**Exploración de la plataforma de identidad de Microsoft**

La plataforma de identidad de Microsoft le ayuda a compilar aplicaciones en las que los usuarios y los clientes inician sesión mediante sus identidades de Microsoft o cuentas de redes sociales, y a proporcionar acceso autorizado a sus API o a las API de Microsoft como Microsoft Graph.

está formada por varios componentes:

* **Servicio de autenticación compatible con los estándares OAuth 2.0 y OpenID Connect** que permite a los desarrolladores autenticar varios tipos de identidades entre las que se incluyen:
  + Cuentas profesionales o educativas, aprovisionadas a través de Azure Active Directory
  + Cuentas personales de Microsoft (por ejemplo, Skype, Xbox y Outlook.com)
  + Cuentas sociales o locales, mediante Azure Active Directory B2C
* **Bibliotecas de código abierto**: bibliotecas de autenticación de Microsoft (MSAL) y compatibilidad con cualquier otra biblioteca que cumpla con los estándares.
* **Portal de administración de aplicaciones**: una experiencia de registro y configuración en Azure Portal, junto con las demás funcionalidades de administración de Azure.
* **PowerShell y API de configuración de aplicaciones**: configuración mediante programación de las aplicaciones con Microsoft Graph API y PowerShell, para que pueda automatizar las tareas de DevOps.

**Exploración de las entidades de servicio**

Al registrar la aplicación con Azure Active Directory, está creando una configuración de identidad para la aplicación que le permite integrarse con Azure Active Directory.Hay 2 identidades:

* **Inquilino único:** solo accesible en el inquilino(Inquilino=dominio,lo que dicen es que te prestan ,tu eres el inquilino ,el dominio que te dan es la llave del hotel)
* **Multiinquilino:** accesible en otros inquilinos

Si registra una aplicación en el portal, se crean automáticamente en el inquilino principal un objeto de aplicación (la instancia única global de la aplicación), así como un objeto de entidad de servicio.Tienen un identificador único global para la aplicación.

**Objeto de aplicación**

El objeto de aplicación se usa como plantilla o plano técnico para crear uno o varios objetos de entidad de servicio. La entidad de servicio se crea en todos los inquilinos en los que se utiliza la aplicación. El objeto de aplicación describe tres aspectos de una aplicación: la forma en que el servicio emite tokens para tener acceso a la aplicación, los recursos a los que la aplicación podría necesitar tener acceso y las acciones que puede realizar la aplicación.

**Objeto de entidad de servicio**

La entidad de seguridad define la directiva de acceso y los permisos para el usuario o la aplicación en el inquilino de Azure Active Directory. Esto habilita características básicas como la autenticación del usuario o de la aplicación durante el inicio de sesión y la autorización durante el acceso a los recursos.

Hay tres tipos de entidad de servicio:

* **Aplicación**:  representación local de aplicación ,define lo que la aplicación puede hacer en el inquilino específico, quién puede acceder a la aplicación y a qué recursos tiene acceso la aplicación.
* **Identidad administrada**: proporcionan una identidad para que usen las aplicaciones al conectarse a recursos compatibles con la autenticación de Azure Active Directory. Se les pueden conceder acceso y permisos, pero no se pueden actualizar ni modificar directamente.
* **Heredado:**  representa una aplicación heredada (la que se ha creado antes de que se introdujeran o crearan registros de aplicaciones mediante experiencias heredadas). puede tener credenciales, nombres de entidad de servicio, direcciones URL de respuesta y otras propiedades que un usuario autorizado puede editar, pero no tiene un registro de aplicación asociado, solo se puede usar en el inquilino donde se creó.

**Relación entre los objetos de aplicación y las entidades de servicio**

El objeto de aplicación es la representación *global* de la aplicación para su uso en todos los inquilinos, y la entidad de servicio es la representación *local* para su uso en un inquilino específico.

Un objeto de aplicación tiene lo siguiente:

* Una relación 1:1 con la aplicación de software y su representacion
* una relación uno a varios con sus objetos de entidad de servicio correspondientes.

**Información sobre los permisos y el consentimiento**

Las aplicaciones que se integran en la plataforma de identidad de Microsoft siguen un modelo de autorización que permite a los usuarios y los administradores controlar el modo en que se puede acceder a los datos.

La plataforma de identidad de Microsoft implementa el protocolo de autorización [OAuth 2.0](https://learn.microsoft.com/es-es/azure/active-directory/develop/active-directory-v2-protocols). OAuth 2.0 es un método a través del cual una aplicación de terceros puede acceder a recursos hospedados en Web en nombre de un usuario.

Aquí he llegado con el primer modulo

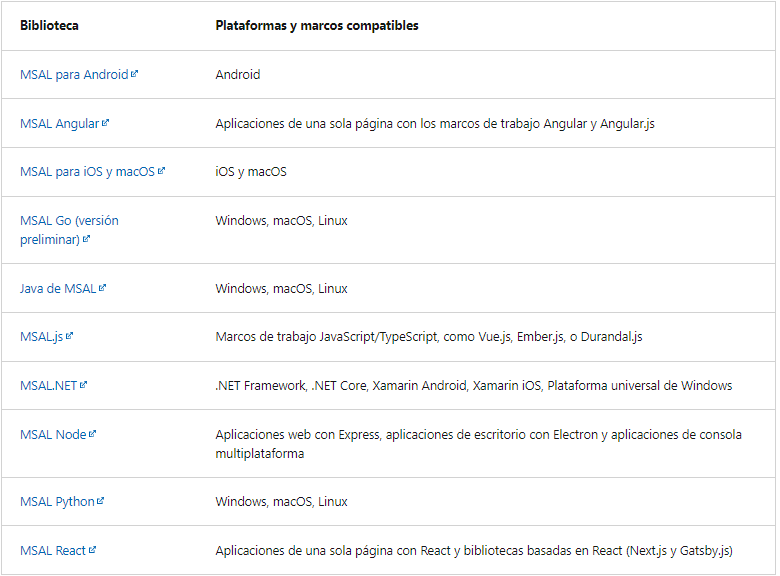
Segundo modulo

**Exploración de la biblioteca de autenticación de Microsoft**

La biblioteca de autenticación de Microsoft (MSAL) se puede usar para proporcionar acceso seguro a Microsoft Graph, otras API de Microsoft, API web de terceros o a su propia API web. MSAL ofrece varias formas de obtener tokens, con una API coherente para muchas plataformas. Usar MSAL brinda las ventajas siguientes:

* No es necesario usar directamente las bibliotecas de OAuth ni el código en el protocolo en la aplicación.
* Adquiere tokens en nombre de un usuario o en nombre de una aplicación (cuando se aplica a la plataforma).
* Mantiene una caché de tokens y actualiza los tokens de forma automática cuando están próximos a expirar. No es necesario que el usuario controle la expiración de los tokens.
* Le ayuda a especificar qué audiencia quiere que inicie sesión en la aplicación.
* Lo ayuda a configurar la aplicación a partir de archivos de configuración.
* Lo ayuda a solucionar problemas de la aplicación mediante la exposición de excepciones, registros y telemetría que requieren acción.

Con MSAL, es posible adquirir un token desde muchos tipos de aplicación:



**Flujos de autenticación**

En la tabla siguiente se muestran algunos de los distintos flujos de autenticación que proporciona la biblioteca de autenticación de Microsoft (MSAL).



**Aplicaciones cliente públicas y confidenciales**

Hay varios tipos de aplicaciones que pueden adquirir tokens de seguridad :

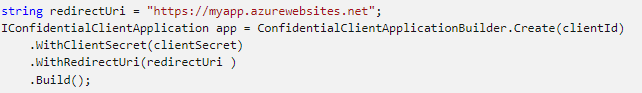
* **Aplicaciones cliente públicas**: aplicaciones que se ejecutan en dispositivos o equipos de escritorio o en un explorador web. No son de confianza para mantener de manera segura secretos de aplicación, por lo que solo tienen acceso a API web en nombre del usuario (solo admiten flujos de cliente públicos). Los clientes públicos no pueden contener secretos de tiempo de configuración, por lo que no tienen secretos de cliente.
* **Aplicaciones cliente confidenciales**: aplicaciones que se ejecutan en servidores (aplicaciones web, aplicaciones de API web o incluso aplicaciones de servicio o demonio). Se consideran de acceso difícil y, por ese motivo, pueden mantener un secreto de aplicación. Los clientes confidenciales pueden contener secretos de tiempo de configuración. Cada instancia del cliente tiene una configuración distinta (incluidos el Id. y el secreto de cliente).

**Inicialización de aplicaciones cliente públicas y confidenciales desde código**

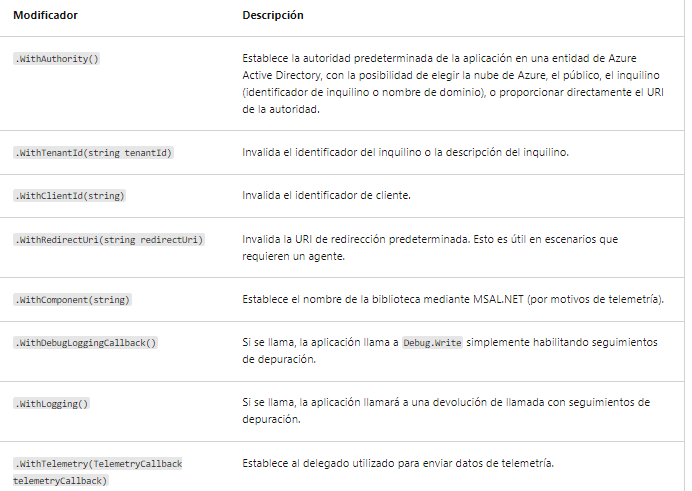
El código siguiente crea una instancia de una aplicación cliente pública e inicia la sesión de los usuarios en la nube pública de Microsoft Azure con sus cuentas profesionales o educativas o bien con sus cuentas Microsoft personales.

IPublicClientApplication app = PublicClientApplicationBuilder.Create(clientId).Build();

En la misma manera, el código siguiente crea una instancia de una aplicación confidencial (una aplicación web ubicada en https://myapp.azurewebsites.net) y controla los tokens de los usuarios en la nube pública de Microsoft Azure con sus cuentas profesionales o educativas o bien con sus cuentas Microsoft personales. La aplicación se identifica con el proveedor de identidades compartiendo un secreto de cliente:

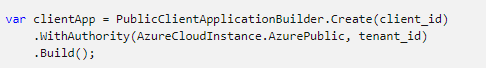


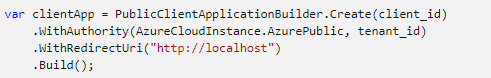
**Modificadores de apps publicas y confidenciales**

****

En los fragmentos de código que usan generadores de aplicaciones, pueden aplicarse como modificadores varios métodos .With (por ejemplo, .WithAuthority y .WithRedirectUri).

* Ejemplo WithAuthority



* Ejemplo.WithRedirectUri:  

**Modificadores específicos de las aplicaciones cliente confidenciales**

****

**Modulo 3**

**Descubrimiento de firmas de acceso compartido**

Una firma de acceso compartido (SAS) es un URI firmado que apunta a uno o varios recursos de almacenamiento e incluye un token que contiene un conjunto especial de parámetros de consulta, que indica cómo puede el cliente tener acceso a los recursos(es un enlace para compartir temporal). Uno de los parámetros de consulta, la firma, se construye a partir de parámetros SAS y se firma con la clave que se usó para crear la SAS. Azure Storage utiliza esta firma para autorizar el acceso al recurso de almacenamiento.

