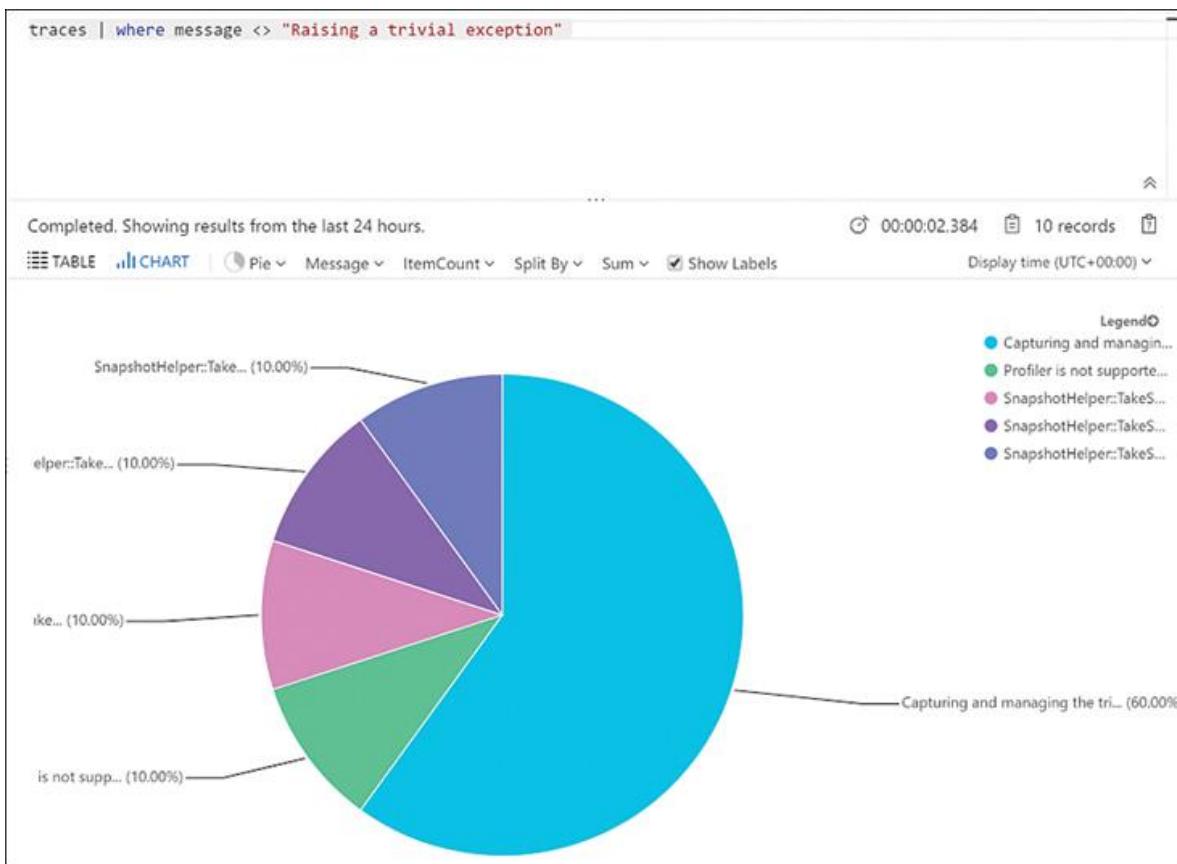
**Figura 4-11** Esquema del espacio de trabajo

Una vez que obtenga los resultados de una consulta, puede refinar fácilmente los resultados de la consulta agregando cláusulas where a la consulta. La forma más sencilla de agregar nuevos criterios de filtrado es expandir uno de los registros en la vista de tabla en la sección de resultados debajo del área de texto de la consulta. Si mueve el mouse sobre cada uno de los campos en un registro, puede ver tres pequeños puntos antes del campo del registro. Si hace clic en el ícono de tres puntos, aparece un menú contextual para incluir o excluir el valor del campo en la cláusula where. Según el ejemplo de la sección anterior, la siguiente consulta obtendría todos los rastros enviados desde la aplicación, excepto aquellos con el mensaje `Raising a trivial exception`.

[Haga clic aquí para ver la imagen del código](#)

```
rastros | donde el mensaje <> "Provocando una excepción trivial"
```

Puede revisar los resultados de esta consulta en formatos de tabla y gráfico. Con los diferentes formatos de visualización, puede obtener una visión diferente de los datos. [La Figura 4-12](#) muestra cómo se trazan los resultados de la consulta anterior en un gráfico circular.



**Figura 4-12** Representación de los resultados de la consulta  
**¿Necesita más revisión? Crear consultas de registro**

La creación de la consulta adecuada para sus necesidades depende en gran medida de los detalles de su solución. Puede revisar los detalles sobre el lenguaje de consulta de Kusto y cómo crear consultas complejas revisando los siguientes artículos:

- Idioma de consulta de Kusto: <https://docs.microsoft.com/en-us/azure/kusto/query/>
- Consultas de registro de Azure Monitor: <https://docs.microsoft.com/en-us/azure/azure-monitor/log-query/query-language>



### Sugerencia para el examen

Cuando intente consultar registros desde Azure Monitor, recuerde que debe habilitar los registros de diagnóstico para Azure App Services. Si recibe el mensaje, *No encontramos ningún registro* cuando intenta consultar los registros de su Servicio de aplicaciones de Azure, eso podría significar que necesita configurar los ajustes de diagnóstico en su Servicio de aplicaciones.

## *Implementar alertas y pruebas web de Application Insights*

Como resultado del análisis de los datos enviados desde su aplicación al Azure Monitor mediante Application Insights, es posible que encuentre algunas situaciones que necesite supervisar con más atención. Con Azure Monitor, puede configurar alertas según el valor de diferentes métricas o registros. Por ejemplo, puede crear una alerta para recibir una notificación cuando su aplicación genere un código de retorno HTTP 502.

También puede configurar Application Insights para monitorear la disponibilidad de su aplicación web. Puede configurar diferentes tipos de pruebas para comprobar la disponibilidad de su aplicación web:

- **Prueba de ping de URL** Esta es una prueba simple para verificar si su aplicación está disponible realizando una solicitud a una única URL para su aplicación.
- **Prueba web de varios pasos** Con Visual Studio Enterprise, puede registrar los pasos que desea usar como verificación para su aplicación. Utiliza este tipo de prueba para comprobar escenarios complejos. El proceso de registrar los pasos en una aplicación web genera un archivo con los pasos registrados. Con este archivo generado, puede crear una prueba web en Application Insights; luego carga el archivo de grabación.
- **Prueba de disponibilidad de pista personalizada** Puede crear su propia prueba de disponibilidad en su código utilizando el `TrackAvailability()` método.

Al crear una prueba de ping de URL, puede verificar no solo el código de respuesta HTTP, sino también el contenido devuelto por el servidor. De esta forma, puede minimizar la posibilidad de falsos positivos. Estos falsos positivos pueden ocurrir si el servidor devuelve un código de respuesta HTTP válido, pero el contenido es diferente debido a errores de configuración. Utilice el siguiente procedimiento para crear una prueba de ping de URL en su Application Insights que verifique la disponibilidad de su aplicación web:

1. Abra Azure Portal (<https://portal.azure.com>).
2. En el cuadro de texto Buscar recursos, servicios y documentos en la parte superior de Azure Portal, escriba **monitor**.

3. Haga clic en Supervisar en la sección Servicios en la lista de resultados.
4. En la hoja Monitor, haga clic en Aplicaciones en la sección Insights.
5. En la hoja Aplicaciones, haga clic en el nombre del recurso de Application Insights donde desea configurar la alerta.
6. En la hoja Applications Insights, haga clic en Disponibilidad en la sección Investigar del menú de navegación en el lado izquierdo de la hoja.
7. En la hoja Disponibilidad, haga clic en Agregar prueba en la parte superior izquierda de la hoja.
8. En la hoja Crear prueba, que se muestra en la [Figura 4-13](#), escriba un nombre para la prueba en el cuadro de texto Nombre de la prueba.

**Create test**

Basic Information

\* Test name  
Give your test a name

Learn more about configuring tests against applications hosted behind a firewall

Test type  
URL ping test

\* URL  
[empty input field]

Parse dependent requests ⓘ

Enable retries for availability test failures. ⓘ

Test frequency ⓘ  
5 minutes

Test locations  
5 location(s) configured

Success criteria  
HTTP response: 200, Test Timeout: 120 seconds

Alerts  
Enabled

**Figura 4-13 Creación de una prueba de URL**

9. Asegúrese de que la prueba de ping de URL esté seleccionada en el menú desplegable Tipo de prueba.
10. En el cuadro de texto URL, escriba la URL de la aplicación que desea probar.
11. Expanda la sección Ubicación de la prueba. Seleccione las ubicaciones desde las que desea realizar la prueba de ping de URL.
12. Deje las otras opciones como están.
13. Haga clic en el botón Crear en la parte inferior del panel.

Cuando configura la prueba de ping de URL, no puede configurar la alerta directamente durante el proceso de creación. Debe finalizar la creación de la prueba y luego puede editar la alerta para definir las acciones que desea realizar cuando se activa la alerta. Utilice el siguiente procedimiento para configurar una alerta asociada con la prueba de ping de URL que configuró anteriormente:

1. En la hoja Disponibilidad, haga clic en los puntos suspensivos junto a la alerta recién creada.
2. En el menú contextual, haga clic en Abrir página de reglas (alertas).
3. En la hoja Administración de reglas de alerta, en la sección Condición, asegúrese de que haya una condición predeterminada con el nombre *siempre que el promedio de ubicaciones fallidas sea mayor o igual a 2 recuentos*.
4. En la sección Grupo de acción, haga clic en el enlace Seleccionar grupo de acción.
5. En el panel Acciones configuradas, haga clic en el botón Crear grupo de acciones.
6. En el panel Seleccione un grupo de acciones para adjuntar a esta regla de alerta, haga clic en Crear grupo de acciones.
7. En el panel Crear grupo de acciones, seleccione un grupo de recursos para guardar este grupo de acciones. Alternativamente, puede crear un nuevo grupo de recursos haciendo clic en el enlace Crear nuevo debajo del menú desplegable Grupo de recursos.
8. Escriba un nombre en el cuadro de texto Nombre del grupo de acciones. Este nombre debe ser único en el grupo de recursos que seleccionó en el paso anterior.
9. Haga clic en el botón Siguiente: Notificaciones en la parte inferior del panel.
10. En la sección Notificaciones, en el menú desplegable Tipo de notificación, seleccione Correo electrónico / Mensaje SMS / Push / Voz.
11. En el panel Correo electrónico / SMS / Push / Voice, seleccione la casilla de verificación Correo electrónico.

12. Escriba una dirección de correo electrónico en el cuadro de texto debajo de la casilla de verificación Correo electrónico.
13. Haga clic en el botón Aceptar en la parte inferior del panel.
14. Escriba un nombre en el cuadro de texto Nombre, junto al menú desplegable Tipo de notificación.
15. Haga clic en Siguiente: Acciones en la parte inferior del panel.
16. Deje la sección Acciones como está. Puede usar esta sección para configurar acciones como llamar a una función de Azure, crear un ticket en un sistema ITSM o iniciar un Runbook de automatización de Azure.
17. Haga clic en el botón Revisar y crear.
18. Haga clic en el botón Crear.
19. En la hoja Administración de reglas de alerta, asegúrese de que el Grupo de acción recién creado se haya agregado correctamente a la lista de Grupos de acción adjunta a la alerta.
20. Haga clic en el botón Guardar en la esquina superior izquierda de la hoja Administración de reglas de alerta.

Ahora puede probar si la prueba de ping de la URL está funcionando correctamente cerrando temporalmente su aplicación de prueba. Después de cinco minutos, debería recibir un mensaje de correo electrónico en la dirección de correo electrónico que configuró en la acción de alerta asociada con la prueba de ping de URL.



### *Sugerencia para el examen*

Recuerde que necesita una licencia de Visual Studio Enterprise para crear pruebas web de varios pasos. Utilice Visual Studio Enterprise para la definición de los pasos que forman parte de la prueba y, a continuación, cargue la definición de la prueba en Azure Application Insights.

### *¿Necesita más revisión? Alertas de Azure Monitor*

Además de crear alertas cuando falla una prueba web, también puede crear alertas basadas en otras condiciones que dependen de la información de eventos almacenada en Application Insights. Puede revisar los detalles sobre cómo crear estas alertas revisando el artículo en <https://docs.microsoft.com/en-us/azure/azure-monitor/platform/alerts-log>.

## *Implementar código que maneje fallas transitorias*

Desarrollar aplicaciones para la nube significa que su aplicación depende de los recursos en la nube para ejecutar su código. Como ya revisamos en capítulos anteriores, estos recursos brindan alta disponibilidad lista para usar y características tolerantes a fallas que hacen que su aplicación sea más resistente. Los servicios en la nube de Azure utilizan equilibradores de carga y hardware redundantes. Aunque tiene la garantía de no sufrir grandes averías, puede haber situaciones que puedan afectar temporalmente su aplicación, como realizar conmutaciones por error automáticas u operaciones de equilibrio de carga. Por lo general, la recuperación de ese tipo de situación transitoria es tan simple como volver a intentar la operación que estaba realizando su aplicación. Por ejemplo, si su aplicación estaba leyendo un registro de una base de datos y obtiene un error de tiempo de espera debido a una sobrecarga temporal de la base de datos,

Lidiar con estas fallas transitorias lo lleva a enfrentar algunos desafíos interesantes. Su aplicación debe responder a estos desafíos para garantizar que ofrece una experiencia confiable a sus usuarios. Estos desafíos son

- **Detectar y clasificar fallas** No todas las fallas que pueden ocurrir durante la ejecución de la aplicación son transitorias. Su aplicación necesita identificar si la falla es transitoria, duradera o una falla de terminal. Incluso el término "falla duradera" depende de la lógica de su aplicación porque la cantidad de tiempo que considera "duradera" depende del tipo de operaciones que realiza su aplicación. Su aplicación también debe lidiar con las diferentes respuestas que provienen de diferentes tipos de servicios. Un error que ocurre al leer datos de un sistema de almacenamiento es diferente de un error que ocurre al escribir datos.
- **Vuelva a intentar la operación cuando sea apropiado** Una vez que su aplicación determina que está tratando con una falla transitoria, la aplicación necesita reintentar la operación. También necesita realizar un seguimiento del número de reintentos de la operación con fallas.
- **Implementar una estrategia de reintentos adecuada**  
**Reintentar** la operación de forma indefinida podría generar otros problemas, como la degradación del rendimiento o el bloqueo de

los recursos que utiliza su aplicación. Para evitar esos problemas de rendimiento, su aplicación debe establecer una estrategia de reinicio que defina el número de reinicios, establece el retraso entre cada reinicio y establece las acciones que su aplicación debe realizar después de un intento fallido. Establecer el número correcto de reinicios y el retraso entre ellos es una tarea compleja que depende de factores como el tipo de recursos, las condiciones de funcionamiento y la propia aplicación.

Puede utilizar las siguientes pautas al implementar un mecanismo de falla transitoria adecuado en su aplicación:

- **Utilice el mecanismo de reinicio integrado existente** Al trabajar con SDK para servicios específicos, el SDK generalmente proporciona un mecanismo de reinicio integrado. Antes de pensar en implementar su mecanismo de reinicio, debe revisar el SDK que está utilizando para acceder a los servicios de los que depende su aplicación y utilizar el mecanismo de reinicio integrado. Estos mecanismos de reinicio integrados se adaptan a las características y requisitos específicos del servicio de destino. Si aún necesita implementar su mecanismo de reinicio para un servicio, como un servicio de almacenamiento o un bus de servicio, debe revisar cuidadosamente los requisitos de cada servicio para asegurarse de administrar correctamente las respuestas con fallas.
- **Determine si la operación es adecuada para reiniciar** Cuando se genera un error, generalmente indica la naturaleza del error. Puede utilizar esta información para determinar si el error es un fallo transitorio. Una vez que determine que su aplicación está lidiando con una falla transitoria, debe determinar si reiniciar la operación puede tener éxito. No debe reiniciar operaciones que indiquen una operación no válida, como un servicio que sufrió un error fatal o continuar buscando un elemento después de recibir un error que indica que el elemento no existe en la base de datos. Debe implementar reinicios de operación si se cumplen las siguientes condiciones:
  - Puede determinar el efecto completo de la operación.
  - Comprende completamente las condiciones del reinicio.
  - Puede validar estas condiciones.

- **Utilice el intervalo y el recuento de reintentos adecuados** Establecer un recuento de reintentos incorrecto podría hacer que la aplicación falle o bloquear recursos que pueden afectar el estado de la aplicación. Si configura el recuento de reintentos demasiado bajo, es posible que su aplicación no tenga tiempo suficiente para recuperarse de la falla transitoria y fallará. Si configura el recuento de reintentos en un valor demasiado alto o demasiado corto, puede bloquear los recursos que usa su aplicación, como subprocessos, conexiones o memoria. Este alto consumo de recursos puede afectar la salud de su aplicación. Al elegir el recuento y el intervalo de reintentos adecuados, debe considerar el tipo de operación que sufrió la falla transitoria. Por ejemplo, si la falla transitoria ocurre durante una operación que es parte de la interacción del usuario, debe usar un intervalo de reinicio corto y contar, lo que evita que su usuario espere demasiado para que su aplicación se recupere de la falla transitoria. Por otro lado, si la falla ocurre durante una operación que es parte de un flujo de trabajo crítico, establecer un recuento e intervalo de reintentos más largos tiene sentido si reiniciar el flujo de trabajo lleva mucho tiempo costoso. A continuación, se muestran algunas de las estrategias más comunes para elegir el intervalo de reinicio:

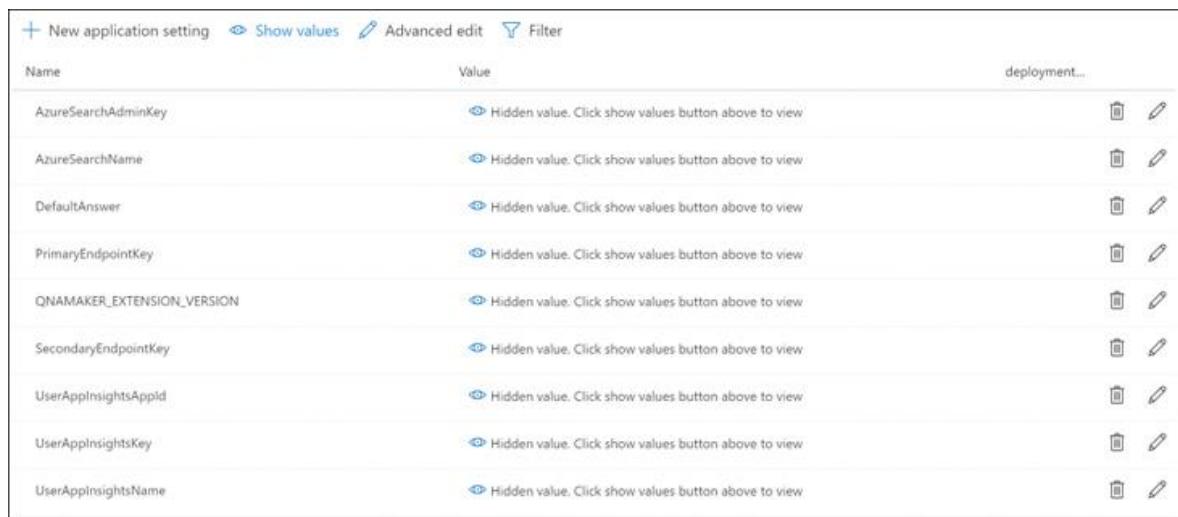
- **Retroceso exponencial** Utiliza un intervalo de tiempo corto para el primer reinicio y luego aumenta exponencialmente el tiempo del intervalo para los reinicios posteriores. Por ejemplo, establece el intervalo inicial en 3 segundos y luego usa 9, 27, 81 para los reinicios posteriores.
- **Intervalos incrementales** Establece un intervalo de tiempo corto para el primer reinicio, luego aumenta gradualmente el tiempo del intervalo para los reinicios posteriores. Por ejemplo, establece el intervalo inicial en 3 segundos y luego usa 5, 8, 13, 21 para los reinicios posteriores.
- **Intervalos regulares** Utiliza el mismo intervalo de tiempo para cada reinicio. Esta estrategia no es apropiada en la mayoría de los casos. Debe evitar usar esta estrategia al acceder a servicios o recursos en Azure. En esos casos, debe

utilizar la estrategia de retroceso exponencial con un patrón de disyuntor.

- **Reintento inmediato** Vuelva a intentarlo tan pronto como ocurra la falla transitoria. No debe utilizar este tipo de reinicio más de una vez. Los reinicios inmediatos son adecuados para fallas máximas, como colisiones de paquetes de red o picos en componentes de hardware. Si el reinicio inmediato no se recupera de la falla transitoria, debe cambiar a otra estrategia de reinicio.
- **Aleatorización** Si su aplicación ejecuta varios reinicios en paralelo, independientemente de la estrategia de reinicio, el uso de los mismos valores de reinicio para todos los reinicios puede afectar negativamente a su aplicación. En general, debe utilizar valores de intervalo de reinicio de inicio aleatorios con cualquiera de las estrategias anteriores. Esto le permite minimizar la probabilidad de que dos subprocesos de aplicación diferentes inicien el mecanismo de reinicio al mismo tiempo en caso de una falla transitoria.
- **Evite los antipatrones** Al implementar su mecanismo de reinicio, hay algunos patrones que debe evitar:
  - Evite implementar capas duplicadas de reinicios. Si su operación se realiza a partir de varias solicitudes a varios servicios, debe evitar implementar reinicios en cada etapa de la operación.
  - Nunca implemente mecanismos de reinicio sin fin. Si su aplicación nunca deja de volver a intentarlo en caso de un error transitorio, la aplicación puede provocar el agotamiento de los recursos o la limitación de la conexión. Debe utilizar el patrón de disyuntor o un número finito de reinicios.
  - Nunca use el reinicio inmediato más de una vez.
- **Probar la estrategia y la implementación de reinicios** Debido a las dificultades al seleccionar el recuento de reinicios y los valores de intervalo correctos, debe probar a fondo la estrategia y la implementación de reinicios. Debe prestar especial atención a los escenarios de alta carga y

simultaneidad. Debe probar esto inyectando fallas transitorias y no transitorias en su aplicación.

- **Administrar la configuración de la política de reinicio** Cuando implementa su mecanismo de reinicios, no debe codificar los valores para el recuento de reinicios y los intervalos. En cambio, puede definir una política de reinicios que contenga el recuento y el intervalo de reinicios, así como el mecanismo que determina si una falla es transitoria o no transitoria. Debe almacenar esta política de reinicio en archivos de configuración para poder ajustar la política. También debe implementar esta configuración de política de reinicio para que su aplicación almacene los valores en la memoria en lugar de volver a leer continuamente el archivo de configuración. Si usa Azure App Service, debe considerar usar la configuración del servicio que se muestra en la [Figura 4-14](#).



The screenshot shows the 'Configuration' blade for an Azure App Service. At the top, there are four buttons: '+ New application setting', 'Show values', 'Advanced edit', and 'Filter'. Below these are three dropdown menus: 'Name', 'Value', and 'deployment...'. A table lists nine application settings:

Name	Value	deployment...
AzureSearchAdminKey	Hidden value. Click show values button above to view	 
AzureSearchName	Hidden value. Click show values button above to view	 
DefaultAnswer	Hidden value. Click show values button above to view	 
PrimaryEndpointKey	Hidden value. Click show values button above to view	 
QNAMAKER_EXTENSION_VERSION	Hidden value. Click show values button above to view	 
SecondaryEndpointKey	Hidden value. Click show values button above to view	 
UserAppInsightsAppId	Hidden value. Click show values button above to view	 
UserAppInsightsKey	Hidden value. Click show values button above to view	 
UserAppInsightsName	Hidden value. Click show values button above to view	 

**Figura 4-14** Configuración de la aplicación Azure App Service

- **Registrar fallas transitorias y no transitorias** Debe incluir un mecanismo de registro en su aplicación cada vez que ocurra una falla transitoria o no transitoria. Una sola falla transitoria no indica un error en su aplicación. Si el número de fallas transitorias está aumentando, esto puede ser un indicador de una falla potencial más significativa o que debe aumentar los recursos asignados al servicio fallido. Debe registrar las fallas transitorias como mensajes de advertencia en lugar de errores. El uso del nivel de registro de errores podría provocar la activación de falsas alertas en su sistema de monitoreo. También debe considerar medir y

registrar el tiempo total que tarda su mecanismo de reintento cuando recupera una operación defectuosa. Esto le permite medir el impacto general de las fallas transitorias en los tiempos de respuesta del usuario, la latencia del proceso y la eficiencia de la aplicación.

#### *¿Necesita más revisión? Manejo de fallas transitorias*

Puede revisar algunas pautas generales para implementar un mecanismo de manejo de fallas transitorias revisando los siguientes artículos:

- <https://docs.microsoft.com/en-us/azure/architecture/best-practices/transient-faults>
- <https://docs.microsoft.com/en-us/aspnet/aspnet/overview/developing-apps-with-windows-azure/building-real-world-cloud-apps-with-windows-azure/transient-fault-manage>

#### *¿Necesita más revisión? Patrones útiles*

Al implementar su mecanismo de reintento, puede utilizar los siguientes patrones:

- Patrón de reintento Puede revisar los detalles y ejemplos de cómo implementar el patrón leyendo el artículo en <https://docs.microsoft.com/en-us/azure/architecture/patterns/retry>.
- Patrón de circuito Puede revisar los detalles y ejemplos de cómo implementar el patrón leyendo el artículo en <https://docs.microsoft.com/en-us/azure/architecture/patterns/circuit-breaker>.



#### *Sugerencia para el examen*

Recuerde probar cuidadosamente su estrategia de reintento. El uso de una estrategia de reintento incorrecta podría hacer que su aplicación agote los recursos necesarios para ejecutar su código. Una estrategia de reintento incorrecta puede conducir potencialmente a bucles infinitos si no usa disyuntores.

## RESUMEN DEL CAPÍTULO

---

- Su aplicación necesita poder administrar fallas transitorias.
- Debe determinar el tipo de falla antes de volver a intentar la operación.
- No debe utilizar el reintento inmediato más de una vez.

- Debe utilizar valores iniciales aleatorios para los períodos de reinicio.
- Debe utilizar el mecanismo SDK integrado cuando esté disponible.
- Debe probar su cuenta de reinicios y su estrategia de intervalo.
- Debe registrar fallas transitorias y no transitorias.
- Puede mejorar el rendimiento de su aplicación agregando un caché a su aplicación.
- Azure Cache para Redis permite el almacenamiento en caché de contenido dinámico.
- Con Azure Cache para Redis, puede crear bases de datos en memoria para almacenar en caché los valores más utilizados.
- Azure Cache para Redis le permite usar patrones de cola de mensajería.
- Las redes de entrega de contenido (CDN) almacenan y distribuyen contenido estático en servidores distribuidos por todo el mundo.
- Las CDN reducen la latencia al entregar el contenido desde el servidor más cercano al usuario.
- Puede invalidar el contenido de la caché configurando un TTL (tiempo de vida) bajo.
- Puede invalidar el contenido de la caché eliminando todo o parte del contenido de la caché.
- Application Insights obtiene información de su aplicación y la envía a Azure.
- Puede utilizar Application Insights con diferentes plataformas e idiomas.
- Application Insights es parte del servicio Azure Monitor.
- Application Insights genera dos tipos de información: métricas y registros.
- Application Insights le permite crear pruebas web para monitorear la disponibilidad de su aplicación.

- Puede configurar alertas y desencadenar diferentes acciones asociadas con las pruebas web.

## EXPERIMENTO MENTAL

---

En este experimento mental, demuestre sus habilidades y conocimiento de los temas cubiertos en este capítulo. Puede encontrar respuestas a este experimento mental en la siguiente sección.

Su empresa tiene una aplicación de línea de negocio (LOB) que ha sido desarrollada por su equipo. Esta aplicación LOB es una aplicación de comercio electrónico que tiene más uso durante los períodos de vacaciones. La aplicación LOB necesita obtener información de sistemas externos. Estás recibiendo algunas quejas sobre la estabilidad y el rendimiento de la aplicación. Responda las siguientes preguntas sobre la resolución de problemas y el rendimiento de la aplicación:

1. Despues de revisar las métricas de su aplicación en Azure Monitor, descubre que no tiene suficientes detalles sobre el rendimiento de los flujos de trabajo de la aplicación interna. ¿Qué debe hacer para obtener información sobre los flujos de trabajo internos?
2. Despues de revisar las métricas de su aplicación en Azure Monitor, encontrará que algunos de los problemas de estabilidad se deben a los sistemas externos. Debe minimizar el efecto en la experiencia del usuario. ¿Qué estrategia deberías utilizar?
3. Debe asegurarse de que el proceso de compra esté funcionando correctamente. Decide configurar una prueba web en Application Insights. ¿Qué tipo de prueba debería configurar?

## RESPUESTAS DEL EXPERIMENTO MENTAL

---

Esta sección contiene la solución al experimento mental. Cada respuesta explica por qué la opción de respuesta es correcta.

1. Debe integrar los instrumentos de Application Insights con su código. Una vez que integre Application Insights con su código, puede rastrear eventos personalizados en su código. Puede definir operaciones dentro de su código para rastrear operaciones complejas compuestas por varias tareas. Esto le permite obtener más información sobre los flujos de trabajo internos ejecutados en la aplicación. La supervisión basada en agentes de Application Insights no proporciona suficiente información.

**2.** Cuando se trata de la experiencia del usuario, debe considerar implementar una estrategia de reinicio que consista en una pequeña cantidad de reinicios con un intervalo de reinicio corto. El uso de este tipo de estrategia le permite minimizar el tiempo que sus usuarios deben esperar su aplicación para recuperarse de una falla transitoria. También puede considerar el uso de un reinicio inmediato como primer reinicio. Si este primer reinicio falla, debe cambiar a otra estrategia de reinicio. No existe una estrategia única para todos, por lo que debe probar su estrategia para asegurarse de brindar la mejor experiencia de usuario.

**3.** El proceso de compra en una aplicación web es un escenario de prueba complejo. En este escenario, debe utilizar una prueba web de varios pasos. Con Visual Studio Enterprise, debe registrar los pasos necesarios para realizar una compra en su aplicación web. Una vez que haya generado el archivo con los pasos registrados, puede crear una prueba web en Application Insights para monitorear el proceso de compra.

# Capítulo 5. Conectarse y consumir servicios de Azure y servicios de terceros

Hoy en día, las empresas utilizan diferentes sistemas para diferentes tareas que suelen ser realizadas por diferentes departamentos. Aunque estos sistemas separados funcionan para resolver una necesidad específica, generalmente actúan como actores independientes en un gran escenario. Estos actores independientes manejan información sobre la empresa que potencialmente puede ser duplicada por otros actores independientes.

Cuando una empresa se da cuenta de que los actores independientes están administrando sus datos, generalmente intentan que todos los actores o sistemas independientes trabajen juntos y comparten información entre ellos. Esta situación es independiente del uso de servicios en la nube o servicios locales. Para que los actores independientes trabajen juntos, es necesario establecer conexiones entre cada actor o servicio que necesita comunicarse con el otro.

Puede utilizar diferentes servicios y técnicas para lograr esta interconexión. Azure proporciona algunos servicios útiles que permiten que diferentes servicios funcionen juntos sin realizar grandes cambios en los servicios interconectados.

## **Habilidades cubiertas en este capítulo:**

- [Habilidad 5.1: Desarrollar una aplicación lógica de servicio de aplicaciones](#)
- [Habilidad 5.2: Implementar la gestión de API](#)
- [Habilidad 5.3: Desarrollar soluciones basadas en eventos](#)
- [Habilidad 5.4: Desarrollar soluciones basadas en mensajes](#)

## **HABILIDAD 5.1: DESARROLLAR UNA APLICACIÓN LÓGICA DE SERVICIO DE APLICACIONES**

---

El intercambio de información entre diferentes aplicaciones es un objetivo para la mayoría de las empresas. Compartir la información

enriquece el proceso interno y crea una mayor comprensión de la información en sí. Al utilizar la aplicación App Service Logic, puede crear flujos de trabajo que interconectan diferentes sistemas en función de condiciones y reglas y facilita el proceso de compartir información entre ellos. Además, puede aprovechar las funciones de Logic Apps para implementar flujos de trabajo de procesos comerciales.

### Esta habilidad cubre cómo

- [Crear una aplicación lógica](#)
- [Cree un conector personalizado para Logic Apps](#)
- [Crea una plantilla personalizada para Logic Apps](#)

#### *Crear una aplicación lógica*

Antes de poder interconectar dos servicios separados, debe comprender completamente qué información necesita compartir entre los servicios. A veces, la información debe sufrir algunas transformaciones antes de que un servicio pueda consumirla. Podría escribir código para realizar esta interconexión, pero esta es una tarea que consume mucho tiempo y es propensa a errores.

Azure proporciona App Service Logic Apps que permite interconectar dos o más servicios que comparten información entre ellos. Un proceso empresarial define esta interconexión entre diferentes servicios. Azure Logic Apps le permite crear escenarios de interconexión complejos mediante el uso de algunos elementos que facilitan el trabajo:

- **Flujos de trabajo** Defina el origen y el destino de la información. Se conecta a diferentes servicios mediante conectores. Un flujo de trabajo define los pasos o acciones que la información debe realizar para entregar la información desde el origen al destino correcto. Utiliza un lenguaje gráfico para visualizar, diseñar, construir, automatizar e implementar un proceso empresarial.
- **Conectores administrados** Un conector es un objeto que permite que su flujo de trabajo acceda a datos, servicios y sistemas. Microsoft proporciona algunos conectores prediseñados para los servicios de Microsoft. Estos conectores son administrados por Microsoft y proporcionan los disparadores y objetos de acción necesarios para trabajar con esos servicios.

- **Activadores** Los activadores son eventos que se activan cuando se cumplen determinadas condiciones. Utiliza un disparador como entrada o punto de partida de un flujo de trabajo. Por ejemplo, cuando llega un nuevo mensaje al buzón de compras de su empresa, puede iniciar un flujo de trabajo que puede acceder a la información del asunto y el cuerpo del mensaje y crear una nueva entrada en el sistema ERP.
- **Acciones** Las acciones son cada uno de los pasos que configura en su flujo de trabajo. Las acciones ocurren solo cuando se ejecuta el flujo de trabajo. El flujo de trabajo comienza a ejecutarse cuando se activa un nuevo disparador.
- **Paquete de integración empresarial** Si necesita realizar integraciones más avanzadas, el paquete de integración empresarial le proporciona capacidades de BizTalk Server.

**Nota:** Azure Logic APP, Azure Functions, Azure APP Service Webjobs y Microsoft Flow

Si necesita implementar flujos de trabajo, Microsoft proporciona algunos productos que puede utilizar para esa tarea. Aunque existe cierta superposición de las características proporcionadas por Logic Apps, Functions, App Service WebJobs y Power Automate, están diseñadas para diferentes escenarios. Puede revisar más detalles sobre los escenarios apropiados para cada producto en <https://docs.microsoft.com/en-us/azure/azure-functions/functions-compare-logic-apps-ms-flow-webjobs>.

Puede usar Azure Logic Apps para diferentes propósitos. La aplicación más obvia para Azure Logic Apps sería implementar procesos comerciales. Aunque no existe un mapeo directo entre las acciones de Azure Logic Apps y Business Process Model Notation (BPMN), puede usar Logic Apps para automatizar algunos procesos comerciales simples o integrarlo con motores de Business Process Model (BPM) para implementar procesos comerciales más complejos. También puede usar Azure Logic Apps para enviar notificaciones cuando ocurren ciertos eventos o crear la estructura de carpetas y permisos en una biblioteca de documentos de SharePoint Online cuando un gerente de proyecto de su empresa crea un nuevo proyecto.

Cuando crea un nuevo flujo de trabajo de la aplicación Azure Logic, debe pensar en cómo comenzará este flujo de trabajo. Este es el detonante de su flujo de trabajo. Un desencadenador puede ser un evento que ocurrió en un servicio, como que se cargó un nuevo archivo en una cuenta de Azure Storage. Un flujo de trabajo también puede comenzar en función de

un cronograma. La programación que configura para iniciar un flujo de trabajo es el desencadenante del flujo de trabajo. Cuando establece su programación y llega el momento adecuado, el motor de Azure Logic Apps crea una nueva instancia de su flujo de trabajo. Puede configurar dos tipos diferentes de horarios:

- **Recurrencia** En este tipo de disparador, se configura un intervalo de tiempo regular. Puede configurar una fecha y hora de inicio, y también puede configurar la zona horaria para su horario. Cuando configura el intervalo de tiempo, puede elegir entre segundos y meses como frecuencia. Por ejemplo, puede configurar una repetición de la ejecución del flujo de trabajo cada 2 minutos o cada 3 semanas. Dependiendo del intervalo que elija, puede seleccionar detalles adicionales para ese intervalo. Para el intervalo de la *semana*, puede seleccionar en qué días se ejecutará el flujo de trabajo. Por el *día* o la *semanaintervalos*, puede seleccionar las horas o los minutos para la ejecución de su flujo de trabajo. El desencadenador de recurrencia no procesa las recurrencias que faltan. Es decir, si falta una recurrencia por cualquier motivo, el desencadenador de recurrencia no reinicia la recurrencia faltante.
- **Ventana deslizante** Este tipo de desencadenante es similar al desencadenante de recurrencia, excepto que no puede configurar los ajustes de programación por adelantado, como días específicos de la semana u horas o minutos de un día. Otra diferencia esencial es que con el desencadenador de ventana deslizante, si falta una recurrencia, el desencadenador de ventana deslizante retrocede y procesa la recurrencia faltante.

