# **AZ-204**

# Module 01: Crear Azure App Service web apps

## Implementar Azure App Service Web Apps

### Explorar Azure App Service

Servicio basado en HTTP, donde podemos alojar aplicaciones web, RestAPI, BackEnd de aplicaciones móviles.

Escalación y Des escalación automática.

**CI/CD**

* Azure DevOps
* GitHub
* Bitbucket
* FTP
* Local Git repository
* And others

### App Service plan

Servicio computacional (por debajo), con diferentes planes. (Servicio de cómputo + Características). Es un recurso que se utiliza para alojar las App Services (siempre debe existir este recurso para alojar apps)

App Service Plan, se miden para calcular el costo:

* CPU
* Memoria
* Disco
* Funcionalidad

Por cada ASP se pueden crear tantas aplicaciones como el plan permita.

Cada plan de App Service define:

* Sistema operativo (Windows, Linux)
* Región (oeste de EE. UU., este de EE. UU., etc.)
* Número de instancias de VM
* Tamaño de las instancias de VM (pequeño, mediano, grande)
* Plan de tarifa (Gratis, Compartido, Básico, Estándar, Premium, PremiumV2, PremiumV3, Aislado y AisladoV2)

El **plan de tarifa** de un plan de App Service determina qué características de App Service obtendrá y cuánto paga por el plan.

**Categorías Plan de tarifa** (determina qué características de App Service obtendrá y cuánto paga por el plan).

* **Proceso de compartido**: Gratis y Compartido, son los más básicos, se ejecutan en recursos compartidos con otros usuarios de Azure. No escala horizontalmente.
* **Dedicated compute (Proceso dedicado)**: (Básico, Estándar, Premium, PremiumV2 y PremiumV3) las aplicaciones se ejecutan en recursos dedicados, las aplicaciones del mismo plan comparten recursos.
* **Aislado**: (Aislado y AisladoV2) proporciona aislamiento de red, y aislamiento de procesos. Máximas posibilidades de escalabilidad horizontal.

**Redundancia por zona**: cuando un data center se cae, dentro de la misma zona, se utilizan otros data centers.

### Escalamiento

Cuando una aplicación tiene muchos Request, por lo que se debe incorporar más recursos.

El **escalamiento horizontal** implica la incrementación de más App Service Plan (más recursos de cómputo) con todo lo que tenga.

El **escalado vertical** es otorgar más o menos recursos, implica para este caso cambiar el plan.

En planes básicos el escalado es manual.

### Ranuras de implementación

Son aplicaciones activas con sus propios nombres de host. Los elementos de contenido y configuraciones de aplicaciones se pueden intercambiar entre dos ranuras de implementación, incluida la ranura de producción. Puede implementar la aplicación en un entorno de ensayo y, a continuación, intercambiar los espacios de ensayo y producción. Nos permiten crear las implementaciones en distintos ambientes sin tener que modificar las aplicaciones.

### App Service Web App

Es la aplicación ejecutándose, dentro de un App Service Plan. (Tipo de aplicación, Lenguaje usado, es contenedor o no).

[Creación de una aplicación web HTML estática mediante Azure Cloud Shell](./Resource/Module01/CreateStaticWebApp.mkv)

### Implementación automatizada y manual

Implementación automatizada: (o implementación continua) proceso que se usa para insertar nuevas características y correcciones de errores en un patrón repetitivo y rápido con un efecto mínimo en los usuarios finales.  
Se admiten varios orígenes:

* **Azure DevOps Services**: inserta el código en Azure DevOps Services, compila, ejecuta pruebas, genera una versión, e inserta el código en una aplicación web de Azure.
* **GitHub**: directamente desde GitHub. Los cambios que se insertan en la rama de producción en GitHub se implementan automáticamente.
* **Bitbucket**: similar a GitHub.

**Implementación manual**

* **Git**: dirección URL de Git que se puede agregar como repositorio remoto. Al insertar en el repositorio remoto, se implementa la aplicación.
* **CLI**: característica de la interfaz de la línea de comandos que empaqueta la aplicación y la implementa. az webapp up puede crear una aplicación web de App Service de forma automática si todavía no ha creado una.
* **Implementación desde un archivo Zip**: use curl o una utilidad HTTP similar para enviar un archivo ZIP de los archivos de la aplicación a App Service.
* **FTP/S**: FTP o FTPS es una manera tradicional de insertar el código en muchos entornos de hospedaje, incluido App Service.

## Registros de aplicaciones Web

[Introducción - Training](https://learn.microsoft.com/es-mx/training/modules/capture-application-logs-app-service/1-introduction)

Forma simple de capturar resultados del registro de aplicaciones básicos sin agregar un SDK especializado, como Azure Application Insights.

Son el resultado de las instrucciones de seguimiento en tiempo de ejecución en el código de aplicación.

Tiene limitaciones de escala, sobre todo porque se usan ***archivos*** para guardar los resultados registrados. Si tiene varias instancias de una aplicación y se comparte el mismo almacenamiento en todas ellas, es posible que se intercalen mensajes procedentes de distintas instancias, lo que dificulta la solución de problemas.

### ASP.NET

Para registrar información en el registro de diagnóstico de aplicaciones, use la clase System.Diagnostics.Trace. Existen cuatro niveles de seguimiento que se pueden usar, y que se corresponden con los niveles de registro error, warning, information y verbose que se muestran en Azure Portal:

* Trace.TraceError("Message"); // Escribe un mensaje de error
* Trace.TraceWarning("Message"); // Escribe un mensaje de advertencia
* Trace.TraceInformation("Message"); // Escribe un mensaje de carácter informativo
* Trace.WriteLine("Message"); // Escribe un mensaje detallado

### Aplicaciones ASP.NET Core

se pueden ejecutar tanto en Windows como en Linux. Para registrar información en los registros de la aplicación de Azure, use la clase **LoggerFactory** y, después, use uno de los seis niveles de registro disponibles:

* logger.LogCritical("Mensaje"); // Escribe un mensaje crítico en el nivel de registro 5
* logger.LogError("Mensaje"); // Escribe un mensaje de error en el nivel de registro 4
* logger.LogWarning("Mensaje"); // Escribe un mensaje de advertencia en el nivel de registro 3
* logger.LogInformation("Mensaje"); // Escribe un mensaje de carácter informativo en el nivel de registro 2
* logger.LogDebug("Message"); // Escribe un mensaje de depuración en el nivel de registro 1
* logger.LogTrace("Mensaje"); // Escribe un mensaje de seguimiento detallado en el nivel de registro 0

En el caso de las aplicaciones ASP.NET Core en Windows, estos mensajes se relacionan con los filtros de Azure Portal del siguiente   
modo:

* Los niveles 4 y 5 son mensajes de "error".
* El nivel 3 es un mensaje de "advertencia".
* El nivel 2 es un mensaje de "carácter informativo".
* Los niveles 0 y 1 son mensajes "detallados".

En cuanto a las aplicaciones ASP.NET Core en Linux, solo se registran los mensajes de "error" (es decir, los niveles 4 y 5).

## **Aplicaciones Node.js**

aplicaciones web basadas en scripts, como las aplicaciones Node.js en Windows o Linux, el registro de aplicaciones se habilita con el método **console()**:

* console.error("Message"): escribe un mensaje en STDERR
* console.log("Message"): escribe un mensaje en STDOUT

Ambos tipos de mensaje se escriben en los registros de nivel de error de Azure App Service.

En la siguiente tabla se resume la compatibilidad de registro en los hosts y entornos de aplicaciones más conocidos.

| **Entorno de la aplicación** | **Host** | **Niveles de registro** | **Ubicación de almacenamiento** |
| --- | --- | --- | --- |
| ASP.NET | Windows | Error, Advertencia, Información, Detallado | Sistema de archivos, Blob Storage |
| ASP.NET Core | Windows | Error, Advertencia, Información, Detallado | Sistema de archivos, Blob Storage |
| ASP.NET Core | Linux | Error | Sistema de archivos |
| Node.js | Windows | Error (STDERR), Información (STDOUT), Advertencia, Detallado | Sistema de archivos, Blob Storage |
| Node.js | Linux | Error | Sistema de archivos |
| Java | Linux | Error | Sistema de archivos |

## **Alternativas a los diagnósticos de aplicaciones**

Azure Application Insights es una extensión de sitio que proporciona más características de supervisión de rendimiento (como datos detallados sobre el uso y el rendimiento), y está pensada para usarse en implementaciones de aplicaciones de producción, además de estar considerada una herramienta de desarrollo útil. Para usar Application Insights hay que incluir código específico dentro de la aplicación mediante el SDK de App Insights. Application Insights también es un servicio facturable, de modo que, dependiendo de la escala de las implementaciones de aplicaciones y de los datos recopilados, puede que haya que hacer una previsión de costes periódicos.

## Azure Cli

Para abrir el flujo de registro,

az webapp log tail --name <app name> --resource-group <resource group name>

Crear un conjunto de credenciales de nivel de usuario

az webapp deployment user set --user-name <name-of-user-to create> --password <new-password>

Abrir el flujo de registro

curl -u {username} https://{sitename}.scm.azurewebsites.net/api/logstream

Para descargar archivos de registro del sistema de archivos

### Autenticación y Autorización

Funcionalidad de App Service, implementación sencilla, que requiere de autenticación.   
Marcos web incluidos:

* Azure AD
* Facebook
* Google
* Twitter

### Características de Networking

Debe controlar el tráfico de red entrante y saliente.  
Dentro de App Service hay dos implementaciones principales

* **Multi-tenant**: Multi inquilino, muchos clientes diferentes en la misma unidad de escalado.
* **Single-Tentant** App Service Environment (ASE) de un único inquilino

# Module 02: Implementando Azure Functions

## Durable Functions

Extensión de Azure Functions que permite escribir funciones con estado en un entorno de proceso sin servidor. La extensión permite definir flujos de trabajo con estado mediante la escritura de funciones del orquestador y entidades con estado mediante la escritura de funciones de entidad con el modelo de programación de Azure Functions.

### Patones utilizados

* **Patrón nº 1: Encadenamiento de funciones** una secuencia de funciones se ejecuta en un orden específico. La salida de una función se aplica a la entrada de otra función.
* **Patrón nº 2: Distribución ramificada de salida y de entrada** se ejecutan en paralelo varias funciones y después se espera a que todas finalicen.
* **Patrón nº 3: Las API de HTTP asincrónico** soluciona el problema de coordinar el estado de las operaciones de larga duración con los clientes externos.
* **Patrón 4: Supervisión** proceso flexible y periódico en un flujo de trabajo, hasta que se cumplen condiciones específicas. Puede usarse cuando se necesita que una API externa modifique su valor de retorno.
* **Patrón nº 5: Interacción humana** Un proceso de aprobación es un ejemplo de un proceso empresarial que implica la interacción humana.
* **Patrón 6: Agregador (entidades con estado)** trata de agregar datos de eventos durante un período de tiempo en una solaentidad direccionable. En este patrón, los datos que se agreguen pueden proceder de varios orígenes. Es posible que el agregador tome medidas según datos de eventos a medida que llegan, y puede que los clientes externos necesiten consultar los datos agregados.

## Azure Functions

Código de diferentes lenguajes) para que se ejecuten en determinado momento. Reciben un trigger(manejado por un Storage Account) que actúa como disparador de la ejecución del código. [Azure Function como núcleo de servicios](/Resource/Module02/AzureFunctions.png)

Se puede desarrollar local y luego subirlo dentro de un Function App. Se usan para ejecutar en un periodo muy corto de tiempo.