**Tipos de firmas de acceso compartido**

Azure Storage admite tres tipos de firmas de acceso compartido:

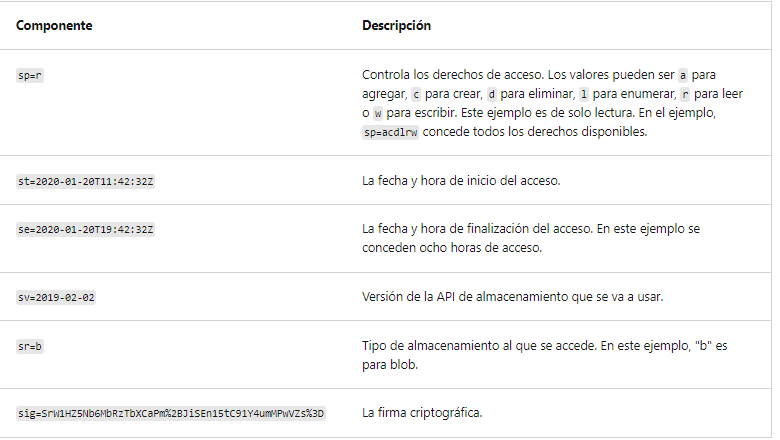
* **SAS de delegación de usuarios(solo blob storahe)**: una SAS de delegación de usuarios se protege con las credenciales de Azure Active Directory y también por los permisos especificados para la SAS. Una SAS de delegación de usuarios solo se aplica a Blob Storage.
* **SAS de servicio(solo servicios)**: una SAS de servicio se protege con la clave de cuenta de almacenamiento. Una SAS de servicio delega el acceso a un recurso en los servicios de Azure Storage siguientes: Blob Storage, Queue Storage, Table Storage o Azure Files.
* **SAS de cuenta(solo cuenta)**: una SAS de cuenta se protege con la clave de cuenta de almacenamiento. SAS de cuenta delega el acceso a los recursos en uno o varios de los servicios de almacenamiento. Todas las operaciones disponibles con una SAS de servicio o delegación de usuarios están también disponibles con una SAS de cuenta.

**Funcionamiento de las firmas de acceso compartido**

Cuando se usa una SAS para acceder a los datos almacenados en Azure Storage, se necesitan dos componentes. El primero es un identificador URI del recurso al que se quiere acceder. El segundo elemento es un token de SAS que ha creado para autorizar el acceso a ese recurso. puede distinguir el URI del token de SAS de la manera siguiente:

* **URI:**https://medicalrecords.blob.core.windows.net/patient-images/patient-116139-nq8z7f.jpg?
* **Token de SAS:**sp=r&st=2020-01-20T11:42:32Z&se=2020-01-20T19:42:32Z&spr=https&sv=2019-02-02&sr=b&sig=SrW1HZ5Nb6MbRzTbXCaPm%2BJiSEn15tC91Y4umMPwVZs%3D

El propio token SAS consta de varios componentes.

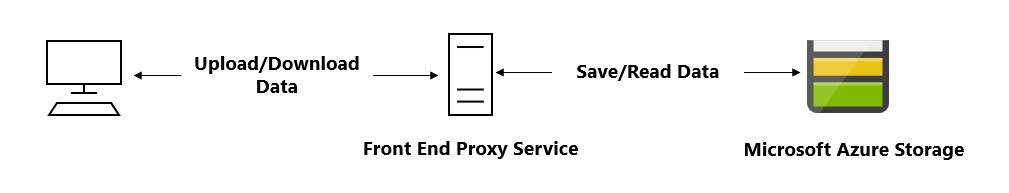
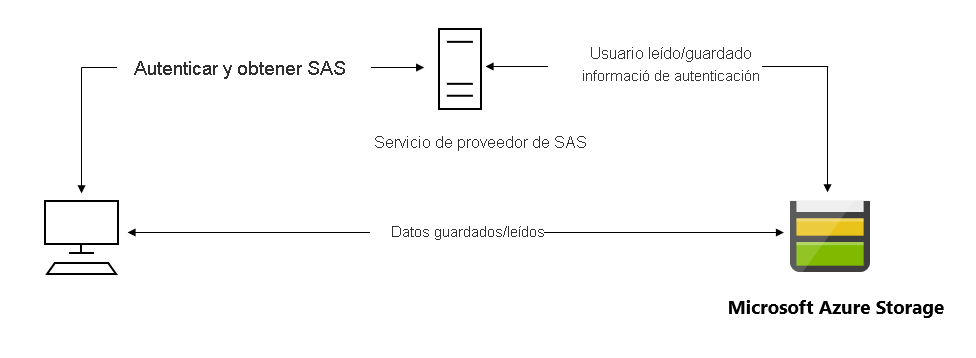


Para reducir los posibles riesgos del uso de una SAS, Microsoft proporciona varias instrucciones:

* Para distribuir de forma segura una SAS y evitar ataques de tipo "Man in the Middle", use siempre HTTPS.
* La SAS más segura es una SAS de delegación de usuarios. Úsela siempre que sea posible, ya que elimina la necesidad de almacenar la clave de la cuenta de almacenamiento en el código. Debe usar Azure Active Directory para administrar las credenciales. Es posible que esta opción no se pueda usar en la solución.
* Intente establecer la hora de expiración en el valor más pequeño que sea útil. Si una clave de SAS se pone en peligro, solo se podrá aprovechar durante un breve período de tiempo.
* Aplique la regla de privilegios mínimos necesarios. Conceda solo el acceso necesario. Por ejemplo, en la aplicación, el acceso de solo lectura es suficiente.
* Hay algunas situaciones en las que una SAS no es la solución correcta. Cuando el uso de una SAS suponga un riesgo inaceptable, cree un servicio de nivel intermedio para administrar los usuarios y su acceso al almacenamiento.

**Determinación de cuándo usar firmas de acceso compartido**

Use una SAS cuando quiera proporcionar acceso seguro a los recursos de la cuenta de almacenamiento a cualquier cliente que no tenga permisos para acceder a esos recursos de otra manera los usuarios leen y escriben sus propios datos en la cuenta de almacenamiento. Existen dos patrones de diseño típicos en los escenarios en los que una cuenta de almacenamiento guarda datos de usuario:

* Los clientes cargan y descargan datos mediante un servicio de proxy front-end que realiza la autenticación. Este servicio de proxy front-end cuenta con la ventaja de permitir la validación de reglas de negocio, pero para grandes cantidades de datos o transacciones de gran volumen, la creación de un servicio que pueda escalarse para satisfacer la demanda puede ser complicada o costosa.
* Un servicio ligero realiza la autenticación del cliente según sea necesario y, luego, genera una SAS. Una vez que la aplicación cliente recibe la SAS, puede obtener acceso a los recursos de la cuenta de almacenamiento directamente con los permisos definidos por la SAS y para el intervalo permitido por ella. La SAS mitiga la necesidad de enrutar todos los datos a través del servicio de proxy front-end.

Muchos servicios reales pueden usar una combinación de estos dos enfoques. Por ejemplo, se pueden procesar y validar algunos datos mediante el proxy de front-end, mientras que otros datos se guardan o leen directamente mediante SAS.

Además, en ciertos escenarios, se requiere una SAS para autorizar el acceso al objeto de origen en las operaciones de copia:

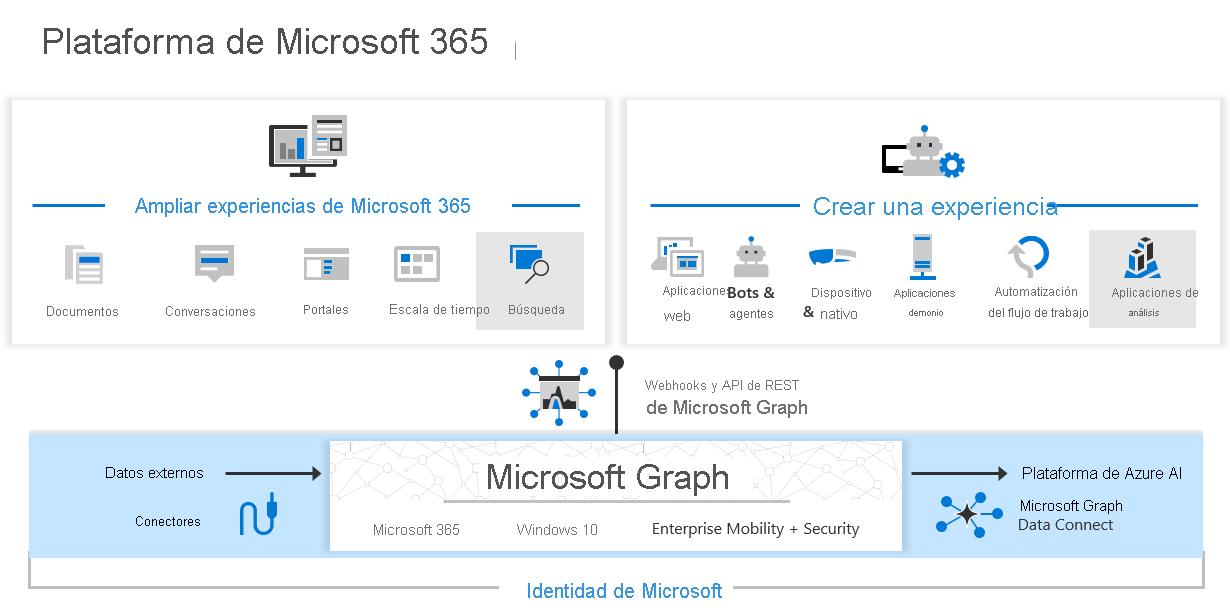
* Cuando copia un blob en otro blob que reside en una cuenta de almacenamiento diferente, debe usar una SAS para autorizar el acceso al blob de origen. Si lo desea, también puede usar una SAS para autorizar el acceso al blob de destino.
* Cuando copia un archivo en otro archivo que se encuentra en una cuenta de almacenamiento diferente, debe usar una SAS para autorizar el acceso al archivo de origen. Si lo desea, también puede usar una SAS para autorizar el acceso al archivo de destino.
* Si va a copiar un blob en un archivo, o un archivo en un blob, tiene que usar una SAS para autorizar el acceso al objeto de origen, incluso si los objetos de origen y destino residen dentro la misma cuenta de almacenamiento.

**Exploración de directivas de acceso almacenadas**

ME FALTA EL FINAL DEL MODULO 3

MODULO 4

Microsoft Graph es la puerta de enlace a los datos y la inteligencia en Microsoft 365. Proporciona un modelo de programación unificado que puede usar para acceder a la ingente cantidad de datos en Microsoft 365, Windows 10 y Enterprise Mobility + Security.



En la plataforma Microsoft 365, tres componentes principales facilitan el acceso y el flujo de datos:

* Microsoft Graph API ofrece un único punto de conexión, https://graph.microsoft.com. Puede usar las API REST o los SDK para acceder al punto de conexión. Microsoft Graph también incluye un conjunto eficaz de servicios que permiten administrar la identidad, el acceso, el cumplimiento y la seguridad de usuarios y dispositivos, y ayudan a proteger a las organizaciones frente a la filtración o la pérdida de datos.
* Los [conectores de Microsoft Graph](https://learn.microsoft.com/es-es/microsoftsearch/connectors-overview) funcionan en la dirección de entrada y **entregan datos externos en la nube de Microsoft en aplicaciones y servicios de Microsoft Graph**, con el fin de mejorar las experiencias de Microsoft 365, como Búsqueda de Microsoft. Existen conectores para muchos orígenes de datos de uso frecuente, como Box, Google Drive, Jira y Salesforce.
* [Microsoft Graph Data Connect](https://learn.microsoft.com/es-es/graph/overview#access-microsoft-graph-data-at-scale-using-microsoft-graph-data-connect) proporciona un conjunto de herramientas para simplificar la **entrega de datos de Microsoft Graph en almacenes de datos de Azure conocidos** de forma segura y escalable. Los datos almacenados en caché funcionan como orígenes de datos para las herramientas de desarrollo de Azure, que puede usar para compilar aplicaciones inteligentes.