Programar la ejecución de su aplicación Azure Logic no es la única forma de iniciar su flujo de trabajo. Puede utilizar otros activadores para iniciar la ejecución del flujo de trabajo. Además de activadores de recurrencia que revisamos anteriormente en esta sección, puede utilizar dos tipos adicionales de activadores:

- **Sondeo** El disparador consulta el sistema o servicio configurado periódicamente para obtener nuevos datos o si ocurrió un nuevo evento. Dependiendo del desencadenante específico, puede configurar el programa de sondeo. Una vez que ocurren los nuevos datos o eventos, el activador crea una nueva instancia de su

flujo de trabajo, recopila la información del sistema y pasa la información a la instancia de flujo de trabajo recién creada.

- **Empujar** El disparador escucha si llegan nuevos eventos o datos al sistema o servicio configurado. Tan pronto como ocurren los nuevos datos o eventos, el disparador crea una nueva instancia de su flujo de trabajo, pasando los datos a la instancia recién creada.

Una vez que comienza el flujo de trabajo, debe seguir algunos pasos para realizar el trabajo que se supone que debe hacer. Cada uno de estos pasos es una acción. Una acción puede ser algo como obtener datos de un servicio OData leyendo información de un archivo de texto almacenado en un servicio SFTP, o incluso transformar el formato de un archivo. También hay otro tipo de acción que es tan importante como las acciones de recopilación o transformación de datos. Establecer el valor de una variable, los bucles, las declaraciones condicionales o de conmutación o la bifurcación de decisiones son otros tipos de acciones que son fundamentales para definir su flujo de trabajo. Al igual que con cualquier otro lenguaje de programación, estas acciones estructurales le permiten controlar el flujo de ejecución de su flujo de trabajo.

Los activadores y las acciones se empaquetan juntos en conectores. Los conectores se utilizan para acceder a datos, eventos y acciones disponibles desde otras aplicaciones o servicios. Azure Logic Apps ofrece miles de conectores, pero todos encajan en una de las siguientes categorías:

- **Integrado** Este tipo de conector contiene los activadores y acciones fundamentales disponibles en Azure Logic Apps. Algunos de los conectores que encajan en esta categoría le permiten programar la ejecución del flujo de trabajo, llamar a otras aplicaciones lógicas o servicios de aplicaciones de Azure, realizar llamadas HTTP y HTTPS a puntos finales, procesar mensajes en lotes o hacer que su aplicación lógica se pueda llamar desde otros servicios. .
- **Administrado** Microsoft desarrolla, implementa y mantiene estos conectores. Utilice estos conectores para acceder a servicios en la nube como Office 365, Azure Blob Storage, SharePoint y muchos otros.

- **Local** Utilice este tipo de conector cuando necesite que su aplicación Azure Logic funcione con los sistemas implementados en su infraestructura local. Con estos conectores, puede acceder a datos de sistemas de archivos, Oracle, MySQL, PostgreSQL, Microsoft SQL Server, IBM DB2, IBM Informix o bases de datos Teradata. Para que estos conectores funcionen correctamente, debe implementar una puerta de enlace de datos local.
- **Cuenta de integración** Le permite conectar su aplicación Azure Logic con socios comerciales de terceros para crear soluciones Business-to-Business (B2B). Las cuentas de integración solo están disponibles a través del paquete de integración empresarial (EIP) en Azure. Utiliza este tipo de conector para transformar los mensajes entre los diferentes sistemas B2B. Puede aplicar decodificación o codificación de archivos AS2, EDIFACT, X12 o Flat, transformaciones Liquid y XML o validaciones XML.
- **ISE** Estos son los conectores que debe utilizar cuando su aplicación Azure Logic debe ejecutarse en un entorno de servicio de integración (ISE). Este es un entorno dedicado donde ejecuta sus Azure Logic Apps. Hay conectores especiales diseñados para trabajar en un ISE. 245 Estos conectores están marcados con la etiqueta CORE si son conectores integrados que puede usar en un ISE o si son conectores administrados que puede usar en un ISE, verá la etiqueta ISE debajo del nombre del conector. Al utilizar un ISE, no se limita a los conectores ISE; también puede utilizar conectores normales en un entorno de servicio de integración.

Además, a la clasificación que revisamos en la lista anterior, los conectores se pueden clasificar como conectores estándar o empresariales. Esta clasificación es esencial porque afecta directamente los costos asociados con la ejecución del flujo de trabajo de la aplicación Azure Logic. Cuando ejecuta un flujo de trabajo, se le cobra cuando usa el flujo de trabajo, excepto cuando su flujo de trabajo se ejecuta dentro de un ISE.

Cuando ejecuta el flujo de trabajo de la aplicación Azure Logic en el entorno público, de múltiples inquilinos y global, solo paga por las acciones que ejecuta su flujo de trabajo. Una acción es cualquiera de los pasos que configuran su flujo de trabajo. Los disparadores, bucles, declaraciones condicionales o cualquiera de las acciones de control

cuentan para el cálculo de los costos de ejecución de su flujo de trabajo. Hay tres niveles de precios:

- **Básico** Este nivel incluye conectores integrados, activadores y acciones de flujo de trabajo de control.
- **Estándar** Esto incluye las acciones definidas en conectores administrados. Cualquier conector personalizado que cree también encaja en esta categoría.
- **Enterprise** Estos son conectores especializados para integrar aplicaciones B2B con su aplicación Azure Logic. Algunos ejemplos de conectores empresariales son SAP, IBM 3270 o IBM MQ.

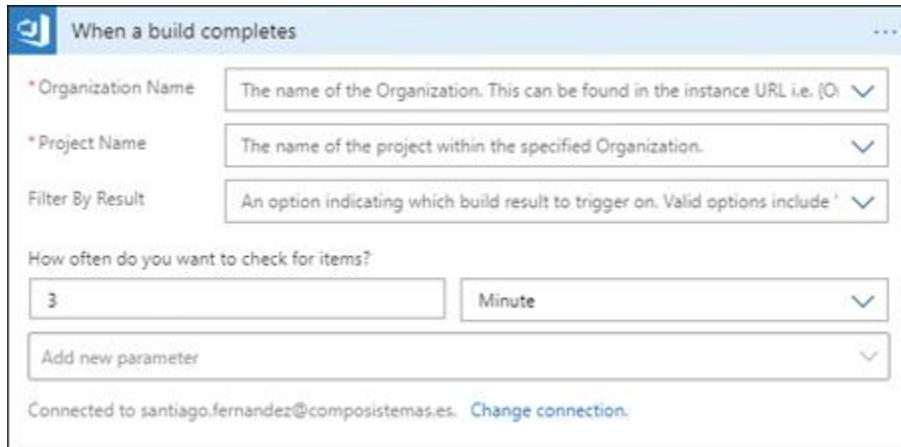
También es importante tener en cuenta que los conectores n locales pueden ser conectores estándar o empresariales. En general, cuando calcule los costos de su aplicación Azure Logic, debe revisar la lista de conectores estándar o empresarial para tener una idea precisa de qué tipo de conector está utilizando en su flujo de trabajo. Puede encontrar una lista completa de conectores revisando el artículo en <https://docs.microsoft.com/en-us/connectors/connector-reference/>.

Ahora que tiene un conocimiento básico de las diferentes partes de un flujo de trabajo de Azure Logic App, es hora de crear su propio flujo de trabajo. El siguiente procedimiento muestra cómo crear un flujo de trabajo de Azure Logic App que escribe un mensaje en la aplicación Microsoft Teams cuando se completa una nueva compilación en Azure DevOps. Para este procedimiento, necesita una cuenta de Azure DevOps con un proyecto configurado que pueda compilar. Si aún no tiene una cuenta de Azure DevOps, puede crear una siguiendo la guía de inicio rápido en <https://docs.microsoft.com/en-us/azure/devops/user-guide/sign-up-invite-teammates?view=azure-devops>. También necesita una suscripción a Microsoft Office 365 con acceso a la aplicación Microsoft Teams. Empiece por crear y configurar la aplicación Azure Logic:

1. Abra Azure Portal (<https://portal.azure.com>).
2. Haga clic en el botón Crear un recurso en la parte superior de Azure Portal.
3. En la hoja Nueva, en la lista de Azure Marketplace en el lado izquierdo de la hoja, haga clic en Integración.

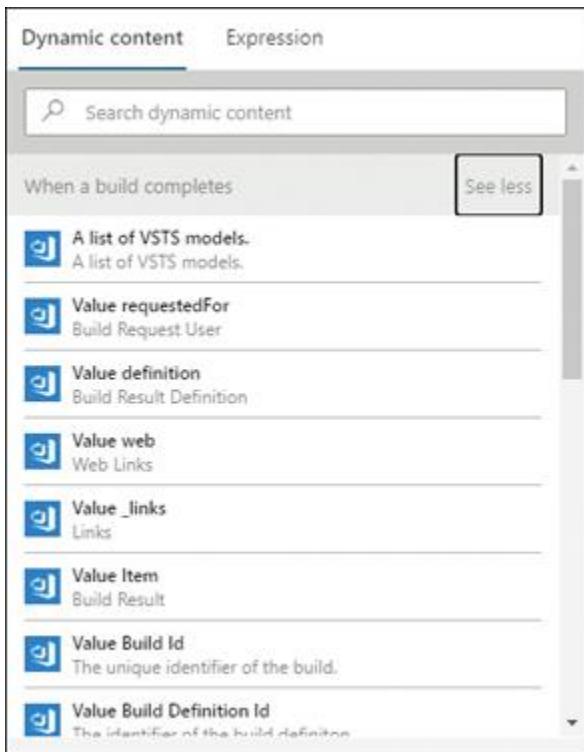
4. En la columna Destacados en el lado derecho de la hoja, haga clic en Aplicación lógica.
5. En la hoja Aplicación lógica, en el menú desplegable Grupo de recursos, seleccione el grupo de recursos donde desea crear su aplicación lógica Azure. Alternativamente, puede crear un nuevo grupo de recursos haciendo clic en el enlace Crear nuevo debajo del menú desplegable del grupo de recursos.
6. Escriba un nombre en el cuadro de texto Nombre de la aplicación lógica.
7. En el control Seleccionar la ubicación, asegúrese de que la opción Región esté seleccionada.
8. Seleccione una ubicación en el menú desplegable Ubicación.
9. Deje la opción Log Analytics establecida en el valor Desactivado.
10. Haga clic en el botón Revisar + Crear en la parte inferior de la hoja.
11. Haga clic en el botón Crear en la parte inferior de la hoja.
12. En la ventana de implementación de Microsoft.EmptyWorkflow, haga clic en el botón Ir a recurso. Este botón aparece una vez que la implementación de su nueva aplicación Azure Logic finaliza correctamente.
13. En la hoja Logic Apps, en Logic App Designer, haga clic en Blank Logic App en la sección Plantillas.
14. En el Diseñador de aplicaciones lógicas, en el cuadro de texto Buscar conectores y desencadenadores, escriba **Azure DevOps**.
15. Haga clic en el ícono de Azure DevOps en el panel Resultados.
16. En la pestaña Desencadenadores, seleccione el desencadenante denominado Cuando se completa una compilación.
17. Haga clic en el botón Iniciar sesión en el elemento Azure DevOps. En este punto, conecte su suscripción de Azure con su cuenta de Azure DevOps.
18. En el panel del activador Cuando se completa una construcción, que se muestra en la Figura 5-1, seleccione el nombre

de su organización en el menú desplegable Nombre de la organización.



**Figura 5-1** Configuración de un desencadenador de Azure Logic Apps

19. En el menú desplegable Nombre del proyecto, seleccione el nombre del proyecto que deseé.
20. Haga clic en el botón Nuevo paso debajo del panel de activación.
21. En el panel Elija una acción, escriba **Equipos** en el cuadro de texto Buscar conectores y acciones.
22. En la pestaña Acciones, haga clic en la acción Publicar un mensaje (V3).
23. En el panel de acción Publicar un mensaje (V3), seleccione un equipo del menú desplegable Equipo.
24. Seleccione General en el menú desplegable Canal.
25. En el área de texto del mensaje, escriba **el**.
26. En el cuadro de diálogo Contenido dinámico, que se muestra en la [Figura 5-2](#), en el lado derecho del área de texto del Mensaje, haga clic en el enlace Ver más. Este enlace muestra la lista de atributos dinámicos que puede agregar a su mensaje.



**Figura 5-2** Contenido dinámico de un activador de conector

27. Desplácese hacia abajo en la lista de atributos dinámicos y haga clic en el atributo dinámico Nombre de definición de creación de valor.
28. Escriba **compilación terminada con estado** en el área de mensajes junto al atributo dinámico.
29. Haga clic en el atributo dinámico Estado del valor.
30. Haga clic en el botón Guardar en la esquina superior izquierda de la hoja Diseñador de Azure Logic Apps.
31. Haga clic en el botón Ejecutar en la esquina superior izquierda de la hoja Diseñador de aplicaciones lógicas de Azure.

En este punto, Azure Logic Apps comienza a escuchar el desencadenador configurado en Azure DevOps. Cuando finaliza una nueva compilación en Azure DevOps, Azure Logic Apps envía un mensaje al canal General en el equipo configurado en su cuenta de Microsoft Teams. Ahora, creará una canalización en Azure DevOps para crear un proyecto de ejemplo. Una vez que finalice la compilación, recibirá un nuevo mensaje en el canal de Microsoft Teams:

1. Navegue hasta el siguiente repositorio de ejemplo de GitHub <https://github.com/MicrosoftDocs/pipelines-dotnet-core> e inicie sesión en su cuenta. Si no tiene una cuenta de GitHub, puede crear una nueva gratis en <https://github.com/join>.
2. En la esquina superior derecha de la página del proyecto, haga clic en el botón Bifurcación.
3. Abra su cuenta de Azure DevOps (<https://dev.azure.com>). Si ve la página genérica de Azure DevOps que describe el servicio en lugar de su cuenta de Azure DevOps, haga clic en el vínculo Iniciar sesión en Azure DevOps debajo del botón Iniciar gratis.
4. En la lista de proyectos de su cuenta de Azure DevOps, haga clic en el nombre del proyecto que configuró en el paso 19 del procedimiento anterior.
5. Haga clic en el elemento Pipelines en la barra de navegación en el lado izquierdo de la página del proyecto.
6. Haga clic en el botón Crear canalización.
7. En la ventana New Pipeline, haga clic en GitHub.
8. Inicie sesión en su cuenta de GitHub.
9. En la ventana Autorizar Azure Pipelines (OAuth), haga clic en el botón Autorizar Azure Pipelines en la parte inferior de la página.
10. En su cuenta de Azure DevOps, en la página Seleccionar un repositorio, haga clic en el proyecto pipelines-dotnet-core.
11. En su cuenta de Azure DevOps, en la página Configure Your Pipeline, haga clic en ASP.NET Core.
12. En la página Revise su canalización YAML, revise los detalles de su canalización.
13. Haga clic en el botón Ejecutar en la esquina superior derecha de la página.
14. Una vez que se ha creado la canalización, aparece una página con los detalles de la ejecución de su canalización. Si todo funciona correctamente, debería ver un trabajo con el estado de éxito en la lista de Trabajos en el panel Trabajo.

En este punto, el agente de Azure DevOps está compilando el proyecto de muestra. Una vez que finalice la compilación, debería recibir un nuevo

mensaje en su canal de Microsoft Teams, como se muestra en la Figura 5-3.



**Figura 5-3** Un mensaje en Microsoft Teams de Azure DevOps

#### *¿Necesita más revisión? Precios de la aplicación Azure Logic*

Calcular los costos asociados con un flujo de trabajo de Azure Logic App puede resultar complicado. Cada una de las iteraciones de un bucle es una o más ejecuciones de acciones. Los diferentes tipos de conectores se cobran de manera diferente, dependiendo de si son Básico, Estándar o Empresarial. Un flujo de trabajo ejecutado en un entorno de servicio de integración tiene su propio precio. Puede leer un artículo con los detalles sobre cómo Azure calcula cuánto tiene que pagar por la ejecución de su flujo de trabajo en <https://docs.microsoft.com/en-us/azure/logic-apps/logic-apps-pricing> .



#### *Sugerencia para el examen*

Cuando trabaja con entornos de servicio de integración (ISE), tiene conectores específicos que se ejecutan dentro del ISE. Estos conectores ISE especiales están marcados con una etiqueta. Aunque los entornos ISE 249 utilizan recursos dedicados para su entorno, aún puede utilizar conectores globales, públicos y multiusuario, como los conectores de Office 365 o Dropbox.

#### *Cree un conector personalizado para Logic Apps*

Microsoft proporciona más de 200 conectores integrados que puede usar en su flujo de trabajo de Azure Logic Apps. A pesar de esta cantidad de conectores, hay ocasiones en las que necesita algunas características específicas que no proporcionan los conectores integrados o administrados, o desea crear un conector para la aplicación de su empresa.

Puede crear conectores personalizados para Microsoft PowerAutomate (anteriormente conocido como Flow), PowerApps y Azure Logic Apps. Aunque no puede compartir conectores de Azure Logic Apps con conectores de Microsoft PowerAutomate y PowerApps, el principio para crear conectores personalizados es el mismo para las tres plataformas. Un conector personalizado es básicamente un contenedor para una API REST o SOAP. Este contenedor permite que Azure Logic Apps interactúe con la

API de la aplicación. La aplicación que desea incluir en el conector personalizado puede ser una aplicación pública, como Amazon Web Services, Google Calendar o la API de su aplicación publicada en Internet. Con la puerta de enlace de datos local, también puede conectar el conector personalizado con una aplicación local implementada en su centro de datos. Cada conector personalizado tiene el siguiente ciclo de vida:

1. **Cree su API** Puede envolver cualquier API REST o SOAP en un conector personalizado. Si está creando su API, debería considerar el uso de Azure Functions, Azure Web Apps o Azure API Apps.
2. **Asegure su API** Necesita autenticar el acceso a su API. Si está implementando su aplicación mediante Azure Functions, Azure Web Apps o Azure API Apps, puede habilitar la autenticación de Azure Active Directory en el portal de Azure para su aplicación. También puede aplicar la autenticación directamente en el código de su API. Puede utilizar cualquiera de los siguientes mecanismos de autenticación:
  1. OAuth 2.0 genérico
  2. OAuth 2.0 para servicios específicos, como Azure Active Directory, Dropbox, GitHub o SalesForce
  3. Autenticación básica
  4. Clave API
3. **Describe la API y define el conector personalizado** Debes proporcionar una descripción de los diferentes puntos finales que tiene tu API. Azure Logic Apps admite dos formatos de documentos legibles por máquina y independientes del idioma que puede usar para documentar esta descripción: OpenAPI (anteriormente conocido como Swagger) o colecciones Postman. Puede crear un conector personalizado a partir de la documentación de la colección de OpenAPI o Postman.
4. **Usar el conector en Azure Logic Apps** Una vez que haya creado el conector personalizado, puede usarlo como un conector integrado administrado regular en su flujo de trabajo. Necesita crear una conexión a su API usando su conector personalizado. Luego, puede usar los activadores y acciones que configuró en su conector personalizado.

**5. Comparta su conector** Una vez que haya creado su conector personalizado, puede compartirlo con otros usuarios de su organización. Este paso es opcional.

**6. Certifique su conector** Si desea compartir su conector personalizado con otros usuarios fuera de su organización, debe enviar el conector personalizado a Microsoft. Luego, Microsoft puede revisar su conector personalizado para asegurarse de que funcione correctamente. Una vez que se revisa y valida el conector, Microsoft lo certifica y puede compartirlo con usuarios fuera de su organización.

Ahora que ha revisado el ciclo de vida de un conector personalizado de la aplicación Azure Logic, va a crear un conector personalizado para conectarse con una API. Para este ejemplo, creará una aplicación Web API 2 simple que simula un sistema de administración de libros. Aunque este ejemplo es bastante simple, cubre algunos puntos clave que debe tener en cuenta al crear un conector personalizado de Azure Logic App. La API que va a implementar en este ejemplo no es apropiada para entornos de producción porque no tiene en cuenta aspectos importantes como el rendimiento o la seguridad. Siga los siguientes pasos para crear la API que utilizará para su conector personalizado:

1. Abra Visual Studio 2019 en su computadora.
2. En la ventana de inicio, haga clic en Crear un nuevo proyecto en la columna Comenzar en el lado derecho de la ventana.
3. En la ventana Crear un proyecto nuevo, en el menú desplegable Todos los idiomas, seleccione C #.
4. En el cuadro de texto Buscar plantillas, escriba **asp.net**.
5. En la lista de resultados, haga clic en Aplicación web ASP.NET (.NET Framework).
6. Haga clic en el botón Siguiente en la esquina inferior derecha de la ventana.
7. En la ventana Configure su nuevo proyecto, escriba un nombre de proyecto, una ubicación y un nombre de solución para su proyecto.
8. Haga clic en el botón Crear en la esquina inferior derecha de la ventana.

9. En la ventana Crear una nueva aplicación web ASP.NET, seleccione la plantilla de API web en la lista de plantillas en el medio del lado izquierdo de la ventana.
10. En el lado derecho de la ventana Crear una nueva aplicación web ASP.NET, en la sección Autenticación, asegúrese de que Autenticación esté configurada en Sin autenticación.
11. Haga clic en el botón Crear en la esquina inferior derecha de la ventana.
12. En la ventana de Visual Studio, haga clic en Herramientas> Administrador de paquetes NuGet> Administrar paquetes NuGet para la solución.
13. En la pestaña Administrador de paquetes NuGet, haga clic en Examinar.
14. Escribe **swashbuckle** y presiona Enter.
15. Haga clic en el paquete Swashbuckle.
16. En el lado derecho de la pestaña NuGet Manager, haga clic en la casilla de verificación junto a su proyecto.
17. Haga clic en el botón Instalar.
18. En la ventana Vista previa de cambios, haga clic en Aceptar.
19. En la ventana Aceptación de licencia, haga clic en el botón Acepto.
20. Repita los pasos 13 a 19 e instale el paquete TRex NuGet.
21. En la ventana del Explorador de soluciones, haga clic con el botón derecho en la carpeta Modelos.
22. En el menú contextual, haga clic en Agregar> Nuevo elemento.
23. En la ventana Agregar nuevo elemento, seleccione Clase de la lista de elementos nuevos.
24. En el cuadro de texto nombre en la parte inferior de la ventana, escriba **Book.cs**.
25. Haga clic en el botón Agregar en la esquina inferior derecha de la ventana.
26. Reemplace el contenido del archivo Book.cs con el contenido del Listado 5-1.

### Listado 5-1 Book.cs

Haga clic aquí para ver la imagen del código

---

```
// C # .NET

using System;
using TRex.Metadata;

namespace <replace_with_your_project_name>
{
    public class libro
    {
        public libro()
        {
            this.Id = Guid.NewGuid();
        }

        [Metadata("ID de devolución de llamada",
        Visibility = VisibilityType.Internal)]
        public int ID { get; }

        [Metadata("Título", "El título del libro")]
        public string Título { get; set; }

        [Metadata("Autor", "El autor del libro")]
        public string Author { get; set; }
    }
}
```

```
    }  
}
```

27. Repita los pasos del 21 al 25 y cree la clase **Callback.cs** .
28. Reemplace el contenido del archivo Callback.cs con el contenido del Listado 5-2 .

**Listado 5-2** *Callback.cs*

Haga clic aquí para ver la imagen del código

---

```
// C # .NET  
  
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Threading.Tasks;  
using System.Net.Http;  
using System.Net.Http.Headers;  
using Newtonsoft.Json;  
  
  
namespace <replace_with_your_project_name>  
.Models  
  
{  
  
    // devolución de llamada de clase pública  
    public async Task<string> GetGuidId()  
    {  
        this.Id = Guid.NewGuid();  
        return this.Id.ToString();  
    }  
}
```

```
    [Metadatos ("ID de devolución de llamada",
    Visibility = VisibilityType.Internal) ]

        ID de guía pública {get; }

    [CallbackUrl]

    [Metadatos ("URL de devolución de llamada",
    Visibility = VisibilityType.Internal) ]

        public Uri Uri {conjunto; obtener; }

    public HttpResponseMessage InvokeAsync <TOutput>
(TOutput triggerOutput)

{

    HttpClient httpClient = new HttpClient ();

    httpClient.DefaultRequestHeaders.Accept.Clear ();

    httpClient.DefaultRequestHeaders.Accept.Add

    (

        new MediaTypeWithQualityHeaderValue
("aplicación / json"));

    return Task.Run (async () => await
httpClient.PostAsJsonAsync (Uri,
JObject.FromObject (triggerOutput))).Result;

}

}
```

}

29. En la ventana del Explorador de soluciones, haga clic con el botón derecho en el nombre de su proyecto. En el menú contextual, haga clic en Agregar> Nueva carpeta.

30. Escriba **Helpers** como el nombre de la nueva carpeta.

31. Repita los pasos del 21 al 25 y cree dos clases adicionales en la carpeta Ayudantes. Las dos clases deben ser **BooksSingleton.cs** y **CallbacksSingleton.cs** .

32. Reemplace el contenido del archivo BooksSingleton.cs con el contenido del [Listado 5-3](#) .

**Listado 5-3** *BooksSingleton.cs*

[Haga clic aquí para ver la imagen del código](#)

---

```
// c # .NET

usando <replace_with_your_project_name>. Modelos;
usando el sistema;
usando System.Collections.Generic;
utilizando System.Linq;

espacio de nombres <replace_with_your_project_name>
.Helpers

{
    Libros públicos sellados de clase
    {
        BooksSingleton de solo lectura estático privado
        m_instance = null;
```

```
Lista de <Libro> _libros estáticos privados de
solo lectura;
```

```
Instancia pública de BooksSingleton estática
```

```
{  
    obtener
```

```
{  
    return m_instance;  
}
```

```
}
```

```
Libros estáticos Singleton ()
```

```
{  
    m_instance = new BooksSingleton ();  
    _libros = nueva Lista <Libro> ();  
}
```

```
Libros privadosSingleton ()
```

```
{  
}
```

```
AddBook public void (libro libro)
```

```
{  
    si (libro! = nulo)  
    {
```

```
        _books.Add (libro);

    }

}

public void ModifyBook (libro libro)

{

    var bookToModify = _books.SingleOrDefault (b
=> b.Id.Equals (book.Id));

    si (bookToModify! = null)

    {

        bookToModify.Author = libro.Author;

        bookToModify.Title = libro.Title;

    }

}

public IEnumerable <Book> GetBooks ()

{

    return _books.ToArray ();

}

public bool DeleteBookById (id de cadena)

{

    bool eliminado = falso;

    Guid guidToRemove = Guid.Parse (id);
```

```

        var booktToRemove = _books.SingleOrDefault
(b => b.Id.Equals (guidToRemove)) ;

        si (booktToRemove! = null)

        {

            _books.Remove (booktToRemove) ;

            eliminado = verdadero;

        }

        volver eliminado;

    }

Public Book GetBookById (id de cadena)

{

    Guid guid = Guid.Parse (id);

    return _books.SingleOrDefault (b =>
b.Id.Equals (guid));

}

}

```

33. Reemplace el contenido del archivo CallbacksSingleton.cs con el contenido del [Listado 5-4](#).

**Listado 5-4** *devoluciones de llamada*

[Haga clic aquí para ver la imagen del código](#)

---

// C # .NET

usando <replace\_with\_your\_project\_name>.Modelos;

```
usando el sistema;
usando System.Collections.Generic;
utilizando System.Linq;
usando System.Web;

espacio de nombres <replace_with_your_project_name>.
Ayudantes

{

    devoluciones de llamada de clase pública

    {

        Private static readonly CallbacksSingleton
m_instance = null;

        Private static readonly List <Callback>
_callbacks;

        Callbacks estáticos públicos Instancia Singleton

        {

            obtener

            {

                return m_instance;

            }

        }

    }

    devoluciones de llamada estáticassSingleton ()

    {
```

```
    m_instance = new CallbacksSingleton () ;

    _callbacks = nueva lista <Callback> () ;

}

devoluciones de llamada privadas

{

}

public void AddCallback (devolución de llamada)

{

    si (devolución de llamada! = nulo)

    {

        // evitar duplicados

        Callback callbackToBeAdded =
_callbacks.FirstOrDefault (c => Uri.

            Compare (c.Uri, callback.Uri,
UriComponents.AbsoluteUri, UriFormat.

            Unescaped,
 StringComparison.CurrentCultureIgnoreCase) == 0);

        si (callbackToBeAdded == null)

            _callbacks.Add (devolución de
llamada);

    }

}
```

```
    public void ModifyCallback (devolución de
        llamada)

    {

        var callbackToModify =
        _callbacks.SingleOrDefault (b =>
            b.Id.Equals (callback.Id));

        si (callbackToModify! = null)

        {

        }

    }

    public IEnumerable <Callback> GetCallbacks ()

    {

        return _callbacks;

    }

    public bool DeleteCallbackById (id de cadena)

    {

        bool eliminado = falso;

        Guid guidToRemove = Guid.Parse (id);

        var callbackToRemove =
        _callbacks.SingleOrDefault (b =>
            b.Id.Equals (guidToRemove));

        si (callbackToRemove! = null)
```

```

    {

        _callbacks.Remove (callbackToRemove);

        eliminado = verdadero;

    }

    volver eliminado;
}

```

```

    GetCallbackById de devolución de llamada pública
(id de cadena)

{

    Guid guid = Guid.Parse (id);

    return _callbacks.SingleOrDefault (b =>
b.Id.Equals (guid));
}

}

```

34. En la ventana del Explorador de soluciones, expanda las carpetas Controllers y elimine el archivo ValuesController.cs.
35. Haga clic con el botón derecho en la carpeta Controladores y luego haga clic en Agregar> Controlador.
36. En Add New Scaffolded Item, seleccione Web API 2 Controller - Empty.
37. Haga clic en el botón Agregar en la esquina inferior derecha de la ventana.
38. En la ventana Agregar controlador, escriba **BooksController** en el cuadro de texto Nombre del controlador.

39. Haga clic en el botón Agregar.

40. Reemplace el contenido del archivo BooksController.cs con el contenido del Listado 5-5.

**Listado 5-5** BooksController.cs

Haga clic aquí para ver la imagen del código

---

```
// c # .NET

usando <replace_with_your_project_name>. Modelos;
usando <replace_with_your_project_name>. Ayudantes;
usando Swashbuckle.Swagger.Annotations;
usando el sistema;
usando System.Collections.Generic;
utilizando System.Linq;
usando System.Net;
usando System.Net.Http;
usando System.Threading.Tasks;
usando System.Web.Http;
utilizando TRex.Metadata;

espacio de nombres <replace_with_your_project_name>.
Controladores

{

    BooksController de clase pública: ApiController
    {

        privado de solo lectura BooksSingleton _books =
        BooksSingleton.Instance;
```

```

    Private static readonly CallbacksSingleton
    _callbacks = CallbacksSingleton.

    Ejemplo;

    // Suscríbete a libros recién creados

    [HttpPost, Ruta ("libros / suscripción")]

        [Metadatos ("Nuevo libro creado", "Se activa
cada vez que se agrega un nuevo libro al
list.", VisibilityType.Important)]

        [Trigger (TriggerType.Subscription, typeof
(Libro), "Libro")]

        [SwaggerResponseRemoveDefaults]

        [SwaggerResponse (HttpStatusCode.Created,
"Suscripción creada")]

        [SwaggerResponse (HttpStatusCode.BadRequest,
"Suscripción no válida
configuración")]

    public IHttpActionResult Subscribe (devolución
de llamada)

    {

        _callbacks.AddCallback (devolución de
llamada);

        return CreatedAtRoute (nombre de (Cancelar
suscripción), nuevo {subscriptionId = callback.
Id}, string.Empty);

    }

```

```

        [HttpDelete, Route ("libros / subscribe /
{callbackID}", Name = nameof

        (Cancelar suscripción))]

        [Metadatos ("Cancelar suscripción", Visibility =
VisibilityType.Internal)]

        [SwaggerResponse (HttpStatusCode.OK)]
```

```

    public IHttpActionResult Unsubscribe (string
callbackID)

{
    _callbacks.DeleteCallbackById (callbackID);

    volver Ok ();
}
```

```

        [HttpGet, Route ("libros / suscripciones")]

        [Metadatos ("Obtener suscripciones", "Obtener
todas las suscripciones actuales")]

        [SwaggerResponse (HttpStatusCode.OK, "Una matriz
de suscripciones",
typeof (Matriz))]
```

```

public IEnumerable <Callback> GetCallbacks ()

{
    return _callbacks.GetCallbacks ();
}
```

```

// OBTENER api / books

[HttpGet, Route ("libros")]

[Metadatos ("Obtener libros", "Obtener todos los
objetos de libros almacenados en la aplicación")]

[SwaggerResponse (HttpStatusCode.OK, "Una matriz
de libros", typeof (Array))]

public IEnumerable <Book> Get ()

{

    return _books.GetBooks ();
}

// OBTENER api / books / 5

[HttpGet, Route ("libros / {id}", Name =
"GetBook")]

[Metadatos ("Obtener un solo libro", "Obtener un
solo objeto de libro por su ID. Puede usar
cualquier cadena válida de GUID")]

[SwaggerResponse (HttpStatusCode.OK, "Un objeto
que representa un libro",

tipo de libro))]

Obtener libro público (ID de cadena)

{

    return _books.GetBookById (id);
}

// POST api / libros

```

```

    [HttpPost, Ruta ("libros")]
        [Metadatos ("Aregar un libro nuevo", "Aregar
un objeto de libro nuevo al sistema. Un valor
el objeto está compuesto por un título y un
autor. ")]
    [SwaggerResponse (HttpStatusCode.Created)]
    Publicación pública de IHttpActionResult
    ([FromBody] Libro libro)

    {
        _books.AddBook (libro);

        foreach (devolución de llamada var en
        _callbacks.GetCallbacks ())
        {
            callback.InvokeAsync (libro);
        }
    }

    return CreatedAtRoute ("GetBook", nuevo {id
    = book.Id}, libro);
}

// PUT api / books / 5

[HttpPut, Ruta ("libros / {id}")]
    [Metadatos ("Modificar un objeto de libro
existente", "Modificar un libro existente. Necesita
para proporcionar los nuevos valores para el
título o autor del libro. Buscas el

```

```

        objeto de libro usando su id")]
public void Put ([FromBody] Libro libro)
{
    _books.ModifyBook (libro);
}

// ELIMINAR api / books / 5
[Metadata ("Eliminar un objeto de libro",
"Eliminar un objeto de libro por su id.")]
[HttpDelete, Route ("libros / {id}")]
public void Delete (id de cadena)
{
    _books.DeleteBookById (id);
}
}

```

41. Abra el archivo SwaggerConfig.cs. Puede encontrar este archivo en la carpeta App\_Start.
42. En el archivo SwaggerConfig.cs, elimine el comentario de la línea `c.PrettyPrint();`
43. Agregue la línea `c.ReleaseTheTREx();` dentro del `EnableSwagger` método, justo después de esta línea:

[Haga clic aquí para ver la imagen del código](#)

```

//c.CustomProvider((defaultProvider) => nuevo
CachingSwaggerProvider
(proveedor predeterminado));

```

44. Descomente esta línea:

Haga clic aquí para ver la imagen del código

```
c.DocExpansion (DocExpansion.List);
```

45. Agregue la línea usando TRex.Metadata al principio del archivo.

En este punto, puede probar su API. Utilice los siguientes pasos para probar su API:

1. En su ventana de Visual Studio 2019, asegúrese de que su proyecto de API esté abierto. Luego presione F5.
2. Una vez que su aplicación web esté cargada en su navegador web, agregue el URI */swagger* a la URL en su navegador web. La dirección final debería ser similar a *https://localhost: 44398 / swagger*.
3. En la lista de métodos de la API, que se muestra en la Figura 5-4, haga clic en el enlace *POST / books*. Esto amplía las opciones para el punto final *POST / books*.

Books		
<b>POST</b>	/books/subscribe	New book created
<b>DELETE</b>	/books/subscribe/{callbackID}	Unsubscribe
<b>GET</b>	/books/subscriptions	Get subscriptions
<b>GET</b>	/books	Get books
<b>POST</b>	/books	Add a new book
<b>DELETE</b>	/books/{id}	Delete a book object
<b>GET</b>	/books/{id}	Get single book
<b>PUT</b>	/books/{id}	Modify an existing book object

**Figura 5-4** Lista de puntos finales de una API web

4. En el área verde del punto final de *POST / libros*, en la sección Parámetros, escriba un objeto JSON en el área de texto del valor. Puede encontrar un ejemplo de la estructura necesaria en la columna Tipo de datos al final de la misma línea.
5. Haga clic en el botón ¡Pruébelo! en la esquina inferior izquierda del área verde del punto final.

6. En la sección Cuerpo de la respuesta, debe obtener la representación JSON de su libro recién creado con una ID válida asignada. También puede comprobar que el libro se ha creado correctamente haciendo clic en el botón ¡Pruébelo! en el área azul del punto final *GET / books*.