Existe 3 planes(hosting) indica el tamaño de la VM, como escala y desescala:

* **Consumo (o Serverless)**: paga por lo que usos, Micrososft escala por consumo, y si no se usa, Microsoft lo desescala. Luego de 20 minutos sin ser usado, el servicio se apaga. Al recibir un Request, se levanta el servicio y responde (lo que hace más lenta la respuesta)
* **Premiun**: se puede elegir (en máquinas virtuales) los recursos, escala y desescala de manera automática según la necesidad, pero, vienen precalentadas (Always On siempre encendido), ya que se paga un precio fijo.
* **Dedicado**: ejecuta las Azure Functions dentro de un App Service Plan(se debe elegir el plan del App Service). Teniendo control sobre el escalado y desescalado (se usan reglas), Always On/Off, precio fijo.

Los planes antes mencionados (Consumo, Premium, Dedicado) son los principales.

También existen 2 planes más de hospedaje:

* **Azure App Service Environment**: o (ASE) entorno completamente aislado y dedicado para ejecutar de forma segura aplicaciones a gran escala.
* **Kubernetes**: (Directo o Azure Arc) entorno aislado y dedicado que se ejecuta sobre la plataforma de Kubernetes.

### Time Out

Por defecto es de 5 minutos, y como máximo puede definirse 10 minutos. Esta propiedad se define en el archivo host.json.  
El tiempo puede ser mayor en planes Premium y Dedicados donde por defecto tienen definido 302 minutos.

### Cuenta de almacenamiento

Al crear una Function App, <mark>SIEMPRE</mark> se debe crear un Storage Account de uso general, ya que las funciones guardan logs de estados.  
Se permiten hasta 3 copias en zona redundante en la zona primaria.

### Controlador de escalado

O Scale Controller (solo para Premium y Por consumo), automáticamente monitorea (revisa la cola de request de) las funciones y determina cual requiere más recursos(instancias) para atender las peticiones, a medida que disminuyen las peticiones, va realizando el desescalaiento.

El máximo (y por defecto) de instancias que se puede escalar es de 200 instancias. Es posible definirle un número menor en el archivo de configuración.

Se debe considerar que las Function App, deben desescalar a todas las funciones contenidas, por lo que se debe considerar que si una función convienen colocarla aislada de otras funciones.

### Controles Personalizados

Los controladores personalizados son servidores web ligeros que reciben eventos del host de Functions. Cualquier lenguaje que admita primitivas de HTTP puede implementar un controlador personalizado.

Los controladores personalizados son más adecuados para las situaciones en las que desea:

* Implemente una aplicación de funciones en un lenguaje que actualmente no se ofrece de manera inmediata, como Go o Rust.
* Implemente una aplicación de funciones en un runtime que actualmente no se incluye de manera predeterminada, como Deno.

### Trigger

CRON utiliza un formato de "{segundo} {minuto} {hora} {día} {mes} {día de la semana}" para las expresiones. El primer "0" significa que corre cuando el segundo es igual a 0. El segundo "15,30,45" significa cuando los minutos son igual a 15, 30 y 45. El tercer "0" significa a la medianoche. Así que la respuesta es a las 12:15, 12:30 y 12:45 todos los días.

### Resumen

La función tiene dos componentes principales, la función (en un lenguaje determinado, por ejemplo, c#)

Archivo de configuración llamado function.json, que define cual es el disparador(trigger) de la función, y cuál es el binding(conexiones a otros servicios) de la función.

Una función tiene 1 único trigger, pero puede tener varios bindings.

Los binding pueden ser:

* IN
* OUT
* IN/OUT

# Module 03: Blob storage

## Blob Storage

Solución de almacenamiento de objetos. Está optimizado para el almacenamiento de cantidades masivas de datos no estructurados. Cuenta con varios servicios de almacenamiento que podemos tener en una cuenta, por lo que es necesario crear una cuenta de Storage. Es el tipo más utilizado, para el desarrollo de aplicaciones, por ejemplo imagenes para mostrar en una pagina web. Se guardan en espacios logicos, es decir, en contenedores.

Ofrece dos niveles de rendimiento de cuentas de almacenamiento, estándar y premium.

* **Estándar:** se trata de la cuenta de uso general v2 estándar y se recomienda para la mayoría de los escenarios en los que se usa Azure Storage.

Dentro del nivel ***Standard***: Cuenta con:  
**Files**: similar a File Server, donde se comparten archivos mediante SMB. Tambien se pueden realizar llamadas REST. SE puede sincronizar tanto con servidores OnPremise como con servidores onCloud.

* **Premium:** (son cuentas exclusivas) no se comparte con otros servicios, para ser más efectivo y con mejor renidimiento. Si crea una cuenta Premium, puede elegir entre **tres tipos de cuenta**:
  + **blobs en bloques**: de bloque, se guardan secuencialemente, se usa para videos, imagenes, videos.
  + **blobs en páginas**: utilizados para guardar discos(no administrados) de maquinas virtuales.
  + **recursos compartidos de archivos**: similar a File server(SMB)

**Append**: utilizados guardar archivos secuenciales, por ejemplo guardar logs, en donde se incorpora información a la ya existente.

**Tables**: Base de datos NoSQL, para guardar información de tipo llave:valor. Escalan dinaicamente, son de respuesta rapidas, siempre y cuando la información que se almacene sea especifica, como ser configuracions.

**Queues**: Cola de mensajes, para encolar mensajes y posteriormente ir desencolando. La idea es desacoplar servicios, para tener bajar interdependencia entre aplicaciones o servicios.

### Permisos de Acceso al Blob Storage

* Permisos a nivel contenedor
* Permisos basados RBAC(Role-Based Acces Control)

La seguridad se aplica en transito, con SSL/TLS.

Toda la información se encripta en reposo, y no se puede deshabilitar, solo se permite asignar quien administrara las llaves de des-encriptacion.

### Ciclo de Vida de Blob Storage

Existen 4 capas de acceso:

* Premiun(visto antes): No es posible cambiar a otras capas.