**Consulta de Microsoft Graph mediante REST**

Microsoft Graph es una API web de RESTful que le permite acceder a los recursos del servicio Microsoft Cloud. Después de registrar la aplicación y obtener los tokens de autenticación para un usuario o servicio, puede realizar solicitudes a Microsoft Graph API.

Microsoft Graph API define la mayoría de sus recursos, métodos y enumeraciones en el espacio de nombres de OData, microsoft.graph, en los [metadatos de Microsoft Graph](https://learn.microsoft.com/es-es/graph/traverse-the-graph#microsoft-graph-api-metadata). Hay varios conjuntos de API definidos en sus subespacios de nombres, como la [API de registros de llamadas](https://learn.microsoft.com/es-es/graph/api/resources/callrecords-api-overview) que define recursos como [callRecord](https://learn.microsoft.com/es-es/graph/api/resources/callrecords-callrecord) en microsoft.graph.callRecords.

A menos que se especifique explícitamente en el tema correspondiente, se supone que los tipos, los métodos y las enumeraciones forman parte del espacio de nombres microsoft.graph.

**Llamada a un método de API REST**

Para leer o escribir en un recurso, como un usuario o un mensaje de correo electrónico, construya una solicitud similar a la siguiente:

{HTTP method} https://graph.microsoft.com/{version}/{resource}?{query-parameters}

Los componentes de una solicitud incluyen:

* {HTTP method}: el método HTTP usado en la solicitud a Microsoft Graph.
* {version}: la versión de Microsoft Graph API que usa la aplicación.
* {resource}: el recurso de Microsoft Graph al que se hace referencia.
* {query-parameters}: opciones de consulta de OData(authentication data) o parámetros del método REST opcionales que personalizan la respuesta.

Después de realizar una solicitud, se devuelve una respuesta que incluye:

* Código de estado: código de estado HTTP que indica acierto o fallo.
* Mensaje de respuesta: los datos que solicitó o el resultado de la operación. El mensaje de respuesta puede estar vacío en el caso de algunas operaciones.
* nextLink: si la solicitud devuelve una gran cantidad de datos, debe paginarlos mediante la dirección URL devuelta en @odata.nextLink.

PARA EL GRAPH

var messages = await graphClient.Me.Messages.Request().Select(m => new { m.Subject,m.Sender}).Filter("<filtercondition>").OrderBy("receivedDateTime").GetAsync();

RUTA 7

MODULO 1

El servicio Azure Key Vault(es la caja fuerte de la caja) admite dos tipos de contenedores: almacenes y grupos de módulos de seguridad de hardware administrados (HSM). Los almacenes permiten almacenar software y claves, secretos y certificados respaldados por HSM. Los grupos HSM administrados solo admiten claves respaldadas por HSM.

Azure Key Vault ayuda a solucionar los problemas siguientes:

* **Administración de secretos**: Azure Key Vault se puede utilizar para almacenar de forma segura y controlar de manera estricta el acceso a los tokens, contraseñas, certificados, claves de API y otros secretos.
* **Administración de claves**: también se puede usar Azure Key Vault como solución de administración de claves. Azure Key Vault facilita la creación y control de las claves de cifrado utilizadas para cifrar los datos.
* **Administración de certificados:** Azure Key Vault también es un servicio que permite aprovisionar, administrar e implementar fácilmente certificados de la Capa de sockets seguros y la Seguridad de la capa de transporte (SSL y TLS) públicos y privados para su uso con Azure y los recursos internos conectados.

Azure Key Vault tiene dos niveles de servicio: Estándar, que realiza el cifrado con una clave de software, y Prémium, que incluye claves protegidas mediante un módulo de seguridad de hardware (HSM).

**ventajas de usar Azure Key Vault**

* **Secretos de aplicación centralizados**: la centralización del almacenamiento de secretos de Azure Key Vault permite controlar su distribución.En lugar de almacenar la cadena de conexión en el código de la aplicación, puede almacenarla de forma segura en Key Vault. Las aplicaciones pueden acceder de manera protegida a la información que necesitan a través de URI. Estos URI permiten que las aplicaciones recuperen versiones específicas de un secreto.
* **Almacenamiento seguro de secretos y claves**: el acceso a un almacén de claves requiere la autenticación y autorización adecuadas para que un autor de llamada (usuario o aplicación) pueda acceder. La autenticación se realiza a través de Azure Active Directory. La autorización puede realizarse mediante el control de acceso basado en rol de Azure (RBAC de Azure) o la directiva de acceso de Key Vault. Se usa el control de acceso basado en rol de Azure cuando se trata de la administración de los almacenes y la directiva de acceso de Key Vault cuando se intenta acceder a los datos almacenados en un almacén. Los almacenes de claves de Azure pueden estar protegidos por software o, con el nivel Premium de Azure Key Vault, por hardware (módulos de seguridad de hardware [HSM]).
* **Supervisión del acceso y uso:** puede supervisar la actividad habilitando el registro para los almacenes. Tiene el control sobre los registros y puede protegerlos de forma segura restringiendo el acceso y, además, puede eliminar los registros que ya no necesita. Azure Key Vault se puede configurar para lo siguiente:
  + Archivar en una cuenta de almacenamiento.
  + Transmitir a un centro de eventos.
  + Envíe los registros a los registros de Azure Monitor.
* **Administración simplificada de secretos de aplicación**: la información de seguridad debe protegerse, debe seguir un ciclo de vida y debe tener una disponibilidad alta. Azure Key Vault simplifica el proceso de cumplimiento de estos requisitos mediante:
  + La eliminación de la necesidad de poseer conocimientos internos sobre los módulos de seguridad de hardware (HSM).
  + El escalado vertical en un breve plazo de tiempo para adaptarse a los picos de uso de la organización.
  + La replicación del contenido de una instancia de Key Vault de una región en una región secundaria. La replicación de datos garantiza la alta disponibilidad y elimina la necesidad de intervención del administrador para desencadenar la conmutación por error.
  + La disposición de opciones estándar de administración de Azure a través del portal, la CLI de Azure y PowerShell.
  + La automatización de determinadas tareas de los certificados que adquiere de entidades de certificación públicas, como la inscripción y la renovación.

**Información sobre los procedimientos recomendados de Azure Key Vault**

Azure Key Vault es una herramienta para almacenar y acceder a los secretos de forma segura. Un secreto es todo aquello cuyo acceso se quiere controlar de forma estrecha, como los certificados, las contraseñas o las claves de API. Un almacén es un grupo lógico de secretos.

**Authentication**

Hay tres formas de autenticarse en Key Vault:

* **Identidades administradas de recursos de Azure**: cuando implementa una aplicación en una máquina virtual en Azure, puede asignar una identidad a la máquina virtual que tiene acceso a Key Vault. También puede asignar identidades a otros recursos de Azure .

Azure rota automáticamente el secreto de cliente de la entidad de servicio asociado con la identidad.(recomendado)

* **Entidad de servicio y certificado**: puede usar una entidad de servicio y un certificado asociado que tenga acceso a Key Vault.
* **Entidad de servicio y secreto**: aunque puede usar una entidad de servicio y un secreto para autenticarse en Key Vault(no recomendado,rotar secretos).

**Cifrado de datos en tránsito**

Azure Key Vault aplica el protocolo Seguridad de la capa de transporte (TLS) para proteger los datos en el tránsito entre Azure Key Vault y los clientes. TLS proporciona una autenticación sólida, privacidad de mensajes e integridad .

Confidencialidad directa total (PFS) protege las conexiones entre los sistemas cliente de los clientes y los servicios en la nube de Microsoft mediante claves únicas.

**Procedimientos recomendados de Azure Key Vault**

* **Uso de almacenes de claves distintos**: se recomienda usar un almacén por aplicación y por entorno (desarrollo, preproducción y producción). Este patrón ayuda a no compartir secretos entre los entornos y, también, a reducir la amenaza en el caso de infracción.
* **Control del acceso al almacén**: los datos de Key Vault son confidenciales y críticos para la empresa; necesita proteger el acceso a sus almacenes de claves permitiéndoselo solo a aplicaciones y usuarios autorizados.
* **Copia de seguridad**: cree copias de seguridad periódicas del almacén al actualizar, eliminar o crear objetos dentro de este.
* **Registro**: asegúrese de activar el registro y las alertas.
* **Opciones de recuperación**: active la [eliminación temporal](https://learn.microsoft.com/es-es/azure/key-vault/general/soft-delete-overview) y la protección de purga si desea protegerse contra la eliminación forzada del secreto.(se queda en una papelera de reciclaje)

**Autenticación en Azure Key Vault**

La autenticación con Key Vault funciona junto con Azure Active Directory, que es responsable de autenticar la identidad de cualquier entidad de seguridad determinada.

hay dos maneras de obtener una entidad de servicio:

* Habilite una **identidad administrada** asignada por el sistema para la aplicación. Con la identidad administrada, Azure administra internamente la entidad de servicio de la aplicación y autentica la aplicación automáticamente con otros servicios de Azure.
* Con su inquilino de Azure AD. El registro también crea un segundo objeto de aplicación que identifica la aplicación en todos los inquilinos.

**Autenticación en Key Vault en el código de la aplicación**

El SDK de Key Vault usa la biblioteca cliente de identidad de Azure, que permite la autenticación sin problemas en Key Vault en entornos con el mismo código.

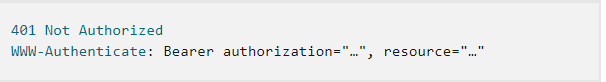


**Autenticación en Key Vault con REST**

Los tokens de acceso deben enviarse al servicio mediante el encabezado de autorización HTTP:



Cuando no se proporciona un token de acceso o cuando el servicio no acepta un token, se devuelve el error HTTP 401 al cliente, que incluirá el encabezado WWW-Authenticate, por ejemplo:



Los parámetros del encabezado WWW-Authenticate son los siguientes:

* authorization: la dirección del servicio de autorización de OAuth2 que puede utilizarse para obtener un token de acceso para la solicitud.
* resource: Nombre del recurso (https://vault.azure.net) que se va a utilizar en la solicitud de autorización.

Secretos --> usuario-contraseña o tmb app –cadenaconexión.

Claves -->cifrado de objetos(blobs,mensajería…).

Certificados-->Conexiones seguras a apps.