Ahora que ha probado que su API web funciona correctamente, debe publicar en un servicio de aplicaciones de Azure. Una alternativa a la implementación de la API web en un servicio de aplicaciones de Azure podría ser el uso de una puerta de enlace de datos local para conectarse a la implementación local de nuestra API web. Utilice los siguientes pasos para implementar la API web en un servicio de aplicaciones de Azure:

1. En su ventana de Visual Studio 2019, asegúrese de haber abierto la API web.
2. En el lado derecho de la ventana de Visual Studio, en la ventana del Explorador de soluciones, haga clic con el botón derecho en el nombre del proyecto.
3. En el menú contextual, haga clic en Publicar. Esto abre la ventana Publicar.
4. En la ventana Publicar, asegúrese de que Azure esté seleccionado en la lista de Destinos en el lado derecho de la ventana.
5. Haga clic en Siguiente.
6. Seleccione Azure App Service (Windows) en la sección Destino específico.
7. Haga clic en Siguiente.
8. En la sección App Service, haga clic en Create A New Azure App Service en la parte inferior de la ventana. Si ya tiene un plan de Azure App Service que desea usar para hospedar el App Service, puede seleccionarlo en el control de árbol en esta misma ventana.
9. En la ventana de App Service (Windows), deje todas las opciones como están y haga clic en el botón Crear. Recuerde eliminar el grupo de recursos y todos sus recursos asociados después de terminar este ejemplo porque estos recursos consumen crédito de su suscripción de Azure.
10. En el control de árbol, expanda el grupo de recursos recién creado y haga clic en Azure App Service recién creado.

11. Haga clic en el botón Finalizar.
12. Haga clic en el botón Publicar.
13. Una vez finalizado el proceso de publicación, Visual Studio abre su navegador web predeterminado con la URL del App Service recién implementado. Esta URL tendrá la estructura `https://<your_app_service_name>.azurewebsites.net`.
14. Asegúrese de que su API esté funcionando correctamente repitiendo los pasos de prueba mostrados anteriormente en esta sección. Esta vez necesitas usar la URL `https://<your_app_service_name>.azurewebsites.net/swagger`.

En este punto, está listo para crear su conector personalizado de la aplicación Azure Logic. Antes de continuar con la creación de su conector personalizado para su API, profundicemos un poco en el código de la API para comprender el conector personalizado de la aplicación Azure Logic que va a crear. La API le permite crear, eliminar, modificar y consultar un libro en la lista de libros que la aplicación puede almacenar. También puede obtener la lista completa de libros almacenados en la API. Como se revisó anteriormente en esta sección, un conector puede tener un disparador que representa un evento que inicia la secuencia de acciones en nuestra carga de trabajo. En nuestra API de prueba, el disparador representa el evento de agregar un nuevo libro a la lista de libros. Puede ver cómo *activamos* el activador en el siguiente fragmento de código extraído del método Post en *BooksController*:

[Haga clic aquí para ver la imagen del código](#)

```
_books.AddBook (libro);

foreach (devolución de llamada var en
_callbacks.GetCallbacks ())

{
    callback.InvokeAsync (libro);
}
```

Como puede ver en el código anterior, después de agregar un nuevo libro a la lista de libros, llamamos al `InvokeAsync` método del objeto de devolución de llamada. La razón para hacer esto es que Azure Logic Apps

funciona con dos tipos de desencadenadores: desencadenadores de extracción y empuje. En este ejemplo, decidimos utilizar un disparador de empuje. Esto significa que la API notifica a la aplicación Azure Logic cuando ocurre el evento. La forma en que la API puede notificar a la aplicación Azure Logic es mediante el patrón de webhook. Por este motivo, debemos proporcionar dos puntos de conexión adicionales a la API, uno para permitir que la aplicación Azure Logic se suscriba a la API y otro para eliminar la suscripción. En el ejemplo, estos puntos finales son `POST /books/subscribe` y `DELETE /books/subscribe/{callbackID}` (puede ver estos puntos finales en la [Figura 5-3](#) anteriormente en esta sección).

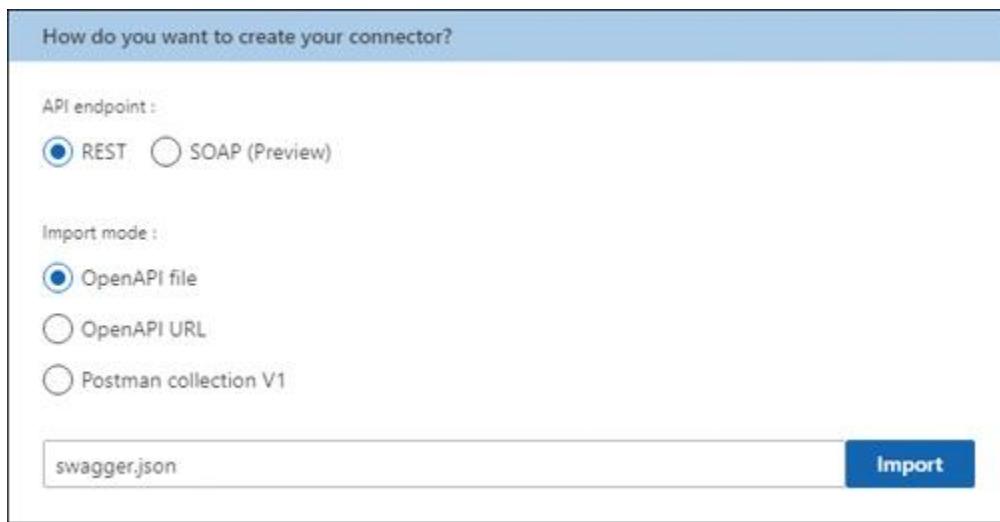
El flujo de trabajo se suscribe a la API cuando el flujo de trabajo cambia de alguna manera; por ejemplo, cuando lo ejecuta por primera vez, cambie los parámetros de entrada al activador o renueve las credenciales para conectarse a su API. Durante el proceso de suscripción, su flujo de trabajo proporciona la URL de devolución de llamada que su API necesita para enviar la información necesaria para el flujo de trabajo. El proceso de cancelación de suscripción ocurre automáticamente cuando el activador, el flujo de trabajo o la suscripción se eliminan o desactivan.

Swagger proporciona toda esta información sobre qué endpoint Azure Logic App debería usar para suscribirse a los eventos en su API, la estructura de los datos que usa su API o los endpoints disponibles para el conector. Podríamos crear manualmente el documento Swagger que describe nuestra API, pero eso sería una tarea que consumiría mucho tiempo y sería propensa a errores. Para facilitar la creación y administración del documento Swagger que describe la API, usamos el paquete Swashbuckle (<https://github.com/domaindrivendev/Swashbuckle>) y biblioteca de metadatos TRex (<https://github.com/nihuae/TRex>). El paquete Swashbuckle agrega Swagger a la API, proporcionando un generador Swagger y una interfaz de usuario para navegar y probar la API. El paquete TRex Metadata Library amplía las capacidades de Swashbuckle, por lo que el documento Swagger generado por Swashbuckle está listo para ser consumido por Logic App Designer.

Ahora que comprende mejor las partes clave de la API que permiten la interacción con Azure Logic Apps, es hora de crear un conector personalizado para su API:

1. En su navegador web, descargue la definición Swagger de su API desde `https://<your_app_service_name>.azurewebsites.net/swagger/docs/v1`.
2. En su navegador web, haga clic con el botón derecho en la definición Swagger de la API, haga clic en Guardar como en el menú contextual y guarde la página en su escritorio como un archivo con el nombre **swagger.json**. Necesita este archivo para un paso posterior.
3. Abra Azure Portal (<https://portal.azure.com>).
4. Haga clic en el botón Crear un recurso en el área superior de Azure Portal.
5. Escriba *aplicaciones lógicas* en el cuadro de texto Buscar en el mercado.
6. Haga clic en Conector personalizado de Logic Apps en la lista de resultados.
7. Haga clic en el botón Crear.
8. En el panel Crear conector personalizado de aplicaciones lógicas, seleccione un grupo de recursos existente en el menú desplegable Grupo de recursos donde desea almacenar su conector personalizado. Alternativamente, puede crear un nuevo grupo de recursos haciendo clic en el enlace Crear nuevo debajo del menú desplegable Grupo de recursos.
9. Escriba un nombre para su conector personalizado en el cuadro de texto Nombre del conector personalizado.
10. En el control Seleccionar la ubicación, asegúrese de que la opción Región esté seleccionada.
11. Seleccione una ubicación en el menú desplegable Ubicación.
12. Haga clic en el botón Revisar + Crear en la parte inferior del panel.
13. Haga clic en el botón Crear.
14. Escriba el nombre de su conector personalizado recién creado en el cuadro de texto de búsqueda en la parte superior de Azure Portal.

15. Haga clic en el nombre de su conector personalizado de la aplicación Azure Logic en los resultados.
16. Haga clic en el botón Editar en la hoja Descripción general del conector personalizado.
17. En la hoja Editar conector personalizado de aplicaciones lógicas, haga clic en el botón Importar en la sección ¿Cómo desea crear su conector? panel, que se muestra en la [Figura 5-5](#).



**Figura 5-5** Importación de la definición de OpenAPI de una API REST

18. Elija el archivo JSON que descargó en el paso 2.
19. En la sección Información general, revise la información de esta sección. No es necesario realizar ningún cambio aquí.

#### **Nota URL base**

En este ejemplo, la definición de punto final ya contiene la URL base correcta en la definición de punto final. Si cambia la propiedad de URL base predeterminada en la sección Información general de su conector personalizado de Azure Logic Apps, recibirá un error 404 cada vez que intente usar su conector personalizado en un flujo de trabajo de Azure Logic Apps>.

20. Haga clic en el enlace Seguridad en la parte superior de la página. También puede encontrar un enlace de seguridad en la esquina inferior izquierda de la página. Puede utilizar cualquiera de estos enlaces para navegar a la sección Seguridad.
21. Revise las opciones en la página de Seguridad. Para este conector personalizado, no está utilizando ninguna

autenticación. Asegúrese de que la opción Sin autenticación esté seleccionada.

22. Haga clic en el enlace Definición en la parte superior de la página. También puede encontrar un enlace de Definición en la esquina inferior izquierda de la página. Puede utilizar cualquiera de estos enlaces para navegar a la sección Definición.

23. Revise la configuración en la página Definición. En esta página, puede encontrar todos los puntos finales que se han definido en el archivo JSON Swagger que importó en el paso 18. Estos puntos finales se traducen en acciones o desencadenadores, según la definición en el archivo JSON. También puede agregar manualmente nuevas acciones y activadores según lo necesite utilizando esta página.

24. Desplácese hacia abajo en la página hasta que vea la sección Activadores.

25. Haga clic en el activador NewBookCreated. Debe haber un círculo rojo con un signo de exclamación al lado del nombre del disparador. Este icono indica que hay un problema con la definición del disparador.

26. En la ventana de definición de disparador NewBookCreated, desplácese hasta el final de la sección Solicitud. En la sección Cuerpo dentro de la sección Solicitud, debería ver el objeto Devolución de llamada con un círculo rojo con un signo de exclamación.

27. Haga clic en el objeto Devolución de llamada.

28. Haga clic en Editar en el menú contextual sobre el objeto de devolución de llamada.

29. En la sección Cuerpo del objeto de devolución de llamada, haga clic en el parámetro URL de devolución de llamada con el círculo rojo con un ícono de signo de exclamación.

30. Haga clic en Editar en el menú contextual sobre el parámetro URL de devolución de llamada.

31. En la definición del parámetro URL de devolución de llamada, en el campo ¿Es obligatorio? sección, seleccione Sí.

32. En la parte inferior de la página de definición de parámetros de URL de devolución de llamada, en la sección Validación,

asegúrese de que haya un ícono verde que indique que la definición es correcta.

33. Haga clic en el ícono Atrás en la esquina superior izquierda de la página de definición de parámetros de URL de devolución de llamada.

34. Haga clic en el ícono Atrás en la esquina superior izquierda de la página de definición del objeto de devolución de llamada.

35. Repita los pasos 27 a 34 para los objetos de devolución de llamada con el signo de exclamación dentro de un ícono de círculo rojo en la sección Referencias en el lado izquierdo de la página Definición.

36. Asegúrese de que no haya un signo de exclamación dentro de un ícono de círculo rojo en la página de Definición.

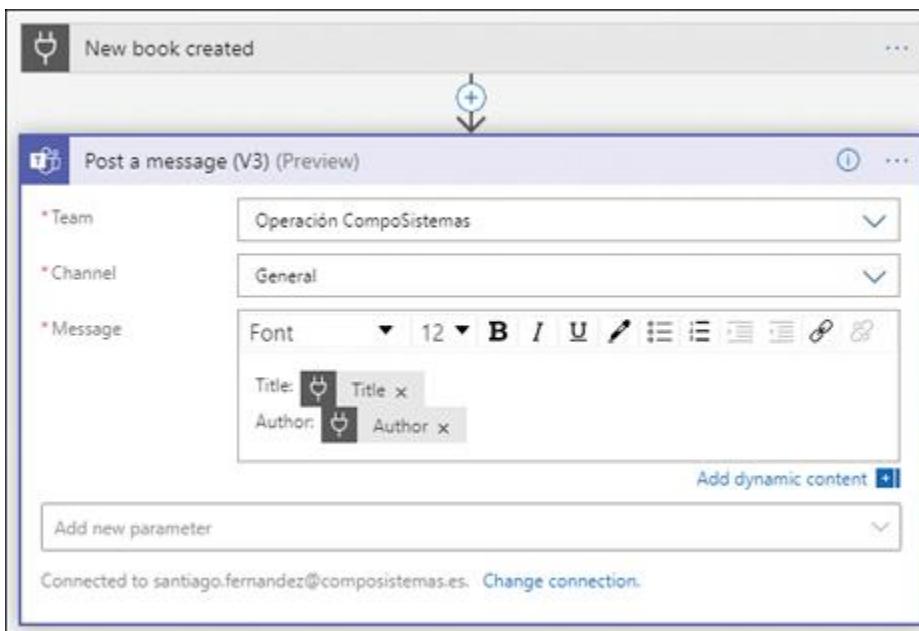
37. Haga clic en el enlace Actualizar conector en la esquina superior derecha de la hoja Editar conector personalizado de aplicaciones lógicas.

En este punto, ha creado correctamente su conector personalizado de Azure Logic Apps. En los siguientes pasos, creará un flujo de trabajo para probar su nuevo conector personalizado. Este flujo de trabajo usa el desencadenador definido en el conector personalizado de la aplicación Azure Logic. Obtiene la información del libro recién creado en la API y coloca la información en un canal de Microsoft Teams:

1. Abra Azure Portal (<https://portal.azure.com>).
2. Haga clic en Crear un recurso en el área superior de Azure Portal.
3. Escriba la **aplicación lógica** en el cuadro de texto Buscar en el mercado.
4. Haga clic en Aplicación lógica en la lista de resultados.
5. Haga clic en el botón Crear.
6. En el panel Aplicación lógica, seleccione un grupo de recursos existente en el menú desplegable del grupo de recursos, donde desea almacenar su aplicación lógica. Alternativamente, puede crear un nuevo grupo de recursos haciendo clic en el enlace Crear nuevo debajo del menú desplegable del grupo de recursos.

7. Escriba un nombre para su aplicación lógica en el cuadro de texto Nombre de la aplicación lógica.
8. En el control Seleccionar la ubicación, asegúrese de que la opción Región esté seleccionada.
9. Seleccione una ubicación en el menú desplegable Ubicación.
10. Haga clic en el botón Revisar + Crear en la parte inferior del panel.
11. Haga clic en el botón Crear.
12. Vaya a la aplicación Azure Logic recién creada.
13. En la hoja Diseñador de aplicaciones lógicas, elija la plantilla Aplicación lógica en blanco. Si no obtiene la hoja Logic Apps Designer tan pronto como abra su Azure Logic App, puede hacer clic en Logic App Designer en el menú de navegación en el lado izquierdo de su Azure Logic App.
14. En el Diseñador de aplicaciones lógicas, haga clic en la pestaña Personalizar.
15. Haga clic en su conector personalizado recién creado.
16. En la sección Activadores, haga clic en Nuevo libro creado. Este es un disparador simple que no requiere parámetros adicionales.
17. Haga clic en el botón Nuevo paso.
18. Escriba **Microsoft Teams** en el cuadro de texto Buscar conectores y acciones en el panel Elegir una acción.
19. Haga clic en el ícono de Microsoft Teams.
20. Haga clic en la acción Publicar un mensaje (V3) (vista previa).
21. Otorgue acceso a su cuenta de Microsoft Teams.
22. En el panel de acción Publicar un mensaje (V3) (vista previa), seleccione un equipo existente en el menú desplegable Equipo.
23. Seleccione el canal General en el menú desplegable Agregar ID de canal de equipos.
24. Haga clic dentro del cuadro de texto Mensaje.
25. En el área de texto del mensaje, escriba **Título:** seguido de un espacio.

26. En el cuadro de diálogo Contenido dinámico, haga clic en el enlace Ver más en la sección Nuevo libro creado.
27. En la sección Nuevo libro creado, haga clic en Título.
28. Escriba una nueva línea en el área de texto del mensaje.
29. En el área de texto del mensaje, escriba **Autor:** seguido de un espacio.
30. Repita los pasos 26 y 27 para la propiedad Author. Su área de mensajes debe ser similar a la de la Figura 5-6.



**Figura 5-6** Enviar un mensaje a Microsoft Teams

31. Haga clic en el botón Guardar en la esquina superior izquierda de la hoja Diseñador de aplicaciones lógicas.

#### *¿Necesita más revisión? Variables*

Puede declarar y utilizar variables en su flujo de trabajo. Estas variables pueden contener datos de sus conectores, valores fijos o el resultado de algunas operaciones que realiza dentro de su flujo de trabajo. Puede obtener más información sobre cómo usar variables en su flujo de trabajo revisando el artículo <https://docs.microsoft.com/en-us/azure/logic-apps/logic-apps-create-variables-store-values> .

En este punto, debería estar listo para probar su nuevo conector personalizado de Azure Logic Apps. Para probar este flujo de trabajo, puede utilizar el mismo procedimiento que utilizó para probar la API anteriormente en esta sección. Crea un libro nuevo en la API. El nuevo

libro debería aparecer en el canal de Microsoft Teams que configuró en el flujo de trabajo de Azure Logic Apps, como se muestra en la [Figura 5-7](#).



**Figura 5-7** Resultado de la ejecución del flujo de trabajo en Microsoft Teams



### *Sugerencia para el examen*

Puede crear conectores personalizados para Azure Logic Apps, Microsoft Flow y Microsoft PowerApps. No puede reutilizar un conector creado para Azure Logic Apps en Microsoft Flow o PowerApps (o viceversa). Puede utilizar la misma definición de OpenAPI para crear un conector personalizado para estos tres servicios.

#### *¿Necesita más revisión? Conector personalizado*

Puede obtener más información sobre conectores personalizados en <https://docs.microsoft.com/en-us/connectors/custom-connectors/>.

### *Crea una plantilla personalizada para Logic Apps*

Una vez que haya creado una aplicación Azure Logic, puede reutilizarla en otras suscripciones de Azure o compartirla con otros colegas. Puede crear una plantilla desde su aplicación Azure Logic en funcionamiento para automatizar los procesos de implementación. Cuando crea una plantilla, está convirtiendo la definición de su aplicación lógica de Azure en una plantilla de Azure Resource Manager (ARM). El uso de plantillas ARM le permite aprovechar la flexibilidad de la plataforma ARM al separar la definición de la aplicación lógica de Azure de los valores usados en la aplicación lógica. Cuando implementa una nueva aplicación de Azure Logic desde una plantilla, puede proporcionar un archivo de parámetros de la misma manera que lo hace con otras plantillas ARM. Azure también proporciona algunas plantillas de aplicaciones lógicas prediseñadas. Puede utilizar estas plantillas como base para crear plantillas.

Puede descargar una plantilla de Azure Logic Apps mediante varios mecanismos:

- **Azure Portal** Puede usar la opción Exportar plantilla en la aplicación Azure Logic en Azure Portal para descargar la plantilla ARM.
- **Visual Studio** Puede usar la extensión de herramientas de Azure Logic Apps para Visual Studio para conectar su suscripción de Azure y descargar una plantilla de sus Azure Logic Apps.
- **PowerShell** Puede usar el módulo LogicAppTemplate PowerShell para descargar una plantilla desde su aplicación Azure Logic.

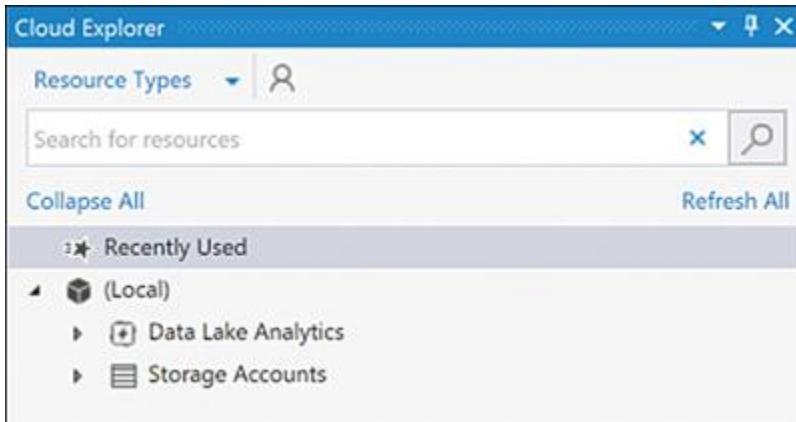
Una plantilla de aplicación lógica es un archivo JSON que consta de tres áreas principales:

- **Recurso de la aplicación lógica** Esta sección contiene información básica sobre la aplicación lógica en sí. Esta información es la ubicación del recurso, los planes de precios y la definición del flujo de trabajo.
- **Definición del flujo de trabajo** Esta sección contiene la descripción del flujo de trabajo, incluidos los desencadenantes y las acciones de su flujo de trabajo. Esta sección también contiene cómo la aplicación lógica ejecuta estos activadores y acciones.
- **Conexiones** Esta sección almacena la información sobre los conectores que utiliza en el flujo de trabajo.

Utilice el siguiente procedimiento para crear una plantilla desde su aplicación Azure Logic con Visual Studio:

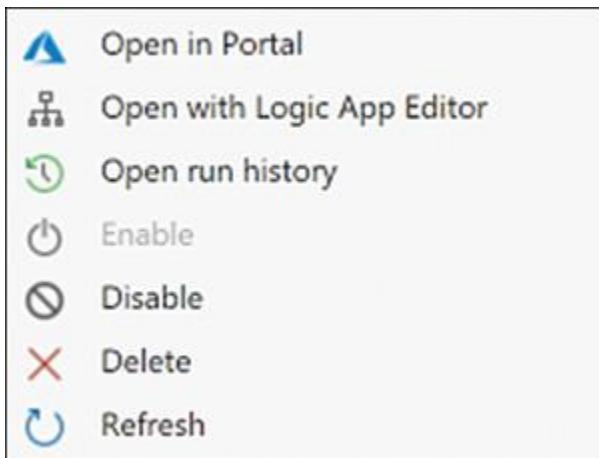
1. Descargue la extensión de la herramienta Azure Logic Apps para Visual Studio 2019 en <https://aka.ms/download-azure-logic-apps-tools-visual-studio-2019>.
2. Instale la extensión de la herramienta Azure Logic Apps.
3. Abra Visual Studio 2019.
4. En la ventana de bienvenida de Visual Studio 2019, haga clic en el vínculo Continuar sin código debajo de la sección Comenzar.
5. En la ventana de Visual Studio, haga clic en Ver> Explorador de la nube.

6. En la ventana del Explorador de la nube, que se muestra en la Figura 5-8, haga clic en el icono de usuario para abrir el Administrador de cuentas.



**Figura 5-8** Ventana del Explorador de la nube

7. Haga clic en el enlace Administrar cuentas.
8. En la sección Todas las cuentas, haga clic en el enlace Iniciar sesión.
9. Inicie sesión con una cuenta que tenga privilegios para acceder a su suscripción de Azure.
10. Asegúrese de que su suscripción de Azure aparezca en la lista de suscripciones en la ventana de Cloud Explorer.
11. Haga clic en el botón Aplicar.
12. En el control de árbol de Cloud Explorer, vaya a Su suscripción> Aplicaciones lógicas.
13. Haga clic con el botón derecho en la aplicación lógica que desea convertir en una plantilla.
14. En el menú contextual que se muestra en la Figura 5-9, haga clic en Abrir con Logic App Editor.



**Figura 5-9** Menú contextual de la herramienta Aplicación lógica

15. En la pestaña Logic App Editor, haga clic en el botón Descargar.
16. Seleccione una ubicación en la que desea descargar el archivo JSON.

En este punto, puede editar y personalizar su plantilla. Una vez que haya terminado con las modificaciones a su plantilla, puede crear un archivo de parámetros para implementar esta plantilla.

*¿Necesita más revisión? Plantillas de aplicaciones lógicas*

Puede obtener más información leyendo los siguientes artículos sobre las plantillas de aplicaciones lógicas:

- **Cree plantillas de aplicaciones lógicas** <https://docs.microsoft.com/en-us/azure/logic-apps/logic-apps-create-azure-resource-manager-templates>
- **Implementar plantillas de aplicaciones lógicas** <https://docs.microsoft.com/en-us/azure/logic-apps/logic-apps-deploy-azure-resource-manager-templates>

## HABILIDAD 5.2: IMPLEMENTAR LA GESTIÓN DE API

---

La mayoría de las aplicaciones y soluciones que puede encontrar o desarrollar hoy en día ofrecen una API para acceder a las funciones disponibles en la solución. En entornos empresariales, es bastante habitual que esas soluciones necesiten comunicarse entre sí mediante sus respectivas API. A veces, necesita exponer sus soluciones a sus clientes para ofrecer sus servicios. En esas situaciones, debe asegurarse de ofrecer una API consistente y segura. Implementar el mecanismo necesario para lograr un nivel de seguridad, consistencia y flexibilidad de nivel

empresarial no es fácil. Si también necesita publicar varios de sus servicios bajo una API común, esta tarea es aún más difícil.

Microsoft proporciona el servicio Azure API Management (APIM). Este servicio le permite crear una API de nivel empresarial para sus servicios de back-end existentes. Con APIM, puede publicar de forma segura sus aplicaciones de back-end, proporcionando a sus clientes una plataforma protegida contra ataques DOS o validaciones de tokens JWT.

### **Esta habilidad cubre cómo**

- [Crea una instancia de APIM](#)
- [Configurar la autenticación para las API](#)
- [Definir políticas para API](#)

#### *Crea una instancia de APIM*

El servicio de administración de API le permite exponer una parte (o todas) de las API que ofrecen sus sistemas de back-end. Al utilizar el servicio APIM, puede unificar todas sus API de back-end en una interfaz común que puede ofrecer a usuarios externos, como clientes o socios, y desarrolladores internos o externos. En general, el servicio APIM es una fachada de las API que configura en su instancia APIM. Gracias a esta función de fachada, puede personalizar la API de front-end que ofrece la instancia APIM sin cambiar la API de back-end.

Al exponer sus sistemas de back-end, no está limitado a los back-end de la API REST. Puede utilizar un servicio de back-end que utilice una API SOAP y luego publicar esta API SOAP como una API REST. Esto significa que puede actualizar sus sistemas de back-end más antiguos sin necesidad de modificar el código y aprovechar el mayor nivel de integración de las API REST.

Utilice el siguiente procedimiento para crear una nueva instancia de APIM:

1. Abra Azure Portal (<https://portal.azure.com>).
2. Haga clic en Crear un recurso en la parte superior de Azure Portal.
3. En la hoja Nueva, haga clic en Integración en la columna Azure Marketplace.

4. Haga clic en Administración de API en la columna Destacados. Si el servicio de administración de API no aparece en la columna Destacado, puede usar el cuadro de texto Buscar en el mercado y buscar el servicio de administración de API.
5. En la hoja del Servicio de administración de API, escriba un nombre para su nueva instancia de APIM.
6. Seleccione una suscripción en el menú desplegable Suscripción.
7. Seleccione un grupo de recursos del menú desplegable Grupo de recursos. Alternativamente, puede crear uno nuevo haciendo clic en el enlace Crear nuevo debajo del menú desplegable.
8. Seleccione una ubicación en el menú desplegable Ubicación.
9. En el cuadro de texto Nombre de la organización, escriba el nombre de su organización. Este nombre aparece en el portal del desarrollador y en las notificaciones por correo electrónico.
10. En el correo electrónico del administrador, escriba el nombre de la cuenta de correo electrónico que debe recibir todas las notificaciones de la instancia APIM. De forma predeterminada, el valor asociado con esta propiedad es la dirección de correo electrónico del usuario que inició sesión.
11. En el nivel de precios, deje seleccionado el nivel de desarrollador.
12. Haga clic en el botón Crear en la parte inferior de la hoja. El proceso de creación de la instancia APIM tarda varios minutos. Cuando su nueva instancia APIM esté lista, recibirá un correo electrónico de bienvenida en la dirección de correo electrónico del administrador que configuró en el paso 10.

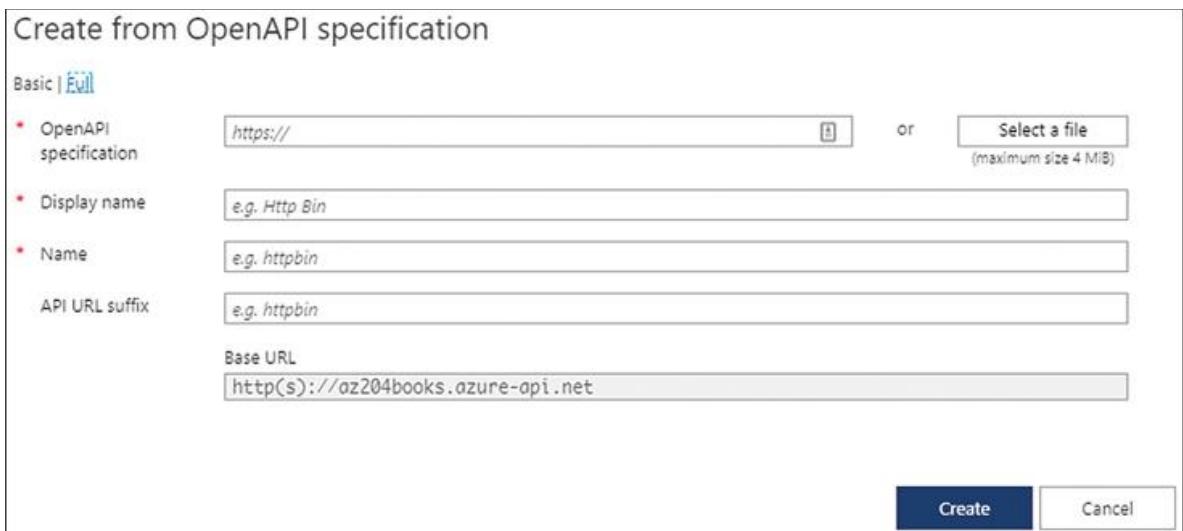
#### **Niveles de precios de nota**

El nivel de precios de desarrollador es apropiado para entornos de prueba y desarrollo, pero no debe usarlo para producción porque el nivel de desarrollador no ofrece funciones de alta disponibilidad y puede verse afectado por desconexiones durante las actualizaciones del nodo. Puede revisar la oferta completa y las funciones disponibles en cada nivel en <https://azure.microsoft.com/en-us/pricing/details/api-management/>.

Una vez que haya creado su instancia APIM, puede comenzar a agregar API a su instancia. En el siguiente procedimiento, agregará dos API. Va a aprovechar la API que creó en la sección “Crear un conector

personalizado para aplicaciones lógicas ” anteriormente en este capítulo. Para la segunda API, creará una definición de API en blanco y agregará solo los métodos que sean adecuados para usted.

1. Abra Azure Portal (<https://portal.azure.com> ).
2. Escriba el nombre de su instancia APIM en el cuadro de texto Buscar en la parte superior del portal.
3. Haga clic en el nombre de su instancia APIM en la lista de resultados.
4. Haga clic en API en el menú de navegación de su hoja de instancia APIM.
5. En la hoja Agregar una nueva API, haga clic en OpenAPI.
6. En el cuadro de diálogo Create From OpenAPI Specification, que se muestra en la Figura 5-10 , en el cuadro de texto OpenAPI Specification, escriba la URL de la definición Swagger de la API que publicó en Azure en la sección “ Crear un conector personalizado para Logic Apps . ” Recuerde que la URL debe ser similar a `https://<your_app_service_name>.azurewebsites.net/swagger/docs/v1` .



**Figura 5-10** Adición de una API de back-end a una instancia de APIM

7. Asegúrese de que Azure rellene automáticamente los cuadros de texto de las propiedades Nombre para mostrar y Nombre. Esto significa que Azure pudo importar los detalles de su API con éxito.

8. Elimine el contenido del cuadro de texto Nombre para mostrar.
9. Escriba **Biblioteca** en el cuadro de texto Nombre para mostrar.
10. Asegúrese de que el cuadro de texto Nombre tenga la **biblioteca de valores**.
11. Escriba **biblioteca** en el campo Sufijo de URL de API. Si va a conectar más de una API de back-end a la instancia de APIM, debe proporcionar un sufijo para cada API. La instancia APIM usa este sufijo para diferenciar entre las diferentes API que conectó a la instancia.
12. Haga clic en el botón Crear en la parte inferior del cuadro de diálogo.

En este punto, ha agregado su primera API de back-end a la instancia de APIM mediante el uso de la especificación OpenAPI de su API de back-end. En los siguientes pasos, agregará una API de back-end sin utilizar ninguna especificación. La creación de los puntos finales de front-end es útil si necesita conectar solo unos pocos puntos de conexión desde su API de back-end o si no tiene la especificación OpenAPI o SOAP de su API en ningún formato:

1. Haga clic en API en el menú de navegación de su hoja de instancia APIM.
2. En la hoja API, haga clic en Agregar API.
3. En la página Agregar una nueva API, haga clic en API en blanco.
4. En el cuadro de diálogo Crear una API en blanco, escriba **API falsa** en el cuadro de texto Nombre para mostrar.
5. Deje la propiedad Name con el valor predeterminado.
6. Escriba <https://fakerestapi.azurewebsites.net> en el cuadro de texto URL del servicio web.
7. Escriba **fakeapi** en el cuadro de texto Sufijo de URL de API.
8. Haga clic en el botón Crear.
9. En la pestaña Diseño de la hoja API con la API falsa recién agregada seleccionada, haga clic en Agregar operación.

10. En el editor Agregar operación, que se muestra en la [Figura 5-11](#), escriba **GetActivities** en el cuadro de texto Nombre para mostrar.

Fake API > Add operation

Frontend

- Display name: GetActivities
- Name: getactivities
- URL: GET /api/activities
- Description:
- Tags: e.g. Booking

**Figura 5-11** Agregar una operación de API a una API en una instancia APIM

11. En el menú desplegable Método URL HTTP, asegúrese de que esté seleccionado el método *GET*.
12. En el cuadro de texto URL, escriba **/ api / activities**.
13. Haga clic en el botón Guardar en la parte inferior del editor.
14. En la hoja de API, asegúrese de que la API falsa esté seleccionada.
15. Haga clic en la pestaña Prueba.
16. Haga clic en la operación GetActivities.
17. Haga clic en el botón Enviar en la parte inferior del panel de operaciones GetActivities. Con este panel, puede probar cada una de las operaciones que están definidas en su API.

En este punto, tiene dos API de back-end conectadas a su instancia APIM. Como puede ver en el ejemplo anterior, no necesita exponer toda la API de back-end. Al agregar las operaciones adecuadas, puede publicar solo aquellas partes de la API de back-end que le resulten útiles. Una vez que haya creado las API en su instancia de APIM, puede otorgar acceso a estas API a sus desarrolladores mediante el portal para desarrolladores. Puede acceder al portal de desarrolladores de APIM utilizando la URL [https://<your\\_APIM\\_name>.developer.azure-api.net/](https://<your_APIM_name>.developer.azure-api.net/).

## *¿Necesita más revisión? Portal para desarrolladores de administración de Azure Api*

El portal para desarrolladores de Azure API Management le permite proporcionar a sus clientes y terceros que deseen integrarse con su API un único punto de contacto para solicitar acceso a su aplicación y proporcionar documentación sobre su API. Puede leer más sobre cómo personalizar la experiencia del desarrollador revisando el siguiente artículo en <https://docs.microsoft.com/en-us/azure/api-management/api-management-howto-developer-portal>.

Tenga en cuenta que debe asociar un producto a su API para publicarlo. Debido a que no asoció sus API a ningún producto, sus API no estarán disponibles para el mundo externo. Puede asociar una API a más de un producto. De forma predeterminada, Azure proporciona dos productos: Starter e Unlimited. Estos productos están asociados con la demostración de la API de Echo que Azure implementa automáticamente cuando crea su instancia de API Management. Utilice el siguiente procedimiento para crear un producto y asociarlo con sus API:

1. Abra Azure Portal (<https://portal.azure.com>).
2. Escriba el nombre de su instancia APIM en el cuadro de texto Buscar en la parte superior central del portal.
3. Haga clic en el nombre de su instancia APIM en la lista de resultados.
4. Haga clic en Productos en el menú de navegación en su hoja de instancia APIM.
5. Haga clic en el botón Agregar en la esquina superior izquierda de la hoja Productos.
6. Escriba un Nombre en el cuadro de texto Nombre para mostrar en el panel Agregar producto.
7. Deje el valor en el cuadro de texto ID como está.
8. Escriba una descripción en el área de texto Descripción.
9. Seleccione el valor publicado en el control del interruptor de estado. Si no selecciona esta opción en este momento, puede publicar más tarde o puede publicar el Producto usando su panel.
10. Haga clic en el botón Seleccionar API en la sección API.
11. En la hoja API, seleccione Biblioteca y API falsas haciendo clic en la casilla de verificación junto al nombre de la API.