Es posible intercambar entre capas

* **Hot**(Dentro de Blob Standard): para acceder a los archivos de manera frecuente. Alto costo de almacenamiento es caro, bajo costo de acceso.
* **Cold**(Dentro de Blob Standard): el acceso es para archivos no tan frecuentes. Menor costo de almacenamiento, y mayor costo de acceso(Comparado a HOT).
* **Archive**(Dentro de Blob Standard): sin acceso al archivo, pero se almacena por temas de cumplimiento o auditorias. Es la capa menos costosa de almacenamiento, pero la más costosa de acceder. Para acceder a un archivo se debe "rehidratar" el archivo(proceso que puede horas en base a la prioridad [Alta o Normal]) pasandolo a capa Cold o Hot.

### Detección de directivas de ciclo de vida de Blob Storage

Colección de reglas en un documento JSON. Cada definición de regla incluye un conjunto de filtros y un conjunto de acciones.

{

"rules": [

{

"name": "rule1",

"enabled": true,

"type": "Lifecycle",

"definition": {...}

},

{

"name": "rule2",

"type": "Lifecycle",

"definition": {...}

}

]

}

**rules**: Se requiere al menos una regla en una directiva. Puede definir hasta 100 reglas en una directiva.

**name**: String Un nombre de regla puede incluir hasta 256 caracteres alfanuméricos. El nombre de regla distingue mayúsculas de minúsculas. Debe ser único dentro de una directiva. True  
**enabled**: Boolean Un valor booleano opcional para permitir que una regla se deshabilite de forma temporal. El valor predeterminado es true si no se establece. False  
**type**: Un valor de enumeración.El tipo válido actual es Ciclo de vida. True  
**definition** Un objeto que define la regla del ciclo de vida Cada definición se compone de un conjunto de filtros y un conjunto de acciones. True

### Acciones de regla

Las acciones se aplican a los blobs filtrados cuando se cumple la condición de ejecución.

**Acción**:

* **tierToCool** Compatible con blockBlob
* **enableAutoTierToHotFromCool** Compatible con blockBlob
* **tierToArchive**: Compatible con blockBlob  
  **delete**Compatible con blockBlob y appendBlob

Para agregar una directiva de administración del ciclo de vida con la CLI de Azure, escriba la directiva en un archivo JSON y, después, llame al comando az storage account management-policy create para crear la directiva.

### Azure File Sync

Permite mantener copia locales de archivos al mismo tiempo que en la nube. Utiliza protocolos como SMV, NFS, FTPS.

### Rehidratación de blob desde el nivel de archivo

Un blob se encuentra en el nivel de acceso de archivo, se considera que está sin conexión. Para leer o modificar los datos de un blob archivado, primero debe rehidratar el blob en un nivel en línea, ya sea el nivel de acceso frecuente o esporádico. Hay dos opciones para rehidratar un blob que se almacena en el nivel de archivo:

* **Copiar un blob archivado en un nivel en línea** Copy blob
* **Cambio del nivel de acceso de un blob a un nivel en línea** Set Blob Tier

#### Prioridad

Puede establecer la prioridad de la operación de rehidratación a través del encabezado x-ms-rehydrate-priority

## Trabajar a nivel de Código

Libreria cliente azure.storage.blob

Por orden de llamada:

1. **Blob Service Client**: se utiliza para acceder a la StorageAccont, se le proporciona el ConnectionString.
2. **Blob Container Client**: ya conectado a la cuenta de Storage, se utiliza este cliente para trabajar con contenedores.
3. **Blob Client**: para trabajar con archivos, permite eliminar, descargar, crear archivos dentro de un contenedor, que a su vez esta dentro de un Storage Account.
4. **Append Blob Client**: para manipular archivos de tipo log.
5. **Block Blob Client**: para trabajar con archivos de tipo bloque.

### Operaciones con metadatos

Los metadatos de un recurso de blob o de un recurso contenedor se pueden recuperar o establecer directamente, sin devolver ni modificar el contenido del recurso.

**Recuperación de propiedades y metadatos**

GET/HEAD

**Establecer encabezados de metadatos**

PUT

### Propiedades HTTP estándar para contenedores y blobs

Los nombres de los encabezados de metadatos están formados por el prefijo de encabezado x-ms-meta- y un nombre personalizado.

Los nombres de propiedades utilizan nombres de encabezado HTTP estándar HTTP/1.1

Los encabezados HTTP estándar admitidos en los contenedores son los siguientes:

* ETag
* Last-Modified

Los encabezados HTTP estándar admitidos en los contenedores son los blobs:

* ETag
* Last-Modified
* Content-Length
* Content-Type
* Content-MD5
* Content-Encoding
* Content-Language
* Cache-Control
* Origin
* Range

# Module 04: Azure Cosmos DB storage

Base de Datos NoSQL, semi estructurada con replicacion global. Es decir, puedo tener múltiples replicas en diferentes regiones. Diseñada para proporcionar una latencia baja, una escalabilidad elástica del rendimiento

**Consistencia**: se refiere a la consistencia de la información en cada replica. Esto afecta directamente a la performance.

Azure Cosmos DB ofrece cinco niveles bien definidos. De más fuerte a más débil, los niveles son:

* **Fuerte**: capacidad de servir solicitudes simultáneamente. Se garantiza que las lecturas devuelven la versión más reciente de un elemento.
* **Uso vinculado**:
* **Sesión**: as escrituras se replican en un mínimo de tres réplicas (en un conjunto de cuatro réplicas) en la región local, con replicación asincrónica en todas las demás regiones.
* **Prefijo coherente**: la escritura se realiza en lotes separados. Solo al leer el lote en conjunto se obtiene la coherencia completa.
* **Ocasional**:

**Baja latencia**: tanto para escritura como para lectura, hay una respuesta menor a los 10ms en el 99% de los casos.

### Jerarquía de servicio

**Azure Cosmo DB Account**: unidad fundamental de distribución global y alta disponibilidad, puede agregar y quitar regiones de Azure en su cuenta en cualquier momento.

**Azure Cosmos DB database**: Una base de datos es análoga a un espacio de nombres. Una base de datos es la unidad de administración de un conjunto de contenedores.