MODULO 2

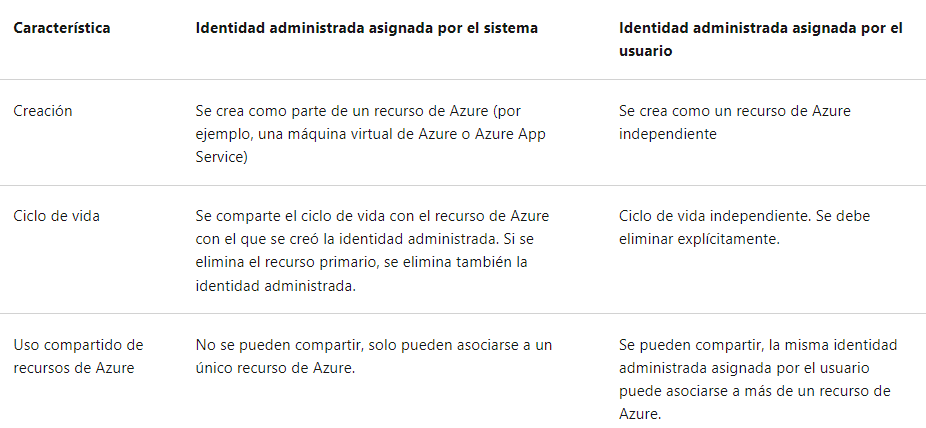
**Exploración de identidades administradas**

Las identidades administradas eliminan la necesidad de administrar las credenciales para los desarrolladores. Proporcionan una identidad administrada automáticamente en Azure Active Directory (Azure AD) para que las aplicaciones la utilicen al conectarse a los recursos que admiten la autenticación de Azure AD.

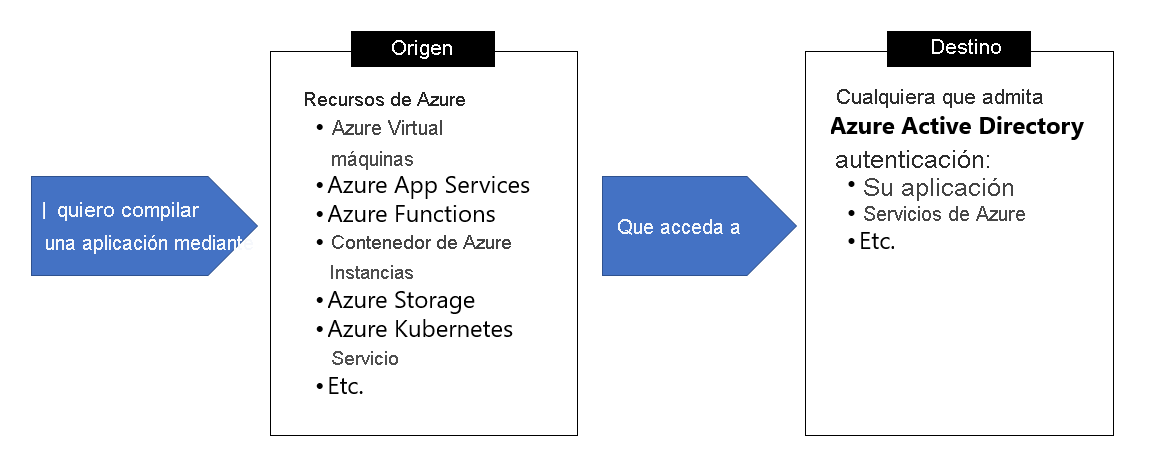
Hay dos tipos :

* Las **identidades administradas asignadas por el sistema**  Azure crea una identidad para la instancia del servicio en el inquilino de Azure AD de confianza de la suscripción de la instancia. Una vez creada la identidad, las credenciales se aprovisionan en la instancia. El ciclo de vida de una identidad administrada asignada por el sistema está vinculado directamente a la instancia de servicio de Azure en que está habilitada. Si se elimina la instancia, Azure limpia automáticamente las credenciales y la identidad en Azure AD.(para un solo recurso)
* **Las identidades administrada asignadas por el usuario**: (para varios elementos)se crean como recursos de Azure independientes. Mediante un proceso de creación, Azure crea una identidad en el inquilino de Azure AD de confianza para la suscripción que se utiliza. Una vez creada la identidad, esta puede asignarse a una o varias instancias de servicio de Azure. El ciclo de vida de una identidad asignada por el usuario no se administra junto con el ciclo de vida de las instancias de servicio de Azure a las que se asigna.(es independiente a sus recursos ,al eliminar los recursos no la eliminas).

**Características de las identidades administradas**



**Cuándo se deben usar las identidades administradas**



**Detección del flujo de autenticación de las identidades administradas**

**Funcionamiento de una identidad administrada asignada por el sistema con una máquina virtual de Azure**

1. Azure Resource Manager recibe una solicitud para habilitar la identidad administrada asignada por el sistema en una máquina virtual.
2. Azure Resource Manager crea una entidad de servicio en Azure Active Directory para la identidad de la máquina virtual. La entidad de servicio se crea en el inquilino de Azure Active Directory en el que confía la suscripción.
3. Azure Resource Manager configura la identidad en la máquina virtual mediante la actualización del punto de conexión de la identidad de Azure Instance Metadata Service con el identificador y el certificado de cliente de la entidad de servicio.
4. Una vez que la máquina virtual tenga una identidad, use la información de la entidad de servicio para conceder a la máquina virtual acceso a los recursos de Azure. Para llamar a Azure Resource Manager, use el control de acceso basado en roles en Azure Active Directory para asignar el rol adecuado a la entidad de servicio de la máquina virtual. Para llamar a Key Vault, conceda a su código acceso al secreto o a la clave específicos en Key Vault.
5. El código que se ejecuta en la máquina virtual puede solicitar un token del punto de conexión de Azure Instance Metadata Service, al que solo puede accederse desde la máquina virtual: http://169.254.169.254/metadata/identity/oauth2/token
6. Se realiza una llamada a Azure Active Directory para solicitar un token de acceso (tal y como se especifica en el paso 5) con el identificador de cliente y el certificado que se configuraron en el paso 3. Azure Active Directory devuelve un token de acceso JSON Web Token (JWT).
7. El código envía el token de acceso en la llamada a un servicio que admite la autenticación de Azure Active Directory.

El azure directory es lo mismo que el azure domain devices pero el primero usa un json y otro usa un jlap

**Funcionamiento de una identidad administrada asignada por el usuario con una máquina virtual de Azure**

1. Azure Resource Manager recibe una solicitud para crear una identidad administrada asignada por el usuario.
2. Azure Resource Manager crea una entidad de servicio en Azure Active Directory para la identidad administrada asignada por el usuario. La entidad de servicio se crea en el inquilino de Azure Active Directory en el que confía la suscripción.
3. Azure Resource Manager recibe una solicitud para configurar la identidad administrada asignada por el usuario en una máquina virtual y actualiza el punto de conexión de identidad de Azure Instance Metadata Service con el certificado y el identificador de cliente de la entidad de servicio de la identidad administrada asignada por el usuario.
4. Una vez que se ha creado la identidad administrada asignada por el usuario, se usa la información de la entidad de servicio para conceder a la identidad acceso a los recursos de Azure. Para llamar a Azure Resource Manager, use el control de acceso basado en roles en Azure Active Directory para asignar el rol adecuado a la entidad de servicio de la identidad asignada por el usuario. Para llamar a Key Vault, conceda a su código acceso al secreto o a la clave específicos en Key Vault.

**Nota**

También puede realizar este paso antes del paso 3.

1. El código que se ejecuta en la máquina virtual puede solicitar un token del punto de conexión de la identidad de Azure Instance Metadata Service, al que solo puede accederse desde la máquina virtual: http://169.254.169.254/metadata/identity/oauth2/token
2. Se realiza una llamada a Azure Active Directory para solicitar un token de acceso (tal y como se especifica en el paso 5) con el identificador de cliente y el certificado que se configuraron en el paso 3. Azure Active Directory devuelve un token de acceso JSON Web Token (JWT).
3. El código envía el token de acceso en la llamada a un servicio que admite la autenticación de Azure Active Directory.

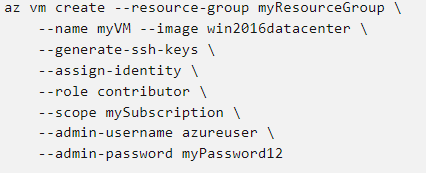
**Configuración de identidades administradas**

**Identidad administrada asignada por el sistema**

Para crear o habilitar una máquina virtual de Azure con la identidad administrada asignada por el sistema, la cuenta necesita la asignación de roles **Colaborador de la máquina virtual**. No se requiere ninguna otra asignación de roles de directorio de Azure AD.

**Habilitación de una identidad administrada asignada por el sistema durante la creación de una máquina virtual de Azure**

En el ejemplo siguiente se crea una máquina virtual llamada *myVM* con una identidad administrada asignada por el sistema, tal y como lo solicita el parámetro --assign-identity, con los valores --role y --scope especificados. Los parámetros --admin-username y --admin-password especifican el nombre de usuario administrativo y la contraseña de la cuenta para el inicio de sesión en la máquina virtual. Actualice estos valores según convenga para su entorno:



**Habilitación de una identidad administrada asignada por el sistema en una máquina virtual de Azure existente**

Use el comando az vm identity assign para asignar la identidad asignada por el sistema a una máquina virtual existente:



**Identidad administrada asignada por el usuario**

Para asignar una identidad asignada por el usuario a una máquina virtual durante su creación, la cuenta necesita las asignaciones de roles **Colaborador de la máquina virtual** y **Operador de identidades administradas**.

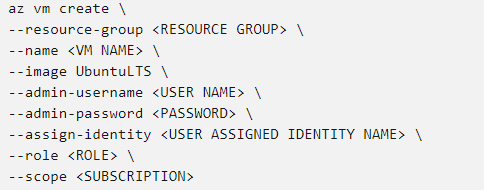
El proceso de habilitación de identidades administradas asignadas por el usuario consta de dos pasos:

1. Creación de la identidad asignada por el usuario
2. Asignación de la identidad a una máquina virtual

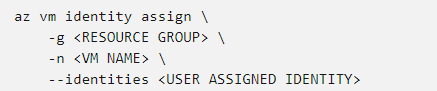
**Creación de una identidad asignada por el usuario**

Cree una identidad administrada asignada por el usuario mediante az identity create. El parámetro -g especifica el grupo de recursos en el que se crea la identidad administrada asignada por el usuario, mientras que el parámetro -n especifica su nombre. 

**Asignación de una identidad administrada asignada por el usuario durante la creación de una máquina virtual de Azure**



**Asignación de una identidad administrada asignada por el usuario a una máquina virtual de Azure existente**



**Compatibilidad de SDK de Azure con identidades administradas de recursos de Azure**

Acepta los sdk de .NET,java,node.js,phyton y Ruby

**Adquisición de un token de acceso**

Una aplicación cliente puede solicitar un token de acceso de solo aplicación de identidades administradas de recursos de Azure para acceder a un recurso determinado. El método recomendado es usar DefaultAzureCredential.DefaultAzureCredential intenta autenticarse automáticamente mediante varios mecanismos, incluidas las variables de entorno o un inicio de sesión interactivo.El tipo de credencial se puede usar en el entorno de desarrollo con sus propias credenciales. También se puede utilizar en el entorno de producción de Azure mediante una identidad administrada. No se requieren cambios en el código al implementar la aplicación.

DefaultAzureCredential intenta realizar la autenticación mediante los siguientes mecanismos, en este orden, y se detiene cuando lo consigue:

1. **Entorno**: DefaultAzureCredential lee la información de la cuenta especificada mediante variables de entorno y la usa para realizar la autenticación.
2. **Identidad administrada**: si la aplicación se implementa en un host de Azure con la identidad administrada habilitada, DefaultAzureCredential realiza la autenticación con esa cuenta.
3. **Visual Studio**: si el desarrollador se ha autenticado mediante Visual Studio, DefaultAzureCredential realiza la autenticación con esa cuenta.
4. **CLI de Azure**: si el desarrollador ha autenticado una cuenta mediante el comando az login de la CLI de Azure, DefaultAzureCredential realiza la autenticación con esa cuenta. Los usuarios de Visual Studio Code pueden autenticar su entorno de desarrollo mediante la CLI de Azure.
5. **Azure PowerShell**: si el desarrollador ha autenticado una cuenta mediante el comando Connect-AzAccount de la CLI de Azure, DefaultAzureCredential realiza la autenticación con esa cuenta.
6. **Explorador interactivo**: si está habilitado, DefaultAzureCredential autenticará de forma interactiva al desarrollador mediante el explorador predeterminado del sistema actual. De forma predeterminada, este tipo de credencial está deshabilitado.