12. Haga clic en el botón Seleccionar en la parte inferior del panel.

13. Haga clic en el botón Crear en la parte inferior del panel Agregar producto.

De forma predeterminada, cuando crea un producto nuevo, solo los miembros del grupo integrado Administradores pueden acceder al producto. Puede configurar esto utilizando la sección Control de acceso en el producto.

#### *¿Necesita más revisión? Revisiones y versiones*

Durante la vida útil de su API, es posible que deba realizar modificaciones agregando, actualizando o eliminando operaciones en su API. Puede realizar estas modificaciones sin interrumpir el uso de su API mediante el uso de revisiones y versiones. Puede revisar cómo trabajar con revisiones y versiones en su API leyendo el artículo en <https://azure.microsoft.com/es-es/blog/versions-revisions/>.

### *Configurar la autenticación para las API*

Una vez que haya importado sus API de back-end, debe configurar la autenticación para acceder a estas API. Cuando configura las opciones de seguridad en la instancia de APIM, la API de back-end delega la seguridad a la instancia de APIM. Esto significa que incluso su API ha implementado su propio mecanismo de autenticación, nunca se usa cuando se accede a la API a través de la instancia APIM.

Esta capacidad de ocultar la autenticación de las API de back-end es útil para unificar toda la seguridad mediante un mecanismo de autenticación consistente y único. Puede administrar las opciones de autenticación asociadas con un producto o API mediante suscripciones. Una suscripción administra las claves que un desarrollador puede usar para acceder a su API. Si una solicitud HTTP realizada a una API protegida por una suscripción no proporciona una clave de suscripción válida, la puerta de enlace APIM rechaza inmediatamente la solicitud sin llegar a su API de back-end. Cuando define una suscripción, puede utilizar tres ámbitos diferentes para aplicarla:

- **Producto** El desarrollador puede acceder a todas las API configuradas en el producto asignado a la suscripción. Tradicionalmente, el desarrollador podía solicitar acceso a los productos utilizando el portal para desarrolladores. Esta ya no es una opción válida. Debe

proporcionar acceso al desarrollador mediante Azure Portal y configurar la suscripción APIM adecuada.

- **Todas las API** El desarrollador puede acceder a todas las API en su instancia APIM utilizando la misma clave de suscripción.
- **API** El desarrollador puede acceder a una única API en su instancia APIM mediante una clave de suscripción. No es necesario que la API sea parte de un producto.

Si usa el alcance de Todas las API, no necesita asociar la API de back-end con una API. La suscripción que usa este alcance permite el acceso directo a todas las API configuradas en su instancia de API Management. Puede utilizar el siguiente procedimiento para crear una suscripción y asociarla a un programa:

1. Abra Azure Portal (<https://portal.azure.com>).
2. Escriba el nombre de su instancia APIM en el cuadro de texto Buscar en la parte superior del portal.
3. Haga clic en el nombre de su instancia APIM en la lista de resultados.
4. Haga clic en Suscripciones en el menú de navegación de su hoja de instancia APIM.
5. Haga clic en el botón Agregar suscripción en la esquina superior izquierda de la hoja Suscripciones.
6. En el panel Nueva suscripción que se muestra en la Figura 5-12, escriba un Nombre para la suscripción. Tenga en cuenta que este nombre solo puede contener letras, números y guiones.

The screenshot shows a configuration page for creating a new API subscription. It includes fields for 'Name' (marked with a red asterisk), 'Display name', and 'Allow tracing' (with options 'No' and 'Yes'). A dropdown menu for 'Scope' is set to 'Product'. Below this, there are two sections: 'User' and 'Product'. The 'User' section has a right-pointing arrow. The 'Product' section has a red exclamation mark icon and a right-pointing arrow, with the text 'Configure required settings' below it.

**Figura 5-12** Creación de una nueva suscripción de administración de API

7. En el menú desplegable Alcance, seleccione el valor del Producto.
8. Haga clic en la propiedad Producto.
9. En el panel Productos, haga clic en el nombre del producto que creó en la sección anterior.
10. Haga clic en el botón Seleccionar en la parte inferior del panel.
11. Haga clic en el botón Guardar en la parte inferior del panel.
12. En la hoja Suscripción, haga clic en los puntos suspensivos al final de la fila de su suscripción recién creada.
13. En el menú contextual, haga clic en Mostrar / Ocultar claves. Puede utilizar cualquiera de estas claves para acceder a las API configuradas en el producto asociado con la Suscripción. Debe utilizar el encabezado Ocp-Apim-Subscription-Key para proporcionar la clave de suscripción en sus solicitudes HTTP.

Cuando está configurando una suscripción, puede asignar diferentes usuarios a la suscripción utilizando los parámetros de Usuarios en el panel Nueva suscripción. Esta es una forma de mejores prácticas de

proporcionar diferentes claves de suscripción a diferentes grupos de usuarios.

#### *¿Necesita más revisión? Otros métodos de autenticación*

El uso de suscripciones y claves de suscripción no es el único mecanismo para proteger el acceso a sus API. API Management le permite utilizar OAuth 2.0, certificados de cliente y listas blancas de IP. Puede utilizar los siguientes artículos para revisar cómo utilizar otros mecanismos de autenticación para proteger sus API:

- **Lista blanca de IP** <https://docs.microsoft.com/en-us/azure/api-management/api-management-access-restriction-policies#RestrictCallerIPs>
- **Autenticación OAuth 2.0 mediante Azure AD** <https://docs.microsoft.com/en-us/azure/api-management/api-management-howto-protect-backend-with-aad>
- **Autenticación mutua mediante certificados de cliente** <https://docs.microsoft.com/en-us/azure/api-management/api-management-howto-mutual-certificates>

### *Definir políticas para API*

Cuando publica una API de back-end mediante el servicio de administración de API, todas las solicitudes realizadas a su instancia de APIM se reenvían a la API de back-end correcta y la respuesta se envía de vuelta al solicitante. Ninguna de estas solicitudes o respuestas se altera o modifica de forma predeterminada, pero puede haber situaciones en las que necesite modificar algunas solicitudes y / o respuestas. Un ejemplo de estas necesidades de modificación es transformar el formato de una respuesta de XML a JSON. Otro ejemplo podría ser la limitación del número de llamadas entrantes de una IP o un usuario en particular.

Una política es un mecanismo que puede utilizar para cambiar el comportamiento predeterminado de la puerta de enlace APIM. Las políticas son documentos XML que describen una secuencia de pasos o declaraciones entrantes y salientes. Cada política se compone de cuatro secciones:

- **Entrante** En esta sección, puede encontrar cualquier declaración que se aplique a las solicitudes de los clientes API administrados.
- **Back End** Esta sección contiene los pasos que deben aplicarse a la solicitud que se debe enviar desde la puerta de enlace API a la API back-end.

- **Saliente** Esta sección contiene declaraciones o modificaciones que debe aplicar a la respuesta antes de enviarla al solicitante.
- En caso de **error** En caso de que haya un error en cualquiera de las otras secciones, el motor deja de procesar los pasos restantes en la sección defectuosa y salta a esta sección.

Cuando esté configurando o definiendo una política, debe tener en cuenta que puede aplicarla en diferentes niveles de alcance:

- **Global** La política se aplica a todas las API en su instancia APIM. Puede configurar políticas globales utilizando el editor de código en el editor de políticas de Todas las API en la hoja de API de su instancia de APIM.
- **Producto** La política se aplica a todas las API asociadas con un producto. Puede configurar políticas de producto en la hoja Políticas del producto en su instancia de API.
- **API** La política se aplica a todas las operaciones configuradas en la API. Puede configurar políticas con ámbito de API utilizando el editor de código en la opción Todas las operaciones en la pestaña Diseño de la API en su instancia APIM.
- **Operación** La política se aplica solo a una operación específica en su API. Puede configurar políticas de ámbito de operación utilizando el editor de código en la operación específica.

Las políticas son un mecanismo poderoso y muy flexible que le permite hacer mucho trabajo útil, como aplicar el almacenamiento en caché a las solicitudes HTTP, realizar el monitoreo de la solicitud y las respuestas, autenticarse con su API de back-end utilizando diferentes mecanismos de autenticación, o incluso interactuar con servicios externos, entre otros. Utilice el siguiente procedimiento para aplicar algunas transformaciones a la API de la biblioteca que configuró en la sección "Crear una instancia APIM" anteriormente en este capítulo:

1. Abra Azure Portal (<https://portal.azure.com>).
2. Escriba el nombre de su instancia APIM en el cuadro de texto Buscar en la parte superior del portal.
3. Haga clic en el nombre de su instancia APIM en la lista de resultados.

4. Haga clic en API en el menú de navegación de su hoja de instancia APIM.
5. Haga clic en API de biblioteca en la hoja API.
6. Haga clic en la operación *Obtener libros*.
7. Haga clic en la pestaña Prueba.
8. Haga clic en el botón Enviar en la parte inferior de la pestaña. Esto debería enviar una solicitud a la API de la biblioteca y obtener resultados similares a los que se muestran en la [Figura 5-13](#). En este procedimiento, transformará los encabezados HTTP dentro de los rectángulos rojos en la [Figura 5-13](#). Si no ve ningún libro cuando ejecuta esta prueba, cree algunos libros usando la aplicación Swagger de su API. Puede revisar cómo hacer esto consultando el proceso de prueba de la API de la biblioteca en la sección "[Crear un conector personalizado para aplicaciones lógicas](#)" anteriormente en este capítulo.

```
HTTP/1.1 200 OK

cache-control: no-cache
content-encoding: gzip
content-length: 258
content-type: application/json; charset=utf-8
date: Sat, 20 Jun 2020 20:19:49 GMT
expires: -1
ocp-apim-trace-location: https://apimstutuoos8fvxig6r2xuy.blob.
8%2F1%2BqMEeAMq9VGXIqmZYRK5mFAkSGyAY2RJvj41g%3D&se=2020-06-21T2
pragma: no-cache
vary: Accept-Encoding,Origin
x-aspnet-version: 4.0.30319
x-powered-by: ASP.NET

[
    {
        "Id": "f2a35131-1b47-420f-b8d0-87bfa8e62968",
        "Title": "AZ203",
        "Author": "Santiago Fernández"
    },
    {
        "Id": "d4cb9a4b-5532-4a39-b007-63436fefef10c",
        "Title": "AZ204",
        "Author": "Santiago Fernández"
    }
]
```

**Figura 5-13** Prueba de una operación de API

9. Haga clic en la pestaña Diseño.
10. Haga clic en Todas las operaciones en la lista de operaciones disponibles para esta API.

11. Haga clic en el icono junto a Políticas en la sección Procesamiento saliente.
12. En el Editor de políticas, mueva el cursor dentro de la sección Saliente, antes de la etiqueta base, y agregue una nueva línea presionando la tecla Intro.
13. Haga clic en el botón Mostrar fragmentos en la esquina superior derecha del Editor de políticas.
14. En la lista de políticas disponibles en el lado derecho del Editor de políticas, navegue hasta Políticas de transformación.
15. Haga clic en la política Establecer encabezado HTTP dos veces para insertar las políticas.
16. Modifique las políticas insertadas con el siguiente contenido:

Haga clic aquí para ver la imagen del código

```
<set-header name = "X-Powered-By" existe-acción = "eliminar" />

<set-header name = "X-AspNet-Version" existe-acción = "eliminar" />
```

17. Agregue una nueva línea debajo de las políticas insertadas.
18. Agregue el siguiente fragmento de código:

Haga clic aquí para ver la imagen del código

```
<set-body> @ {

    var respuesta = context.Response.Body.As <cadena> ();

    var arrayString = "{\"Biblioteca \": " + respuesta + "}";

    JObject libros = JObject.Parse (arrayString);

    JArray modifiedBooks = new JArray ();

    foreach (libro JObject en libros ["Biblioteca"]. ToObject
<JArray> ())

    {

        book.Add ("URL", "https://az204books.azure-
api.net/library/books/" + libro ["Id"]);

        modifiedBooks.Add (libro);

    }

}
```

```
        }

        return (cadena) modifiedBooks.ToString
( Newtonsoft.Json.Formatting.None ) ;

} </set-body>
```

19. Haga clic en el botón Guardar en la parte inferior del Editor de políticas.

20. Repita los pasos 6 a 8 para aplicar las políticas de transformación. Debería notar que faltan los encabezados X-Powered-By y X-AspNet-Version. Además, debería ver que todos los libros tienen una URL de propiedad adicional que apunta a la URL del libro.

Como puede ver en el ejemplo anterior, las políticas del servicio API Management son convincentes. Incluso puede usar el código C # para realizar modificaciones elaboradas a las solicitudes y respuestas realizadas a su API. Aunque este ejemplo muestra parte del poder de usar políticas con el servicio APIM, no debe usar este ejemplo para un entorno de producción, ya que faltan algunas verificaciones críticas en esta política de ejemplo. Debido a que creamos esta política para Todas las operaciones en la API de la biblioteca, cualquier llamada realizada a una operación diferente de Obtener libros fallará.

#### *¿Necesita más revisión? Más sobre políticas*

Hay muchas cosas útiles que puede hacer con las pólizas, demasiadas para cubirlas en esta sección. Si desea obtener más información sobre las políticas de APIM, puede revisar los siguientes artículos:

- **Manejo de errores en API Management** <https://docs.microsoft.com/en-us/azure/api-management/api-management-error-handling-policies>
- **Cómo establecer o editar las políticas de administración de la API de Azure** <https://docs.microsoft.com/en-us/azure/api-management/set-edit-policies>
- **Depura tus API mediante el seguimiento de solicitudes** <https://docs.microsoft.com/es-es/azure/api-management/api-management-howto-api-inspector>

## HABILIDAD 5.3: DESARROLLAR SOLUCIONES BASADAS EN EVENTOS

---

Uno de los principios fundamentales del desarrollo de código es reutilizar tanto como sea posible. Para que sea posible reutilizar el código, debe asegurarse de que el código esté lo más débilmente acoplado posible, lo que reduce al mínimo las dependencias con otras partes del código u otros sistemas.

Con este principio en mente, para hacer que los sistemas acoplados débilmente se comuniquen, es necesario utilizar algún tipo de comunicación. Las arquitecturas controladas por eventos permiten la comunicación entre sistemas separados al compartir información a través de eventos.

En general, un evento es un cambio significativo en el estado del sistema que ocurre en el contexto del sistema. Un ejemplo de un evento podría ser cuando un usuario agrega un artículo al carrito de compras en una aplicación de comercio electrónico, o cuando un dispositivo de IoT recopila la información de sus sensores.

Azure proporciona diferentes servicios, como Event Grid, centros de notificación o centros de eventos, para cubrir las diferentes necesidades al implementar arquitecturas impulsadas por eventos.

### **Esta habilidad cubre cómo**

- Implementar soluciones que usen Azure Event Grid
- Implementar soluciones que usan Azure Notification Hubs
- Implementar soluciones que usen Azure Event Hub

#### *Implementar soluciones que usen Azure Event Grid*

Azure Event Grid le permite crear una aplicación utilizando una arquitectura sin servidor al proporcionar una plataforma segura para administrar eventos. Puede usar Azure Event Grid para conectarse a varios tipos de orígenes de datos, como Azure Blob Storage, Azure Subscription, Event Hubs, IoT Hubs y otros; Azure Event Grid también le permite usar diferentes controladores de eventos para administrar estos eventos. También puede crear sus eventos personalizados para integrar su aplicación con Azure Event Grid. Antes de que pueda comenzar a usar

Azure Event Grid en su solución, hay algunos conceptos básicos que debemos revisar:

- **Evento** Se trata de un cambio de estado en el origen (por ejemplo, en Azure Blob Storage o cuando ocurre un evento cuando se agrega un nuevo blob a Azure Blob Storage).
- **Origen del evento** Este es el servicio o la aplicación cuando ocurre el evento. Hay una fuente de eventos para cada tipo de evento.
- **Controlador de eventos** Esta es la aplicación o servicio que reacciona al evento.
- **Temas** Estos son los puntos finales a los que la fuente del evento puede enviar los eventos. Puede utilizar temas para agrupar varios eventos relacionados.
- **Suscripciones a eventos** Cuando se agrega un nuevo evento a un tema, ese evento puede ser procesado por uno o más controladores de eventos. La suscripción a eventos es un punto final o un mecanismo integrado para distribuir los eventos entre los diferentes controladores de eventos. Además, puede usar suscripciones para filtrar eventos entrantes.

Una consideración importante que debe tener en cuenta es que un evento no contiene la información completa sobre el evento en sí. El evento solo contiene información relevante para el evento, como la fuente del evento, la hora en que tuvo lugar y un identificador único. Por ejemplo, cuando se agrega un nuevo blob a una cuenta de Azure Blob Storage, el nuevo evento de blob no contiene el blob. En su lugar, el evento contiene una referencia al blob en la cuenta de Azure Blob Storage.

Cuando necesite trabajar con eventos, configure un origen de eventos para enviar eventos a un tema. Cualquier sistema o controlador de eventos que necesite procesar esos eventos se suscribe a ese tema. Cuando surgen nuevos eventos, el origen del evento inserta el evento en el tema configurado en el servicio Azure Event Grids. Cualquier controlador de eventos suscrito a ese tema lee el evento y lo procesa de acuerdo con su programación interna. No es necesario que la fuente del evento tener controladores de eventos suscritos al tema; la fuente del evento empuja el evento al tema y lo olvida. Los siguientes pasos muestran cómo crear un tema personalizado. Luego, creará aplicaciones

de consola usando C # para enviar eventos al tema y procesar estos eventos.

1. Abra Azure Portal (<https://portal.azure.com> ).
2. En el cuadro de texto Buscar recursos, servicios y documentos en el área superior de Azure Portal, escriba **evento** .
3. Haga clic en Event Grid Topic en la lista de resultados.
4. En la hoja Temas de la cuadrícula de eventos, haga clic en el botón Agregar en la esquina superior izquierda de la hoja.
5. En el panel Crear tema, seleccione una suscripción en el menú desplegable Suscripción.
6. Seleccione un grupo de recursos en el menú desplegable Grupo de recursos. Alternativamente, puede crear un nuevo grupo de recursos haciendo clic en el enlace Crear nuevo debajo del menú desplegable.
7. En el cuadro de texto Nombre, escriba un nombre para el tema de la cuadrícula de eventos.
8. Seleccione una ubicación en el menú desplegable Ubicación.
9. Haga clic en el botón Revisar + Crear en la parte inferior del panel.
10. Haga clic en el botón Crear.

Cuando Azure Resource Manager termina de crear su nuevo tema de Event Grid, puede suscribirse al tema para procesar los eventos. Además, puede enviar sus eventos personalizados a este tema. Utilice los siguientes pasos para publicar eventos personalizados en su tema de cuadrícula de eventos recién creado:

1. Abra Visual Studio 2019.
2. En la ventana de inicio, haga clic en Crear un proyecto nuevo.
3. En la ventana Crear un nuevo proyecto, seleccione la plantilla Aplicación de consola (.NET Core).
4. Haga clic en el botón Siguiente en la esquina inferior derecha de la ventana.
5. Escriba un nombre de proyecto.
6. Seleccione una ubicación para su solución.

7. Haga clic en el botón Crear.
8. Haga clic en Herramientas> Administrador de paquetes NuGet> Administrar paquetes NuGet para la solución.
9. En la pestaña NuGet - Solución, haga clic en Examinar.
10. En el cuadro de texto Buscar, escriba **Microsoft.Azure.EventGrid**.
11. Haga clic en Microsoft.Azure.EventGrid en la lista de resultados.
12. En el lado derecho de la pestaña NuGet - Solución, haga clic en la casilla de verificación junto al nombre de su proyecto.
13. Haga clic en el botón Instalar.
14. En la ventana Vista previa de cambios, haga clic en el botón Aceptar.
15. En la ventana Aceptación de licencia, haga clic en el botón Acepto.
16. Repita los pasos del 10 al 15 e instale el paquete Microsoft.Extensions.Configuration.Json NuGet.
17. En la ventana del Explorador de soluciones, haga clic con el botón derecho en el nombre de su proyecto.
18. En el menú contextual, haga clic en Agregar> Nuevo elemento.
19. En Agregar nuevo elemento, escriba **json** en el cuadro de texto Buscar.
20. Haga clic en la plantilla Archivo JSON.
21. Escriba **appsettings.json** en el cuadro de texto Nombre.
22. Haga clic en el botón Agregar en la esquina inferior derecha de la ventana.
23. En la ventana del Explorador de soluciones, haga clic en el archivo appsettings.json.
24. En la ventana de propiedades, establezca la configuración Copiar en el directorio de salida en Copiar siempre.
25. Abra el archivo appsettings.json y reemplace el contenido del archivo con el contenido del Listado 5-6. Puede obtener la clave de

acceso de la hoja Clave de acceso en su Tema de la cuadrícula de eventos.

**Listado 5-6** archivo appsettings.json

Haga clic aquí para ver la imagen del código

---

```
{  
    "EventGridAccessKey": "  
        <Your_EventGridTopic_Access_Key> ",  
    "EventGridTopicEndpoint": "https://  
        <Your_EventGrid_Topic>. <region_name>-1.eventgrid.  
            azure.net/api/events "  
}
```

26. En la ventana del Explorador de soluciones, haga clic con el botón derecho en el nombre de su proyecto.
27. En el menú contextual, haga clic en Agregar> Nuevo elemento.
28. En la ventana Agregar nuevo elemento, seleccione Clase de la lista de elementos nuevos.
29. En el cuadro de texto del nombre en la parte inferior de la ventana, escriba **NewItemCreatedEvent.cs**.
30. Haga clic en el botón Agregar en la esquina inferior derecha de la ventana.
31. Reemplace el contenido del archivo NewItemCreatedEvent.cs con el contenido del Listado 5-7 .

**Listado 5-7** NewItemCreatedEvent.cs

Haga clic aquí para ver la imagen del código

---

```
// c # .NET  
  
utilizando Newtonsoft.Json;  
  
  
espacio de nombres <your_project_name>
```

```
{  
    clase NewItemCreatedEvent  
    {  
        [JsonProperty (PropertyName = "nombre")]  
        itemName cadena pública;  
    }  
}
```

32. Abra el archivo Program.cs.

33. Agregue las siguientes declaraciones de uso:

Haga clic aquí para ver la imagen del código

```
utilizando Microsoft.Azure.EventGrid;  
  
utilizando Microsoft.Azure.EventGrid.Models;  
  
utilizando Microsoft.Extensions.Configuration;  
  
usando System.Collections.Generic;
```

34. Reemplace el contenido del `Main`método con el contenido del Listado 5-8.

**Listado 5-8** Método principal de Program.cs

Haga clic aquí para ver la imagen del código

---

```
// C # .NET  
  
Constructor de IConfigurationBuilder = new  
ConfigurationBuilder (). AddJsonFile ("appsettings.  
json");  
  
IConfigurationRoot configuration = builder.Build ();
```

```
string topicEndpoint = configuración
["EventGridTopicEndpoint"];

string apiKey = configuración ["EventGridAccessKey"];


string topicHostname = new Uri (topicEndpoint) .Host;

TopicCredentials topicCredentials = new TopicCredentials
(apiKey);

Cliente EventGridClient = nuevo EventGridClient
(topicCredentials);

Lista <EventGridEvent> eventos = nueva Lista
<EventGridEvent> ();

events.Add (new EventGridEvent ()

{

    Id = Guid.NewGuid () . ToString () ,

    EventType = "MyCompany.Items.NewItemCreated" ,

    Datos = new NewItemCreatedEvent ()

    {

        itemName = "Elemento 1"

    } ,

    EventTime = DateTime.Now ,

    Asunto = "Tienda A" ,

    DataVersion = "3.7"

}) ;
```

```
client.PublishEventsAsync (topicHostname, eventos)
    .GetAwaiter ().GetResult ();

Console.WriteLine ("Eventos publicados en el tema Event
Grid");

Console.ReadLine ();
```

En este punto, su aplicación de consola publica eventos en el tema Event Grid que creó anteriormente. Presione F5 para ejecutar su aplicación de consola y asegurarse de que todo se compile y funcione correctamente; todavía no podrá ver el mensaje publicado. Use los siguientes pasos para crear una función de suscriptor de Azure que se conecte al tema de Event Grid y procese estos eventos:

1. Abra Visual Studio 2019.
2. En la ventana de inicio, haga clic en Crear un proyecto nuevo.
3. En la ventana Crear un nuevo proyecto, haga clic en la plantilla Azure Functions.
4. Haga clic en Siguiente.
5. Escriba un nombre de proyecto.
6. Seleccione una ubicación para su proyecto.
7. Haga clic en Crear.
8. En la ventana Crear una nueva aplicación de Azure Functions, haga clic en Activador de cuadrícula de eventos.
9. En el menú desplegable Cuenta de almacenamiento en el lado derecho de la ventana, haga clic en Examinar.
10. En la ventana de Azure Storage, seleccione una cuenta de Azure Storage de su suscripción para usarla con la función de Azure. Como alternativa, puede crear una nueva cuenta de almacenamiento de Azure haciendo clic en el vínculo Crear una cuenta de almacenamiento en la parte inferior de la ventana.
11. Haga clic en el botón Agregar.
12. Haga clic en Crear.
13. Cree una nueva clase C # vacía llamada `NewItemCreatedEventData`.

14. Reemplace el contenido del `NewItemCreatedEventArgs.cs` archivo con el contenido del [Listado 5-9](#).

**Listado 5-9** `NewItemCreatedEvent.cs`

[Haga clic aquí para ver la imagen del código](#)

---

// c # .NET

utilizando `Newtonsoft.Json`;

espacio de nombres `<your_project_name>`

{

    clase `NewItemCreatedEvent`

{

        [JsonProperty (PropertyName = "nombre")]

        itemName cadena pública;

}

}

15. Reemplace el contenido de `Function1.cs` con el contenido del [Listado 5-10](#).

**Listado 5-10** `Function1.cs`

[Haga clic aquí para ver la imagen del código](#)

---

// c # .NET

utilizando `Microsoft.Azure.WebJobs`;

utilizando `Microsoft.Azure.EventGrid.Models`;

utilizando `Microsoft.Azure.WebJobs.Extensions.EventGrid`;

utilizando `Microsoft.Extensions.Logging`;

```
utilizando Newtonsoft.Json.Linq;

espacio de nombres <your_project_name>

{

    clase pública estática Function1

    {

        [FunctionName ("EventGridTrigger")]

        Public static void Run ([EventGridTrigger]
EventGridEvent eventGridEvent,

        ILogger log)

        {

            log.LogInformation ("Trriger de cuadrícula
de eventos C # que maneja eventos de cuadrícula de
eventos.");

            log.LogInformation ($ "Nuevo evento
recibido: {eventGridEvent.Data}");


            si (eventGridEvent.Data es
StorageBlobCreatedEventData)

            {

                var eventData =
(StorageBlobCreatedEventData) eventGridEvent.Data;

                log.LogInformation ($ "Obtuve datos de
eventos BlobCreated, URI de blob {eventData.
```

```

        Url} ") ;

    }

        else if (eventGridEvent.EventType.Equals
("MyCompany.Items.NewItemCreated"))

    {

        NewItemCreatedEventData eventData =
( JObject ) eventGridEvent.Data) .

        ToObject <NewItemCreatedEventData> () ;

        log.LogInformation ($ "Evento
personalizado de elemento nuevo, Nombre {eventData.

        nombre del artículo}");

    }

}

}

```

16. Publique la función de Azure en su suscripción de Azure. Use el procedimiento en la siguiente URL para publicar una función de Azure en Azure: <https://docs.microsoft.com/en-us/azure/azure-functions/functions-develop-vs#publish-to-azure>.
17. Abra su tema de Event Grid en Azure Portal.
18. En la hoja Descripción general del tema de Event Grid, haga clic en el botón Suscripción al evento.
19. En la hoja Crear suscripción a eventos, que se muestra en la Figura 5-14, escriba un Nombre para la suscripción.

EVENT SUBSCRIPTION DETAILS	
Name	<input type="text"/>
Event Schema	<input type="text"/> Event Grid Schema <span style="float: right;">▼</span>
TOPIC DETAILS	
Pick a topic resource for which events should be pushed to your destination. <a href="#">Learn more</a>	
Topic Type	 Event Grid Topic
Source Resource	<a href="#">az204EventTopic</a>
EVENT TYPES	
Pick which event types get pushed to your destination. <a href="#">Learn more</a>	
Filter to Event Types	<input type="button" value="▼ Add Event Type"/>
ENDPOINT DETAILS	
Pick an event handler to receive your events. <a href="#">Learn more</a>	
Endpoint Type	 Azure Function <a href="#">(change)</a>
Endpoint	<a href="#">Select an endpoint</a>

**Figura 5-14** Creación de una suscripción mediante un punto final de WebHook

20. En el menú desplegable Tipo de extremo, seleccione Función de Azure.
21. Haga clic en el vínculo Seleccionar un punto de conexión debajo del tipo de punto de conexión de la función de Azure.
22. En el panel Seleccionar función de Azure, en el menú desplegable Aplicación de función, seleccione la función de Azure que publicó anteriormente en esta sección.
23. Salga del menú desplegable Slot con el valor de Producción.
24. Asegúrese de que el nombre de su función de Azure aparezca en el menú desplegable Función.
25. Haga clic en el botón Confirmar selección.
26. Haga clic en el botón Crear.

En este punto, debería poder publicar y procesar eventos utilizando el tema de la cuadrícula de eventos que creó anteriormente. Utilice los siguientes pasos para asegurarse de que todo funcione correctamente:

1. Abra la aplicación de la consola del editor en Visual Studio 2019.
2. Ejecute la aplicación de consola para publicar un evento en el tema.
3. Abra el portal de Azure y navegue hasta su función de Azure.
4. En la hoja Azure Functions, haga clic en Supervisar en el control de árbol.
5. En la hoja Monitor, haga clic en el botón Configurar para configurar la integración de Application Insights. Necesita esta integración para poder capturar registros de invocación.
6. En la hoja Application Insights, deje seleccionada la opción Crear nuevo recurso. Como alternativa, puede utilizar una instancia de Application Insights existente mediante la opción Seleccionar recurso existente.
7. Haga clic en el botón Aceptar.
8. Debería poder ver una lista de invocaciones cuando se haya llamado a la función porque llegó un nuevo evento al tema de la cuadrícula de eventos.
9. Haga clic en una de las invocaciones exitosas; obtendrá un resultado similar al de la Figura 5-15.

Invocation Details		
	<a href="#">Run query in Application Insights</a>	
Timestamp	Message	Type
2020-06-21 10:39:37.607	Executing 'EventGridTrigger' (Reason='EventGrid trigger fired at 2020-06-21T10:39:37.6075878+00:00', Id=05fa7328-314e-4795-a66f-39393cb9e102)	Information
2020-06-21 10:39:37.608	C# Event Grid trigger handling EventGrid Events.	Information
2020-06-21 10:39:37.608	New event received: { "name": "Item 1" }	Information
2020-06-21 10:39:37.608	New item Custom Event, Name Item 1	Information
2020-06-21 10:39:37.608	Executed 'EventGridTrigger' (Succeeded, Id=05fa7328-314e-4795-a66f-39393cb9e102)	Information

**Figura 5-15** Mensajes de registro de un procesamiento de eventos exitoso

**Nota Supervisión de funciones de Azure**

Debe tener habilitada la integración de Application Insight para poder ver los mensajes de registro generados desde la función de Azure. Consulte el artículo sobre cómo supervisar Azure Functions mediante Application Insights en <https://docs.microsoft.com/en-us/azure/azure-functions/functions-monitoring> .

La función de Azure que usamos en este ejemplo puede administrar no solo eventos personalizados, sino también eventos de una cuenta de almacenamiento de Azure. Como ejercicio, puede crear una nueva suscripción que escuche solo los eventos de la cuenta de almacenamiento de Azure y use la función de Azure que publicó anteriormente en esta sección para administrar los eventos producidos por la cuenta de almacenamiento de Azure.

Otra consideración importante que debe tener en cuenta cuando agrega un controlador a una suscripción de Azure Event Grid es la validación del controlador. Dependiendo del tipo de controlador que use, este proceso de validación lo realiza automáticamente el SDK, o debe implementarlo manualmente. Cuando utiliza un punto final HTTP como controlador de eventos, debe ocuparse de la verificación de la suscripción. Este proceso de verificación consiste en un código de verificación enviado por el servicio Event Grid al punto final del webhook. Su aplicación debe responder al servicio Event Grid utilizando el mismo código de verificación. Puede encontrar un ejemplo detallado de cómo realizar esta

verificación revisando el código disponible en <https://github.com/Azure-Samples/azure-event-grid-viewer>.

#### *¿Necesita más revisión? Políticas de letra muerta y reintento*

Cuando trabaja con arquitecturas controladas por eventos, puede haber situaciones en las que el evento no se pueda entregar al controlador de eventos. En esas situaciones, es apropiado establecer una estrategia de reintento para intentar recuperar el evento antes de que expire. Puede obtener más información sobre estas políticas de reintento y la administración de *mensajes no entregados* en <https://docs.microsoft.com/en-us/azure/event-grid/manage-event-delivery>.



#### *Sugerencia para el examen*

Event Grid es uno de los servicios que proporciona Azure para intercambiar información entre diferentes sistemas. Estos sistemas publican y consumen eventos de Event Grid, lo que le permite desacoplar los diferentes elementos de su arquitectura. Asegúrese de comprender completamente el papel que desempeña cada elemento en el intercambio de información mediante Event Grid.

#### *Implementar soluciones que usan Azure Notification Hubs*

El desarrollo de aplicaciones a las que se puede acceder mediante dispositivos móviles puede ser un desafío porque, por lo general, debe permitir el acceso a su aplicación desde diferentes plataformas móviles. El desafío se vuelve aún mayor porque las diferentes plataformas móviles utilizan diferentes sistemas de notificación para enviar eventos. Debe tratar con el Servicio de notificaciones push de Apple (APNS), el Servicio de mensajería en la nube de Google Firebase (FCM) o el Servicio de notificación de Windows (WNS), y estas son solo las plataformas móviles líderes en el mercado. Hay muchas otras plataformas móviles que puede utilizar para enviar notificaciones a su aplicación móvil.

Azure Notification Hubs proporciona una capa de abstracción que puede usar para conectarse a diferentes plataformas móviles de notificaciones push. Gracias a esta abstracción, puede enviar el mensaje de notificación al Notification Hub, que administra el mensaje y lo entrega a la plataforma adecuada. También puede definir y utilizar plantillas multiplataforma. Con estas plantillas, se asegura de que su solución envíe mensajes coherentes independientemente de la plataforma móvil que esté utilizando.

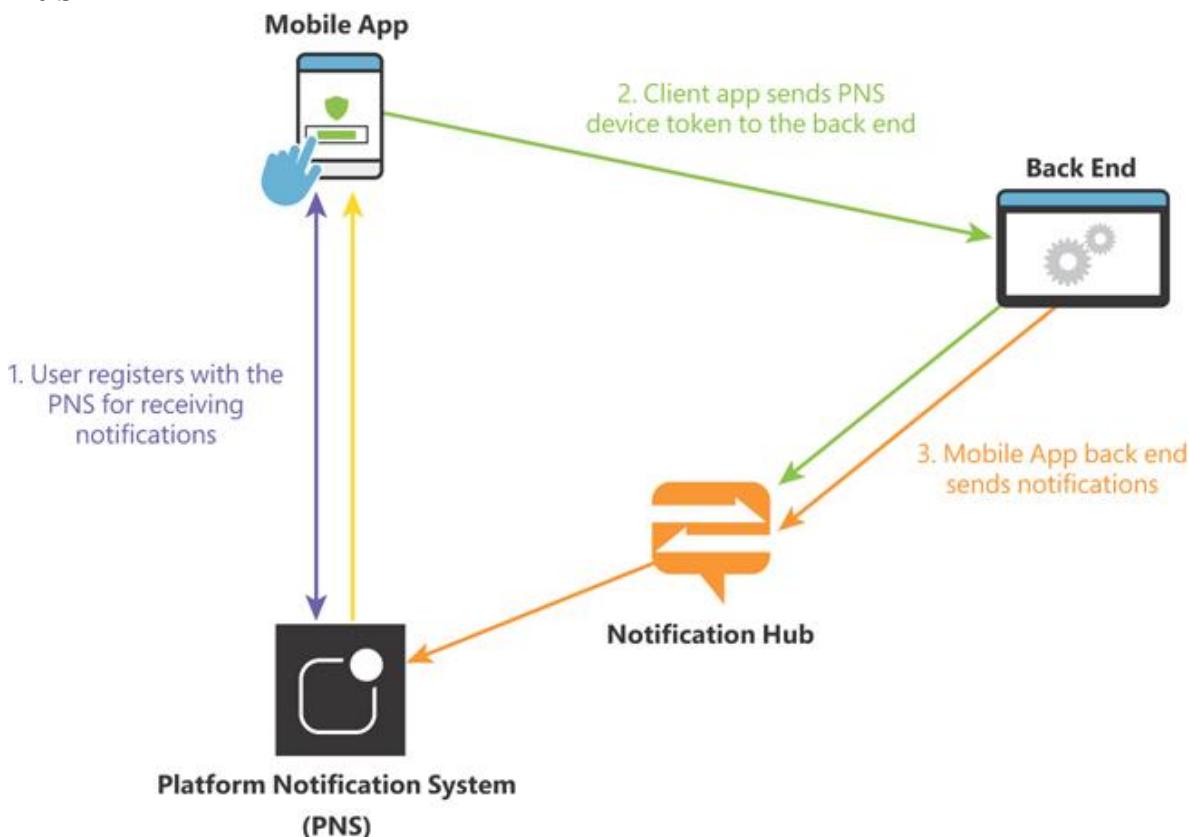
Cuando desarrolla una aplicación móvil, existe una alta probabilidad de que necesite enviar información a sus usuarios cuando no están usando la aplicación. Al hacerlo, utiliza las conocidas notificaciones push. Este mecanismo de comunicación asincrónica le permite interactuar con sus usuarios cuando están desconectados. Para que esta interacción suceda, algunos actores clave son parte de esta comunicación asincrónica:

- **El cliente de la aplicación móvil** Esta es su aplicación móvil real, que se ejecuta en el dispositivo de su usuario. El usuario debe registrarse en el Platform Notification System (PNS) para recibir notificaciones. Esto genera un controlador PNS que se almacena en el back-end de la aplicación móvil para enviar notificaciones.
- **El back-end de la aplicación móvil** Este es el back-end para el cliente de su aplicación y almacena el controlador PNS que el cliente recibió del PNS. Con este controlador, su servicio de back-end puede enviar notificaciones automáticas a todos los usuarios registrados.
- **Un sistema de notificación de plataforma (PNS)** Estas plataformas envían la notificación real al dispositivo del usuario. Los PNS dependen de la plataforma y cada proveedor tiene su propio PNS. Apple tiene el Servicio de notificaciones push de Apple, Google usa Firebase Cloud Messaging y Microsoft usa el Servicio de notificaciones de Windows.