**Azure Cosmos DB containers**: Permite la creacion de tablas, Store Procedures, funciones definidas por el usuario, triggers. Es la unidad de escalabilidad del rendimiento y almacenamiento aprovisionados. Un contenedor se divide de forma horizontal y luego se replica en varias regiones.

**Azure Cosmos DB items**: En función de la API que use, un elemento de Azure Cosmos DB puede representar un documento de una colección, una fila de una tabla, o un nodo o un borde de un grafo. Se refiere a los registros(document, row, node, edge).

### API para Cosmos DB

Azure Cosmos DB ofrece varias API de base de datos, entre las que se incluyen las siguientes:

* **Azure Cosmos DB para NoSQL** almacena datos en formato de documento, permiten realizar consultas mediante la sintaxis del Lenguaje de consulta estructurado (SQL).
* **Azure Cosmos DB for MongoDB** almacena los datos en una estructura de documentos a través del formato BSON.
* **Azure Cosmos DB para PostgreSQL** Almacena datos en un único nodo o distribuidos en una configuración de varios nodos.
* **Azure Cosmos DB for Apache Cassandra** almacena los datos en el esquema orientado a columnas, escalado horizontal y altamente distribuido para almacenar grandes volúmenes de datos
* **Azure Cosmos DB for Table** almacena datos en formato clave-valor.
* **Azure Cosmos DB for Apache Gremlin** permite a los usuarios realizar consultas de grafos y almacenar datos como bordes y vértices.

### Unidades de solicitud (Request Units [RU])

Azure Cosmos DB normaliza el costo de todas las operaciones de base de datos y este se expresa en unidades de solicitud (o RU, para abreviar). Una unidad de solicitud representa los recursos del sistema, como CPU e IOPS, y la memoria que se necesitan para realizar las operaciones de base de datos

El tipo de cuenta de Azure Cosmos DB que usa determina el modo en que se cobrarán las RU consumidas.

Hay tres modos en los que se puede crear una cuenta:

* **Modo de rendimiento aprovisionado**:se realiza por segundos en incrementos de 100 RU/segundo. Para escalar el rendimiento aprovisionado para la aplicación, puede aumentar o disminuir el número de RU en cualquier momento, en incrementos o decrementos de 100 RU.
* **Modo sin servidor**: no es necesario aprovisionar rendimiento al crear recursos, factura el número de unidades de solicitud consumidas por las operaciones de base de datos.
* **Modo de escalabilidad automática**: puede escalar de forma automática e instantánea el rendimiento (RU/s) de la base de datos o del contenedor en función de su uso. Adecuado para cargas de trabajo críticas que tienen patrones de tráfico variables o imprevisibles y requieren SLA para el alto rendimiento y la escala.

[Creación de CosmosDB Account + DB + Container + Item](./Resource/Module04/CreateCosmosDBforNoSQL.mkv)

# Module 05: Contenedores (IAAS)

## Azure Container Registry(ACR)

Permite guardar en un repositorio privado imagenes para utilizar en nuestra organizacion. Almacenar las imágenes de contenedor y automatizar compilaciones e implementaciones.. Basado en Docker Registry 2.0, soporte para Docker, Helm, OCI (Open Container Initiative).

### Niveles de ACR

Ofrecen precios predecibles y varias opciones para alinearse con la capacidad y los patrones de uso de su registro de Docker privado:

* **Básico**: optimizado para los costos para que los desarrolladores aprendan sobre Azure Container Registry. Los registros básicos tienen las mismas funcionalidades de programación que Estándar y Premium (como integración de la autenticación de Azure Active Directory, la eliminación de imágenes y webhooks).
* **Estándar**: Mismas funcionalidades del nivel basico, pero con más almacenamiento y mayor rendimiento en las imagenes. Ideal para la mayoria de necesidades en escenarios productivos.
* **Premiun**: Mayor cantidad de almacenamiento incluido y operaciones simultaneas, por lo que permiten trabajar con escenarios de mayor volumen. Agrega replicacion geografica, vinculo privado con puntos de conexion privados.

### Procedimientos recomendados

* **Implementación cercana a la red**: Cree el registro de contenedor en la misma región de Azure en la que implementa los contenedores.
* **Replicación geográfica en varias regiones** Permite minimizar la latencia mediante la replicación geográfica del registro. ambién puede configurar **webhooks** regionales para que le envíen notificaciones cuando se produzcan eventos en réplicas concretas, como cuando se insertan imágenes.
* **Espacios de nombres del repositorio**: puede permitir el uso compartido de un registro único en varios grupos de su organización. Azure Container Registry admite espacios de nombres anidados, lo que permite el aislamiento del grupo.

### Almacenamiento

ya sea Básico, Estándar o Premium, se gozan de características avanzadas:

**Cifrado en reposo**: Azure cifra automáticamente una imagen antes de almacenarla y la descifra sobre la marcha cuando usted o sus aplicaciones y servicios extraen la imagen.

**Almacenamiento regional**: almacena datos en la región donde se crea el registro, para ayudar a los clientes a cumplir los requisitos de cumplimiento y residencia de datos.

**Redundancia de zona**: característica del nivel Premium y usa zonas de disponibilidad de Azure para replicar el registro en un mínimo de tres zonas independientes en cada región habilitada.

**Almacenamiento escalable**: permite crear tantos repositorios, imágenes, capas o etiquetas como necesite, hasta el límite de almacenamiento del registro.

**Azure Container Registry Tasks** (ACR Tasks) para simplificar la creación, la prueba, el envío de cambios y la implementación de imágenes en Azure. Cree tareas de varios pasos para automatizar la compilación, prueba y aplicación de revisiones de varias imágenes de contenedor en paralelo en la nube. Configure las tareas de compilación para automatizar el sistema operativo del contenedor y la canalización de aplicaciones de revisión de marcos, y compile imágenes de forma automática cuando el equipo guarde el código en el control de origen.

Escenarios de uso:

**Tarea rapida**: compile e inserte una sola imagen de contenedor en un registro de contenedor a petición en Azure, sin tener que realizar una instalación local del motor de Docker.