Para ejecutar el skd de azure identities = dotnet add package Azure.Identity

**Autenticación con DefaultAzureCredential**



**Especificación de una identidad administrada asignada por el usuario con DefaultAzureCredential**

// When deployed to an azure host, the default azure credential will authenticate the specified user assigned managed identity.

string userAssignedClientId = "<your managed identity client Id>";

var credential = new DefaultAzureCredential(new DefaultAzureCredentialOptions { ManagedIdentityClientId = userAssignedClientId });

var blobClient = new BlobClient(new Uri("https://myaccount.blob.core.windows.net/mycontainer/myblob"), credential);

**Definición de un flujo de autenticación personalizado con ChainedTokenCredential**

Aunque DefaultAzureCredential generalmente es la forma más rápida de empezar a desarrollar aplicaciones para Azure, es posible que los usuarios más avanzados quieran personalizar las credenciales que se tienen en cuenta al autenticarse. ChainedTokenCredential permite a los usuarios combinar varias instancias de credenciales para definir una cadena personalizada de credenciales:

// Authenticate using managed identity if it is available; otherwise use the Azure CLI to authenticate.

var credential = new ChainedTokenCredential(new ManagedIdentityCredential(), new AzureCliCredential());

var eventHubProducerClient = new EventHubProducerClient("myeventhub.eventhubs.windows.net", "myhubpath", credential);

Modulo 3

Azure App Configuration proporciona un servicio para administrar la configuración de la aplicación y las marcas de características de forma centralizada.

App Configuration ofrece las siguientes ventajas:

* Un servicio totalmente administrado que puede configurar en cuestión de minutos
* Representaciones y asignaciones de claves flexibles
* Etiquetado con etiquetas
* Reproducción de la configuración en un momento dado
* Interfaz de usuario dedicada para la administración de marcas de características
* Comparación de dos conjuntos de configuraciones en dimensiones definidas de forma personalizada
* Seguridad mejorada mediante las identidades administradas de Azure
* Cifrado de información confidencial en reposo y en tránsito
* Integración nativa con plataformas conocidas

App Configuration complementa Azure Key Vault, que se usa para almacenar secretos de aplicación. App Configuration facilita la implementación de los siguientes escenarios:

* La administración centralizada y la distribución de datos de configuración jerárquicos para diferentes entornos y regiones geográficas
* Los cambios de configuración dinámica sin necesidad de volver a implementar o reiniciar una aplicación
* El control de la disponibilidad de las características en tiempo real

La forma más fácil de agregar un almacén de App Configuration a una aplicación es mediante una biblioteca de cliente que Microsoft proporciona:



**Creación de claves y valores emparejados**

Azure App Configuration almacena los datos de configuración como pares clave-valor.

**Claves**

Las claves sirven como nombre de los pares clave-valor y se usan para almacenar y recuperar los valores correspondientes.Azure App Configuration No analiza las claves para averiguar cómo están estructurados sus nombres o para aplicar ninguna regla en ellas.

Las claves almacenadas en App Configuration distinguen entre mayúsculas y minúsculas y son cadenas basadas en Unicode.

**Diseño de espacios de nombres de clave**

Hay dos enfoques generales en relación con la nomenclatura de las claves que se usa en los datos de configuración: plano o jerárquico. Desde el punto de vista del uso de la aplicación, estos métodos son similares, pero la nomenclatura jerárquica ofrece varias ventajas:

* Es más fácil de leer. En lugar de una secuencia larga de caracteres, los delimitadores de una función de nombre de clave jerárquico funciona como los espacios de una frase.
* Es más fácil de administrar. Un nombre de clave jerárquico representa grupos lógicos de datos de configuración.
* Es más fácil de usar. Resulta más fácil escribir una consulta cuyos patrones coinciden con las claves de una estructura jerárquica y que recupera solo una porción de los datos de configuración.

En los ejemplos siguientes se muestra cómo puede estructurar los nombres de clave en una jerarquía:

Según los servicios de los componentes.



Según las regiones de implementación.



**Claves de etiqueta**

Los valores de clave de App Configuration pueden tener un atributo de etiqueta. Las etiquetas se utilizan para diferenciar los valores de clave con la misma clave. Una clave *app1* con las etiquetas *A* y *B* forma dos claves independientes en un almacén de App Configuration. De forma predeterminada, la etiqueta de un valor de clave está vacía o es null.

Las etiquetas proporcionan una manera cómoda de crear variantes de una clave. Un uso habitual de las etiquetas consiste en especificar varios entornos para la misma clave:

Key = AppName:DbEndpoint & Label = Test

Key = AppName:DbEndpoint & Label = Staging

Key = AppName:DbEndpoint & Label = Production

**Valores de clave de versión**

App Configuration no crea valores de clave de versión automáticamente a medida que se modifican. Use etiquetas como una manera de crear varias versiones de un valor de clave.

**Valores de clave de consulta**

Cada valor de clave se identifica por su clave más una etiqueta que puede ser null. Si desea consultar los valores de las claves en un almacén de App Configuration, especifique un modelo. El almacén de App Configuration devuelve todos los valores de claves que coincidan con el modelo, así como sus valores y atributos correspondientes.

**Valores**

Los valores asignados a las claves también son cadenas unicode. Puede usar todos los caracteres unicode para los valores. Hay un tipo de contenido opcional definido por el usuario que se asocia a cada valor. Utilice este atributo para almacenar información, por ejemplo, un esquema de codificación, sobre un valor que ayuda a la aplicación a procesarlo correctamente.

Los datos de configuración que se encuentran en un almacén de App Configuration, entre los que se incluyen todas las claves y los valores, se cifran tanto en reposo como en tránsito. App Configuration no es una solución de reemplazo para Azure Key Vault. No almacene secretos de aplicación en ella.

Octava ruta de aprendizaje(Resumen chatgpt)

**Descripción del servicio API Management**

El servicio API Management en Azure proporciona funcionalidades esenciales para garantizar el éxito de un programa de API. Se compone de una puerta de enlace de API, un plano de administración y un portal para desarrolladores. La puerta de enlace acepta llamadas de API, verifica credenciales, aplica cuotas y límites, y almacena en caché respuestas. El plano de administración se utiliza para configurar el programa de API, definir esquemas, empaquetar API en productos y gestionar usuarios. El portal para desarrolladores es un sitio web personalizable con documentación de API, consola interactiva y análisis. Los productos se utilizan para presentar las API a los desarrolladores, mientras que los grupos permiten administrar la visibilidad de los productos. Los desarrolladores son cuentas de usuario que pueden suscribirse a productos y las directivas son declaraciones que se ejecutan en las solicitudes o respuestas de una API.

**Exploración de las puertas de enlace de API**

La puerta de enlace de API Management viene en dos opciones: administrada y autohospedada. La administrada es el componente predeterminado en Azure, donde todo el tráfico fluye a través de Azure. La autohospedada, por otro lado, es una versión en contenedor de la puerta de enlace administrada y se utiliza en escenarios híbridos y multicloud, donde se ejecutan las puertas de enlace fuera de Azure junto con los back-end de API.

La puerta de enlace de API Management ayuda a desacoplar a los clientes de los servicios, simplifica la administración y mejora la seguridad al proteger los servicios expuestos públicamente.

**Exploración de directivas de API Management**

Las directivas en Azure API Management permiten cambiar el comportamiento de una API mediante la configuración. Son una colección de declaraciones que se ejecutan secuencialmente en las solicitudes o respuestas de la API.

Las directivas se aplican en la puerta de enlace entre el consumidor de la API y la API administrada. Pueden modificar las solicitudes de entrada y las respuestas de salida. Las directivas están organizadas en secciones: inbound, backend, outbound y on-error. Cada sección contiene instrucciones que se ejecutan en orden para una solicitud y una respuesta.

Las directivas se definen en un documento XML que describe la secuencia de declaraciones. Pueden utilizarse expresiones de directiva, que son instrucciones de C#, para acceder a variables de contexto y realizar acciones personalizadas.

Las directivas se pueden aplicar a nivel global, de API o de operación. Si hay múltiples directivas configuradas, se aplican en un orden determinado. Además, se pueden filtrar y modificar los elementos de datos de las respuestas en función de condiciones específicas.

Las directivas proporcionan una forma flexible de controlar el tráfico y personalizar el comportamiento de la API sin necesidad de escribir código adicional ni modificar los servicios back-end.

**Creación de directivas avanzadas**

En esta unidad, se exploran varias directivas avanzadas en Azure API Management:

1. Flujo de control: La directiva "choose" permite aplicar instrucciones de directiva basadas en la evaluación de expresiones booleanas. Se pueden usar elementos "when" y "otherwise" para establecer condiciones y ejecutar instrucciones específicas según el resultado de la evaluación.

2. Reenvío de solicitud: La directiva "forward-request" reenvía la solicitud entrante al servicio backend especificado en el contexto de la solicitud. Se puede establecer un tiempo de espera y controlar el seguimiento de redirecciones.

3. Limitar la simultaneidad: La directiva "limit-concurrency" evita que se ejecuten simultáneamente un número de solicitudes mayor que el especificado. Si se supera este número, las nuevas solicitudes producirán un error con un código de estado 429.

4. Registro en centro de eventos: La directiva "log-to-eventhub" envía mensajes a un centro de eventos definido por una entidad de registro. Se utiliza para guardar información de contexto de solicitud o respuesta para su análisis posterior.

5. Simular respuesta: La directiva "mock-response" se utiliza para simular respuestas de API. Anula la ejecución normal de la canalización y devuelve una respuesta simulada al llamador.

6. Reintentar: La directiva "retry" permite volver a intentar ejecutar las directivas secundarias hasta que se cumpla una condición o se agote el número de intentos especificado.

7. Devolución de respuesta: La directiva "return-response" anula la ejecución de la canalización y devuelve una respuesta personalizada o predeterminada al autor de la llamada. Se pueden establecer encabezados, cuerpo y estado de respuesta.

Cada directiva tiene su propia estructura y atributos específicos que se utilizan para personalizar el comportamiento de la API y controlar el flujo de la solicitud y respuesta.

**Protección de las API mediante certificados**

En esta unidad, se explora cómo se pueden utilizar los certificados para proteger las API en Azure API Management:

1. Autenticación de cliente TLS: La puerta de enlace de API Management puede autenticar y validar los certificados de cliente incluidos en las solicitudes. Se pueden establecer condiciones basadas en propiedades como la entidad de certificación (CA), la huella digital, el asunto y la fecha de expiración.

2. Configuración de la puerta de enlace: La configuración para permitir certificados de cliente se realiza en la página de dominios personalizados en Azure Portal. En el nivel de plan de consumo, es necesario habilitar explícitamente el uso de certificados de cliente.

3. Directivas de autorización de certificados: Se pueden crear directivas de autorización en la puerta de enlace de API Management para controlar el acceso basado en los certificados de cliente. Estas directivas se configuran en el archivo de directiva de procesamiento de entrada.

4. Comprobación de huella digital: Se puede comprobar la huella digital de un certificado de cliente en la directiva para garantizar su validez. Se puede realizar una comparación con una huella digital deseada para permitir o denegar el acceso.