Incluso si su aplicación móvil está dirigida a una sola plataforma, implementar notificaciones push requiere una gran cantidad de esfuerzo. Esto se debe a que algunos sistemas de notificación de plataforma solo se centran en enviar la notificación al dispositivo del usuario, pero no se ocupan de requisitos como notificaciones específicas o notificaciones de transmisión. Otro requisito para la mayoría de los PNS esesos tokens de dispositivo deben actualizarse cada vez que lanza una nueva versión de su aplicación. Esta operación requiere que su back-end maneje una gran cantidad de tráfico y actualizaciones de la base de datos simplemente para mantener actualizadas las fichas del dispositivo. Si necesita admitir diferentes plataformas móviles, estas tareas se vuelven aún más complicadas.

Microsoft le proporciona el Centro de notificaciones de Azure. Este servicio proporciona notificaciones push multiplataforma al back-end de su aplicación móvil, lo que le permite abstraerse de los detalles de la

gestión de cada sistema de notificación de plataforma para proporcionar una API coherente para interactuar con Notification Hub. Cuando necesite agregar notificaciones push a su aplicación móvil, integre el servicio Notification Hub con su servicio de back-end alojado en el Servicio de aplicaciones móviles. [La Figura 5-16](#) muestra el flujo de trabajo para enviar notificaciones automáticas a los usuarios que utilizan Notification Hub.



**Figura 5-16** Flujo de trabajo de notificaciones push usando Notification Hub

#### *Nota Integración de Notification Hub*

Microsoft también proporciona un SDK para facilitar la integración directa entre su código nativo (iOS, Android o Windows) o multiplataforma (Xamarin o Córdoba) y Azure Notification Hub, sin usar su back-end. El inconveniente de este enfoque es que Mobile Apps Client SDK elimina todas las etiquetas que puede asociar con el dispositivo por motivos de seguridad. Si necesita estas etiquetas para realizar notificaciones segmentadas, debe registrar los dispositivos de sus usuarios utilizando el back-end.

La interacción entre su aplicación móvil back-end y Notification Hub se realiza mediante el SDK de aplicaciones móviles para aplicaciones web ASP.NET o Node.js. Antes de tu back-endLa aplicación puede enviar notificaciones automáticas, necesita conectar su App Service con su

Notification Hub. Utilice el siguiente procedimiento para realizar esta conexión:

1. Inicie sesión en Azure Portal (<http://portal.azure.com>).
2. En la parte superior del portal, haga clic en Crear un recurso.
3. En la hoja Nuevo, en el cuadro de texto Buscar en el mercado, escriba **notificación**.
4. Haga clic en Centro de notificaciones en la lista de resultados.
5. En la hoja Notification Hub, haga clic en el botón Crear.
6. En la hoja Crear centro de notificaciones, seleccione su suscripción en el menú desplegable Suscripción.
7. Seleccione su grupo de recursos en el menú desplegable Grupo de recursos. Alternativamente, puede crear un nuevo grupo de recursos haciendo clic en el enlace Crear nuevo.
8. Escriba un espacio de nombres en el cuadro de texto Espacio de nombres de Notification Hub. Un espacio de nombres es un grupo de uno o más centros.
9. Escriba un nombre en el cuadro de texto Centro de notificaciones.
10. Seleccione la ubicación del centro de notificaciones en el menú desplegable Ubicación.
11. Deje el nivel de precios como Gratis.
12. Haga clic en el botón Crear en la parte inferior de la hoja.
13. Una vez que se haya creado el Centro de notificaciones, escriba el nombre de su nuevo Centro de notificaciones en el cuadro de texto Buscar recursos, servicios y documentos en la parte superior de Azure Portal.
14. En este punto, puede configurar la integración del Notification Hub con cada sistema de notificación de plataforma que desee utilizar para enviar notificaciones.

Ahora que tiene su Notification Hub listo, puede registrar su PNS con su Notification Hub y enviar notificaciones a su aplicación móvil. Este paso requiere que tenga una cuenta de desarrollador asociada con el PNS que desea utilizar. El siguiente procedimiento muestra cómo crear una aplicación en Firebase console:

1. Abra la consola de Firebase (<https://console.firebaseio.google.com> ).
2. Haga clic en el botón Agregar proyecto.
3. Escriba un nombre para su proyecto.
4. Haga clic en el botón Continuar.
5. En la página de Google Analytics, desactive Google Analytics para este proyecto.
6. Haga clic en el botón Crear proyecto.
7. Una vez que su proyecto esté listo, haga clic en el botón Continuar. Esto lo reenvía a la página principal de su proyecto.
8. En la página principal de su proyecto, haga clic en el ícono de Android debajo del título Comenzar agregando Firebase a su aplicación.
9. En Agregar Firebase a su aplicación de Android, escriba un nombre de paquete. Necesita este nombre de paquete en el siguiente ejemplo.
10. Haga clic en el botón Registrar aplicación.
11. Haga clic en el botón Descargar google-services.json. Necesitará este archivo más adelante en esta sección.
12. Haga clic en el botón Siguiente.
13. Vuelva a hacer clic en el botón Siguiente.
14. Haga clic en el botón Continuar a la consola.
15. En la ventana de la consola, haga clic en el icono de engranaje junto a la descripción general del proyecto en el panel de navegación en el lado izquierdo de la consola.
16. En el menú contextual, haga clic en Configuración del proyecto.
17. Haga clic en la pestaña Mensajería en la nube.
18. En la sección Credenciales del proyecto, copie el token asociado con la clave del servidor. Necesitará esto en un paso posterior.
19. Vaya al portal de Azure (<https://portal.azure.com> ).

20. Escriba el nombre de su Centro de notificaciones en el cuadro de texto Buscar recursos, servicios y documentos en la parte superior de Azure Portal.

21. Haga clic en la opción Google (GCM / FCM) en la sección Configuración en el menú de navegación en el lado izquierdo de su hoja Notification Hub.

22. Pegue la clave del servidor que copió en el paso 18 en el cuadro de texto Clave API.

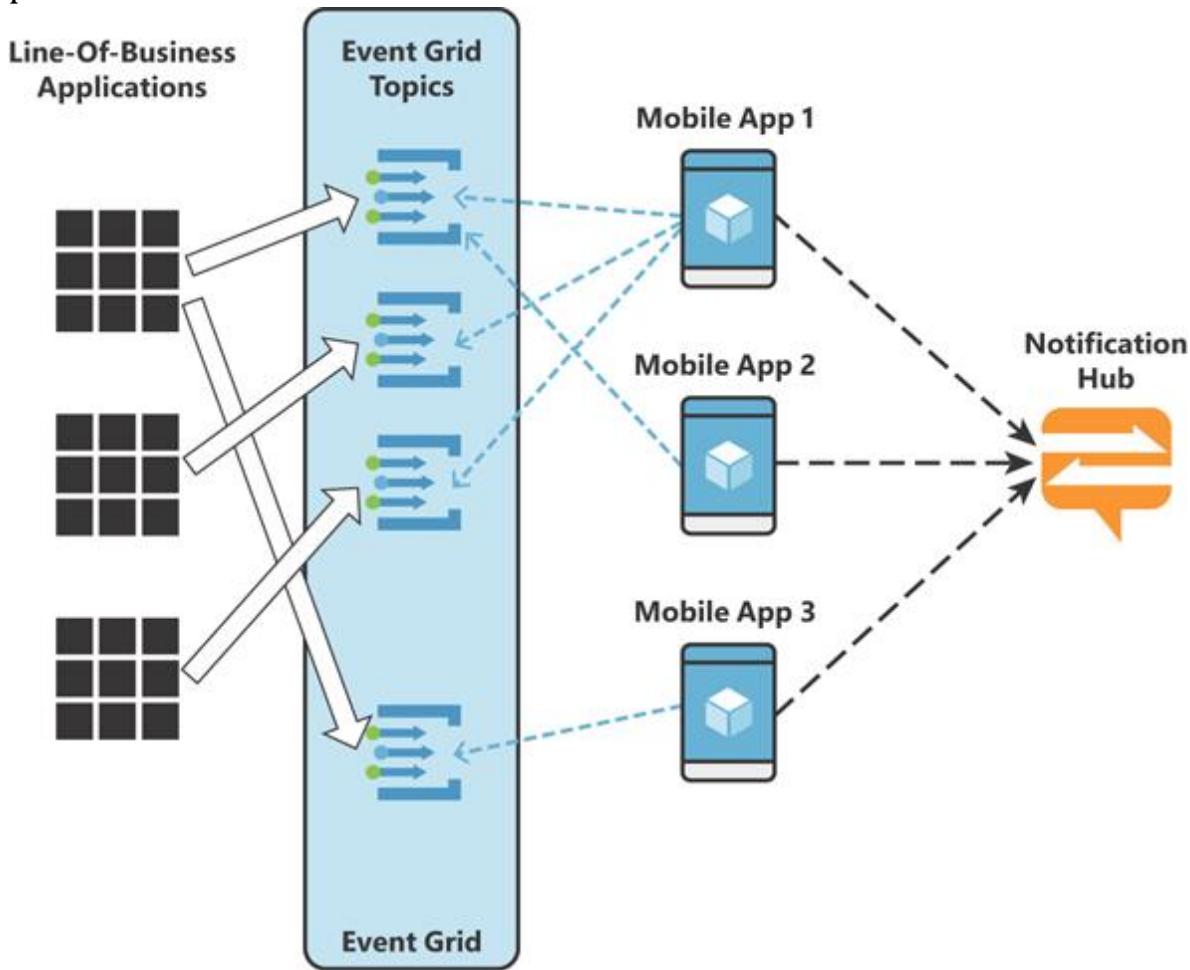
23. Haga clic en el botón Guardar.

En este punto, ha configurado su Notification Hub para enviar notificaciones a su aplicación de Android mediante Firebase. En aras de la brevedad, omito cómo programar una aplicación móvil para recibir notificaciones del Notification Hub. Puede encontrar ejemplos de cómo realizar esta gestión de notificaciones para cada plataforma revisando los siguientes artículos:

- **iOS** <https://docs.microsoft.com/en-us/azure/notification-hubs/ios-sdk-204>
- **Android** <https://docs.microsoft.com/en-us/azure/notification-hubs/notification-hubs-android-push-notification-google-fcm-get-started>
- **Windows Universal** <https://docs.microsoft.com/en-us/azure/notification-hubs/notification-hubs-windows-store-dotnet-get-started-wns-push-notification>

Cuando necesite agregar soporte de notificaciones push a su solución, debe pensar en el centro de notificaciones como parte de una arquitectura más grande. Un ejemplo de esto podría ser una solución que necesita conectar sus aplicaciones de línea de negocio con una aplicación móvil. En tal escenario, una posible arquitectura podría ser utilizar temas de Event Grid. Las aplicaciones de línea de negocio serían los editores de eventos para el tema apropiado, y luego puede implementar uno o más servicios de aplicaciones de Azure que estén suscritos a estos temas. Cuando una de las aplicaciones de línea de negocio publica un evento en el tema Event Grid, su Azure App Service, que actúa como un controlador de eventos, puede procesar el evento y enviar una notificación a sus usuarios móviles mediante el Centro de notificaciones de Azure. Figura 5-17 muestra un esquema de esta arquitectura. Como

puede ver en esa figura, el componente clave de la arquitectura es el servicio Event Grid y la implementación de una arquitectura impulsada por eventos.



**Figura 5-17** Diagrama de la arquitectura basada en eventos, incluidos los centros de notificación

*¿Necesita más revisión? Implementación de arquitectura de muestra*

Puede revisar una implementación de arquitectura de muestra utilizando mensajes de Service Bus en lugar de Event Grid leyendo el artículo en <https://docs.microsoft.com/en-us/azure/notification-hubs/notification-hubs-enterprise-push-notification-architectura>.

*Implementar soluciones que usen Azure Event Hub*

Azure Event Grid es un servicio excelente para implementar soluciones impulsadas por eventos, pero es solo una parte de una canalización más compleja. Aunque Event Grid es apropiado para trabajar con programación reactiva impulsada por eventos, no es la mejor solución

cuando necesita ingerir millones de eventos por segundo con baja latencia.

Azure Event Hub es una solución más adecuada cuando necesita un servicio que pueda recibir y procesar millones de eventos por segundo y proporcionar procesamiento de eventos de baja latencia. Azure Event Hub es la puerta principal de una canalización de big data que procesa millones de eventos. Una vez que Azure Event Hub recibe los datos, puede enviar el evento a Azure Event Grid, almacenar la información en una cuenta de Azure Blob Storage o almacenar los datos en Azure Data Lake Storage.

Cuando trabaja con centros de eventos, envía eventos al centro. La entidad que envía eventos al centro de eventos se conoce como editor de eventos. Un editor de eventos puede enviar eventos al centro de eventos mediante cualquiera de estos protocolos: AMQP 1.0, Kafka 1.0 (o posterior) o HTTPS.

Puede publicar eventos en el centro de eventos enviando un solo evento o agrupando varios eventos en una operación por lotes. Independientemente, si publica un solo evento o un lote de ellos, está limitado a un tamaño máximo de 1 MB de datos por publicación. Cuando Azure Event Hub almacena un evento, distribuye los diferentes eventos en diferentes particiones según la clave de partición proporcionada como uno de los datos del evento. Con este patrón, Azure Event Hub garantiza que todos los eventos que comparten la misma clave de partición se entreguen en la misma partición.

Una partición almacena los eventos a medida que llegan a la partición. De esta manera, los eventos más nuevos se agregan al final de la partición. No puede eliminar eventos de una partición. En su lugar, debe esperar a que expire el evento para eliminarlo de la partición. Como cada partición es independiente de otras particiones en el centro de eventos, las tasas de crecimiento son diferentes de una partición a otra. Puede definir el número de particiones que contiene su centro de eventos durante la creación del centro de eventos. Puede crear entre 2 y 32 particiones, aunque puede ampliar el límite de 32 si se pone en contacto con el equipo de Azure Event Hub. Tenga en cuenta que una vez que cree el centro de eventos y establezca el número de particiones, no podrá cambiar este número más adelante. Al planificar la cantidad de particiones que se asignarán al centro de eventos,

Puede conectar aplicaciones de receptor de eventos a un centro de eventos mediante grupos de consumidores. Un grupo de consumidores es equivalente a un flujo descendente en una arquitectura de procesamiento de flujo. Con los grupos de consumidores, puede tener diferentes consumidores o receptores de eventos, accediendo a diferentes vistas (estado, posición o desplazamiento) de las particiones en el centro de eventos. Los consumidores de eventos se conectan al centro de eventos mediante el protocolo AMQP que envía el evento al cliente tan pronto como hay nuevos datos disponibles.

El siguiente procedimiento muestra cómo crear un Centro de eventos de Azure:

1. Abra Azure Portal (<https://portal.azure.com>).
2. Expanda el menú de navegación haciendo clic en el ícono con tres líneas paralelas en la esquina superior izquierda de Azure Portal.
3. Haga clic en Todos los servicios en el menú de navegación.
4. En el cuadro de texto Buscar todo, escriba **evento**.
5. Haga clic en Event Hubs en la lista de resultados.
6. En la hoja Event Hubs, haga clic en el botón Agregar en la esquina superior izquierda de la hoja.
7. En el panel Crear espacio de nombres, asegúrese de seleccionar la suscripción correcta en el menú desplegable Suscripción.
8. Seleccione un grupo de recursos del menú desplegable Grupo de recursos. Alternativamente, puede crear un nuevo grupo de recursos haciendo clic en el enlace Crear nuevo debajo del menú desplegable.
9. Escriba un nombre para el espacio de nombres de Event Hub.
10. Seleccione una ubicación en el menú desplegable Ubicación.
11. Seleccione el nivel Básico en el menú desplegable Nivel de precios.
12. Deje las Unidades de rendimiento como 1.
13. Haga clic en el botón Revisar + Crear en la parte inferior del panel.

14. Haga clic en el botón Crear.
15. Navegue hasta el espacio de nombres de Event Hub recién creado.
16. En la hoja Descripción general de la hoja del espacio de nombres de Event Hub, haga clic en el botón Event Hub.
17. En el panel Crear centro de eventos, escriba un nombre para el centro de eventos.
18. Deje el recuento de particiones en 2. Recuerde que no puede cambiar este valor una vez creado el centro de eventos.
19. Haga clic en el botón Crear.
20. Haga clic en Políticas de acceso compartido en el menú de navegación en el lado izquierdo del espacio de nombres de Event Hub.
21. Haga clic en el `RootManageSharedAccessKey`.
22. Copie el valor de la clave principal de la cadena de conexión. Necesita este valor para el paso 9 en el siguiente procedimiento.

Una vez que haya creado el espacio de nombres de su centro de eventos y su centro, puede comenzar a enviar y consumir eventos desde el centro. Utilice el siguiente procedimiento para crear dos aplicaciones de consola, una para enviar eventos y otra para recibir eventos:

1. Abra Visual Studio 2019.
2. En la pantalla de bienvenida, haga clic en Crear un proyecto nuevo.
3. Seleccione la plantilla Aplicación de consola (.NET Core).
4. Haga clic en el botón Siguiente.
5. Escriba un nombre de proyecto.
6. Seleccione una ubicación para el proyecto.
7. Haga clic en el botón Crear.
8. Instale el paquete Microsoft.Azure.EventHubs NuGet.
9. Reemplace el contenido del archivo Program.cs con el contenido del [Listado 5-11](#). Recibió la cadena de conexión del

espacio de nombres de Event Hub en el último paso del procedimiento anterior.

**Listado 5-11** Function1.cs

Haga clic aquí para ver la imagen del código

---

```
// C # .NET

using System;
using System.Text;
using System.Threading.Tasks;
using Microsoft.Azure.EventHubs;

espacio de nombres <your_project_name>

{
    programa de clase

    {
        eventHubClient estático privado EventHubClient;

        private const string EventHubConnectionString = "
<Your_event_hub_"

        namespace_connection_string> ";

        private const string EventHubName = "
<your_event_hub_name> ";

        privado const int numMessagesToSend = 100;

    }

    static void Main (cadena [] argumentos)
    {
    }
}
```

```
var connectionStringBuilder = new
EventHubsConnectionStringBuilder (
    EventHubConnectionString)
{
    EntityPath = EventHubName
};

eventHubClient =
EventHubClient.CreateFromConnectionString (
    connectionStringBuilder.ToString ());
}

para (var i = 0; i <numMessagesToSend; i++)
{
    intentar
    {
        var message = $"Mensaje {i}";
        Console.WriteLine ($"Enviando mensaje:
{mensaje}");
        eventHubClient.SendAsync (new EventData
(Encoding.UTF8.GetBytes (mensaje)));
    }
    catch (excepción de excepción)
    {
        Console.WriteLine ($"DateTime.Now>
Excepción: {excepción.Message}");
    }
}
```

```

        Task.Delay (10);

    }

    Console.WriteLine ($ "{numMessagesToSend} "
mensajes enviados.);

    eventHubClient.CloseAsync ();

    Console.WriteLine ("Presione ENTER para
salir.");
```

Console.ReadLine ();

}

}

En este punto, puede presionar F5 y ejecutar la aplicación de consola. Esta consola de aplicación envía 100 mensajes al centro de eventos que configuró en el `EventHubName` constante. En el siguiente procedimiento, creará otra consola de aplicación para implementar un host de procesador de eventos. Event Processor Host es un agente que le ayuda a recibir eventos desde el centro de eventos. El procesador de eventos gestiona automáticamente los puntos de control persistentes y la recepción de eventos paralelos. El host del procesador de eventos requiere una cuenta de almacenamiento de Azure para procesar los puntos de control persistentes.

#### **Nota Requisitos de ejemplo**

Debe crear un contenedor de Azure Blob Storage para ejecutar este ejemplo. Puede revisar cómo crear un contenedor de blobs y cómo obtener la clave de acceso leyendo los siguientes artículos:

- **Cree un contenedor** <https://docs.microsoft.com/en-us/azure/storage/blobs/storage-quickstart-blobs-portal#create-a-container>
- **Obtenga las claves de acceso** <https://docs.microsoft.com/en-us/azure/storage/common/storage-account-manage#access-keys>

Siga estos pasos para crear la aplicación de consola que implementa Event Processor Host:

1. Abra Visual Studio 2019.
2. En la pantalla de bienvenida, haga clic en Crear un proyecto nuevo.
3. Seleccione la plantilla Aplicación de consola (.NET Core).
4. Haga clic en el botón Siguiente.
5. Escriba un nombre de proyecto.
6. Seleccione una ubicación para el proyecto.
7. Haga clic en el botón Crear.
8. Instale los siguientes paquetes de NuGet:
  1. Microsoft.Azure.EventHubs
  2. Microsoft.Azure.EventHubs.Processor
9. Cree una nueva clase C # vacía y **asígnelle el nombre SimpleEventProcessor**. En pasos posteriores, esta clase implementa la interfaz IEventProcessor que contiene la firma de los métodos necesarios para el Procesador de eventos.
10. Reemplace el contenido del archivo SimpleEventProcessor.cs con el contenido del [Listado 5-12](#).

#### **Listado 5-12** SimpleEventProcessor.cs

[Haga clic aquí para ver la imagen del código](#)

---

```
// c # .NET

utilizando Microsoft.Azure.EventHubs;

utilizando Microsoft.Azure.EventHubs.Processor;

usando el sistema;

usando System.Collections.Generic;
```

```
usando System.Text;

usando System.Threading.Tasks;

espacio de nombres <your_project_name>

{

    clase pública SimpleEventProcessor: IEventProcessor

    {

        Tarea pública CloseAsync (contexto
PartitionContext, motivo CloseReason)

        {

            Console.WriteLine ($ "Procesador apagándose.
Partición

'{context.PartitionId}', Razón: '{razón}'.

");

            return Task.CompletedTask;
        }

        Tarea pública OpenAsync (contexto
PartitionContext)

        {

            Console.WriteLine ($ "SimpleEventProcessor
inicializado. Partición: '{context.

PartitionId}' ");

            return Task.CompletedTask;
        }
    }
}
```

```

    Public Task ProcessErrorAsync (contexto
PartitionContext, error de excepción)

    {

        Console.WriteLine ($ "Error en la partición:
{context.PartitionId}, Error:

{mensaje de error}");

        return Task.CompletedTask;

    }

    Public Task ProcessEventsAsync (contexto
PartitionContext, IEnumerable

<EventData> mensajes)

{

    foreach (var eventData en los mensajes)

    {

        var data = Encoding.UTF8.GetString
(eventData.Body.Array, eventData.

Body.Offset, eventData.Body.Count);

        Console.WriteLine ($ "Mensaje recibido.
Partición: '{context.

PartitionId}', Datos:' {datos} ''");

    }

    return context.CheckpointAsync ();

}

```

}

11. Reemplace el contenido del archivo Program.cs con el contenido del Listado 5-13.

**Listado 5-13** Program.cs

Haga clic aquí para ver la imagen del código

---

```
// C # .NET

utilizando Microsoft.Azure.EventHubs;

utilizando Microsoft.Azure.EventHubs.Processor;

usando el sistema;

espacio de nombres <your_project_name>

{

    programa de clase

    {

        cadena const privada EventHubConnectionString =
                "
<su_evento_hub_namespace_connection_string> ";

        private const string EventHubName = "
<your_event_hub_name> ";

        private const string StorageContainerName = "
<your_container_name> ";

        private const string StorageAccountName = "
<your_storage_account_name> ";

        private const string StorageAccountKey = "
<your_storage_account_access_key> ";

        cadena privada estática de solo lectura
StorageConnectionString = cadena.Format(
```

```
    $  
"DefaultEndpointsProtocol = https;  
  
        AccountName =  
{StorageAccountName};  
  
        AccountKey =  
{StorageAccountKey} ");  
  
  
    static void Main (cadena [] argumentos)  
{  
  
    Console.WriteLine ("Registrando EventProcessor  
...");  
  
  
    var eventProcessorHost = nuevo  
EventProcessorHost (  
  
    EventHubName,  
  
    PartitionReceiver.DefaultConsumerGroupName,  
  
    EventHubConnectionString,  
  
    StorageConnectionString,  
  
    StorageContainerName);  
  
  
    // Registra el host del procesador de eventos y  
comienza a recibir mensajes  
  
    eventProcessorHost.RegisterEventProcessorAsync  
<SimpleEventProcessor> ();  
  
  
    Console.WriteLine ("Recibiendo. Presione ENTER  
para detener el trabajador.");
```

```

Console.ReadLine();

// Elimina el host del procesador de eventos
eventProcessorHost.UnregisterEventProcessorAsync
();

}

}

```

Ahora puede presionar F5 y ejecutar su aplicación de consola. La aplicación de consola se registra como un procesador de eventos y comienza a esperar eventos no procesados en el centro de eventos. Debido a que el tiempo de vencimiento predeterminado para los eventos en el centro de eventos es un día, debe recibir todos los mensajes enviados por su aplicación de consola de publicación en el ejemplo anterior. Si ejecuta la aplicación de la consola del editor de eventos sin detener la aplicación de la consola del procesador de eventos, debería poder ver los mensajes en la consola del procesador de eventos casi en tiempo real a medida que la consola de publicación de eventos los envía al centro de eventos. Este sencillo ejemplo también muestra cómo el centro de eventos distribuye los eventos en las diferentes particiones.



### *Sugerencia para el examen*

Azure Event Hub es un servicio apropiado para procesar grandes cantidades de eventos con baja latencia. Debe considerar el centro de eventos como el punto de partida en una canalización de procesamiento de eventos. Puede utilizar el centro de eventos como origen de eventos del servicio Event Grid.

### *¿Necesita más revisión? Conceptos de Event Hubs*

El servicio Azure Event Hub está diseñado para funcionar con canalizaciones de big data en las que necesita procesar millones de eventos por segundo. En esos escenarios, tomar una mala decisión al planificar la implementación de un centro de eventos puede tener un gran efecto en el rendimiento. Puede obtener más información sobre el servicio de centro de

eventos leyendo el artículo en <https://docs.microsoft.com/en-in/azure/event-hubs/event-hubs-features>.

## HABILIDAD 5.4: DESARROLLAR SOLUCIONES BASADAS EN MENSAJES

---

En la habilidad anterior, revisamos cómo usar servicios basados en eventos en los que un editor envía una notificación o evento ligero al sistema de gestión de eventos y se olvida de cómo se maneja el evento o si incluso se procesa.

En esta sección, revisaremos cómo desarrollar soluciones basadas en mensajes utilizando los servicios de Azure. En términos generales, un mensaje son datos sin procesar producidos por un servicio con el objetivo de ser almacenados o procesados en otro lugar. Esto significa que el editor de los mensajes tiene la expectativa de que algún otro sistema o suscriptor procese el mensaje. Debido a esta expectativa, el suscriptor debe notificar al editor sobre el estado del mensaje.

### Esta habilidad cubre cómo

- Implementar soluciones que utilicen Azure Service Bus
- Implementar soluciones que usen colas de Azure Queue Storage

### *Implementar soluciones que utilicen Azure Service Bus*

Azure Service Bus es un agente de mensajes de integración de nivel empresarial que permite que diferentes aplicaciones se comuniquen entre sí de manera confiable. Un mensaje es un dato sin procesar que una aplicación envía de forma asíncrona al intermediario para ser procesados por otra aplicación conectada al intermediario. El mensaje puede contener información JSON, XML o de texto.

Hay algunos conceptos que debe revisar antes de comenzar a trabajar con Azure Service Bus:

- **Espacio de nombres** Es un contenedor para todos los componentes de la mensajería. Un solo espacio de nombres puede contener varias colas y temas. Puede utilizar espacios de nombres como contenedores de aplicaciones que asocian una única solución

a un único espacio de nombres. Los diferentes componentes de su solución se conectan a los temas y colas en el espacio de nombres.

- **Cola** Una cola es el contenedor de mensajes. La cola almacena el mensaje hasta que la aplicación receptora recupera y procesa el mensaje. La cola de mensajes funciona como una pila FIFO (primero en entrar, primero en salir). Cuando llega un nuevo mensaje a la cola, el servicio Service Bus asigna una marca de tiempo al mensaje. Una vez que se procesa el mensaje, el mensaje se mantiene en almacenamiento redundante. Las colas son apropiadas para escenarios de comunicación punto a punto en los que una sola aplicación necesita comunicarse con otra sola aplicación.
- **Tema** Utiliza temas para enviar y recibir mensajes. La diferencia entre las colas y los temas es que los temas pueden tener varias aplicaciones que reciben mensajes que se utilizan en escenarios de publicación / suscripción. Un tema puede tener varias suscripciones en las que cada suscripción de un tema recibe una copia del mensaje enviado al tema.

Utilice el siguiente procedimiento para crear un espacio de nombres de Azure Service Bus; luego, puede crear un tema en el espacio de nombres. Vamos a utilizar ese tema para crear dos aplicaciones de consola para enviar y recibir los mensajes del tema:

1. Abra Azure Portal (<https://portal.azure.com>).
2. Haga clic en Crear un recurso en la parte superior del portal.
3. Haga clic en Integración en la columna Azure Marketplace.
4. Haga clic en Service Bus en la columna Destacados.
5. En el panel Crear espacio de nombres, asegúrese de seleccionar la suscripción correcta en el menú desplegable Suscripción.
6. Seleccione un grupo de recursos en el menú desplegable Grupo de recursos. Alternativamente, puede crear un nuevo grupo de recursos haciendo clic en el enlace Crear nuevo debajo del control del menú desplegable.
7. Escriba un nombre para el Bus de servicio en el cuadro de texto Nombre del espacio de nombres.

8. Seleccione una ubicación en el menú desplegable Ubicación.
9. Seleccione el nivel Estándar en el menú desplegable Nivel de precios. No puede crear temas en el nivel de precios Básico; debe utilizar al menos el nivel Estándar.
10. Haga clic en el botón Revisar + Crear en la parte inferior del panel.
11. Haga clic en el botón Crear.
12. Vaya al recurso una vez que Azure Resource Manager finalice la implementación de su nuevo espacio de nombres de Service Bus.
13. En la hoja Descripción general del espacio de nombres de Service Bus, haga clic en el botón Tema.
14. En el panel Crear tema, que se muestra en la [Figura 5-18](#), escriba un Nombre para el tema.

The screenshot shows the 'Create topic' configuration page. It includes fields for Name (mandatory), Max topic size (1 GB), Message time to live (set to 14 days, 0 hours, 0 minutes, 0 seconds), Enable duplicate detection (checked), Duplicate detection window (0 days, 0 hours, 0 minutes, 30 seconds), and Enable partitioning (unchecked).

Days	Hours	Minutes	Seconds
14	0	0	0

Days	Hours	Minutes	Seconds
0	0	0	30

**Figura 5-18** Creación de un tema nuevo

15. Deje los parámetros Tamaño máximo de tema y Tiempo de vida del mensaje como están.
16. Marque Habilitar detección de duplicados. Esta opción asegura que el tema no almacene mensajes duplicados durante la ventana de detección configurada.

17. Haga clic en el botón Crear.
18. Haga clic en Políticas de acceso compartido en el menú de navegación en el lado izquierdo del espacio de nombres de Service Bus.
19. Haga clic en la `RootManageSharedAccessKey` política.
20. Copie la cadena de conexión principal. Va a utilizar la cadena de conexión más adelante en esta sección.
21. Haga clic en Temas en el menú de navegación en el lado izquierdo del espacio de nombres de Service Bus.
22. Haga clic en su tema.
23. En la hoja Descripción general del tema de Service Bus, haga clic en el botón Suscripción.
24. En el panel Crear suscripción, que se muestra en la [Figura 5-19](#), escriba un nombre para la suscripción.

**Create subscription**

Service Bus

**SUBSCRIPTION SETTINGS**

Name \* ⓘ

Max delivery count \* ⓘ

Auto-delete after idle for ⓘ

Days	Hours	Minutes	Seconds
14	0	0	0

Never auto-delete

Forward messages to queue/topic ⓘ

**MESSAGE SESSIONS**

Service bus sessions allow ordered handling of unbounded sequences of related messages. With sessions enabled a subscription can guarantee first-in-first-out delivery of messages. [Learn more.](#)

Enable sessions

**MESSAGE TIME TO LIVE AND DEAD-LETTERING**

Message time to live (default) ⓘ

Days	Hours	Minutes	Seconds
14	0	0	0

Move expired messages to the dead-letter subqueue

Move messages that cause filter evaluation exceptions to the dead-letter subqueue

**Figura 5-19** Creación de una nueva suscripción

25. Escriba **10** en el cuadro de texto Recuento máximo de entregas. Este es el número de reintentos para entregar un mensaje antes de mover el mensaje a la cola de mensajes no entregados.
26. Deje las otras propiedades como están.
27. Haga clic en el botón Crear en la parte inferior del panel.

Ahora vas a crear dos aplicaciones de consola. Una aplicación de consola va a publicar mensajes en el tema de Service Bus; la otra aplicación de consola se suscribirá al tema de Service Bus, procesará el mensaje y actualizará el mensaje procesado. Utilice el siguiente procedimiento para crear la aplicación de consola que publica mensajes en el tema de Service Bus:

1. Abra Visual Studio 2019.

2. En la pantalla de bienvenida, haga clic en Crear un proyecto nuevo.
3. Seleccione la plantilla Aplicación de consola (.NET Core).
4. Haga clic en Siguiente.
5. Escriba un nombre de proyecto.
6. Seleccione una ubicación para el proyecto.
7. Haga clic en Crear.
8. Instale el paquete Microsoft.Azure.ServiceBus NuGet
9. Reemplace el contenido del archivo Program.cs con el contenido del [Listado 5-14](#). Recuerde que copió la cadena de conexión necesaria para este código en el paso 20 del ejemplo anterior.

**Listado 5-14** Program.cs

Haga clic aquí para ver la imagen del código

---

```
// C # .NET

utilizando Microsoft.Azure.ServiceBus;

usando el sistema;

usando System.Text;

espacio de nombres <your_project_name>

{

    programa de clase

    {

        const string ServiceBusConnectionString =
            " <su_servicio_bus_conexión_cadena> ";

        const string TopicName = " <your_topic_name> ";

        const int numberOfMessagesToSend = 100;
    }
}
```

```
static ITopicClient topicClient;

static void Main (cadena [] argumentos)
{
    topicClient = new TopicClient
    (ServiceBusConnectionString, TopicName);

    Console.WriteLine ("Presione la tecla ENTER para
salir después de enviar todos los
mensajes. ");
    Console.WriteLine ();

    // Enviar mensajes.

    intentar
    {
        para (var i = 0; i <numberOfMessagesToSend;
i++)
        {
            // Crea un nuevo mensaje para enviar al
tema.

            string messageBody = $ "Mensaje {i}
{DateTime.Now}";

            var mensaje = nuevo mensaje
(Encoding.UTF8.GetBytes (messageBody));
        }
    }
}
```

```

        // Escribe el cuerpo del mensaje en la
        consola.

        Console.WriteLine ($ "Enviando mensaje:
{messageBody} ");

        // Envía el mensaje al tema.

        topicClient.SendAsync (mensaje);

    }

}

catch (excepción de excepción)

{

    Console.WriteLine ($ "{DateTime.Now} ::

Exception: {exception.Message}");

}

Console.ReadKey ();

topicClient.CloseAsync ();

}

}

```

Ahora puede presionar F5 y publicar mensajes en el tema. Una vez que publique los mensajes, debería poder ver un aumento en la columna Recuento de mensajes en la hoja Descripción general de su tema de Service Bus. Los siguientes pasos muestran cómo crear la segunda aplicación de consola que se suscribe al tema y procesa los mensajes en el tema:

1. Abra Visual Studio 2019.

2. En la ventana Inicio, haga clic en Crear un proyecto nuevo.
3. Seleccione la plantilla Aplicación de consola (.NET Core).
4. Haga clic en Siguiente.
5. Escriba un nombre de proyecto.
6. Seleccione una ubicación para el proyecto.
7. Haga clic en Crear.
8. Instale el paquete Microsoft.Azure.ServiceBus NuGet.
9. Reemplace el contenido del archivo Program.cs con el contenido del [Listado 5-15](#).