Para compilar una imagen de contenedor desde el código de ejemplo.

az acr build --registry $ACR\_NAME --image helloacrtasks:v1 .

Para crear un almacen de claves

az keyvault create --resource-group $RES\_GROUP --name $AKV\_NAME

**Tareas desencadenadas automáticamente**: habilite uno o varios desencadenadores para compilar una imagen:

* Desencadenar al actualizar el código fuente
* Desencadenar al actualizar la imagen base
* Desencadenar de acuerdo con una programación

**Tarea de varios pasos**: amplíe la funcionalidad de inserción y compilación de una única imagen de ACR Tasks con flujos de trabajo basados en varios contenedores y varios pasos.

## DockerFile

es un script que contiene una serie de instrucciones que se usan para compilar una imagen de Docker. Suelen incluir la siguiente información:

* La imagen base o primaria que usamos para crear la nueva imagen.
* Los comandos para actualizar el sistema operativo base e instalar otro software.
* Los artefactos de compilación que se incluirán, como una aplicación desarrollada.
* Los servicios que se expondrán, como la configuración de red y del almacenamiento.
* El comando que se ejecutará cuando se inicie el contenedor.

**WebHooks**

## Container Instances

Permite trabajar con un contenedor o con una aplicación que esta contenida en varias aplicaciones (uso más comun).

* **Container Groups**: recurso de nivel superior de Azure Container Instances es el **grupo de contenedores**. Un grupo de contenedores es una colección de contenedores que se programan en la misma máquina host. Los contenedores de un grupo comparten un ciclo de vida, los recursos, la red local y los volúmenes de almacenamiento. Es similar en concepto a un pod en Kubernetes.
* **Implementación**: dos maneras comunes de implementar un grupo de varios contenedores:
* **plantilla de Resource Manager**: Se recomienda cuando se necesite implementar recursos adicionales de un servicio de Azure (por ejemplo, un recurso compartido de Azure Files) al implementar instancias de contenedor.
* **archivo YAML**: (similar a Docker Compose) Se recomienda cuando la implementación incluya solo instancias de contenedor.
* **Asignación de recursos**: Si bien es un modo serverless se especifica CPUs, memoria y opcionalmente GPUs, a un grupo de contenedores mediante la adición de las solicitudes de recursos de las instancias del grupo.
* **Redes**: se comparten direcciones IP y puertos.
* **Storage**: se utilizan "volumenes" concepto de contenedores, es un recurso de file server, como un espacio de disco. Que contienen información de aplicaciones. Sirve para tener un espacio de almacenamiento persistente. En Azure puede ser dentro de una Maquina Virtual, o tambien, puede ser un Azure Files dentro de un Storage Account.
* **Escenarios frecuentes**: útiles en los casos en los que se quiere dividir una sola tarea funcional en varias imágenes de contenedor. Luego, estas imágenes las pueden entregar diferentes equipos y pueden tener diversos requisitos de recursos. Considerar a docker build o docker push para compilar e insertar una sola imagen de contenedor(sin instalacion local).

## Comandos utilizados en CliShell

az group create para crear un grupo de recusos.  
az container create crea un contenedor.  
az container show comprueba el estado del contenedor ya creado.  
az group delete quita el grupo de recursos, el registro de contenedor y las imágenes de contenedor almacenadas allí.

### Directivas de reinicio

Cuando se crea un grupo de contenedores, se puede especificar una de tres configuraciones:

**Always**: Los contenedores del grupo de contenedores siempre se reinician. (valor de configuración predeterminado).

**Never**: Nunca se reinician los contenedores del grupo de contenedores. Los contenedores se ejecutan al menos una vez.

**OnFailure**: Los contenedores del grupo de contenedores se reinician solo cuando se produce un error en el proceso ejecutado en el contenedor (cuando se cierra con un código de salida distinto de cero). Los contenedores se ejecutan al menos una vez.

Se especifica con el parámetro --restart-policy al llamar a az container create.

## Azure Container Apps

Es un cluster ya preparado(por debajo se ejecuta Azure Kubernetes Services), que requiere de cierta condiguracion para funcionar. Permite aplicar microsevicios o aplicaciones contenerizadas en modos ServerLess. Admite cualquier imagen de contenedor x86-64 (linux/amd64) basada en Linux. No hay ninguna imagen de contenedor base necesaria y, si un contenedor se bloquea, se reinicia de forma automática.

Las aplicaciones basadas en Azure Container Apps se pueden escalar de forma dinámica en función del tráfico HTTP, el procesamiento controlado por eventos, la CPU o la carga de memoria

**Autenticación** integrada con proveedores de identidades federados. Esta característica solo se debe usar con HTTPS.

El componente de **middleware de autenticación y autorización** es una característica de la plataforma que se ejecuta como contenedor sidecar en cada réplica de la aplicación. Cuando está habilitado, cada solicitud HTTP entrante pasa por el nivel de seguridad antes de que la aplicación lo controle. El módulo de autenticación y autorización se ejecuta en un contenedor independiente, aislado del código de la aplicación.

Existen dos flujos de inicio de sesión:

* **Sin el SDK del proveedor**: la aplicación delega el inicio de sesión federado a Container Apps.
* **Con el SDK del proveedor**: la aplicación inicia manualmente la sesión del usuario en el proveedor y, luego, envía el token de autenticación a Container Apps para la validación.

### Microservicios

Las arquitecturas de microservicios permiten desarrollar, actualizar, controlar versiones y escalar áreas básicas de funcionalidad de forma independiente en un sistema general.

**Distributed Application Runtime (Dapr)**: simplifican la creación de aplicaciones distribuidas basadas en microservicios. Incluye características como la observabilidad, la publicación/suscripción y la invocación de servicio a servicio con TLS mutuo, reintentos, etc. Proporciona funcionalidades para habilitar la intercomunicación de aplicaciones mediante la mensajería a través de pub/sub o llamadas de servicio a servicio confiables y seguras.