5. Comprobación de emisor y sujeto: Además de la huella digital, se pueden comprobar propiedades adicionales del certificado, como el emisor y el sujeto. Esto ayuda a asegurar que los certificados provienen de entidades confiables.

La utilización de certificados proporciona una capa adicional de seguridad y autenticación mutua entre los clientes y la puerta de enlace de API Management, garantizando que solo las solicitudes con certificados válidos sean procesadas.

Modulo 2

**Creación de una puerta de enlace de API**

La creación de una puerta de enlace de API con Azure API Management implica los siguientes pasos:

1. Crear una puerta de enlace de API Management: Este servicio se encuentra en la nube de Azure y actúa como una puerta de enlace entre Internet y sus API. Proporciona una serie de ventajas como documentación de API, límites de frecuencia de acceso, seguimiento del estado, formatos modernos como JSON, conexiones a cualquier API, análisis y seguridad.

2. Registrar una API web existente en la puerta de enlace: Puede registrar sus API existentes en la puerta de enlace de API Management. Esto permite que sus API estén disponibles para los consumidores a través del punto de conexión proporcionado por la puerta de enlace.

3. Llamar a la API mediante la puerta de enlace: Una vez que la API está registrada en la puerta de enlace, los consumidores pueden llamar a la API utilizando la URL del punto de conexión de la puerta de enlace. Esto proporciona un enfoque centralizado y simplificado para que los consumidores accedan a sus API.

Al configurar Azure API Management, puede aprovechar sus características y funcionalidades para mejorar la documentación, el rendimiento, la seguridad y la administración de sus API, facilitando la integración de los consumidores y el control de acceso a sus servicios.

**Importación y publicación de una API**

Para importar y publicar una API en Azure API Management, los pasos principales son los siguientes:

1. Importar la API: Puede importar una API existente en Azure API Management utilizando diferentes marcos y estándares como HTTP, WebSocket, GraphQL, OpenAPI, WADL, WSDL, e incluso API de recursos de Azure como Aplicaciones lógicas, App Service, Funciones y más. Puede realizar la importación a través de Azure Portal proporcionando el tipo de API y la especificación correspondiente.

2. Visualizar la API: Una vez importada, puede utilizar la puerta de enlace de API para visualizar y probar la API. Puede interactuar con los conjuntos de datos y pasar parámetros para ver cómo se procesan los datos mediante las API importadas. Como administrador, puede acceder a esta funcionalidad a través de Azure Portal.

3. Configurar directivas: Las directivas son declaraciones que se ejecutan secuencialmente en la solicitud o respuesta de una API, lo que permite cambiar su comportamiento. Puede aplicar diferentes directivas a las API para realizar conversiones, limitar la frecuencia de llamadas, establecer encabezados y más. Puede configurar las directivas en Azure Portal, específicamente en la pestaña Diseño de la configuración de la API.

4. Asociar la API a un producto: En Azure API Management, los productos son colecciones de API y se pueden asignar a diferentes productos. Puede asignar una API a más de un producto, lo que le permite definir diferentes reglas de acceso, cuotas de uso y condiciones de uso para cada producto. Puede asociar las API con productos a través de Azure Portal, lo que le permite administrar los derechos de acceso de los asociados y proveedores.

Es importante tener en cuenta que los pasos y opciones pueden variar según el plan de tarifa seleccionado en Azure API Management. Los productos predeterminados incluyen Starter y Unlimited, pero también puede crear nuevos productos según sus necesidades.

Al seguir estos pasos, puede importar, configurar y publicar su API en Azure API Management, brindando un control y acceso adecuados a sus asociados y proveedores.

**Llamada a una API con una clave de suscripción**

Las claves de suscripción son utilizadas para restringir el acceso a una API publicada a través de una puerta de enlace. Estos son los puntos clave a tener en cuenta:

- Una clave de suscripción es una cadena única generada automáticamente que se puede incluir en los encabezados de solicitud o como una cadena de consulta en la URL.

- Cada suscripción tiene una clave principal y una clave secundaria, lo que permite regenerar una clave sin interrumpir el servicio.

- Las claves de suscripción están asociadas a diferentes ámbitos, como todas las API, una sola API o un producto, lo que brinda un control detallado sobre los permisos y directivas.

- Las aplicaciones que llaman a una API protegida deben incluir la clave de suscripción en la solicitud, ya sea en el encabezado de la solicitud o como una cadena de consulta en la URL.

- Las claves de suscripción se pueden obtener solicitando una suscripción como parte del flujo de trabajo de Azure API Management.

- Es importante asegurarse de incluir una clave de suscripción válida en todas las solicitudes HTTP cuando se llame a puntos de conexión protegidos por una suscripción. Si la clave no se proporciona, la puerta de enlace de la API devolverá una respuesta de acceso denegado (401).

En resumen, las claves de suscripción son utilizadas para proteger las API y controlar el acceso a través de Azure API Management. Al incluir una clave de suscripción válida en las solicitudes, se puede acceder y utilizar la API de manera autorizada.

**Elección de la directiva adecuada de API Management**

Las directivas de API Management permiten controlar el comportamiento de una API sin necesidad de modificar su código. Aquí hay algunos puntos clave:

- Las directivas son instrucciones que se ejecutan en orden y se componen en estructuras XML.

- Las directivas se ejecutan en cuatro momentos diferentes: entrada, back-end, salida y en caso de error.

- Los ámbitos de las directivas son: global (afecta a todas las API), producto (afecta a todas las API de un producto), API (afecta a una única API) y operación (afecta a una única operación dentro de una API).

- La etiqueta <base /> determina el orden de ejecución de las directivas en un ámbito determinado.

- Hay varias directivas de uso frecuente que se pueden utilizar para restringir el acceso, controlar la autenticación, habilitar solicitudes entre dominios y realizar transformaciones en las respuestas.

- También hay directivas avanzadas que permiten un comportamiento personalizado, como el flujo de control, el reenvío de solicitudes, el control de errores y el almacenamiento de valores en variables.

En resumen, las directivas de API Management ofrecen flexibilidad para modificar y controlar el comportamiento de las API sin necesidad de cambiar su código subyacente, lo que facilita la implementación de diferentes funcionalidades y restricciones en la administración de API.

**Configuración de una directiva de almacenamiento en caché**

La configuración de una caché en Azure API Management puede mejorar el rendimiento de las respuestas de la API. Aquí hay algunos puntos clave:

- Para configurar una caché, se utilizan las directivas "cache-lookup" y "cache-store".

- La directiva "cache-lookup" se utiliza para comprobar si hay una respuesta almacenada en caché para una solicitud.

- La directiva "cache-store" se utiliza para almacenar las respuestas en la caché.

- También es posible almacenar valores individuales en la caché utilizando las directivas "cache-lookup-value" y "cache-store-value".

- Se pueden utilizar etiquetas de variación (vary-by) para garantizar que las respuestas de la caché sean relevantes para la solicitud original.

- La caché interna de API Management se utiliza de forma predeterminada, pero también se puede utilizar una caché externa, como Azure Redis Cache, si se prefiere tener un mayor control sobre la configuración de la caché.

La configuración de una caché en API Management puede ayudar a reducir el tiempo de respuesta de las solicitudes y mejorar el rendimiento general de la API.

Modulo 3

**Eliminación de información técnica de las respuestas de API**

Para garantizar la seguridad de una API y proteger los datos personales, es importante eliminar información técnica sensible de las respuestas. En Azure API Management, se pueden configurar directivas para eliminar encabezados de respuesta no seguros que puedan exponer información confidencial.

Algunos encabezados que se deben eliminar incluyen "X-Powered-By", que revela la tecnología de plataforma utilizada, y otros encabezados que puedan proporcionar detalles sobre el servidor.

Para configurar API Management y eliminar estos encabezados de respuesta, se deben seguir los siguientes pasos:

1. Crear una puerta de enlace de API Management en Azure Portal. Esto implica crear el recurso de API Management y asignar propiedades como el Fully Qualified Domain Name (FQDN) y el plan de tarifas.

2. Registrar una API web existente en la puerta de enlace. Se debe agregar la API web a API Management para poder aplicar las directivas y otras herramientas de seguridad. La API ya debe estar alojada en Azure App Service.

3. Aplicar una directiva que elimine los encabezados no seguros de todas las respuestas. Esto se puede hacer mediante la configuración de una directiva de salida en la puerta de enlace, que elimine los encabezados específicos que se desean ocultar.

Al eliminar los encabezados de respuesta no seguros, se protege la información técnica y se reduce el riesgo de exposición de datos sensibles a usuarios malintencionados.

**Enmascaramiento de URL con una directiva de transformación**

El enmascaramiento de URL es una práctica importante para proteger la seguridad de una API. En Azure API Management, se puede lograr esto utilizando una directiva de transformación llamada "Enmascarar URL en el contenido".

Cuando se realiza una llamada a la API del censo gubernamental, la respuesta contiene vínculos URL que pueden revelar detalles sobre la URL desde la cual se realizó la llamada. Esto podría permitir a usuarios malintencionados acceder a datos sensibles al eludir la puerta de enlace de API Management y utilizar un punto de conexión menos seguro. Como desarrollador, es recomendable enmascarar estas URL en el cuerpo de la respuesta de la API.

Para lograr esto, se puede utilizar la directiva de transformación "Enmascarar URL en el contenido", que reescribe los vínculos en el cuerpo de la respuesta para que apunten a través de la puerta de enlace de API Management en lugar de la URL original.

En Azure API Management, existen diversas transformaciones disponibles, como convertir entre JSON y XML, buscar y reemplazar cadenas en el cuerpo de la solicitud o respuesta, establecer encabezados y parámetros de consulta, y reescribir URL. En este caso, se utilizará la transformación "Enmascarar URL en el contenido" para modificar los vínculos y asegurar que apunten a través de la puerta de enlace de API Management.

Al enmascarar las URL en el contenido de las respuestas, se protege la seguridad de la API y se evita la exposición de información sensible a usuarios no autorizados.

**Limitación de las solicitudes de API**

La limitación de las solicitudes de API es una práctica común para evitar la sobreutilización de una API y garantizar una respuesta rápida a todos los usuarios. En Azure API Management, se pueden aplicar dos tipos de limitaciones utilizando directivas: limitación por suscripción y limitación por clave.

La limitación por suscripción establece límites de frecuencia para una operación de API específica, sin distinguir entre clientes. Esto significa que todas las solicitudes a la API u operación se limitan de la misma manera. Por ejemplo, se puede configurar un límite de 3 llamadas en un período de renovación de 15 segundos para todas las operaciones de la API. Esto garantiza que los clientes no puedan exceder ese límite y recibirán un error 429 si lo hacen.

La limitación por clave permite establecer límites de frecuencia diferentes para cada valor de solicitud del cliente, generalmente basado en la dirección IP del cliente. Esto proporciona a todos los clientes el mismo ancho de banda para llamar a la API. Por ejemplo, se puede configurar un límite de 10 llamadas en un período de renovación de 60 segundos para cada dirección IP de solicitud. Esto asegura que cada dirección IP tenga su propio límite de frecuencia y evita que un cliente agote todas las solicitudes antes de que otro pueda usarlas.

Es importante tener en cuenta que la directiva <rate-limit-by-key> para limitación por clave no está disponible cuando la puerta de enlace de API Management está en el nivel de consumo. En ese caso, se puede utilizar la directiva <rate-limit> en su lugar.

La limitación de las solicitudes de API es una estrategia eficaz para controlar el uso de la API y evitar problemas como costos adicionales o una disminución en la capacidad de respuesta debido a la sobreutilización.