**Listado 5-15** Program.cs

[Haga clic aquí para ver la imagen del código](#)

---

```
// c # .NET

utilizando Microsoft.Azure.ServiceBus;
usando el sistema;
usando System.Text;
usando System.Threading;
usando System.Threading.Tasks;

espacio de nombres <your_project_name>
{

    programa de clase
    {

        const string ServiceBusConnectionString = "
<your_service_bus_
connection_string> ";

        const string TopicName = " <your_topic_name> ";
    }
}
```

```

        const string SubscriptionName = "
<your_subscription_name> ";

        static ISubscriptionClient subscriptionClient;

        static void Main (cadena [] argumentos)

        {

            subscriptionClient = new SubscriptionClient
(ServiceBusConnectionString,

TopicName,
SubscriptionName);

            Console.WriteLine ("Presione la tecla ENTER para
salir después de recibir todos los

mensajes. ");

```

// Configure las opciones del controlador de mensajes en términos de manejo de excepciones,

número de mensajes simultáneos para entregar,

etc.

```

var messageHandlerOptions = new
MessageHandlerOptions (

```

ExceptionReceivedHandler)

```

{
    // Número máximo de llamadas simultáneas a
    la devolución de llamada

    // ProcessMessagesAsync (), establecido en 1
    para simplificar.

```

```
        // Configúrelo de acuerdo con la cantidad de
        mensajes que desea enviar la aplicación

        // proceso en paralelo.

        MaxConcurrentCalls = 1,

        // Indica si la bomba de mensajes debe
        completarse automáticamente

        // los mensajes después de regresar de la
        devolución de llamada del usuario.

        // False a continuación indica que la
        devolución de llamada del usuario maneja el

        // operación como en ProcessMessagesAsync
        ().

        Autocompletar = falso

    };

    // Registra la función que procesa mensajes.

    subscriptionClient.RegisterMessageHandler
    (ProcessMessagesAsync,
     messageHandlerOptions);

Console.ReadKey ();

subscriptionClient.CloseAsync ();

}
```

```
    static async Task ProcessMessagesAsync (Mensaje de
mensaje, CancellationToken

        simbólico)

    {

        // Procesar el mensaje.

        Console.WriteLine ($ "Mensaje recibido:
SequenceNumber: {mensaje.

            SystemProperties.SequenceNumber} Cuerpo:
{Encoding.UTF8.GetString (message.

            Cuerpo) }");



        // Completa el mensaje para que no se vuelva a
recibir.

        // Esto se puede hacer solo si el
subscriptionClient se crea en

        // Modo ReceiveMode.PeekLock (que es el
predeterminado).

        esperar subscriptionClient.CompleteAsync
(message.SystemProperties.LockToken);

        // Nota: Use el cancellationToken pasado según
sea necesario para determinar si el

        // subscriptionClient ya se ha cerrado.

        // Si subscriptionClient ya se ha cerrado, puede
optar por no

        // llama a CompleteAsync () o AbandonAsync ()
etc.

        // para evitar excepciones innecesarias.

    }
```

```

    // Utilice este controlador para examinar las
    excepciones recibidas en la bomba de mensajes.

        Tarea estática ExceptionReceivedHandler
        (ExceptionReceivedEventArgs

            exceptionReceivedEventArgs)

        {

            Console.WriteLine ($ "El controlador de mensajes
encontró una excepción

            {exceptionReceivedEventArgs.Exception}. ");

            var context =
exceptionReceivedEventArgs.ExceptionReceivedContext;

            Console.WriteLine ("Contexto de excepción para
la resolución de problemas:");

            Console.WriteLine ($ "- Punto final:
{contexto.Punto final}");

            Console.WriteLine ($ "- Ruta de la entidad:
{context.EntityPath}");

            Console.WriteLine ($ "- Acción de ejecución:
{context.Action}");

            return Task.CompletedTask;

        }

    }

}

```

Ahora puede presionar F5 y ejecutar la aplicación de consola. A medida que la aplicación de consola procesa los mensajes del tema, puede ver que el recuento de mensajes de la suscripción está disminuyendo.

*¿Necesita más revisión? Funciones avanzadas de Service Bus*

Puede obtener más información sobre Service Bus en los siguientes artículos:

- **Colas, temas y suscripciones** <https://docs.microsoft.com/en-us/azure/service-bus-messaging/service-bus-queues-topics-subscriptions>
- **Mejoras en el rendimiento de Service Bus** <https://docs.microsoft.com/en-us/azure/service-bus-messaging/service-bus-performance-improvements>
- **Filtros de tema y acciones** <https://docs.microsoft.com/en-us/azure/service-bus-messaging/topic-filters>

## *Implementar soluciones que usen colas de Azure Queue Storage*

Azure Queue Storage es el primer servicio que Microsoft lanzó para administrar las colas de mensajes. Aunque Azure Service Bus y Azure Queue Storage comparten algunas características, como proporcionar servicios de cola de mensajes, Azure Queue Storage es más apropiado cuando su aplicación necesita almacenar más de 80 GB de mensajes en una cola. Otra característica importante del servicio Azure Queue Storage que debe tener en cuenta es que, aunque las colas del servicio funcionan como una pila FIFO (primero en entrar, primero en salir), el orden del mensaje no está garantizado.

### **Nota Azure Queue Storage frente a Azure Service Bus**

Puede revisar una lista completa de diferencias entre estos dos servicios de cola en <https://docs.microsoft.com/en-us/azure/service-bus-messaging/service-bus-azure-and-service-bus-queues-comparado-contrastado>.

El tamaño máximo de un solo mensaje que puede enviar a una cola de Azure es de 64 KB, aunque el tamaño total de la cola puede superar los 80 GB. Solo puede acceder a una cola de Azure mediante la API de REST o mediante el SDK de almacenamiento de Azure de .NET. Estos son los pasos para crear una cuenta de almacenamiento en cola de Azure y una cola para enviar y recibir mensajes:

1. Abra Azure Portal (<https://portal.azure.com>).
2. Haga clic en Crear un recurso en la parte superior del portal.
3. Haga clic en Almacenamiento en la columna Azure Marketplace.
4. Haga clic en Cuenta de almacenamiento: blob, archivo, tabla, cola en la columna Destacados.

5. En la hoja Crear cuenta de almacenamiento, seleccione una suscripción en el menú desplegable Suscripción.
6. Seleccione un grupo de recursos en el menú desplegable Grupo de recursos.
7. Escriba un nombre de cuenta de almacenamiento.
8. Seleccione una ubicación en el menú desplegable Ubicación.
9. Seleccione Almacenamiento con redundancia local en el menú desplegable Replicación.
10. Deje las otras propiedades como están.
11. Haga clic en el botón Revisar + Crear.
12. Haga clic en el botón Crear.
13. Haga clic en el botón Ir a recurso una vez que finalice la implementación.
14. Haga clic en Claves de acceso en el menú de navegación de la hoja de la cuenta de Azure Storage.
15. Copie la cadena de conexión de la sección key1. Necesitará este valor más adelante en esta sección.

En este punto, puede crear colas en su cuenta de Azure Storage mediante Azure Portal. También puede agregar mensajes a la cola mediante Azure Portal. Este enfoque es útil para propósitos de desarrollo o prueba, pero no es adecuado para aplicaciones. Siga los siguientes pasos para crear una aplicación de consola que cree una nueva cola en su cuenta de Azure Storage. La aplicación también envía y lee mensajes de la cola:

1. En la pantalla de bienvenida, haga clic en Crear un proyecto nuevo.
2. Seleccione la plantilla Aplicación de consola (.NET Core).
3. Haga clic en Siguiente.
4. Escriba un nombre de proyecto.
5. Seleccione una ubicación para el proyecto.
6. Haga clic en Crear.
7. Instale los siguientes paquetes de NuGet:
  1. Azure.Almacenamiento.Común

## 2. Azure.Storage.Queue

8. Reemplace el contenido del archivo Program.cs con el contenido del Listado 5-16.

### **Listado 5-16** Program.cs

Haga clic aquí para ver la imagen del código

---

```
// C # .NET

utilizando Azure.Storage.Queues;

utilizando Azure.Storage.Queues.Models;

usando el sistema;

espacio de nombres <your_project_name>

{

    programa de clase

    {

        private const string connectionString = "
<your_storage_account_"

        connection_string> ";

        private const string queueName = "az204queue";

        privado const int maxNumOfMessages = 10;

        static void Main (cadena [] argumentos)

        {

            QueueClient queueClient = new QueueClient
(connectionString, queueName);

            // Crea la cola
```

```
queueClient.CreateIfNotExists ();

// Envío de mensajes a la cola.

para (int i = 0; i <maxNumOfMessages; i ++)

{

    queueClient.SendMessageAsync ($ "Mensaje {i}"
{DateTime.Now} );

}

// Obtener la longitud de la cola

QueueProperties queueProperties =
queueClient.GetProperties ();

¿En t? cachedMessageCount =
queueProperties.ApproximateMessagesCount;

// Leer mensajes de la cola sin eliminar el
mensaje

Console.WriteLine ("Leer mensajes de la cola sin
eliminarlos

de la cola ");

PeekedMessage [] peekedMessages =
queueClient.PeekMessages ((int)
cachedMessageCount);

foreach (PeekedMessage peekedMessage en
peekedMessages)

{
```

```
        Console.WriteLine ($ "Mensaje leído de la
cola: {peekedMessage.

                                                Mensaje de texto}");
```

```
        // Obtener la longitud de la cola
queueProperties = queueClient.GetProperties
();

¿En t? queueLength =
queueProperties.ApproximateMessagesCount;

Console.WriteLine ($ "Longitud actual de la
cola {queueLength}");
```

```
}
```

```
        // Leer mensajes eliminándolos de la cola
Console.WriteLine ("Leer mensaje de la
eliminación de la cola");

QueueMessage [] messages =
queueClient.ReceiveMessages ((int)

cachedMessageCount);

foreach (mensaje QueueMessage en mensajes)

{
    Console.WriteLine ($ "Mensaje leído de la
cola: {mensaje.

                                                Mensaje de texto}");
```

```
    // Necesitas procesar el mensaje en menos de
30 segundos.
```

```

        queueClient.DeleteMessage
(message.MessageId, message.PopReceipt);

        // Obtener la longitud de la cola
queueProperties = queueClient.GetProperties
();

¿En t? queueLength =
queueProperties.ApproximateMessagesCount;

Console.WriteLine ($ "Longitud actual de la
cola {queueLength}");

}

}

```

Presione F5 para ejecutar la aplicación de consola que envía y lee mensajes de la cola. Puede ver cómo se agregan los mensajes a la cola utilizando Azure Portal y navegando a su cuenta de Azure Storage> Colas> az204queue. Debería ver una cola similar a la que se muestra en la [Figura 5-20](#).

ID	MESSAGE TEXT	INSERTION TIME	EXPIRATION TIME	DEQUEUE COUNT	SIZE
350ba362-a307-40ce-b4b3-bd960069e298	Message 8	22/06/2020 0:28:47	22/6/2020 0:28:47	29/6/2020 0:28:47	0
ea3fd335-2009-4fcf-8210-dd3adf2a3e82	Message 4	22/06/2020 0:28:47	22/6/2020 0:28:48	29/6/2020 0:28:48	0
61291a8f-a1fc-420d-a670-7d46451dd18e	Message 0	22/06/2020 0:32:34	22/6/2020 0:32:34	29/6/2020 0:32:34	0
8496df6d-4e04-45a8-890d-ef84519b6379	Message 4	22/06/2020 0:32:34	22/6/2020 0:32:35	29/6/2020 0:32:35	0
2b7f169b-2905-4535-a74a-ebdbb456329f	Message 8	22/06/2020 0:32:34	22/6/2020 0:32:35	29/6/2020 0:32:35	0
e2922180-d74c-48d3-8e82-b7b3ddc7f956	Message 3	22/06/2020 0:32:34	22/6/2020 0:32:35	29/6/2020 0:32:35	0
37a0d855-bc45-4b28-aab4-60f930f18ac2	Message 1	22/06/2020 0:32:34	22/6/2020 0:32:35	29/6/2020 0:32:35	0
4cf62756-fbae-4b76-8864-eb0e5c49edce	Message 2	22/06/2020 0:32:34	22/6/2020 0:32:35	29/6/2020 0:32:35	0
43a53fa6-ac1a-4cf0-b380-44c105300c51	Message 6	22/06/2020 0:32:34	22/6/2020 0:32:35	29/6/2020 0:32:35	0
558cd99-475e-4296-a379-bfa1ad7f184b	Message 7	22/06/2020 0:32:34	22/6/2020 0:32:35	29/6/2020 0:32:35	0
781bd7bf-e5b4-462b-9bc1-406055b9c065	Message 5	22/06/2020 0:32:34	22/6/2020 0:32:35	29/6/2020 0:32:35	0
8c7a0f2c-5ddb-44f2-8294-801b35348b4c	Message 9	22/06/2020 0:32:34	22/6/2020 0:32:35	29/6/2020 0:32:35	0

**Figura 5-20** Creación de una nueva suscripción

*¿Necesita más revisión? Patrón Publicar-Suscribir*

Aunque el servicio Azure Queue Storage no ofrece la capacidad de crear suscripciones a las colas, puede implementar fácilmente el patrón de publicación-suscripción para comunicar

aplicaciones mediante Azure Queue Storage. Puede aprender a implementar este patrón revisando el artículo en <https://docs.microsoft.com/en-us/learn/modules/communicate-between-apps-with-azure-queue-storage/>.

## RESUMEN DEL CAPÍTULO

---

- Azure App Service Logic Apps le permite interconectar diferentes servicios sin necesidad de crear un código específico para la interconexión.
- Los flujos de trabajo de aplicaciones lógicas definen los pasos necesarios para intercambiar información entre aplicaciones.
- Microsoft proporciona conectores para obtener y enviar información desde y hacia diferentes servicios.
- Los disparadores son eventos que se activan en los sistemas de origen.
- Las acciones son cada uno de los pasos que se realizan en un flujo de trabajo.
- Azure Logic Apps proporciona un editor gráfico que facilita el proceso de creación de flujos de trabajo.
- Puede crear sus conectores personalizados para conectar su aplicación con Azure Logic Apps.
- Un conector personalizado es un contenedor para una API REST o SOAP.
- Puede crear conectores personalizados para Azure Logic Apps, Microsoft Flow y Microsoft PowerApps.
- No puede reutilizar conectores personalizados creados para Microsoft Flow o Microsoft PowerApps con Azure Logic Apps.
- Puede exportar sus Logic Apps como plantillas de Azure Resource Manager.
- Puede editar y modificar las plantillas de Logic Apps en Visual Studio.
- El servicio de gestión de API le permite publicar sus API REST o SOAP de back-end utilizando un front-end común y seguro.
- Debe crear suscripciones en el servicio APIM para autenticar el acceso a la API.

- Necesita crear un producto para publicar una API de back-end.
- Puede publicar solo algunas operaciones de sus API de back-end.
- Las políticas APIM le permiten modificar el comportamiento de la puerta de enlace APIM.
- Un evento es un cambio en el estado de una entidad.
- En una arquitectura impulsada por eventos, el editor no tiene la expectativa de que un suscriptor procese o almacene el evento.
- Azure Event Grid es un servicio para implementar arquitecturas basadas en eventos.
- Un tema de Event Grid es un punto final al que un servicio de editor puede enviar eventos.
- Los suscriptores son servicios que leen eventos de un tema de Event Grid.
- Puede configurar varios tipos de servicios como orígenes de eventos o suscriptores de eventos en Azure Event Grid.
- Puede crear eventos personalizados para enviarlos a Event Grid.
- Puede suscribirse a su aplicación personalizada con un tema de Event Grid mediante WebHooks.
- Azure Notification Hub es un servicio que unifica las notificaciones push en plataformas móviles.
- Puede conectar los servicios de notificaciones push de los diferentes fabricantes al Centro de notificaciones de Azure.
- Azure Event Hub es el punto de entrada para las canalizaciones de eventos de Big Data.
- Azure Event Hub está especializado en la ingestión de millones de eventos por segundo con baja latencia.
- Puede usar Azure Event Hub como fuente de eventos para el servicio Event Grid.
- Puede usar AMQP, Kafka y HTTPS para conectarse a Azure Event Hub.

- En una arquitectura basada en mensajes, la aplicación del editor tiene la expectativa de que el suscriptor procese o almacene el mensaje.
- El suscriptor debe cambiar el estado una vez que se procesa el mensaje.
- Un mensaje son datos sin procesar enviados por un editor que deben ser procesados por un suscriptor.
- Los mensajes de Azure Service Bus y Azure Queue son servicios de agente de mensajes.

## EXPERIMENTO MENTAL

---

En este experimento mental, demuestre sus habilidades y conocimiento de los temas cubiertos en este capítulo. Puede encontrar respuestas a este experimento mental en la siguiente sección.

Su organización tiene varias aplicaciones de línea de negocio (LOB) implementadas en entornos locales y de Azure. La información administrada por algunas de estas aplicaciones LOB se superpone entre aplicaciones. Todas sus aplicaciones LOB le permiten utilizar SOAP o REST API para conectarse a las aplicaciones.

Su organización necesita implementar algunos procesos comerciales que requieren compartir información entre las aplicaciones LOB. Responda las siguientes preguntas sobre la conexión de servicios de Azure y aplicaciones de terceros:

1. Debe implementar un proceso empresarial que requiera que una aplicación implementada en Azure comparta información con una aplicación implementada en el centro de datos local de su empresa. ¿Cómo puede implementar este proceso empresarial?
2. Su empresa necesita compartir información administrada por una de las aplicaciones LOB con un socio. La aplicación LOB utiliza una API SOAP para acceder a los datos. Debe asegurarse de que el socio esté autenticado antes de acceder a la información. Su socio debe obtener la información de su aplicación en formato JSON, por lo que también debe asegurarse de que la información proporcionada por su aplicación se publique mediante una API REST. ¿Qué servicio debería utilizar?
3. Una de las aplicaciones LOB de su empresa se está volviendo obsoleta. Su empresa decide desarrollar una nueva aplicación web para

reemplazar la aplicación LOB heredada. Está diseñando la arquitectura de la nueva aplicación web. Necesita implementar una arquitectura desacoplada que necesita procesar millones de eventos por segundo. ¿Qué servicio debería utilizar?

## RESPUESTAS DEL EXPERIMENTO MENTAL

---

Esta sección contiene la solución al experimento mental. Cada respuesta explica por qué la opción de respuesta es correcta.

1. Debe utilizar Azure Logic Apps para implementar el proceso empresarial. Azure Logic Apps le permite crear flujos de trabajo que se pueden usar para implementar su proceso empresarial. Puede conectar Azure Logic Apps con sus aplicaciones LOB locales mediante la puerta de enlace de datos local. También necesita crear conectores personalizados para que Azure Logic Apps pueda trabajar con sus aplicaciones LOB.
2. Debe utilizar el servicio de gestión de API. Este servicio le permite compartir sus API respaldadas con socios y desarrolladores externos de forma segura. Con las políticas APIM, también puede convertir el mensaje XML proporcionado por la API SOAP en documentos JSON necesarios para las API REST. Puede usar Azure AD, autenticación de certificado mutuo o claves de API para autenticar el acceso a la API.
3. Debe utilizar Azure Event Hub. Este servicio está especialmente diseñado para ingerir millones de eventos por segundo. Una vez que el servicio ha ingerido los eventos, reenvía el evento a otros servicios como Azure Storage, Azure Data Lake o Azure Event Grid. El punto crítico aquí para elegir Azure Event Hub en lugar de Event Grid es la cantidad de eventos que deben ingerirse. Otra pista para elegir Event Hub en lugar de Azure Queue Storage o Azure Service Bus es que necesita procesar eventos en lugar de mensajes. Azure Queue o Azure Service Bus son servicios destinados a usarse en arquitecturas basadas en mensajes.

# Índice

## SIMBOLOS

\$ esquema en ARM, [13](#)

## A

control de acceso. Ver [autenticación](#) ; [autorización](#)

claves de acceso para cuentas de almacenamiento, [154](#)

niveles de acceso para Blob de almacenamiento, [117 - 118](#) , [120 - 124](#)

cuenta SAS, [155](#)

creación manera, [157 - 158](#)

URI parámetros, [156 - 157](#)

cuentas (Cosmos DB), creando, [77 - 78](#)

ACI (Azure Container Instancia), que ejecutan las imágenes de contenedores, [26 - 27](#)

ACR (Azure Registro de contenedores), publishing imágenes de contenedores, [24 - 25](#)

conjuntos de acciones, [118](#)

comportamiento

definido, [242](#)

flujos de trabajo y, [244](#)

Directorio Activo. Ver [Azure Active Directory](#)

funciones de actividad, [63 - 64](#) , [68 - 69](#)

activadores de actividad, [64](#)

AllowInsecureHttp, [137](#)

APIM (Administración de API de Azure), [268 - 278](#)

agregando API a, [270 - 272](#)

asociar API y productos, [272 - 273](#)

autenticación para API, [273 - 275](#)

creando instancias, [269 - 273](#)

políticas para API, [275 - 278](#)

niveles de precios, [270](#)  
API. Consulte también [APIM \(Azure API Management\)](#)  
creación de conectores personalizados, [249 - 266](#)  
seleccionando para Cosmos DB, [76 - 78](#)  
Aplicación de configuración, [175 - 183](#)  
tiendas de acceso, [178 - 182](#)  
creación de tiendas, [176](#)  
pares clave-valor, [177](#)  
Servicio de aplicaciones. Ver [Azure App Service](#)  
diagnóstico de aplicaciones, [32 - 33](#)  
Application Insights, [219 - 227](#)  
accediendo, [222 - 223](#)  
agregar a aplicaciones, [221 - 222](#)  
métricas y eventos personalizados, [223 - 225](#)  
enviar mensajes a, [225 - 226 , 227](#)  
visualización de métricas personalizadas, [226](#)  
pruebas y alertas web, [231 - 234](#)  
aplicaciones. Ver [aplicaciones web](#)  
AppSettings.cs  
[Listado 2-8 , 106](#)  
[El listado 2-13 , 110 - 111](#)  
[Listado 2-19 , 121](#)  
[Listado 3-22 , 178 - 179](#)  
Archivo de configuración AppSettings.json  
[Listado 2-7 , 105](#)  
[Listado 2-12 , 110](#)  
[Listado 2-18 , 121](#)  
[Listado 5-6 , 281](#)  
nivel de almacenamiento de archivos, [117 , 120 - 124](#)  
archivado para Blob Storage, [117 - 120](#)  
Plantillas ARM

creando, [12](#) - [21](#)  
personalizado para Logic Apps, [266](#) - [268](#)  
definido, [13](#)  
implementación de aplicaciones web, [35](#)  
autenticación  
para API, [273](#) - [275](#)  
Azure Active Directory, [167](#) - [172](#)  
definido, [127](#)  
para puntos finales, [60](#) - [61](#)  
basado en formulario, [128](#)  
Marco de identidad, [130](#)  
OAuth2, [128](#) - [154](#)  
servidores de autorización, creando, [135](#) - [146](#)  
aplicaciones cliente, creando, [149](#) - [152](#)  
servidores de recursos, la creación, [146](#) - [148](#)  
autenticación, *continuación*  
papeles en, [133](#) - [134](#)  
pruebas, [153](#)  
Pasos de adquisición de tokens, [133](#) - [135](#)  
para aplicaciones web, [131](#) - [132](#)  
ejecutando imágenes de contenedor, [26](#) - [27](#)  
firmas de acceso compartido, [154](#) - [166](#)  
acceso a cuentas de almacenamiento, [163](#) - [166](#)  
SAS cuenta la creación manera, [157](#) - [158](#)  
explicar parámetros SAS Uri, [156](#) - [157](#)  
Servicio SAS creación manera, [159](#) - [161](#)  
parámetros de servicio SAS URI, [158](#) - [159](#)  
Políticas de acceso almacenadas y [161](#)  
tipos de, [155](#)  
delegación de usuario SAS creación manera, [161](#) - [163](#)  
basado en token, [128](#) - [130](#)

autorización  
definido, [127](#)  
RBAC (controles de acceso basados en roles), [172](#) - [174](#)  
servidores de autorización en OAuth2, [133](#) , [135](#) - [146](#)  
AuthorizationCodeProvider, [138](#)  
Código de error AuthorizationPermissionMismatch, [163](#)  
Authorize.cshtml ( [Listado 3-13](#) ), [145](#)  
AuthorizeEndpointPath, [137](#)  
AuthorizeError.cshtml ( [Listado 3-14](#) ), [146](#)  
reglas de ajuste de escala automático para aplicaciones web, [41](#) - [46](#)  
disponibilidad  
de VM (máquinas virtuales), [7](#)  
de aplicaciones web, [231](#) - [234](#)  
estado disponible (arrendamientos), [115](#)  
AzCopy, [102](#)  
Azure Active Directory, [167](#) - [172](#)  
autenticación, [168](#) - [172](#)  
registro de aplicaciones web, [167](#) - [168](#)  
Administración de API de Azure (APIM), [268](#) - [278](#)  
agregando API a, [270](#) - [272](#)  
asociar API y productos, [272](#) - [273](#)  
autenticación para API, [273](#) - [275](#)  
creando instancias, [269](#) - [273](#)  
políticas para API, [275](#) - [278](#)  
niveles de precios, [270](#)  
Azure aplicación de configuración, [175](#) - [183](#)  
tiendas de acceso, [178](#) - [182](#)  
creación de tiendas, [176](#)  
pares clave-valor, [177](#)  
Servicio de aplicaciones de Azure, [27](#)  
reglas de ajuste de escala automático para aplicaciones web, [41](#) - [46](#)

configurar los ajustes de la aplicación web, [38 - 41](#)  
conectando a Notification Hub, [288 - 289](#)  
creación de aplicaciones web, [28 - 32](#)  
implementación de código en aplicaciones web, [35 - 38](#)  
habilitar el registro de diagnóstico, [32 - 35](#)  
Aplicaciones lógicas, [241 - 268](#)  
creando, [242 - 249](#)  
conectores personalizados, [249 - 266](#)  
plantillas personalizadas, [266 - 268](#)  
niveles de precios, [245 , 248](#)  
configuración, [175 - 176](#)  
Azure Blob Storage, [101 - 124](#)  
archivo y retención de datos, [117 - 120](#)  
.NET ejemplo Core, [109 - 114](#)  
caliente, fresco, los niveles de almacenamiento de archivos, [120 - 124](#)  
arrendamientos, [114 - 117](#)  
mover elementos entre el almacenamiento de cuentas /  
contenedores, [102 - 104 , 109 - 114](#)  
parámetros de servicio SAS URI, [158 - 159](#)  
configuración y recuperación de propiedades / metadatos, [104 - 109](#)  
Azure Cache para Redis, [212 - 219](#)  
accediendo, [214 - 218](#)  
reglas de almacenamiento en caché, [209](#)  
creación de base de datos, [213 - 214](#)  
patrones de implementación, [212 - 213](#)  
niveles de precios, [212](#)  
Servicios en la nube de Azure, reglas de ajuste de escala automático, [43](#)  
Azure Recipiente de instancia (ACI), se ejecutan las imágenes de  
contenedores, [26 - 27](#)  
Azure Registro de contenedores (ACR), publishing imágenes de  
contenedores, [24 - 25](#)  
Emulador de Azure Cosmos DB, [78](#)

Funciones duraderas de Azure, [63](#) - [72](#)  
Azure Evento cuadrícula, [279](#) - [287](#)  
temas personalizados, [279](#) - [280](#)  
procesamiento de eventos, [282](#) - [285](#)  
publicación de eventos por temas, [280](#) - [282](#)  
Azure Hub Evento, [291](#) - [298](#)  
Azure Front Door, reglas de almacenamiento en caché, [210](#) - [212](#)  
Funciones de Azure, [46](#)  
Funciones duraderas de Azure, [63](#) - [72](#)  
enlaces de entrada y salida, [46](#) - [52](#)  
disparadores, [52](#) - [63](#)  
versiones de, [55](#)  
Servicio de metadatos de instancia de Azure (IMDS), [192](#)  
Azure Key Vault, [176](#) , [183](#) - [191](#)  
Azure Monitor, [227](#) - [231](#)  
Integración de Azure App Service, [34](#)  
Log Analytics, [229](#) - [231](#)  
Análisis métrico, [227](#) - [228](#)  
Notificación Azure concentradores, [287](#) - [291](#)  
Azure Pipelines, [36](#)  
Azure cola de almacenamiento, [305](#) - [309](#)  
Azure repositorio, [36](#)  
Plantillas de Azure Resource Manager (ARM)  
creando, [12](#) - [21](#)  
definido, [13](#)  
implementación de aplicaciones web, [35](#)  
Azure Service Bus, [299](#) - [305](#)  
Azure Service Fabric, reglas de ajuste de escala automático, [43](#)  
Explorador de Azure Storage, [102](#)  
azureauth.properties ( [Listado 1-1](#) ),[4](#)  
SDK de Azure.Storage.Blobs, [114](#)

## B

Errores de BadRequest, [99](#)  
expresiones vinculantes, [50](#)  
fijaciones  
en funciones, [46 - 52](#)  
desencadenantes versus, [46 - 47](#)  
Almacenamiento de blobs, [101 - 124](#)  
.NET ejemplo Core, [109 - 114](#)  
archivo y retención de datos, [117 - 120](#)  
caliente, fresco, los niveles de almacenamiento de archivos, [120 - 124](#)  
arrendamientos, [114 - 117](#)  
mover elementos entre el almacenamiento de cuentas /  
contenedores, [102 - 104](#), [109 - 114](#)  
parámetros de servicio SAS URI, [158 - 159](#)  
configuración y recuperación de propiedades / metadatos, [104 - 109](#)  
Book.cs ( [Listado 5-1](#) ), [251](#)  
BooksController.cs ( [Listing 5-5](#) ), [256 - 259](#)  
BooksSingleton.cs ( [Listado 5-3](#) ), [253 - 254](#)  
nivel de consistencia de obsolescencia limitada (Cosmos DB), [92](#)  
estado de ruptura (arrendamientos), [115](#)  
estado roto (arrendamientos), [115](#)  
conectores integrados, [244](#)  
procesos de negocio, Logic Apps y, [243](#)

## C

Caché aparte, [213](#)  
almacenamiento en caché  
con Azure Cache para Redis, [212 - 219](#)  
con CDN (redes de entrega de contenido), [202 - 206](#)  
configurar políticas para, [207 - 212](#)  
Callback.cs ( [Listado 5-2](#) ), [252](#)

CallbacksSingleton.cs ( [Ficha de 5-4](#) ), [254 - 256](#)  
API de Cassandra, [76 - 77](#)  
niveles de consistencia, [93](#)  
CDN (redes de entrega de contenido)  
reglas de almacenamiento en caché, [208 - 210](#)  
creando, [202 - 206](#)  
gestión de certificados con la norma API keyvault, [183 - 191](#)  
cambiar notificaciones de feeds en Cosmos DB, [98 - 101](#)  
Propiedad ChangeFeedPolicy, [95](#)  
elementos secundarios, dependencias versus, [21](#)  
funciones de cliente, [64 , 66 - 67](#)  
ID de cliente, [192](#)  
clientes en OAuth2, [133 , 149 - 152](#)  
Clients.cs ( [Listado 3-10](#) ), [141 - 142](#)  
seguridad en la nube, [175 - 196](#)  
Azure aplicación de configuración, [175 - 183](#)  
tiendas de acceso, [178 - 182](#)  
creación de tiendas, [176](#)  
pares clave-valor, [177](#)  
API keyvault, [183 - 191](#)  
identidades gestionadas, [191 - 196](#)  
sincronización en la nube, implementación de aplicaciones web, [35](#)  
Common.cs  
[Listado 2-9 , 106 - 107](#)  
[Listado 2-14 , 111](#)  
[Listado 2-20 , 122](#)  
[Listado 3-23 , 179 - 180](#)  
configurando  
Autenticación API, [273 - 275](#)  
enlaces en funciones, [46 - 52](#)  
políticas de almacenamiento en caché, [207 - 212](#)

perfiles (Azure CDN), [202](#) - [205](#)  
disparadores en funciones, [52](#) - [63](#)  
VM (máquinas virtuales) para acceso remoto, [7](#)- [12](#)  
configuración de la aplicación web, [38](#) - [41](#)  
cadenas de conexión para aplicaciones web, [38](#) , [40](#)  
conexiones entre servicios. Ver [intercambio de información](#)  
conectores  
creación personalizada, [249](#) - [266](#)  
definido, [242](#)  
tipos de, [244](#) - [245](#)  
niveles de consistencia en Cosmos DB, [91](#) - [94](#)  
nivel de consistencia de prefijo consistente (Cosmos DB), [92](#)  
grupos de consumidores, [292](#)  
contenedor de imágenes  
creando con acoplable, [21](#) - [24](#)  
la publicación de Azure Registro de contenedores, [24](#) de - [25](#) de  
corriendo con Azure Container Instancia, [26](#) de - [27](#) de  
contenedores (Cosmos DB), creando, [94](#) - [97](#)  
contenido, tipos de, [201](#)  
almacenamiento en caché de contenido, [213](#)  
Redes de entrega de contenido (CDN)  
reglas de almacenamiento en caché, [208](#) - [210](#)  
creando, [202](#) - [206](#)  
contentVersion en ARM, [13](#)  
despliegue continuo de aplicaciones web, [35](#)  
contribuyentes (RBAC), [173](#)  
nivel de almacenamiento fresco, [117](#) , [120](#) - [124](#)  
Cosmos DB, [75](#) - [101](#)  
niveles de consistencia, [91](#) - [94](#)  
creando cuenta, [77](#) - [78](#)  
creación de contenedores, [94](#) - [97](#)

disparador de operación de datos ( [Listado 1-10](#) ), [53](#)  
emulador para [78](#)  
Ejemplo de API de MongoDB, [90 - 91](#)  
esquemas de partición, [79 - 81](#)  
seleccionar API, [76 - 78](#)  
del lado del servidor de programación, [98 - 101](#)  
Ejemplo de API de SQL, [81 - 90](#)  
conectores personalizados, creación, [249 - 266](#)  
roles personalizados (RBAC), [174](#)  
plantillas personalizadas, creación, [266 - 268](#)  
temas personalizados, crear, [279 - 280](#)  
pruebas de disponibilidad de pistas personalizadas, [231](#)

## D

acceso a datos en Cosmos DB  
Ejemplo de API de MongoDB, [90 - 91](#)  
Ejemplo de API de SQL, [81 - 90](#)  
archivar para Blob almacenamiento de datos, [117 - 120](#)  
el intercambio de datos. Ver [intercambio de información](#)  
operaciones de datos, como tipo de disparador, [53 - 54](#)  
retención de datos para Blob Storage, [117 - 120](#)  
seguridad de datos. Ver [seguridad](#)  
bases de datos  
en Azure Cache para Redis  
accediendo, [214 - 218](#)  
creando, [213 - 214](#)  
en Cosmos DB, [94 - 95](#)  
rendimiento dedicado, [95](#)  
dependencias en ARM, [20 - 21](#)  
desplegando  
código para aplicaciones web, [35 - 38](#)

VM (máquinas virtuales), [2-7](#)  
diagnósticos de implementación, [33](#)  
ranuras de despliegue, [37](#)  
registro detallado de errores, [32](#)  
registro de diagnóstico, habilitación, [32 - 35](#)  
transacciones distribuidas, [213](#)  
Estibador, la creación de imágenes de contenedores, [21 - 24](#)  
Componer ventana acopable, [23 de - 24 de](#)  
Dockerfile ( [Listado 1-5](#) ), [23](#)  
.NET Core  
Blob ejemplo de almacenamiento, [109 - 114](#)  
aplicaciones de consola, creación de máquinas virtuales (máquinas virtuales), [4](#)  
Versiones SDK, [114](#)  
descarga de archivos de registro, [34](#)  
DSA (aceleración dinámica del sitio), [206](#)  
funciones duraderas. Ver [funciones duraderas de Azure](#)  
contenido dinámico, [201](#)

## MI

bordes en Gremlin API, [77](#)  
emuladores, Cosmos DB, [78](#)  
habilitar el registro de diagnóstico, [32 - 35](#)  
Paquete de integración empresarial, [242](#)  
niveles de error para archivos de registro, [33](#)  
Rejilla evento, [279 - 287](#)  
temas personalizados, [279 - 280](#)  
procesamiento de eventos, [282 - 285](#)  
publicación de eventos por temas, [280 - 282](#)  
controladores de eventos, [279](#)  
Hub evento, [291 - 298](#)

fuentes de eventos, [279](#)  
suscripciones a eventos, [279](#)  
soluciones basadas en eventos de intercambio de información, [278 - 298](#)  
Azure Evento cuadrícula, [279 - 287](#)  
temas personalizados, [279 - 280](#)  
procesamiento de eventos, [282 - 285](#)  
publicación de eventos por temas, [280 - 282](#)  
Azure Hub Evento, [291 - 298](#)  
Notificación Azure concentradores, [287 - 291](#)  
eventos  
definido, [279](#)  
en tabiques, [292](#)  
procesamiento, [282 - 285](#)  
publicación  
a Event Hub, [291 - 292](#)  
a los temas (en Event Grid), [280 - 282](#)  
nivel de consistencia eventual (Cosmos DB), [92](#)  
estado vencido (arrendamientos), [115](#)  
extensiones para VM (máquinas virtuales), [3](#)

## F

seguimiento de solicitudes fallidas, [32](#)  
juegos de filtros, [118](#)  
Errores de acceso prohibido, [188](#)  
autenticación basada en formularios, [128](#)  
Puerta principal, reglas de almacenamiento en caché, [210 - 212](#)  
FTP, implementación de aplicaciones web, [35](#)  
teclas de función, [61](#)  
Función1.cs  
[Listado 5-10](#), [283 - 284](#)