**Container App Envaironment**: ambiente donde se crean y ejecuta Container Apps.

**Administración de Revisiones**: Una revisión es una instantánea inmutable de una versión de la aplicación de contenedor.az containerapp update para generar versiones, que permiten volver a revisiones anteriores, tambien se puede ejecutar 2 revisiones a la vez, hasta validar el funcionamiento de la nueva revision, y asi apagar la más antigua. Puede controlar qué revisiones están activas y el tráfico externo que se dirige a cada revisión activa. Los nombres de revisión se usan para identificar una revisión, y en la URL de la revisión.

az containerapp revision list : enumera todas las revisiones asociadas a la aplicación de contenedor.

# Module 06: Implementación de la autenticación y autorización de usuarios

## Plataforma de identidad(v2.0)

Es un conjunto de herramientas que incluye un servicio de autenticación, bibliotecas de código abierto y herramientas de administración de aplicaciones.

Permite realizar configuraciones desde el portal. Soporta Auth2.0 y OpenAI lo que permite autenticarse desde múltiples servicios corporativos, de servicios federados.

La plataforma esta formada por:

* Servicio de autenticación compatible con los estándares OAuth 2.0 y OpenID Connect, permite autenticar:
  + Cuentas profesionales o educativas
  + Cuentas personales de Microsoft
  + Cuentas sociales o locales
* Bibliotecas de código abierto
* Portal de administración de aplicaciones
* PowerShell y API de configuración de aplicaciones

**Service Principal**: objeto dentro del Active Directory. Al referirnos a aplicaciones creamos un Service Principal para registrar aplicaciones.

**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. De forma similar a una clase en la programación orientada a objetos, el objeto de aplicación tiene algunas propiedades estáticas que se aplican a todas las entidades de servicio creadas (o instancias de 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
* las acciones que puede realizar la aplicación

Al crear una aplicación en el portar se crea:  
ObjectID: contiene toda la configuración de seguridad en el tenant de la aplicación.  
Application ID: Id dentro del Azure Active Directory.

1 Object Id = 1 aplicación

1 aplicación = múltiples aplication ID

## Permisos

Se representa como un valor de cadena. Admite dos tipos de permisos:

**Delegados**: se utilizan en aplicaciones que tienen un usuario con la sesión iniciada, el usuario o un administrador dan su consentimiento para los permisos que la aplicación requiere.

**Aplicación**: los usan las aplicaciones que se ejecutan sin la presencia de un usuario con la sesión iniciada, aplicaciones que se ejecutan como demonios o servicios en segundo plano. Solo un administrador puede dar su consentimiento para los permisos de acceso exclusivo de aplicación.

**Consentimientos**:

* **Usuario estático**: debe especificar todos los permisos que necesita en la configuración de la aplicación en Azure Portal.
* **Incremental y dinámico**: solicita permisos de forma incremental, puede solicitar un conjunto mínimo de permisos por adelantado y solicitar más con el tiempo, a medida que el cliente use características de aplicación adicionales.
* **Administrador**: si la aplicación necesita acceder a determinados permisos con privilegios elevados.

## Acceso condicional

Permite que los desarrolladores y clientes empresariales protejan los servicios de diversas formas, entre las que se incluyen las siguientes:

* Autenticación multifactor
* Autorización para que solo los dispositivos inscritos en Intune accedan a servicios específicos
* Restricción de ubicaciones de usuario e intervalos IP

## MSAL (Microsoft Auth Library)

Liberia que se utiliza dentro de la aplicación a nivel de código. Soporta .net, JavaScript, Python, Android y iOS. Permite que los desarrolladores adquieran tokens desde la Plataforma de identidad de Microsoft para autenticar usuarios y acceder a API web protegidas.

es posible adquirir un token desde muchos tipos de aplicación: aplicaciones web, API web, aplicaciones de una sola página (JavaScript), aplicaciones móviles y nativas, así como aplicaciones del lado servidor y demonios.

### Flujos de autenticación

**Código de autorización**Las aplicaciones nativas y web obtienen tokens de forma segura en el nombre del usuario.  
**Credenciales de cliente**Las aplicaciones de servicio se ejecutan sin interacción del usuario.  
**En nombre de**La aplicación llama a una API web o de servicio, que a su vez llama a Microsoft Graph.  
**Implícita**Se usa en aplicaciones basadas en explorador.  
**Código del dispositivo**Habilita el inicio de sesión en un dispositivo mediante otro dispositivo que tiene un explorador.  
**Integrado en Windows**Los equipos Windows adquieren de forma silenciosa un token de acceso cuando están unidos a un dominio.  
**Interactive**Las aplicaciones móviles y de escritorio llaman a Microsoft Graph nombre de un usuario.  
**Nombre de usuario y contraseña**La aplicación inicia la sesión de un usuario con su nombre de usuario y contraseña.

**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).

**Aplicaciones cliente confidenciales**: se ejecutan en servidores (aplicaciones web, aplicaciones de API web o incluso aplicaciones de servicio o demonio). Se consideran de acceso difícil pueden mantener un secreto de aplicación.

## Firmas compartidas / Identidad Administrada

Una Firma de acceso compartido (SAS) es un URI que concede derechos de acceso restringidos a recursos de Azure Storage. Eliminan la necesidad de administrar las credenciales para los desarrolladores.

Existen tres tipos de firmas de acceso compartido

* **SAS de Delegación de Usuario**: la más recomendable. 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**: 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**: se protege con la clave de cuenta de almacenamiento. Delega el acceso a los recursos en uno o varios de los servicios de almacenamiento. 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. Las aplicaciones pueden usar identidades administradas para obtener tokens de Azure AD sin necesidad de administrar credenciales.

## Microsoft Graph

Nos brinda información sobre todo el ecosistema de Microsoft 365, proporciona un modelo de programación unificado para acceder ad atos en Microsoft 365, Windows 10 y Enterprise Mobility + Security. Facilita el acceso y el flujo de datos y cómo crear consultas mediante REST y código. Crear aplicaciones para organizaciones y consumidores que interactúen con millones de usuarios.