**¿Qué es API Management?**

API Management es un servicio de Azure que permite a las organizaciones publicar APIs para socios externos y desarrolladores internos. Proporciona las herramientas necesarias para gestionar y controlar el acceso a las APIs, así como para garantizar su seguridad, seguimiento y análisis.

Los componentes principales de API Management son:

1. Puerta de enlace de API: Es el punto de entrada para las solicitudes de API. Se encarga de enrutar las solicitudes a los back-end correspondientes, autenticar y autorizar las solicitudes, aplicar límites de uso y transformar las respuestas según sea necesario.

2. Azure Portal: Es la interfaz de administración donde se configura y se gestiona el programa de API. Permite definir el esquema de las APIs, crear productos que agrupen las APIs, establecer políticas de uso, obtener información analítica y administrar usuarios.

3. Portal para desarrolladores: Es el lugar donde los desarrolladores pueden acceder a la documentación de la API, probarla en una consola interactiva, obtener claves de API para su suscripción y acceder a análisis sobre su uso personal.

API Management ofrece una serie de beneficios, como la protección de la infraestructura, la incorporación rápida de socios, la creación de programas internos de API y la capacidad de administrar y controlar el acceso a las APIs de manera centralizada.

En resumen, API Management permite a las organizaciones compartir sus servicios y datos a través de APIs de manera segura y controlada, facilitando la colaboración y la innovación con socios externos y desarrolladores internos.

**Creación de suscripciones en Azure API Management**

Al publicar una API con API Management, se puede controlar el acceso a ella utilizando claves de suscripción. Las suscripciones permiten segmentar el acceso de los usuarios a la API, y las claves de suscripción son utilizadas como autorización para habilitar dicho acceso.

Cada vez que un cliente realiza una solicitud a una API protegida, debe incluir una clave de suscripción válida en la solicitud HTTP. Estas claves son únicas y se generan automáticamente, estando relacionadas directamente con una suscripción. Las suscripciones pueden tener diferentes alcances, como todas las API, una sola API o un producto que contiene una o varias API.

Cada suscripción cuenta con una clave principal y una clave secundaria, lo que permite regenerar una clave sin tiempo de inactividad. Es recomendable utilizar la clave secundaria en las aplicaciones mientras se cambia la clave principal.

Las claves de suscripción se pueden obtener mediante el envío de una solicitud de suscripción y, una vez aprobada, se envían de manera segura a los desarrolladores. Las aplicaciones que llaman a una API protegida deben incluir la clave de suscripción en todas las solicitudes, ya sea en el encabezado de la solicitud o como un parámetro de cadena de consulta en la URL.

Es importante destacar que si una clave de suscripción obligatoria no se incluye en la solicitud, se recibirá una respuesta de acceso denegado (401) de la puerta de enlace de API.

En resumen, las claves de suscripción permiten controlar el acceso a las APIs publicadas con API Management, asegurando que solo los usuarios autorizados puedan utilizar los servicios ofrecidos.

**Uso de certificados de cliente para proteger el acceso a una API**

Al utilizar API Management, se puede configurar la puerta de enlace de API para que acepte certificados de cliente, lo que permite la autenticación mutua de TLS entre el cliente y la puerta de enlace. Esto se logra mediante el uso de directivas de autorización en el nivel de puerta de enlace.

Algunas propiedades de los certificados que se pueden comprobar incluyen:

- Entidad de certificación (CA): Se puede permitir únicamente certificados firmados por una CA específica.

- Huella digital: Se pueden permitir certificados que contengan una huella digital específica.

- Sujeto: Se pueden permitir certificados con un firmante especificado.

- Fecha de expiración: Se pueden permitir certificados que no hayan expirado.

Existen diferentes métodos para comprobar la autenticidad de un certificado. Por ejemplo, se puede verificar si el certificado proviene de una entidad de certificación de confianza o si ha sido entregado personalmente por un asociado.

En el caso del plan de consumo de API Management, es necesario habilitar explícitamente el uso de certificados de cliente en el panel de Dominios personalizados.

Para crear las directivas de autorización de certificados, se deben agregar las directivas correspondientes en el archivo de directiva de procesamiento de entrada de la puerta de enlace de API Management. Estas directivas pueden incluir la comprobación de la huella digital de un certificado de cliente, la comprobación de la huella digital en relación con certificados cargados en API Management, o la comprobación del emisor y el sujeto de un certificado de cliente.

En resumen, al configurar API Management para aceptar certificados de cliente, se habilita la autenticación mutua de TLS, lo que proporciona una capa adicional de seguridad al controlar el acceso a la puerta de enlace de API mediante certificados verificados.

**Creación de una nueva API en API Management a partir de una aplicación de función**

En una arquitectura basada en microservicios, Azure API Management es útil para ensamblar y administrar microservicios individuales en un solo producto. Los microservicios son servicios pequeños e independientes que se desarrollan, implementan y escalan de forma independiente. Azure Functions es un servicio sin servidor que permite implementar microservicios. Puede escribir funciones en varios lenguajes y utilizar diferentes plantillas para activar su código en respuesta a eventos como solicitudes HTTP, mensajes en una cola o programaciones predefinidas.

Azure API Management es un servicio en la nube que ayuda a publicar, proteger, transformar, mantener y supervisar API. Permite publicar API para desarrolladores externos, asociados e internos, y controla tareas como autenticación, autorización, límites de velocidad, transformación de solicitudes y respuestas, registro y seguimiento, y administración de versiones. Al publicar Azure Functions a través de API Management, se puede implementar una arquitectura de microservicios, donde cada función representa un microservicio. API Management permite integrar estos microservicios en una aplicación distribuida integrada, y se pueden aplicar directivas de administración para agregar almacenamiento en caché y garantizar requisitos de seguridad.

El nivel de consumo de API Management es especialmente adecuado para arquitecturas basadas en microservicios y sistemas sin servidor. Utiliza una arquitectura basada en recursos compartidos asignados dinámicamente y ofrece beneficios como la ausencia de infraestructura para administrar, capacidad inactiva, alta disponibilidad, escalado automático y precios basados en el uso. Es una opción ideal para exponer recursos sin servidor como API.

En la siguiente unidad, se creará una instancia de API Management y se le agregará una función para implementar un microservicio en la arquitectura sin servidor.

**Las ventajas de usar Azure API Management para crear la API**

Al utilizar Azure API Management, se pueden abordar los desafíos de administrar una arquitectura de microservicios al integrar y presentar diferentes microservicios a las aplicaciones cliente de manera coherente a través de una única dirección URL.

Los desafíos de la arquitectura de microservicios incluyen el acoplamiento de las aplicaciones cliente a los microservicios individuales, la posibilidad de presentar los microservicios en diferentes nombres de dominio o direcciones IP, la dificultad de aplicar reglas y estándares de API coherentes en todos los microservicios, y la dependencia de los equipos individuales para implementar la seguridad correctamente.

API Management ayuda al ensamblar múltiples API, funciones y servicios en un producto integrado que proporciona un punto de entrada único para las aplicaciones cliente. Al hacerlo, las aplicaciones cliente están acopladas a la API que expresa la lógica empresarial en lugar de la implementación técnica subyacente de los microservicios individuales. API Management actúa como intermediario, reenviando las solicitudes al microservicio correcto y ocultando los diferentes URI donde se alojan los microservicios. Además, se pueden utilizar directivas de API Management para aplicar reglas coherentes, como transformar respuestas de XML a JSON, y aplicar requisitos de seguridad consistentes.

Azure API Management permite importar aplicaciones de función de Azure como API nuevas o agregarlas a API existentes, generando automáticamente una clave de host que se asigna a un valor con nombre en API Management.

En la siguiente unidad, se integrará el microservicio de detalles del pedido con la API Online Store creada anteriormente, presentando ambos microservicios en el mismo dominio como parte de una API integrada.

**Compilación de aplicaciones sin servidor con Go(modulo 4)**

**Controladores personalizados**

Los controladores personalizados en Azure Functions son servidores web que reciben eventos del host de Functions y permiten escribir código en el lenguaje de su elección para responder a esos eventos. Con los controladores personalizados, puede utilizar prácticamente cualquier lenguaje que admita primitivas HTTP.

El flujo de procesamiento de una solicitud a través de un controlador personalizado y el host de Functions es el siguiente:

1. Cuando se produce un evento que desencadena la ejecución de una función (como una solicitud HTTP), se envía una solicitud al host de Functions.

2. El host de Functions crea una carga de solicitud que contiene información sobre el desencadenador, los datos de enlace de entrada y otros metadatos relevantes.

3. La función se ejecuta en el controlador personalizado, donde se implementa la lógica de procesamiento de la solicitud.

4. El controlador personalizado devuelve una respuesta al host de Functions.

5. El host de Functions procesa los datos de salida de la función y los pasa al enlace de salida correspondiente para su procesamiento adicional o entrega al destino adecuado.

En resumen, los controladores personalizados permiten utilizar cualquier lenguaje compatible con HTTP para implementar la lógica de procesamiento de eventos en Azure Functions, brindando flexibilidad y elección en el desarrollo de funciones.

**Aplicación web**

Ultima ruta de aprendizaje(Resumen de chatgpt)

Azure Cache for Redis es un servicio proporcionado por Microsoft que ofrece un almacén de datos en memoria basado en el software de Redis. Redis es una tecnología que mejora el rendimiento y la escalabilidad de las aplicaciones que utilizan almacenes de datos de back-end de manera intensiva. Permite procesar grandes volúmenes de solicitudes de aplicación al mantener los datos frecuentemente accedidos en la memoria del servidor, lo que permite realizar operaciones rápidas de lectura y escritura. Redis es una solución crítica de almacenamiento de datos de baja latencia y alto rendimiento en aplicaciones modernas.

Azure Cache for Redis ofrece tanto la versión de código abierto (OSS Redis) como la versión comercial de Redis Labs (Redis Enterprise) como servicio administrado. Proporciona instancias seguras y dedicadas de servidores Redis, así como una compatibilidad completa con la API de Redis. Este servicio está disponible en Azure, y todas las aplicaciones, tanto las que se encuentran dentro como fuera de Azure, pueden utilizarlo.

**Exploración de Azure Cache for Redis**

Azure Cache for Redis se utiliza en diversos escenarios para mejorar el rendimiento de las aplicaciones mediante el soporte de patrones de arquitectura comunes. Algunos de los escenarios más comunes incluyen:

1. Caché de datos: Permite cargar datos en la caché solo cuando sea necesario, utilizando el patrón cache-aside. Cuando se realizan cambios en los datos, la caché se actualiza y se distribuye a otros clientes.

2. Caché de contenido: Se utiliza para acceder rápidamente a contenido estático, como encabezados, pies de página y banners de páginas web generadas a partir de plantillas. Esto es más eficiente en comparación con el acceso a los almacenes de datos de back-end.

3. Almacén de sesión: Se utiliza para asociar información a un usuario, como carros de compra y otros datos del historial de usuarios en aplicaciones web. Almacenar demasiados datos en una cookie puede afectar el rendimiento, por lo que utilizar una caché en memoria como Azure Cache for Redis puede ser más rápido que interactuar con una base de datos relacional completa.

4. Almacenamiento en cola de trabajos y mensajes: Se utiliza para encolar tareas cuando las operaciones asociadas a una solicitud tardan en ejecutarse. Las operaciones de ejecución prolongada se ponen en cola y se procesan secuencialmente, a menudo por otro servidor.