## Listado 5-11 , 293 - 294

function.json

vinculaciones ( Listado 1-8 ), 51

activadores del temporizador ( Listado 1-11 ), 56

funciones

en BRAZO, 14 , 19

Funciones de Azure, 46

Funciones duraderas de Azure, 63 - 72

enlaces de entrada y salida, 46 - 52

disparadores, 52 - 63

versiones de, 55

## GRAMO

regiones geográficas para máquinas virtuales (máquinas virtuales), 3

Repositorio de Git, implementación de aplicaciones web, 35

API de Gremlin, 77

grupos (RBAC), 172

## H

validación del controlador, 286

Clase HomeController ( Listado 4-3 ), 223 - 225

HomeController método RedisCache ( Listado 4-1 ), 216 - 217

escala horizontal, 42

claves de host, 61

propiedades de acogida, 62

Particiones "calientes", 80

nivel de almacenamiento en caliente, 117 , 120 - 124

Activadores HTTP, 53 , 58 - 62

conexiones híbridas para aplicaciones web, 29

|

IaaS (infraestructura como servicio), [1](#)  
VM (máquinas virtuales), [2](#)  
configurar para acceso remoto, [7](#)- [12](#)  
creación de plantillas ARM, [12](#) - [21](#)  
la creación de imágenes de contenedores con estibador, [21](#) de - [24](#) de  
aprovisionamiento [2](#)-[7](#)  
la publicación de imágenes de contenedores en Azure Registro de  
contenedores, [24](#) de - [25](#) de  
ejecutan las imágenes de contenedores con Azure Container  
ejemplo, [26](#) - [27](#)  
Marco de identidad, [130](#)  
Métodos IfNotExists en Azure Cosmos DB SDK, [86](#) - [87](#)  
imágenes. Ver [imágenes de contenedores](#)  
IMDS (servicio de metadatos de instancia de Azure), [192](#)  
patrones de implementación para Azure caché para Redis, [212](#) - [213](#)  
Propiedad IndexingPolicy, [95](#)  
intercambio de información  
APIM (Administración de API de Azure), [268](#) - [278](#)  
agregando API a, [270](#) - [272](#)  
asociar API y productos, [272](#) - [273](#)  
autenticación para API, [273](#) - [275](#)  
creando instancias, [269](#) - [273](#)  
políticas para API, [275](#) - [278](#)  
niveles de precios, [270](#)  
soluciones basadas en eventos, [278](#) - [298](#)  
Azure Evento cuadrícula, [279](#) - [287](#)  
Azure Hub Evento, [291](#) - [298](#)  
Notificación Azure concentradores, [287](#) - [291](#)  
Aplicaciones lógicas, [241](#) - [268](#)  
creando, [242](#) - [249](#)

conectores personalizados, [249](#) - [266](#)  
plantillas personalizadas, [266](#) - [268](#)  
niveles de precios, [245](#) , [248](#)  
soluciones basadas en mensajes, [298](#) - [309](#)  
Azure cola de almacenamiento, [305](#) - [309](#)  
Azure Service Bus, [299](#) - [305](#)  
seguridad de información. Ver [seguridad](#)  
Infraestructura como servicio (IaaS), [1](#)  
VM (máquinas virtuales), [2](#)  
configurar para acceso remoto, [7](#)- [12](#)  
creación de plantillas ARM, [12](#) - [21](#)  
la creación de imágenes de contenedores con estibador, [21](#) de - [24](#) de  
aprovisionamiento [2](#)-[7](#)  
la publicación de imágenes de contenedores en Azure Registro de  
contenedores, [24](#) de - [25](#) de  
ejecutan las imágenes de contenedores con Azure Container  
ejemplo, [26](#) - [27](#)  
enlaces de entrada en funciones, [46](#) - [52](#)  
instancias (APIM), creando, [269](#) - [273](#)  
instrumentación con Application Insights, [219](#) - [227](#)  
accediendo, [222](#) - [223](#)  
agregar a aplicaciones, [221](#) - [222](#)  
métricas y eventos personalizados, [223](#) - [225](#)  
enviar mensajes a, [225](#) - [226](#) , [227](#)  
visualización de métricas personalizadas, [226](#)  
Conectores de cuenta de integración, [244](#)  
servicios interconectados. Ver [intercambio de información](#)  
Conectores ISE (entorno de servicio de integración),  
[245](#), [249](#)

J

cola de mensajes y trabajos, [213](#)

Token web JSON (JWT), [129](#) - [130](#)

## K

gestión de claves con la API keyvault, [183](#) - [191](#)  
pares clave-valor, creando, [177](#)  
API keyvault, [183](#) - [191](#)  
Kudu, [36 años](#)

## L

estado arrendado (arrendamientos), [115](#)  
cobro de arrendamientos, [54](#)  
arrienda para Blob Storage, [114](#) - [117](#)  
políticas de gestión del ciclo de vida para Blob Storage, [118](#) - [120](#)  
Servicios de aplicaciones Linux, [32](#)  
listados  
la adición de servidor de autorización OAuth, [136](#) - [137](#)  
agregar información secreta a la página de inicio, [195](#)  
AppSettings.cs, [178](#) - [179](#)  
AppSettings.cs clase C #, [106](#) , [110](#) - [111](#) , [121](#)  
Archivo de configuración AppSettings.json, [105](#) , [110](#) , [121](#)  
Archivo AppSettings.json, [281](#)  
Plantilla ARM para implementar VM, [14](#) - [18](#)  
código de autorización para el delegado de OnCreate, [140](#)  
código de autorización para el delegado de OnReceive, [140](#)  
sección de concesión del código de autorización, [152](#)  
Authorize.cshtml, [145](#)  
AuthorizeError.cshtml, [146](#)  
Código de función de actividad de Azure Durable Functions, [68](#) - [69](#)  
Archivo de configuración JSON de la función de actividad de Azure Durable Functions, [68](#) , [69](#)  
Código de función de cliente de Azure Durable Functions, [66](#)

Código de función del orquestador de Azure Durable Functions , [67](#)  
azureauth.properties, [4](#)  
Libro.cs, [251](#)  
BooksController.cs, [256 - 259](#)  
LibrosSingleton.cs, [253 - 254](#)  
Devolución de llamada.cs, [252](#)  
CallbacksSingleton.cs, [254 - 256](#)  
Clients.cs, [141 - 142](#)  
Common.cs, [179 - 180](#)  
Common.cs C # clase, [106 - 107 , 111 , 122](#)  
configurar un disparador de temporizador en function.json, [56](#)  
configurar el disparador CosmosDB, [53](#)  
configurar el disparador HTTP, [58 - 59](#)  
configurar enlaces de entrada y salida, [48 - 49](#)  
configurar enlaces de entrada y salida en function.json, [51](#)  
Ejemplo de API SQL de Cosmos DB, [81 - 86](#)  
Ejemplo de API de SQL de Cosmos DB: Address.cs, [89](#)  
Ejemplo de API de SQL de Cosmos DB: Device.cs, [89](#)  
Ejemplo de API de SQL de Cosmos DB: Person.cs, [88 - 89](#)  
Procedimiento almacenado de la API SQL de Cosmos DB, [98 - 99](#)  
crear, eliminar, actualizar, y la lectura de artículos clave de bóveda, [185 - 188](#)  
la creación de un servicio de contraseña director, [26 de - 27 de](#)  
Ejemplo de Dockerfile, [23](#)  
archivo de configuración JSON de función de cliente de funciones duraderas, [66 - 67](#)  
archivo de configuración JSON de función de orquestador de funciones duraderas, [67](#)  
ejemplo de documento JSON, [79](#)  
Función1.cs, [283 - 284 , 293 - 294](#)  
conseguir en secreto de la bóveda clave, [194 - 195](#)

HomeController clase, [223](#) - [225](#)  
Método HomeController RedisCache, [216](#) - [217](#)  
método de índice en ManageController.cs, [150](#) - [151](#)  
definición de la política de gestión del ciclo de vida, [119](#)  
MeController.cs, [147](#)  
NewItemCreatedEvent.cs, [281](#) , [283](#)  
OAuthController.cs, [143](#) - [144](#)  
OnGrantClientCredentials delegado, [139](#) - [140](#)  
Delegado de OnGrantResourceOwnerCredentials, [139](#)  
OnValidateClientAuthentication delegado, [138](#) - [139](#)  
Delegado de OnValidateClientRedirectUri, [138](#)  
Paths.cs, [142](#) - [143](#)  
Program.cs, [4-6](#), [162](#) - [163](#) , [180](#) - [181](#) , [297](#) , [302](#) - [305](#) , [307](#) - [308](#)  
Program.cs Clase C #, [107](#) - [108](#) , [112](#) - [113](#) , [122](#) - [124](#)  
Extensión Program.cs, [164](#) - [166](#) , [171](#)  
Program.cs Método principal, [282](#)  
Modificaciones de Program.cs, [8- 11](#) , [115](#) - [116](#)  
Vista de RedisCache, [217](#) - [218](#)  
token de actualización para el delegado de OnCreate, [140](#)  
token de actualización para el delegado OnReceive, [140](#)  
configuración de metadatos definidos por el usuario, [108](#) - [109](#)  
SimpleEventProcessor.cs, [295](#) - [296](#)  
Startup.WebApi.cs, [148](#)  
usando enlaces en JavaScript, [52](#)  
usando el disparador del temporizador con JavaScript, [56](#) - [57](#)  
funciones del entorno local, resolución de problemas, [57](#)  
Log Analytics, [229](#) - [231](#)  
flujos de registro, [34](#)  
Inicio sesión  
registro de diagnóstico, habilitación, [32](#) - [35](#)  
fallas transitorias, [237](#)

Aplicaciones lógicas, [241](#) - [268](#)  
creando, [242](#) - [249](#)  
conectores personalizados, [249](#) - [266](#)  
plantillas personalizadas, [266](#) - [268](#)  
niveles de precios, [245](#) , [248](#)  
particiones lógicas  
definido, [79](#)  
claves de partición, [79](#) - [81](#)  
limitaciones de tamaño, [79](#)

## METRO

Método ManageController.cs índice ( [Listing 3-17](#) ), [150](#) - [151](#)  
conectores gestionados, [242](#) , [244](#)  
identidades gestionadas, [191](#) - [196](#)  
en RBAC, [172](#)  
tipos de, [192](#)  
Identidad de servicio administrado. Ver [identidades administradas](#)  
MeController.cs ( [Listado 3-15](#) ), [147](#)  
soluciones basadas en mensajes de intercambio de  
información, [298](#) - [309](#)  
Azure cola de almacenamiento, [305](#) - [309](#)  
Azure Service Bus, [299](#) - [305](#)  
metadatos en Blob Storage, [104](#) - [109](#)  
Análisis métrico, [227](#) - [228](#)  
SDK de Microsoft.Azure.Storage.Blob, [114](#)  
aplicaciones móviles, las notificaciones push, [287](#) - [291](#)  
API de MongoDB, [77](#)  
niveles de consistencia, [94](#)  
ejemplo de uso, [90](#) - [91](#)  
vigilancia  
con Application Insights, [219](#) - [227](#)

accediendo, [222 - 223](#)  
agregar a aplicaciones, [221 - 222](#)  
métricas y eventos personalizados, [223 - 225](#)  
enviar mensajes a, [225 - 226](#), [227](#)  
visualización de métricas personalizadas, [226](#)  
pruebas y alertas web, [231 - 234](#)  
con Azure Monitor, [227 - 231](#)  
Log Analytics, [229 - 231](#)  
Análisis métrico, [227 - 228](#)  
Mover elementos Blob de almacenamiento, [102 - 104](#), [109 - 114](#)  
autenticación multifactor, [127](#)  
pruebas web de varios pasos, [231](#)

## NORTE

espacios de nombres (en Azure Service Bus)  
creando, [299 - 300](#)  
definido, [299](#)  
nombrar máquinas virtuales (máquinas virtuales), [3](#)  
interfaces de red para máquinas virtuales (máquinas virtuales), [3](#)  
grupos de seguridad de red  
gestión, [12](#)  
para máquinas virtuales (máquinas virtuales), [8](#)  
NewItemCreatedEvent.cs  
[Listado 5-7 , 281](#)  
[Listado 5-9 , 283](#)  
Notificación concentradores, [287 - 291](#)

## O

Autentificación OAuth2, [128 - 154](#)  
servidores de autorización, creando, [135 - 146](#)  
aplicaciones cliente, creando, [149 - 152](#)

servidores de recursos, la creación, [146](#) - [148](#)  
papeles en, [133](#) - [134](#)  
pruebas, [153](#)  
Pasos de adquisición de tokens, [133](#) - [135](#)  
para aplicaciones web, [131](#) - [132](#)  
OAuthController.cs ( [Listado 3-12](#) ), [143](#) - [144](#)  
Código de autorización de delegado de OnCreate ( [Listado 3-6](#) ), [140](#)  
Token de actualización de delegado de OnCreate ( [Listado 3-8](#) ), [140](#)  
OnGrantClientCredentials delegado ( [Listado 3-5](#) ), [139](#) - [140](#)  
Delegado de OnGrantResourceOwnerCredentials ( [Listado 3-4](#) ), [139](#)  
conectores locales, [244](#)  
Código de autorización de delegado OnReceive ( [Listado 3-7](#) ), [140](#)  
Token de actualización de delegado OnReceive ( [Listado 3-9](#) ), [140](#)  
OnValidateClientAuthentication delegado ( [Listing 3-3](#) ), [138](#) - [139](#)  
Delegado de OnValidateClientRedirectUri ( [Listado 3-2](#) ), [138](#)  
sistemas operativos  
para máquinas virtuales (máquinas virtuales), [2](#)  
para aplicaciones web, [28](#)  
optimizar el rendimiento. Ver [optimización del rendimiento](#)  
activadores de orquestación, [64](#)  
funciones de orquestador, [64](#) , [67](#)  
enlaces de salida en funciones, [46](#) - [52](#)  
salidas en ARM, [14](#)  
propietarios (RBAC), [173](#)

## PAG

PaaS (plataforma como servicio), [1](#)  
Funciones de Azure, [46](#)  
Funciones duraderas de Azure, [63](#) - [72](#)  
enlaces de entrada y salida, [46](#) - [52](#)  
disparadores, [52](#) - [63](#)

aplicaciones web, [27](#)  
reglas de ajuste de escala automático, [41 - 46](#)  
configurar ajustes, [38 - 41](#)  
creando, [28 - 32](#)  
implementar código en, [35 - 38](#)  
habilitar el registro de diagnóstico, [32 - 35](#)  
parámetros en ARM, [13 , 18 - 19](#)  
claves de partición, [79 - 81](#)  
esquemas de particionamiento para Cosmos DB, [79 - 81](#)  
particiones, eventos en, [292](#)  
Paths.cs ( [Ficha de 3-11](#) ), [142 - 143](#)  
optimización del rendimiento, [201 - 219](#)  
almacenamiento en caché  
con Azure Cache para Redis, [212 - 219](#)  
configurar políticas para, [207 - 212](#)  
CDN (redes de entrega de contenido)  
reglas de almacenamiento en caché, [208 - 210](#)  
creando, [202 - 206](#)  
fallas transitorias, [234 - 238](#)  
permisos (RBAC), [172](#)  
particiones físicas, [79](#)  
Plataforma como servicio (PaaS), [1](#)  
Funciones de Azure, [46](#)  
Funciones duraderas de Azure, [63 - 72](#)  
enlaces de entrada y salida, [46 - 52](#)  
disparadores, [52 - 63](#)  
aplicaciones web, [27](#)  
reglas de ajuste de escala automático, [41 - 46](#)  
configurar ajustes, [38 - 41](#)  
creando, [28 - 32](#)  
implementar código en, [35 - 38](#)

habilitar el registro de diagnóstico, [32 - 35](#)  
PNS (sistema de notificación de plataforma), [287](#)  
políticas para API, [275 - 278](#)  
Activadores de sondeo, [244](#)  
post-disparadores, [100](#)  
pre-disparadores, [100](#)  
niveles de precios  
para Azure API Management, [270](#)  
para Azure Cache para Redis, [212](#)  
para aplicaciones lógicas, [245 , 248](#)  
para aplicaciones web, [29 , 31 - 32](#)  
ID principales, [192](#)  
procesamiento de eventos, [282 - 285](#)  
condiciones de perfil en autoescalado, [43 - 44](#)  
perfiles (Azure CDN), configurar, [202 - 205](#)  
Program.cs  
extensión  
[Listado 3-20 , 164 - 166](#)  
[Listado 3-21 , 171](#)  
[Listado 1-2 ,4-6](#)  
[Listado 1-3 ,8- 11](#)  
[Listado 2-10 , 107 - 108](#)  
[El listado 2-15 , 112 - 113](#)  
[El listado 2-16 , 115 - 116](#)  
[Listado 2-21 , 122 - 124](#)  
[Listado 3-19 , 162 - 163](#)  
[Listado 3-24 , 180 - 181](#)  
[Listado 5-13 , 297](#)  
[Listado 5-14 , 302 - 303](#)  
[Listado 5-15 , 303 - 305](#)  
[Listado 5-16 , 307 - 308](#)

Método principal ( [Listado 5-8](#) ), [282](#)  
propiedades  
en Blob Storage, [104](#) - [109](#)  
para contenedores Cosmos DB, [95](#) - [96](#)  
en la API de Gremlin, [77](#)  
proveedores, [137](#) - [138](#)  
aprovisionamiento de máquinas virtuales (máquinas virtuales), [2-7](#)  
IP públicas para máquinas virtuales (máquinas virtuales), [8](#)  
publicación  
imágenes de contenedores a Azure Registro de contenedores, [24](#) - [25](#)  
eventos  
a Event Hub, [291](#) - [292](#)  
a los temas (en Event Grid), [280](#) - [282](#)  
notificaciones push, [287](#) - [291](#)  
Empujar gatillos, [244](#)  
Pitón, [102](#)

## Q

colas  
Azure cola de almacenamiento, [305](#) - [309](#)  
Autobús de servicio de Azure, [299](#)  
límites de cuota para máquinas virtuales (máquinas virtuales), [3](#)

## R

RBAC (controles de acceso basados en roles), [172](#) - [174](#)  
lectores (RBAC), [173](#)  
Desencadenantes de recurrencia, [243](#)  
Redis, [212](#) - [219](#)  
accediendo, [214](#) - [218](#)  
reglas de almacenamiento en caché, [209](#)  
creación de base de datos, [213](#) - [214](#)

patrones de implementación, [212](#) - [213](#)  
niveles de precios, [212](#)  
Ver RedisCache ( [Listado 4-2](#) ), [217](#) - [218](#)  
RefreshTokenProvider, [138](#)  
registro de aplicaciones web en Azure Active Directory, [167](#) - [168](#)  
acceso remoto, configuración de máquinas virtuales (máquinas virtuales) para, [7](#)- [12](#)  
grupos de recursos  
en BRAZO, [12](#) , [13](#)  
para máquinas virtuales (máquinas virtuales), [3](#)  
propietarios de recursos en OAuth2, [133](#)  
proveedores de recursos en ARM, [12](#) - [13](#)  
servidores de recursos en OAuth2, [133](#) , [146](#) - [148](#)  
recursos  
en BRAZO, [12](#) , [14](#)  
número de, [3](#)  
relacionados, [3](#)  
retención para Blob Storage, [117](#) - [120](#)  
reintento de operaciones, [234](#) - [238](#)  
asignación de roles (RBAC), [173](#)  
definiciones de roles (RBAC), [173](#)  
controles de acceso basados en roles (RBAC), [172](#) - [174](#)  
reglas de enrutamiento, el almacenamiento en caché en, [210](#) - [211](#)  
ejecutan las imágenes de contenedores con Azure Container  
ejemplo, [26](#) - [27](#)

## S

SAS (firmas de acceso compartido), [154](#) - [166](#)  
acceso a cuentas de almacenamiento, [163](#) - [166](#)  
SAS cuenta la creación manera, [157](#) - [158](#)  
explicar parámetros SAS Uri, [156](#) - [157](#)  
Servicio SAS creación manera, [159](#) - [161](#)

parámetros de servicio SAS URI, [158 - 159](#)  
Políticas de acceso almacenadas y [161](#)  
tipos de, [155](#)  
delegación de usuario SAS creación manera, [161 - 163](#)  
Reglas de [escalamiento](#), [45](#)  
Reglas de escalamiento horizontal, [45](#)  
reglas de escalado para aplicaciones web, [41 - 46](#)  
\$ esquema en ARM, [13](#)  
alcance (RBAC), [173](#)  
gestión secreta con API keyvault, [183 - 191](#)  
seguridad  
autenticación  
servidores de autorización, creando, [135 - 146](#)  
Azure Active Directory, [167 - 172](#)  
aplicaciones cliente, creando, [149 - 152](#)  
definido, [127](#)  
basado en formulario, [128](#)  
Marco de identidad, [130](#)  
OAuth2, [128 - 154](#)  
servidores de recursos, la creación, [146 - 148](#)  
firmas de acceso compartido, [154 - 166](#)  
pruebas, [153](#)  
basado en token, [128 - 130](#)  
autorización  
definido, [127](#)  
RBAC (controles de acceso basados en roles), [172 - 174](#)  
mejores prácticas, almacenamiento de cadenas de conexión, [216](#)  
soluciones de nube, [175 - 196](#)  
Azure aplicación de configuración, [175 - 183](#)  
API keyvault, [183 - 191](#)  
identidades gestionadas, [191 - 196](#)

dimensiones de [127](#)  
directores de seguridad (RBAC), [172](#)  
reglas de seguridad para máquinas virtuales (máquinas virtuales), [8](#)  
seleccionando  
API para Cosmos DB, [76 - 78](#)  
niveles de coherencia para Cosmos DB, [91 - 94](#)  
programación del lado del servidor en el Cosmos DB, [98 - 101](#)  
contraseñas de entidad de servicio, creación, [26 - 27](#)  
directores de servicio (RBAC), [172](#)  
servicio SAS, [155](#)  
creación manera, [159 - 161](#)  
URI parámetros, [158 - 159](#)  
servicios en Docker, [24](#)  
nivel de coherencia de sesión (Cosmos DB), [92](#)  
firmas de acceso compartido (SAS), [154 - 166](#)  
acceso a cuentas de almacenamiento, [163 - 166](#)  
SAS cuenta la creación manera, [157 - 158](#)  
explicar parámetros SAS Uri, [156 - 157](#)  
Servicio SAS creación manera, [159 - 161](#)  
parámetros de servicio SAS URI, [158 - 159](#)  
Políticas de acceso almacenadas y [161](#)  
tipos de, [155](#)  
delegación de usuario SAS creación manera, [161 - 163](#)  
autorización de clave compartida, [154](#)  
rendimiento compartido, [95](#)  
SimpleEventProcessor.cs ( [Listado 5-12](#) ), [295 - 296](#)  
Disparadores de ventana deslizante, [243](#)  
autenticación de redes sociales para aplicaciones web, [130](#)  
API de SQL, [76](#)  
niveles de consistencia, [93](#)  
ejemplo de uso, [81 - 90](#)

Consultas SQL, [88](#)  
SSIS (servicio de integración de SQL Server), [102](#)  
Configuración de SSL para aplicaciones web, [40 - 41](#)  
Startup.WebApi.cs ( [Listado 3-16](#) ), [148](#)  
funciones con estado. Ver [funciones duraderas de Azure](#)  
contenido estático, [201](#)  
cuentas de almacenamiento  
firmas de acceso compartido, [154 - 166](#)  
para máquinas virtuales (máquinas virtuales), [3](#)  
soluciones de almacenamiento  
Almacenamiento de blobs, [101 - 124](#)  
archivo y retención de datos, [117 - 120](#)  
.NET ejemplo Core, [109 - 114](#)  
caliente, fresco, los niveles de almacenamiento de archivos, [120 - 124](#)  
arrendamientos, [114 - 117](#)  
mover elementos entre el almacenamiento de cuentas /  
contenedores, [102 - 104](#) , [109 - 114](#)  
parámetros de servicio SAS URI, [158 - 159](#)  
configuración y recuperación de propiedades / metadatos, [104 - 109](#)  
Cosmos DB, [75 - 101](#)  
niveles de consistencia, [91 - 94](#)  
creando cuenta, [77 - 78](#)  
creación de contenedores, [94 - 97](#)  
emulador para [78](#)  
Ejemplo de API de MongoDB, [90 - 91](#)  
esquemas de partición, [79 - 81](#)  
seleccionar API, [76 - 78](#)  
del lado del servidor de programación, [98 - 101](#)  
Ejemplo de API de SQL, [81 - 90](#)  
Políticas de acceso almacenado, [161](#)  
procedimientos almacenados en Cosmos DB, [98 - 101](#)

tiendas (Configuración de la aplicación de Azure)  
accediendo, [178 - 182](#)  
crear, [176](#)  
almacenamiento de archivos de registro, [33 - 34](#), [35](#)  
fuerte nivel de consistencia (Cosmos DB), [92](#)  
implementaciones de suscripción en ARM, [13](#)  
suscripciones  
para autenticación API, [273 - 275](#)  
suscripciones a eventos, [279](#)  
llaves de partición sintéticas, [80 - 81](#)  
identidades administradas asignadas por el sistema, [192](#)

## T

API de tabla, [76](#)  
niveles de consistencia, [93](#)  
etiquetado de imágenes de contenedores, [24](#)  
telemetría con Application Insights, [219 - 227](#)  
accediendo, [222 - 223](#)  
agregar a aplicaciones, [221 - 222](#)  
métricas y eventos personalizados, [223 - 225](#)  
enviar mensajes a, [225 - 226](#), [227](#)  
visualización de métricas personalizadas, [226](#)  
referencia de plantilla en ARM, [19](#)  
plantillas (ARM)  
creando, [12 - 21](#)  
personalizado para Logic Apps, [266 - 268](#)  
definido, [13](#)  
implementación de aplicaciones web, [35](#)  
probar la autenticación OAuth2, [153](#)  
rendimiento para contenedores Cosmos DB, [95](#)  
temporizadores como tipo de disparo, [53](#), [55 - 57](#)

TimeToLive (TTL) Propiedad, [95](#) , [207](#) - [208](#) , [209](#)  
autenticación basada en token, [128](#) - [130](#)  
SAS cuenta la creación manera, [157](#) - [158](#)  
Servicio SAS creación manera, [159](#) - [161](#)  
Pasos de adquisición de tokens, [133](#) - [135](#)  
delegación de usuario SAS creación manera, [161](#) - [163](#)  
TokenEndpointPath, [137](#)  
temas  
Bus de servicio de Azure  
creando, [299](#) - [300](#)  
definido, [299](#)  
Cuadrícula de eventos  
creación personalizada, [279](#) - [280](#)  
definido, [279](#)  
publicación de eventos a, [280](#) - [282](#)  
fallas transitorias, [234](#) - [238](#)  
desencadenantes  
consolidaciones versus, [46](#) - [47](#)  
en Cosmos DB, [98](#) - [101](#)  
definido, [242](#)  
para funciones duraderas, [64](#)  
en funciones, [52](#) - [63](#)  
flujos de trabajo y, [243](#) - [244](#)  
resolución de problemas de funciones del entorno local, [57](#)  
TTL (TimeToLive) Propiedad, [95](#) , [207](#) - [208](#) , [209](#)

U

Propiedad UniqueKeyPolicy, [95](#) - [96](#)  
Pruebas de ping de URL, [231](#) - [234](#)  
administradores de acceso de usuario (RBAC), [173](#)  
delegación de usuarios SAS, [155](#)

creación manera, [161 - 163](#)  
caché de sesión de usuario, [213](#)  
identidades administradas asignadas por el usuario, [192](#)  
funciones definidas por el usuario en Cosmos DB, [98 - 101](#)  
usuarios (RBAC), [172](#)

## V

variables  
en BRAZO, [14](#)  
en flujos de trabajo, [266](#)  
versiones de Azure Functions, [55](#)  
escala vertical, [42](#)  
vértices en la API de Gremlin, [77](#)  
redes virtuales para máquinas virtuales (máquinas virtuales), [3](#)  
VM (máquinas virtuales), [2](#)  
reglas de ajuste de escala automático, [43](#)  
configurar para acceso remoto, [7- 12](#)  
creación de plantillas ARM, [12 - 21](#)  
la creación de imágenes de contenedores con estibador, [21 de - 24 de](#)  
aprovisionamiento [2-7](#)  
la publicación de imágenes de contenedores en Azure Registro de  
contenedores, [24 de - 25 de](#)  
ejecutan las imágenes de contenedores con Azure Container  
ejemplo, [26 - 27](#)  
Integración de redes virtuales para aplicaciones web, [29](#)

## W

Archivos WAR, implementación de aplicaciones web, [35](#)  
aplicaciones web, [27](#)  
reglas de ajuste de escala automático, [41 - 46](#)  
comprobando disponibilidad, [231 - 234](#)  
seguridad de los datos de configuración, [175 - 183](#)

tiendas de acceso, [178 - 182](#)  
creación de tiendas, [176](#)  
pares clave-valor, [177](#)  
configurar ajustes, [38 - 41](#)  
creando, [28 - 32](#)  
implementar código en, [35 - 38](#)  
habilitar el registro de diagnóstico, [32 - 35](#)  
OAuth2 autenticación para, [131 - 132](#)  
registrándose en Azure Active Directory, [167 - 168](#)  
autenticación de redes sociales para, [130](#)  
diagnóstico del servidor web, [32](#)  
registro del servidor web, [32](#)  
webhooks como tipo de activador, [53 , 58 - 62](#)  
flujos de trabajo  
acciones y, [244](#)  
creando, [245 - 248](#)  
definido, [242](#)  
dispara y, [243 - 244](#)  
variables en, [266](#)

## Z

Archivos ZIP, implementación de aplicaciones web, 35

# Ref. De examen AZ-204 Desarrollo de soluciones para Microsoft Azure

## LISTA DE URL

*Capítulo 1: Desarrollar la infraestructura de Azure como solución informática de servicio*

<https://management.core.windows.net/>

<https://management.azure.com/>

<https://login.windows.net/>

<https://graph.windows.net/>

<https://portal.azure.com>

<https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#>

<https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#>

<https://schema.management.azure.com/schemas/2018-05-01/subscriptionDeploymentTemplate.json#>

<https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#>

<https://schema.management.azure.com/schemas/2015-01-01/deploymentParameters.json#>

<https://shell.azure.com>

[http://\\$SP\\_NAME](http://$SP_NAME)

[http://\\$SP\\_NAME](http://$SP_NAME)

[https://<your\\_app\\_service\\_name>.azurewebsites.net](https://<your_app_service_name>.azurewebsites.net)

<https://docs.microsoft.com/en-us/azure/app-service/containers/app-service-linux-intro>

<https://docs.microsoft.com/en-us/azure/devops/repos/git/creatingrepo>

<https://portal.azure.com>

<https://portal.azure.com>

<https://portal.azure.com>

`http://<your_function_app>.azurewebsites.net/api/<your_function_name>`

`https://<your_function_app>.azurewebsites.net/api/<your_function_name>`

`https://<your_function_app_name>.azurewebsites.net/admin/host/status`

`https://<your_function_app_name>.azurewebsites.net/api/<your_function_name>`

`https://<your_function_app_name>.azurewebsites.net/api/devices/{id: int?}`

<https://www.asp.net/web-api/overview/web-api-routing-and-actions/attribute-routing-in-web-api-2#constraints>

<https://docs.microsoft.com/en-us/azure/app-service/app-service-authentication-how-to#access-user-claims>

`http://localhost:7071/api/orchestrators/OrchestratorFunction`

<https://support.microsoft.com/en-us/help/2721672/microsoft-server-software-support-for-microsoft-azure-virtual-machines>

<https://docs.microsoft.com/en-us/azure/virtual-machines/linux/endorsed-distros>

<https://docs.microsoft.com/en-us/azure/virtual-machines/windows/manage-availability>

<https://docs.microsoft.com/en-us/azure/virtual-network/security-overview>

<https://docs.microsoft.com/en-us/azure/virtual-network/tutorial-restrict-network-access-to-resources>

<https://docs.microsoft.com/en-us/azure/templates/>

<https://docs.microsoft.com/en-us/azure/azure-resource-manager/resource-group-template-functions>

[https://docs.docker.com/develop/develop-images/dockerfile\\_best-practices/](https://docs.docker.com/develop/develop-images/dockerfile_best-practices/)

<https://docs.microsoft.com/en-us/azure/aks/tutorial-kubernetes-prepare-app>

<https://azure.github.io/AppService/2019/11/01/App-Service-Integration-with-Azure-Monitor.html>

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

<https://docs.microsoft.com/en-us/azure/app-service/deploy-zip>

<https://docs.microsoft.com/en-us/azure/app-service/deploy-content-sync>

<https://docs.microsoft.com/en-us/azure/app-service/deploy-local-git>

<https://docs.microsoft.com/en-us/azure/app-service/deploy-staging-slots>

<https://docs.microsoft.com/en-us/azure/app-service/configure-common>

<https://docs.microsoft.com/en-us/azure/azure-monitor/platform/autoscale-best-practices>

<https://docs.microsoft.com/en-us/azure/architecture/best-practices/auto-scaling#related-patterns-and-guidance>

<https://docs.microsoft.com/en-us/azure/azure-functions/functions-triggers-bindings#supported-bindings>

<https://docs.microsoft.com/en-us/azure/azure-functions/install-update-binding-extensions-manual>

<https://docs.microsoft.com/en-us/azure/azure-functions/functions-bindings-expressions-patterns>

<https://docs.microsoft.com/en-us/azure/azure-functions/functions-versions>

<https://docs.microsoft.com/en-us/azure/azure-functions/functions-bindings-http-webhook#trigger---hostjson-properties>

<https://docs.microsoft.com/en-us/azure/azure-functions/functions-scale>

`http://localhost:7071/runtime/webhooks/durabletask/instances`

`http://localhost:7071/runtime/webhooks/durabletask/instances`

`http://localhost:7071/runtime/webhooks/durabletask/instances`

`http://localhost:7071/runtime/webhooks/durabletask/instances`

`http://localhost:7071/runtime/webhooks/durabletask/instances`

<https://docs.microsoft.com/en-us/azure/azure-functions/durable/durable-functions-concepts>

<https://docs.microsoft.com/en-us/azure/active-directory/develop/howto-create-service-principal-portal>

*Capítulo 2: Desarrollar para almacenamiento de Azure*

`http://portal.azure.com`

<http://portal.azure.com>

<http://portal.azure.com>

<https://github.com/Hashnode/mern-starter.git>

<http://localhost:8000>

<http://portal.azure.com>

<http://portal.azure.com>

<https://portal.azure.com>

<https://azure.microsoft.com/en-us/features/storage-explorer/>

<https://docs.microsoft.com/en-us/dotnet/api/microsoft.azure.storage.blob.blobcontainerproperties>

<http://portal.azure.com>

<https://docs.microsoft.com/en-us/azure/cosmos-db/local-emulator>

<https://docs.microsoft.com/en-us/azure/cosmos-db/partition-data#physical-partitions>

<https://docs.microsoft.com/en-us/azure/cosmos-db/partitioning-overview>

<https://docs.microsoft.com/en-us/azure/cosmos-db/sql-api-query-reference>

<https://docs.microsoft.com/en-us/azure/cosmos-db/create-graph-dotnet>

<https://docs.microsoft.com/en-us/azure/cosmos-db/create-cassandra-dotnet>

<https://docs.microsoft.com/en-us/azure/cosmos-db/consistency-levels-tradeoffs>

<https://docs.microsoft.com/en-us/azure/cosmos-db/consistency-levels-tradeoffs>

<https://docs.microsoft.com/en-us/azure/cosmos-db/consistency-levels-across-apis>

<https://docs.microsoft.com/en-us/azure/cosmos-db/databases-containers-items#azure-cosmos-containers>

<https://docs.microsoft.com/en-us/azure/cosmos-db/how-to-time-to-live>

<https://docs.microsoft.com/en-us/azure/cosmos-db/unique-keys>

<https://docs.microsoft.com/en-us/azure/cosmos-db/change-feed-design-patterns>

<https://docs.microsoft.com/en-us/azure/cosmos-db/how-to-write-stored-procedures-triggers-udfs>

<https://docs.microsoft.com/en-us/azure/cosmos-db/how-to-write-javascript-query-api>

<https://docs.microsoft.com/en-us/azure/cosmos-db/how-to-use-stored-procedures-triggers-udfs>

<https://docs.microsoft.com/en-us/azure/cosmos-db/how-to-configure-cosmos-db-trigger-connection-policy>

<https://docs.microsoft.com/en-us/azure/cosmos-db/change-feed>

<https://docs.microsoft.com/en-us/azure/storage/blobs/storage-quickstart-blobs-python>

<https://docs.microsoft.com/en-us/azure/machine-learning/team-data-science-process/move-data-to-azure-blob-using-ssis>

<https://blogs.msdn.microsoft.com/windowsazurestorage/2012/06/12/introducing-asynchronous-cross-account-copy-blob/>

<https://docs.microsoft.com/en-us/rest/api/storageservices/lease-blob>

<https://docs.microsoft.com/en-us/rest/api/storageservices/lease-container>

<https://docs.microsoft.com/en-us/azure/storage/blobs/storage-blob-storage-tiers>

<https://docs.microsoft.com/en-us/azure/storage/blobs/storage-lifecycle-management-concepts>

*Capítulo 3: Implementar la seguridad de Azure*

<https://developers.google.com/identity/sign-in/web/devconsole-project>

[https://localhost:44395 / signin-google](https://localhost:44395/signin-google)

<https://localhost:44317>

<https://localhost:44317>

[https://localhost:44317 / Inicio / Iniciar sesin](https://localhost:44317/Inicio/Iniciar sesin)

[https://localhost:44317 / Administrar](https://localhost:44317/Administrar)