5. Transacciones distribuidas: Permite ejecutar una serie de comandos en un almacén de datos de back-end como una operación única y atómica. Todos los comandos deben ser exitosos o revertirse al estado inicial. Azure Cache for Redis admite la ejecución de un lote de comandos como una transacción única.

Azure Cache for Redis ofrece diferentes niveles de servicio:

1. Básico: Una memoria caché de OSS Redis que se ejecuta en una sola máquina virtual. No tiene contrato de nivel de servicio (SLA) y es ideal para cargas de trabajo de desarrollo y pruebas no críticas.

2. Estándar: Una memoria caché de OSS Redis que se ejecuta en dos máquinas virtuales en una configuración replicada.

3. Premium: Memorias caché de OSS Redis de alto rendimiento que ofrecen mejor rendimiento, menor latencia, mayor disponibilidad y más características. Se implementan en máquinas virtuales más eficientes en comparación con los niveles Básico y Estándar.

4. Enterprise: Memorias caché de alto rendimiento con la tecnología del software Redis Enterprise de Redis Labs. Admite módulos de Redis y ofrece una mayor disponibilidad que el nivel Premium.

5. Enterprise Flash: Memorias caché de gran tamaño y rentabilidad basadas en el software Redis Enterprise de Redis Labs. Amplía el almacenamiento de datos de Redis en memoria no volátil en una máquina virtual, lo que reduce el costo total de la memoria por GB.

**Configuración de Azure Cache for Redis**

El proceso de creación y configuración de una instancia de Azure Cache for Redis implica los siguientes pasos:

1. Asignar un nombre único global para la instancia de caché, que se utilizará para generar una dirección URL pública.

2. Seleccionar la ubicación de la instancia de caché y asegurarse de que coincida con la ubicación de la aplicación que utilizará la caché.

3. Elegir el tipo de caché adecuado en función de las necesidades de tamaño, rendimiento y características. Puede consultar más detalles en la documentación de precios de Azure Cache for Redis.

4. Si se requiere, habilitar la agrupación en clústeres para dividir automáticamente el conjunto de datos entre varios nodos.

5. Para acceder a la instancia de Redis Cache como cliente, utilizar la herramienta de línea de comandos de Redis para interactuar con ella. La herramienta está disponible para plataformas Windows y otras plataformas a través de Azure Cache for Redis o las herramientas de línea de comandos de Redis para Windows.

6. Redis admite varios comandos, como ping, set, get, exists, type, incr, incrby, del y flushdb, que permiten interactuar con la caché y realizar operaciones como establecer valores, obtener valores, eliminar claves, etc.

7. Es posible agregar un tiempo de expiración a los valores almacenados en la caché mediante la configuración de un período de vida (TTL) para una clave. Cuando el TTL expira, la clave se elimina automáticamente de la caché.

8. Para conectarse a una instancia de Azure Cache for Redis desde un cliente, se necesita el nombre de host, el puerto y una clave de acceso. Esta información se puede obtener desde Azure Portal en la página de configuración y claves de acceso.

Es recomendable regenerar las claves de acceso periódicamente por motivos de seguridad, y se pueden usar tanto la clave principal como la secundaria para acceder a la caché.

Microsoft recomienda usar los niveles Estándar o Premium en sistemas de producción. El nivel básico es un sistema de nodo único sin replicación de datos ni Acuerdo de Nivel de Servicio.

**Interacción con Azure Cache for Redis mediante .NET**

Recuerde que Azure Cache for Redis es una solución eficaz para mejorar el rendimiento y la escalabilidad de las aplicaciones mediante el uso de una caché en memoria.

Para interactuar con Azure Cache for Redis mediante .NET, se utiliza una biblioteca cliente como StackExchange.Redis. Aquí se presenta un resumen de los pasos y conceptos clave:

1. Conexión a la caché de Redis: Se utiliza una cadena de conexión que incluye la dirección del host, el número de puerto y una clave de acceso. Esta cadena se pasa a StackExchange.Redis para establecer una conexión al servidor de Redis.

[cache-name].redis.cache.windows.net:6380,password=[password-here],ssl=True,abortConnect=False

2. Creación de una conexión: Se crea una instancia de la clase ConnectionMultiplexer de StackExchange.Redis, que administra las conexiones eficientemente. Se puede crear mediante ConnectionMultiplexer.Connect o ConnectionMultiplexer.ConnectAsync, pasando la cadena de conexión o ConfigurationOptions.

using StackExchange.Redis;

...

var connectionString = "[cache-name].redis.cache.windows.net:6380,password=[password-here],ssl=True,abortConnect=False";

var redisConnection = ConnectionMultiplexer.Connect(connectionString);

3. Acceso a una base de datos de Redis: Una vez que se tiene una instancia de ConnectionMultiplexer, se obtiene una instancia de IDatabase mediante el método GetDatabase(). Esta instancia permite ejecutar métodos para interactuar con la caché, como almacenar valores, obtener valores y realizar otras operaciones comunes.

IDatabase db = redisConnection.GetDatabase();

4. Ejecución de otros comandos: Además de los métodos básicos de almacenamiento y recuperación de valores, se pueden ejecutar otros comandos en la caché utilizando el método Execute o ExecuteAsync de IDatabase. Estos comandos se pasan como texto y se puede obtener el resultado correspondiente.Ej:

var result = db.Execute("ping");

Console.WriteLine(result.ToString()); // displays: "PONG"

5. Almacenamiento de valores más complejos: Redis es compatible con cadenas binarias, pero también es posible almacenar objetos serializados en formato XML o JSON. Se pueden utilizar bibliotecas como Newtonsoft.Json para convertir objetos en cadenas y viceversa.

6. Limpieza de la conexión: Una vez que se ha finalizado el uso de la conexión de Redis, se debe llamar al método Dispose() en la instancia de ConnectionMultiplexer para cerrar las conexiones y finalizar la comunicación con el servidor.

Es importante recordar que la biblioteca StackExchange.Redis ofrece una amplia gama de métodos y operaciones para interactuar con Azure Cache for Redis de manera eficiente y flexible.

Modulo 2

**Exploración de instancias de Azure Content Delivery Network**

Azure Content Delivery Network (CDN) es una solución global que ofrece entrega rápida de contenido mediante el almacenamiento en caché del contenido en nodos físicos distribuidos estratégicamente. Algunas características y ventajas de Azure CDN incluyen:

- Mejor rendimiento y experiencia para los usuarios finales al reducir los tiempos de carga y minimizar los recorridos de red.

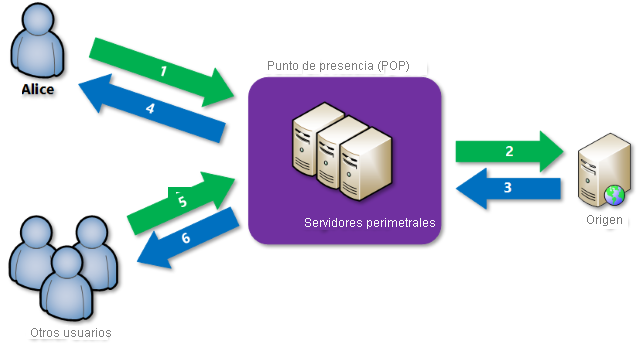
- Escalabilidad para manejar cargas instantáneas pesadas y eventos de alto tráfico.

- Distribución de las solicitudes de usuario y entrega del contenido desde servidores perimetrales, lo que reduce la carga en el servidor de origen.

¿Como funciona?

1.Utiliza una dirección URL especial (<endpoint name>.azureedge.net) para enrutar las solicitudes a la ubicación POP más cercana al usuario.

2. Almacena en caché archivos en los servidores perimetrales y los entrega desde la memoria caché para una respuesta más rápida.



- Requiere la creación de un perfil de CDN que contiene puntos de conexión de CDN para configurar el comportamiento de la entrega de contenido.

- Tiene límites predeterminados en el número de perfiles de CDN, puntos de conexión y dominios personalizados según la suscripción de Azure.

- Ofrece características como aceleración de sitios dinámicos, reglas de caché, soporte para dominios personalizados de HTTPS, registros de diagnóstico de Azure, compresión de archivos y filtrado geográfico.

En resumen, Azure CDN proporciona una manera eficiente de entregar contenido a los usuarios finales en todo el mundo, mejorando el rendimiento, la escalabilidad y la experiencia del usuario.

**Control del comportamiento de la memoria caché en Azure Content Delivery Network**

El control del comportamiento de almacenamiento en caché es importante para asegurar que el contenido almacenado en caché esté actualizado. Azure CDN ofrece opciones de configuración para el almacenamiento en caché, como reglas de caché y almacenamiento en caché de cadena de consulta. Estas opciones permiten establecer reglas globales o personalizadas para determinar cómo se almacena en caché y se responde a las solicitudes de contenido.

El período de vida de un recurso almacenado en caché se define mediante el encabezado Cache-Control en la respuesta HTTP del servidor de origen. Azure CDN establece valores predeterminados de período de vida, pero estos valores pueden ser reemplazados mediante reglas de almacenamiento en caché. Los valores predeterminados varían según la optimización de entrega web generalizada, grandes archivos y streaming multimedia.

Para actualizar el contenido en la red CDN, se puede incluir una cadena de versión en la URL del recurso o depurar contenido almacenado en caché en los nodos perimetrales. La depuración del contenido puede realizarse de forma individual en cada punto de conexión o en todos los puntos de conexión a la vez. Se pueden utilizar caracteres comodín o la raíz para especificar los recursos que se desean depurar. La CLI de Azure proporciona comandos para purgar o cargar previamente los recursos almacenados en caché en un punto de conexión.

Además, Azure CDN ofrece la capacidad de filtrado geográfico para autorizar o bloquear contenido en países o regiones específicas, utilizando códigos de país o región. Sin embargo, las opciones de filtrado geográfico varían según el plan de tarifa utilizado.

En resumen, Azure CDN ofrece opciones y herramientas para controlar el almacenamiento en caché, actualizar el contenido y aplicar filtrado geográfico, lo que permite mejorar la entrega de contenido y la experiencia del usuario.

**Interacción con instancias de Azure Content Delivery Network mediante .NET**

La interacción con las instancias de Azure Content Delivery Network (CDN) mediante .NET se puede realizar utilizando la biblioteca de Azure CDN para .NET. Para comenzar, es necesario crear un cliente de CDN mediante la clase CdnManagementClient.

El proceso general implica la creación y administración de perfiles y puntos de conexión de CDN. Para listar los perfiles y puntos de conexión existentes, se utiliza el método ListByResourceGroup para obtener una lista de perfiles y luego se recorre esa lista para mostrar la información correspondiente.

Para crear un perfil de CDN, se utiliza el método Create de la clase Profiles, pasando los parámetros necesarios, como el nombre del perfil, la ubicación y el SKU. Si el perfil ya existe, se verifica antes de crearlo.

Una vez creado el perfil, se procede a crear un punto de conexión utilizando el método Create de la clase Endpoints. Se especifican los parámetros requeridos, como los orígenes, si se permite el uso de HTTP o HTTPS y la ubicación. Al igual que con el perfil, se verifica si el punto de conexión ya existe antes de crearlo.

Otra tarea común es purgar el contenido de un punto de conexión. Esto se puede lograr utilizando el método PurgeContent de la clase Endpoints, pasando el nombre del punto de conexión y una lista de rutas de acceso que se desean purgar.

az group create --name az204-redis-rg --location East US

En resumen, la interacción con las instancias de Azure CDN mediante .NET implica la creación de un cliente de CDN, la lista y creación de perfiles y puntos de conexión, y la posibilidad de purgar el contenido de un punto de conexión existente.