<http://www.w3.org/1999/xhtml>

<http://www.w3.org/1999/xhtml>

<http://portal.azure.com>

<https://azure.microsoft.com/en-us/features/storage-explorer/>

<https://portal.azure.com>

<https:// {cuenta de almacenamiento}.blob.core.windows.net>

<https:// {cuenta de almacenamiento}.blob.core.windows.net>

<https://portal.azure.com>

<https://portal.azure.com>

<https://portal.azure.com>

<https://portal.azure.com>

<https://portal.azure.com>

<https://docs.microsoft.com/en-us/azure/azure-app-configuration/use-key-vault-references-dotnet-core>

<https://portal.azure.com>

`https:// {keyVaultName}.vault.azure.net`

`https:// {keyVaultName}.vault.azure.net`

`https:// {keyvault-name}.vault.azure.net / {object-type} / {object-name} / {object-version}`

<https://portal.azure.com>

<https://docs.microsoft.com/en-in/azure/app-service/app-service-web-get-started-dotnet>

`https:// {keyVaultName}.vault.azure.net / secrets / {secretName}`

<https://portal.azure.com>

<https://docs.microsoft.com/en-us/azure/key-vault/quick-create-portal>

<https://tools.ietf.org/html/rfc6749>

<https://docs.microsoft.com/en-us/aspnet/aspnet/overview/owin-and-katana/owin-oauth-20-authorization-server>

<https://docs.microsoft.com/en-us/rest/api/storageservices/define-stored-access-policy>

<https://docs.microsoft.com/en-us/azure/storage/common/storage-stored-access-policy-define-dotnet>

<https://docs.microsoft.com/en-us/rest/api/storageservices/delegate-access-with-shared-access-signature>

<https://docs.microsoft.com/en-us/azure/active-directory/develop/vs-active-directory-dotnet-what-happened>

<https://docs.microsoft.com/en-us/azure/role-based-access-control/custom-roles>

<https://docs.microsoft.com/en-us/azure/azure-app-configuration/quickstart-feature-flag-aspnet-core>

<https://docs.microsoft.com/en-us/azure/azure-app-configuration/enable-dynamic-configuration-aspnet-core>

<https://docs.microsoft.com/en-us/azure/azure-app-configuration/howto-best-practices>

<https://docs.microsoft.com/en-us/azure/key-vault/about-keys-secrets-and-certificates>

*Capítulo 4: Supervisar, solucionar problemas y optimizar las soluciones de Azure*

[https://<your\\_app\\_service\\_name>.azurewebsites.net](https://<your_app_service_name>.azurewebsites.net)

<https://portal.azure.com>

[https://<nombre\\_de\\_su\\_punto\\_final>.azureedge.net](https://<nombre_de_su_punto_final>.azureedge.net)

[https://<nombre\\_de\\_su\\_punto\\_final>.azureedge.net](https://<nombre_de_su_punto_final>.azureedge.net)

<https://app.contoso.com>

<https://portal.azure.com>

<https://portal.azure.com>

<https://docs.microsoft.com/en-us/azure/frontdoor/quickstart-create-front-door>

<https://portal.azure.com>

<https://portal.azure.com>

<https://portal.azure.com>

<https://portal.azure.com>

<https://visualstudio.microsoft.com/free-developer-offers/>

<https://portal.azure.com>

<https://portal.azure.com>

<https://portal.azure.com>

<https://docs.microsoft.com/en-us/azure/cdn/cdn-how-caching-works>

<https://docs.microsoft.com/en-us/azure/frontdoor/front-door-caching>

<https://stackexchange.github.io/StackExchange.Redis/Basics>

<https://stackexchange.github.io/StackExchange.Redis/Transactions>

<https://stackexchange.github.io/StackExchange.Redis/KeysValues>

<https://docs.microsoft.com/en-us/azure/azure-monitor/app/api-custom-events-metrics>

<https://docs.microsoft.com/en-us/azure/kusto/query/>

<https://docs.microsoft.com/en-us/azure/azure-monitor/log-query/query-language>

<https://docs.microsoft.com/en-us/azure/azure-monitor/platform/alerts-log>

<https://docs.microsoft.com/en-us/azure/architecture/best-practices/transient-faults>

<https://docs.microsoft.com/en-us/aspnet/aspnet/overview/developing-apps-with-windows-azure/building-real-world-cloud-apps-with-windows-azure/transient-fault-manage>

<https://docs.microsoft.com/en-us/azure/architecture/patterns/retry>

<https://docs.microsoft.com/en-us/azure/architecture/patterns/circuit-breaker>

*Capítulo 5: Conectarse y consumir servicios de Azure y servicios de terceros*

<https://docs.microsoft.com/en-us/connectors/connector-reference/>

<https://docs.microsoft.com/en-us/azure/devops/user-guide/sign-up-invite-teammates?view=azure-devops>

<https://portal.azure.com>

<https://github.com/MicrosoftDocs/pipelines-dotnet-core>

<https://github.com/join>

<https://dev.azure.com>

<https://localhost: 44398 / swagger>

[https://<your\\_app\\_service\\_name>.azurewebsites.net](https://<your_app_service_name>.azurewebsites.net)

[https://<your\\_app\\_service\\_name>.azurewebsites.net / swagger](https://<your_app_service_name>.azurewebsites.net / swagger)

<https://github.com/domaindrivendev/Swashbuckle>

<https://github.com/nihau/TREx>

[https://<your\\_app\\_service\\_name>.azurewebsites.net / swagger / docs / v1](https://<your_app_service_name>.azurewebsites.net / swagger / docs / v1)

<https://portal.azure.com>

<https://portal.azure.com>

<https://aka.ms/download-azure-logic-apps-tools-visual-studio-2019>

<https://portal.azure.com>

<https://portal.azure.com>

[https://<your\\_app\\_service\\_name>.azurewebsites.net / swagger / docs / v1](https://<your_app_service_name>.azurewebsites.net/swagger/docs/v1)

<https://fakerestapi.azurewebsites.net>

[https://<your\\_APIM\\_name>.developer.azure-api.net /](https://<your_APIM_name>.developer.azure-api.net/)

<https://portal.azure.com>

<https://portal.azure.com>

<https://portal.azure.com>

<https://az204books.azure-api.net/library/books/>

<https://portal.azure.com>

[https://<Tu\\_Tema\\_Grid\\_Eventos>.<nombre\\_region>-1.eventgrid.azure.net/api/events](https://<Tu_Tema_Grid_Eventos>.<nombre_region>-1.eventgrid.azure.net/api/events)

<https://docs.microsoft.com/en-us/azure/azure-functions/functions-develop-vs#publish-to-azure>

<https://github.com/Azure-Samples/azure-event-grid-viewer>

<http://portal.azure.com>

<https://console.firebaseio.google.com>

<https://portal.azure.com>

<https://docs.microsoft.com/en-us/azure/notification-hubs/ios-sdk-204>

<https://docs.microsoft.com/en-us/azure/notification-hubs/notification-hubs-android-push-notification-google-fcm-get-started>

<https://docs.microsoft.com/en-us/azure/notification-hubs/notification-hubs-windows-store-dotnet-get-started-wns-push-notification>

<https://portal.azure.com>

<https://portal.azure.com>

<https://portal.azure.com>

<https://docs.microsoft.com/en-us/azure/azure-functions/functions-compare-logic-apps-ms-flow-webjobs>

<https://docs.microsoft.com/en-us/azure/logic-apps/logic-apps-pricing>

<https://docs.microsoft.com/en-us/azure/logic-apps/logic-apps-create-variables-store-values>

<https://docs.microsoft.com/en-us/connectors/custom-connectors/>

<https://docs.microsoft.com/en-us/azure/logic-apps/logic-apps-create-azure-resource-manager-templates>

<https://docs.microsoft.com/en-us/azure/logic-apps/logic-apps-deploy-azure-resource-manager-templates>

<https://azure.microsoft.com/en-us/pricing/details/api-management/>

<https://docs.microsoft.com/en-us/azure/api-management/api-management-howto-developer-portal>

<https://azure.microsoft.com/es-es/blog/versions-revisions/>

<https://docs.microsoft.com/en-us/azure/api-management/api-management-access-restriction-policies#RestrictCallerIPs>

<https://docs.microsoft.com/en-us/azure/api-management/api-management-howto-protect-backend-with-aad>

<https://docs.microsoft.com/en-us/azure/api-management/api-management-howto-mutual-certificates>

<https://docs.microsoft.com/en-us/azure/api-management/api-management-error-handling-policies>

<https://docs.microsoft.com/en-us/azure/api-management/set-edit-policies>

<https://docs.microsoft.com/es-es/azure/api-management/api-management-howto-api-inspector>

<https://docs.microsoft.com/en-us/azure/azure-functions/functions-monitoring>

<https://docs.microsoft.com/en-us/azure/event-grid/manage-event-delivery>

<https://docs.microsoft.com/en-us/azure/notification-hubs/notification-hubs-enterprise-push-notification-architecture>

<https://docs.microsoft.com/en-us/azure/storage/blobs/storage-quickstart-blobs-portal#create-a-container>

<https://docs.microsoft.com/en-us/azure/storage/common/storage-account-manage#access-keys>

<https://docs.microsoft.com/en-in/azure/event-hubs/event-hubs-features>

<https://docs.microsoft.com/en-us/azure/service-bus-messaging/service-bus-queues-topics-subscriptions>

<https://docs.microsoft.com/en-us/azure/service-bus-messaging/service-bus-performance-improvements>

<https://docs.microsoft.com/en-us/azure/service-bus-messaging/topic-filters>

<https://docs.microsoft.com/en-us/azure/service-bus-messaging/service-bus-azure-and-service-bus-queues-compared-contrasted>

<https://docs.microsoft.com/en-us/learn/modules/communicate-between-apps-with-azure-queue-storage/>

## Fragmentos de código

Muchos títulos incluyen código de programación o ejemplos de configuración. Para optimizar la presentación de estos elementos, vea el libro electrónico en modo horizontal de una sola columna y ajuste el tamaño de fuente al valor más pequeño. Además de presentar el código y las configuraciones en el formato de texto ajustable, hemos incluido imágenes del código que imitan la presentación que se encuentra en el libro impreso; por lo tanto, cuando el formato reajustable pueda comprometer la presentación del listado de código, verá un enlace "Haga clic aquí para ver la imagen del código". Haga clic en el enlace para ver la imagen del código de fidelidad de impresión. Para volver a la página anterior vista, haga clic en el botón Atrás en su dispositivo o aplicación.

```
dotnet add package Microsoft.Azure.Management.Fluent
```

---

```
subscription=<subscription-id>
client=<client-id>
key=<client-secret>
tenant=<tenant-id>
managementURI=https://management.core.windows.net/
baseURL=https://management.azure.com/
authURL=https://login.windows.net/
graphURL=https://graph.windows.net/
```

---

```
//dotnet core 2.2
using System;
using Microsoft.Azure.Management.Compute.Fluent;
using Microsoft.Azure.Management.Compute.Fluent.Models;
using Microsoft.Azure.Management.Fluent;
using Microsoft.Azure.Management.ResourceManager.Fluent;
using Microsoft.Azure.Management.ResourceManager.Fluent.Core;

namespace ch1_1_1
{
    class Program
    {
        static void Main(string[] args)
        {
            //Create the management client. This will be used for all the operations
            //that we will perform in Azure.
            var credentials = SdkContext.AzureCredentialsFactory
                .FromFile("./azureauth.properties");

            var azure = Azure.Configure()
                .WithLogLevel(HttpLoggingDelegatingHandler.Level.Basic)
                .Authenticate(credentials)
                .WithDefaultSubscription();

            //First of all, we need to create a resource group where we will add all
            //the resources
            // needed for the virtual machine
            var groupName = "az204-ResourceGroup";
            var vmName = "az204VMTesting";
            var location = Region.USWest2;
            var vNetName = "az204VNET";
            var vNetAddress = "172.16.0.0/16";
            var subnetName = "az204Subnet";
            var subnetAddress = "172.16.0.0/24";
            var nicName = "az204NIC";
            var adminUser = "azureadminuser";
            var adminPassword = "Pa$$w0rd!2019";

            Console.WriteLine($"Creating resource group {groupName} ...");
            var resourceGroup = azure.ResourceGroups.Define(groupName)
                .WithRegion(location)
                .Create();

            //Every virtual machine needs to be connected to a virtual network.
            Console.WriteLine($"Creating virtual network {vNetName} ...");
```

```

var network = azure.Networks.Define(vNetName)
    .WithRegion(location)
    .WithExistingResourceGroup(groupName)
    .WithAddressSpace(vNetAddress)
    .WithSubnet(subnetName, subnetAddress)
    .Create();

//Any virtual machine need a network interface for connecting to the
//virtual network
Console.WriteLine($"Creating network interface {nicName} ...");
var nic = azure.NetworkInterfaces.Define(nicName)
    .WithRegion(location)
    .WithExistingResourceGroup(groupName)
    .WithExistingPrimaryNetwork(network)
    .WithSubnet(subnetName)
    .WithPrimaryPrivateIPAddressDynamic()
    .Create();

//Create the virtual machine
Console.WriteLine($"Creating virtual machine {vmName} ...");
azure.VirtualMachines.Define(vmName)
    .WithRegion(location)
    .WithExistingResourceGroup(groupName)
    .WithExistingPrimaryNetworkInterface(nic)
    .WithLatestWindowsImage("MicrosoftWindowsServer", "WindowsServer",
"2012-R2-Datacenter")
    .WithAdminUsername(adminUser)
    .WithAdminPassword(adminPassword)
    .WithComputerName(vmName)
    .WithSize(VirtualMachineSizeTypes.StandardDS2V2)
    .Create();
{

}
}
}

```

---

```
//dotnet core 2.2
using System;
using Microsoft.Azure.Management.Compute.Fluent;
using Microsoft.Azure.Management.Compute.Fluent.Models;
using Microsoft.Azure.Management.Network.Fluent;
using Microsoft.Azure.Management.Fluent;
using Microsoft.Azure.Management.ResourceManager.Fluent;
using Microsoft.Azure.Management.ResourceManager.Fluent.Core;
using Microsoft.Azure.Management.Network.Fluent.Models;

namespace ch1_1_2
{
    class Program
    {
        static void Main(string[] args)
        {
            //Create the management client. This will be used for all the operations
            //that we will perform in Azure.
            var credentials = SdkContext.AzureCredentialsFactory
                .FromFile("./azureauth.properties");
```

```
var azure = Azure.Configure()
    .WithLogLevel(HttpLoggingDelegatingHandler.Level.Basic)
    .Authenticate(credentials)
    .WithDefaultSubscription();

//First of all, we need to create a resource group where we will add all
//the resources
// needed for the virtual machine
var groupName = "az204-ResourceGroup";
var vmName = "az204VMTesting";
var location = Region.USWest2;
var vNetName = "az204VNET";
var vNetAddress = "172.16.0.0/16";
var subnetName = "az204Subnet";
var subnetAddress = "172.16.0.0/24";
var nicName = "az204NIC";
var adminUser = "azureadminuser";
var adminPassword = "Pa$$w0rd!2019";
var publicIPName = "az204publicIP";
var nsgName = "az204VNET-NSG";

Console.WriteLine($"Creating resource group {groupName} ...");
var resourceGroup = azure.ResourceGroups.Define(groupName)
    .WithRegion(location)
    .Create();

//Every virtual machine needs to be connected to a virtual network.
Console.WriteLine($"Creating virtual network {vNetName} ...");
var network = azure.Networks.Define(vNetName)
    .WithRegion(location)
    .WithExistingResourceGroup(groupName)
    .WithAddressSpace(vNetAddress)
    .WithSubnet(subnetName, subnetAddress)
    .Create();

//You need a public IP to be able to connect to the VM from the Internet
Console.WriteLine($"Creating public IP {publicIPName} ...");
var publicIP = azure.PublicIPAddresses.Define(publicIPName)
    .WithRegion(location)
    .WithExistingResourceGroup(groupName)
    .Create();

//You need a network security group for controlling the access to the VM
Console.WriteLine($"Creating Network Security Group {nsgName} ...");
var nsg = azure.NetworkSecurityGroups.Define(nsgName)
```

```

    .WithRegion(location)
    .WithExistingResourceGroup(groupName)
    .Create();

    //You need a security rule for allowing the access to the VM from the
    //Internet
    Console.WriteLine($"Creating a Security Rule for allowing the remote
access");
    nsg.Update()
        .DefineRule("Allow-RDP")
            .AllowInbound()
            .FromAnyAddress()
            .FromAnyPort()
            .ToAnyAddress()
            .ToPort(3389)
            .WithProtocol(SecurityRuleProtocol.Tcp)
            .WithPriority(100)
            .WithDescription("Allow-RDP")
            .Attach()
        .Apply();

    //Any virtual machine needs a network interface for connecting to the
    //virtual network
    Console.WriteLine($"Creating network interface {nicName} ...");
    var nic = azure.NetworkInterfaces.Define(nicName)
        .WithRegion(location)
        .WithExistingResourceGroup(groupName)
        .WithExistingPrimaryNetwork(network)
        .WithSubnet(subnetName)
        .WithPrimaryPrivateIPAddressDynamic()
        .WithExistingPrimaryPublicIPAddress(publicIP)
        .WithExistingNetworkSecurityGroup(nsg)
        .Create();

    //Create the virtual machine
    Console.WriteLine($"Creating virtual machine {vmName} ...");
    azure.VirtualMachines.Define(vmName)
        .WithRegion(location)
        .WithExistingResourceGroup(groupName)
        .WithExistingPrimaryNetworkInterface(nic)
        .WithLatestWindowsImage("MicrosoftWindowsServer", "WindowsServer",
"2012-R2-Datacenter")
            .WithAdminUsername(adminUser)
            .WithAdminPassword(adminPassword)

```

```
        .WithComputerName(vmName)
        .WithSize(VirtualMachineSizeTypes.StandardDS2V2)
        .Create();

    }
}

{
"$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
"contentVersion": "",
"parameters": { },
"variables": { },
"functions": [ ],
"resources": [ ],
"outputs": { }
}
```

---

```
{  
  
"$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",  
    "contentVersion": "1.0.0.0",  
    "parameters": {  
        "virtualNetworks_az204VNET_name": {  
            "defaultValue": "az204demoVNET",  
            "type": "string"  
        },  
        "networkInterfaces_az204NIC_name": {  
            "defaultValue": "az204demoNIC",  
            "type": "string"  
        },  
        "virtualMachines_az204VMTesting_name": {  
            "defaultValue": "az204demoVM",  
            "type": "string"  
        },  
        "subnets_az204Subnet_name": {  
            "defaultValue": "az204demoSubnet",  
            "type": "string"  
        },  
        "virtualMachines_az204VMTesting_id": {  
            "defaultValue": "[concat(parameters('virtualMachines_az204VMTesting_name'),  
                '_OSDisk1_1')]",  
            "type": "string"  
        },  
        "virtualMachines_adminUser": {  
            "defaultValue": "azureadminuser",  
            "type": "string"  
        },  
    }  
}
```

```
"virtualMachines_adminpassword": {
    "defaultValue": "Pa$$w0rd",
    "type": "securestring"
},
},
"variables": {
    "osDiskName": "_OSDisk1_1_39c654d89d88405e968db84b722002d1"
},
"resources": [
{
    "type": "Microsoft.Compute/virtualMachines",
    "name": "[parameters('virtualMachines_az204VMTesting_name')]",
    "apiVersion": "2018-06-01",
    "location": "westus2",
    "tags": {},
    "scale": null,
    "properties": {
        "hardwareProfile": {
            "vmSize": "Standard_DS2_v2"
        },
        "storageProfile": {
            "imageReference": {
                "publisher": "MicrosoftWindowsServer",
                "offer": "WindowsServer",
                "sku": "2012-R2-Datacenter",
                "version": "latest"
            },
            "osDisk": {
                "osType": "Windows",
                "name": "[concat(parameters('virtualMachines_az204VMTesting_name'), variables('osDiskName'))]",
                "createOption": "FromImage",
                "caching": "ReadWrite"
            },
            "dataDisks": []
        },
        "osProfile": {
            "computerName": "[parameters(
                'virtualMachines_az204VMTesting_name')]",
            "adminUsername": "azureadminuser",
            "adminPassword": "Pa$$w0rd",
            "windowsConfiguration": {
                "provisionVMAgent": true,
                "enableAutomaticUpdates": true
            },
            "secrets": [
                {
                    "name": "virtualMachines_adminpassword",
                    "value": "Pa$$w0rd"
                }
            ]
        }
    }
}];
```

```
        "secrets": [],
        "allowExtensionOperations": true
    },
    "networkProfile": {
        "networkInterfaces": [
            {
                "id": "[resourceId('Microsoft.Network/networkInterfaces', parameters('networkInterfaces_az204NIC_name'))]",
                "properties": {
                    "primary": true
                }
            }
        ]
    },
    "dependsOn": [
        "[resourceId('Microsoft.Network/networkInterfaces', parameters('networkInterfaces_az204NIC_name'))]"
    ],
    {
        "type": "Microsoft.Network/networkInterfaces",
        "name": "[parameters('networkInterfaces_az204NIC_name')]",
        "apiVersion": "2018-10-01",
        "location": "westus2",
        "tags": {},
        "scale": null,
        "properties": {
            "ipConfigurations": [
                {
                    "name": "primary",
                    "properties": {
                        "privateIPAllocationMethod": "Dynamic",
                        "subnet": {
                            "id": "[resourceId('Microsoft.Network/virtualNetworks/subnets',
                                parameters('virtualNetworks_az204VNET_name'),
                                parameters('subnets_az204Subnet_name'))]"
                            },
                            "primary": true,
                            "privateIPAddressVersion": "IPv4"
                        }
                    }
                }
            ],
        }
    ],

```

```
        "dnsSettings": {
            "dnsServers": [],
            "appliedDnsServers": []
        },
        "enableAcceleratedNetworking": false,
        "enableIPForwarding": false,
        "primary": true,
        "tapConfigurations": []
    },
    "dependsOn": [
        "[resourceId('Microsoft.Network/virtualNetworks/subnets', parameters('virtualNetworks_az204VNET_name')), parameters('subnets_az204Subnet_name'))]"
    ]
},
{
    "type": "Microsoft.Network/virtualNetworks",
    "name": "[parameters('virtualNetworks_az204VNET_name')]",
    "apiVersion": "2018-10-01",
    "location": "westus2",
    "tags": {},
    "scale": null,
    "properties": {
        "resourceGuid": "145e7bfc-8b00-48cf-8fa1-082448a30bae",
        "addressSpace": {
            "addressPrefixes": [
                "172.16.0.0/16"
            ]
        },
        "dhcpOptions": {
            "dnsServers": []
        },
        "subnets": [
            {
                "name": "[parameters('subnets_az204Subnet_name')]",
                "properties": {
                    "addressPrefix": "172.16.0.0/24"
                }
            }
        ],
        "virtualNetworkPeerings": [],
        "enableDdosProtection": false,
        "enableVmProtection": false
    },
    "dependsOn": []
},
```

```
{  
    "type": "Microsoft.Network/virtualNetworks/subnets",  
  
    "name": "[concat(parameters('virtualNetworks_az204VNET_name'), '/',  
parameters('subnets_az204Subnet_name'))]",  
    "apiVersion": "2018-10-01",  
    "scale": null,  
    "properties": {  
        "addressPrefix": "172.16.0.0/24"  
    },  
    "dependsOn": [  
  
        "[resourceId('Microsoft.Network/virtualNetworks',  
parameters('virtualNetworks_az204VNET_name'))]"  
    ]  
}  
}  
]  
}
```

```
{
    "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentParameters.json#",
    "contentVersion": "1.0.0.0",
    "parameters": {
        "virtualNetworks_az204VNET_name": {
            "value": "az204demoVNET"
        },
        "networkInterfaces_az204NIC_name": {
            "value": "az204demoNIC"
        },
        "virtualMachines_az204VMTesting_name": {
            "value": "az204demoVM"
        },
        "subnets_az204Subnet_name": {
            "value": "az204demoSubnet"
        },
        "virtualMachines_az204VMTesting_id": {
            "value": "[concat(parameters(
                'virtualMachines_az204VMTesting_name'),
                '_OSDisk1_1_39c654d89d88405e968db84b722002d1'))]"
        },
        "virtualMachines_adminUser": {
            "value": "azureadminuser"
        },
        "virtualMachines_adminpassword": {
            "value": "Pa$$w0rd"
        }
    }
}

#!/bin/bash
#Azure CLI template deployment
az group create --name AZ204-ResourceGroup --location "West US"
az group deployment create \
    --name AZ204DemoDeployment \
    --resource-group AZ204-ResourceGroup \
    --template-file az204-template.json \
    --parameters @az204-parameters.json

docker build --tag=<tag_name>[:<version>] <dockerfile_dir>
```

---

```
# Use an official Python runtime as a parent image
FROM python:2.7-slim

# Set the working directory to /app
WORKDIR /app

# Copy the current directory contents into the container at /app
COPY . /app

# Install any needed packages specified in requirements.txt
RUN pip install --trusted-host pypi.python.org -r requirements.txt

# Make port 80 available to the world outside this container
EXPOSE 80

# Define an environment variable
ENV NAME World

# Run app.py when the container launches
CMD ["python", "app.py"]

az acr login ---name <acr_name>
docker tag foobar <acr_name>.azurecr.io/<repository_name>/<image_name>
docker push <acr_name>.azurecr.io/<repository_name>/<image_name>
```

---

```
#!/bin/bash

#Some variable definition useful for the script
ACR_NAME=az204demo
SP_NAME=az204demo_sp
IMAGE_TAG=az204demo.azurecr.io/develop/foobar:latest
RESOURCE_GROUP=AKSdemo-RG
APP_NAME=foobar
APP_DNS_NAME=prueba

#Get the registry ID. You will need this ID for creating the authorization to the
#service principal
ACR_ID=$(az acr show --name $ACR_NAME --query id --output tsv)

#Get the ACR Login server
ACR_SERVER=$(az acr show --name $ACR_NAME --query loginServer --output tsv)

#Get the service principal password. We will grant pull only privileges to the service
#principal
echo "Generating Service Principal password"
SP_PASS=$(az ad sp create-for-rbac --name http://$SP_NAME --scopes $ACR_ID
--role acrpull --query password --output tsv)

#Get the App ID associated to the service principal
SP_ID=$(az ad sp show --id http://$SP_NAME --query appId --output tsv)

echo "Service principal ID: $SP_ID"
echo "Service principal password: $SP_PASS"

#Create the container in the Container Instance service
az container create --resource-group $RESOURCE_GROUP --name $APP_NAME --image
$IMAGE_TAG --cpu 1 --memory 1 --registry-login-server $ACR_SERVER --registry-username
$SP_ID --registry-password $SP_PASS --dns-name-label $APP_DNS_NAME --ports 80

az webapp log download --resource-group <Resource group name> --name <App name>
az webapp log tail --resource-group <Resource group name> --name <App name>
<?php
    $testing_var1 = getenv('APPSETTING_testing-var1')
    $connection_string = getenv('SQLAZURECONNSTR_testing-connsq11')
?>
System.Configuration.ConfigurationManager.AppSettings["testing-var1"]
System.Configuration.ConfigurationManager.ConnectionStrings["testing-connsq11"]
```

```
func extensions install -package Microsoft.Azure.WebJobs.ServiceBus

// C# ASP.NET Core
using System;
using System.IO;
using Microsoft.Azure.WebJobs;
using Microsoft.Extensions.Logging;
using Microsoft.Azure.WebJobs.Extensions.SignalRService;
using Microsoft.Azure.WebJobs.Extensions.EventGrid;
using Microsoft.Azure.EventGrid.Models;
using System.Threading.Tasks;

namespace Company.Functions
{
    public static class BlobTriggerCSharp
    {
        [FunctionName("BlobTriggerCSharp")]
        public static Task Run(
            [EventGridTrigger]EventGridEvent eventGridEvent,
            [Blob("{data.url}", FileAccess.Read, Connection = "ImagesBlobStorage")] Stream
imageBlob,
            [CosmosDB(
                databaseName: "GIS",
                collectionName: "Processed_images",
                ConnectionStringSetting = "CosmosDBConnection")] out dynamic document,
            [SignalR(HubName = "notifications")] IAsyncCollector<SignalRMessage> signalRMessages,
            ILogger log)
        {
            document = new { Description = eventGridEvent.Topic,
id = Guid.NewGuid() };

            log.LogInformation($"C# Blob trigger function Processed event\n Topic: {eventGridEvent.
Topic} \n Subject: {eventGridEvent.Subject} ");
            return signalRMessages.AddAsync(
                new SignalRMessage
                {
                    Target = "newMessage",
                    Arguments = new [] { eventGridEvent.Subject }
                });
        }
    }
}
```

---

```
{  
    "disabled": false,  
    "bindings": [  
        {  
            "name": "eventGridEvent",  
            "type": "eventGridTrigger",  
            "direction": "in"  
        },  
        {  
            "name": "imageBlob",  
            "type": "blob",  
            "connection": "ImagesBlobStorage",  
            "direction": "in",  
            "path": "{data.url}"  
        },  
        {  
            "name": "document",  
            "type": "cosmosDB",  
            "direction": "out",  
            "databaseName": "GIS",  
            "collectionName": "Processed_images",  
            "connectionStringSetting": "CosmosDBConnection",  
            "createIfNotExists": true  
        },  
        {  
            "name": "signalRMessages",  
            "type": "signalR",  
            "direction": "out",  
            "hubName": "notifications"  
        }  
    ]  
}
```

---

```
// NodeJS. Index.js
const uuid = require('uuid/v4');
module.exports = async function (context, eventGridEvent) {
    context.log('JavaScript Event Grid trigger function processed a request.');
    context.log("Subject: " + eventGridEvent.subject);
    context.log("Time: " + eventGridEvent.eventTime);
    context.log("Data: " + JSON.stringify(eventGridEvent.data));

    context.bindings.document = JSON.stringify({
        id: uuid(),
        Description: eventGridEvent.topic
    });

    context.bindings.signalRMessages = [
        {
            "target": "newMessage",
            "arguments": [ eventGridEvent.subject ]
        }
    ];

    context.done();
};
```

---

```
// C# ASP.NET Core
using System.Collections.Generic;
using Microsoft.Azure.Documents;
using Microsoft.Azure.WebJobs;
using Microsoft.Azure.WebJobs.Host;
using Microsoft.Extensions.Logging;
namespace Company.Function
{
    public static class CosmosDBTriggerCSharp
    {
        [FunctionName("CosmosDBTriggerCSharp")]
        public static void Run([CosmosDBTrigger(
            databaseName: "databaseName",
            collectionName: "collectionName",
            ConnectionStringSetting = "AzureWebJobsStorage",
            LeaseCollectionName = "leases",
            CreateLeaseCollectionIfNotExists = true)]IReadOnlyList<Document> input, ILogger log)
        {
            if (input != null && input.Count > 0)
            {
                log.LogInformation("Documents modified " + input.Count);
                log.LogInformation("First document Id " + input[0].Id);
                log.LogInformation("Modified document: " + input[0]);
            }
        }
    }
}

{second} {minute} {hour} {day} {month} {day-of-week}

{
    "disabled": false,
    "bindings": [
        {
            "name": "myTimer",
            "type": "timerTrigger",
            "direction": "in",
            "schedule": "0 */5 * * *",
            "useMonitor": true,
            "runOnStartup": true
        }
    ]
}
```

---

---

```
//NodeJS. Index.js file
module.exports = async function (context, myTimer) {
    var timeStamp = new Date().toISOString();

    if(myTimer.isPastDue)
    {
        context.log('JavaScript is running late!');
    }
    context.log('JavaScript timer trigger Last execution: ', myTimer.ScheduleStatus.Last);
    context.log('JavaScript timer trigger Next execution: ', myTimer.ScheduleStatus.Next);
};


```

[http://<your\\_function\\_app>.azurewebsites.net/api/<your\\_function\\_name>](http://<your_function_app>.azurewebsites.net/api/<your_function_name>)

---

```
// C# ASP.NET Core
using System.Security.Claims;
using System;
using System.IO;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Azure.WebJobs;
using Microsoft.Azure.WebJobs.Extensions.Http;
using Microsoft.AspNetCore.Http;
using Microsoft.Extensions.Logging;
using Newtonsoft.Json;

namespace Company.Function
{
    public static class HttpTriggerCSharp
    {
        [FunctionName("HttpTriggerCSharp")]
        public static async Task<IActionResult> Run(
            [HttpTrigger(AuthorizationLevel.Anonymous, "get", "post", Route = "devices/{id:int?}")] HttpRequest req,
            int? id,
            ILogger log)
        {
            log.LogInformation("C# HTTP trigger function processed a request.");
        }
    }
}
```

```
//We access to the parameter in the address by adding a function parameter
//with the same name
log.LogInformation($"Requesting information for device {id}");

//If you enable Authentication / Authorization at Function App level,
//information
//about the authenticated user is automatically provided in the
//HttpContext
ClaimsPrincipal identities = req.HttpContext.User;
string username = identities.Identity?.Name;

log.LogInformation($"Request made by user {username}");

string name = req.Query["name"];

string requestBody = await new StreamReader(req.Body).ReadToEndAsync();
dynamic data = JsonConvert.DeserializeObject(requestBody);
name = name ?? data?.name;

//We customize the output binding
return name != null

    ? (ActionResult)new JsonResult(new { message = $"Hello, {name}", 
username = username, device = id})
    : new BadRequestObjectResult("Please pass a name on the query string or
in the request body");
}
}
}

https://<your_function_app_name>.azurewebsites.net/admin/host/status
HttpTrigger(AuthorizationLevel.Anonymous...
ClaimsPrincipal identities = req.HttpContext.User;
string username = identities.Identity?.Name;

https://<your_function_app_name>.azurewebsites.net/api/<your_function_name>
https://<your_function_app_name>.azurewebsites.net/api/devices/{id:int?}
HTTP 200 OK in case of Function 1.x runtime
HTTP 204 No Content in case of Function 2.x runtime
func extensions install -p <package_name> -v <package_version>
```

---

```
// NodeJS. HTTPTriggerDurable/index.js
const df = require("durable-functions");

module.exports = async function (context, req) {
    context.log('JavaScript Durable Functions example');
    const client = df.getClient(context);
    const instanceId = await client.startNew(req.params.functionName, undefined,
req.body);

    context.log('Started orchestration with ID = '${instanceId}'.');

    return client.createCheckStatusResponse(context.bindingData.req, instanceId);
};

{
    "disabled": false,
    "bindings": [
        {
            "authLevel": "anonymous",
            "type": "httpTrigger",
            "direction": "in",
            "name": "req",
            "route": "orchestrators/{functionName}",
            "methods": [
                "get",
                "post"
            ]
        },
        {
            "type": "http",
            "direction": "out",
            "name": "$return"
        },
        {
            "name": "context",
            "type": "orchestrationClient",
            "direction": "in"
        }
    ]
}
```

---

---

```
// NodeJS. OrchestratorFunction/index.js
const df = require("durable-functions");

module.exports = df.orchestrator(function*(context) {
    context.log("Starting workflow: chain example");

    const order = yield context.df.callActivity("GetOrder");
    const savedOrder = yield context.df.callActivity("SaveOrder", order);

    return savedOrder;
});

{
    "disabled": false,
    "bindings": [
        {
            "type": "orchestrationTrigger",
            "direction": "in",
            "name": "context"
        }
    ]
}

// NodeJS. GetOrder/index.js
module.exports = async function (context) {
    //Create a mock order for testing
    var order = {
        "id" : Math.floor(Math.random() * 1000),
        "name" : "Customer",
        "date" : new Date().toJSON()
    }
    context.log(order);
    return order;
};
```

---

---

```
{  
    "disabled": false,  
    "bindings": [  
        {  
            "type": "activityTrigger",  
            "direction": "in",  
            "name": "name"  
        }  
    ]  
}  


---



```
// NodeJS. SaveOrder/index.js  
module.exports = async function (context) {  
  
    //Saves the order object received from other activities to a CosmosDB document  
    context.bindings.orderDocument = JSON.stringify({  
        "id": `${context.bindings.order.id}`,  
        "customerName": context.bindings.order.name,  
        "orderDate": context.bindings.order.date,  
        "cosmosDate": new Date().toJSON()  
    });  
    context.done();  
};
```


```

```
{  
    "disabled": false,  
    "bindings": [  
        {  
            "type": "activityTrigger",  
            "direction": "in",  
            "name": "order"  
        },  
        {  
            "name": "orderDocument",  
            "type": "cosmosDB",  
            "databaseName": "ERP_Database",  
            "collectionName": "Orders",  
            "createIfNotExists": true,  
            "connectionStringSetting": "CosmosDBStorage",  
            "direction": "out"  
        }  
    ]  
}  
{  
    "name": "context",  
    "type": "orchestrationClient",  
    "direction": "in"  
}  
  
const client = df.getClient(context);  
  
{  
    "id": "789e7eb945a04ab78e74e9216870af28",  
    "statusQueryGetUri": "http://localhost:7071/runtime/webhooks/durabletask/  
instances...",  
    "sendEventPostUri": "http://localhost:7071/runtime/webhooks/durabletask/  
instances...",  
    "terminatePostUri": "http://localhost:7071/runtime/webhooks/durabletask/  
instances...",  
    "rewindPostUri": "http://localhost:7071/runtime/webhooks/durabletask/  
instances...",  
    "purgeHistoryDeleteUri": "http://localhost:7071/runtime/webhooks/durabletask/  
instances."  
}  
  
const order = yield context.df.callActivity("GetOrder");  
const savedOrder = yield context.df.callActivity("SaveOrder", order);
```