Tres componentes principales facilitan el acceso y el flujo de datos:

* **Microsoft Graph API** ofrece un único punto de conexión. Puede usar las API REST o los SDK para acceder al punto de conexión. 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.
* **Conectores de Microsoft Graph** funcionan en la dirección de entrada y entregan datos externos en la nube de Microsoft en aplicaciones y servicios, 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** conjunto de herramientas para simplificar la entrega de datos de Microsoft Graph en almacenes de datos de Azure conocidos de forma segura y escalable.

# Module 07:Implementar seguridad en soluciones en la nube

## Azure Key Vault

Permite gestionar secretos. Asiste en proteger mejor sus aplicaciones y cómo configurar y recuperar secretos mediante la CLI de Azure.

Un **secreto** es todo aquello cuyo acceso desea controlar de forma estricta, como las claves API, las contraseñas, los certificados o las claves criptográficas.

Para realizar cualquier operación con Key Vault, deberá autenticarse. Hay tres formas de autenticarse en Key Vault:

* **Identidades administradas** de recursos de Azure: cuando implementa una aplicación en una máquina virtual,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. Este enfoque es un procedimiento recomendado.
* **Entidad de servicio y certificado**: puede usar una entidad de servicio y un certificado asociado que tenga acceso a Key Vault. No se recomienda este enfoque porque el propietario o el desarrollador de la aplicación debe girar el certificado.
* Entidad de servicio y secreto:puede usar una entidad de servicio y un secreto para autenticarse en Key Vault, No se recomienda este enfoque porque Es difícil girar automáticamente el secreto de arranque

### Cifrado en transito

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. Los clientes negocian una conexión TLS con Azure Key Vault. TLS proporciona una autenticación sólida, privacidad de mensajes e integridad (lo que permite la detección de la manipulación, interpretación y falsificación de mensajes), interoperabilidad, flexibilidad de algoritmo, y facilidad de implementación y uso.

**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

* **Uso de almacenes de claves distintos**: usar un almacén por aplicación y por entorno (desarrollo, preproducción y producción).
* **Control del acceso al almacén**: 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**: activar el registro y las alertas.
* **Opciones de recuperación**: active la eliminación temporal y la protección de purga si desea protegerse contra la eliminación forzada del secreto.

### Autenticación

(Azure Key Vault)Funciona junto con Azure Active Directory, que es responsable de autenticar la identidad de cualquier entidad de seguridad determinada.

En el caso de las aplicaciones, hay dos maneras de obtener una entidad de servicio:

* Habilite una **identidad administrada** asignada por el sistema para la aplicación, lo que permite la administración interna entre la entidad de servicio de la aplicación y la autentica automáticamente con otros servicios.
* (Si nos es posible usar identidad administrada) registre la aplicación con su inquilino de Azure AD. El registro también crea un segundo objeto de aplicación que identifica la aplicación en todos los inquilinos.

## Identidades administradas

Se utilizan cuando dos servicios se tienen que comunicar entre si, y un servicio necesita permisos del otro, sin necesidad de administrar las credenciales.

La aplicación puede tener dos tipos de identidades:

* **Identidad asignada por el sistema** está asociada al almacén de configuración. Solo puede tener una identidad asignada por el sistema.
* **Identidad asignada por el usuario** puede tener varias identidades asignadas por el usuario.

## Implementando App Configuration

Es un servicio donde se concentran todas las configuraciones de nuestras aplicaciones. Con lo que se evita que cada aplicación busque sus propias configuraciones, en cambio, se puede mediante el uso de un único App Config, centralizar todas las configuraciones.  
Usa una librería especifica, que se usa mediante código

Existe para:

* .NET
* JavaSpring
* Para otros lenguajes se puede usar una APIREST

### Clave-Valor

Las **claves** sirven como nombre de los pares clave-valor y se usan para almacenar y recuperar los valores correspondientes.

Los datos de configuración, se cifran tanto en reposo como en tránsito.

# Module 08: Implementación de API Management

Es un servicio que esta en el medio, entre las aplicación y las API, sirve para protección y limitación de las API. Es un gateway ya que enruta las peticiones hacia las API correspondientes. Por lo que gestiona APIs, políticas, parámetros, suscripción para desarrolladores.  
Cada API consta de una o varias operaciones y se puede agregar a uno o varios productos.

### Componentes

Se compone de una puerta de enlace API, un plano de administración y un portal para desarrolladores.

### Productos

Se usan para administrar la visibilidad de productos a los desarrolladores. Tiene los siguientes grupos invariables del sistema:

* **Administradores**: controlan las instancias del servicio API Management y crean las API, las operaciones y los productos que los desarrolladores usan.
* **Desarrolladores**: usuarios autenticados del portal para desarrolladores que compilan aplicaciones mediante las API.
* **Invitados**: usuarios del portal para desarrolladores no autenticados. Se les concede determinado acceso de solo lectura con la posibilidad de ver API pero no llamarlas.

### Puertas de enlace de API

La puerta de enlace de API Management (también denominada plano de datos o tiempo de ejecución) es el componente de servicio responsable de las solicitudes de API de proxy, la aplicación de directivas y la recopilación de telemetría. Se ubica entre los clientes y los servicios. Actúa como un proxy inverso, enrutando las solicitudes de los clientes a los servicios. También puede realizar diversas tareas transversales como la autenticación, la terminación SSL y la limitación de velocidad.

### Puertas de enlace Administradas y autohospedadas

* **Administrada**: es el componente de puerta de enlace predeterminado que se implementa en Azure para cada instancia de API Management en cada nivel de servicio. Todo el tráfico de API fluye a través de Azure, independientemente de dónde se hospeden los back-end que implementan las API.
* **Autohospedada** : versión opcional y en contenedores de la puerta de enlace administrada predeterminada. Útil para escenarios híbridos y de varias nubes en los que es necesario ejecutar las puertas de enlace fuera de Azure en los mismos entornos donde se hospedan los back-end de API.

### Directivas / Politicas

Permiten al publicador cambiar el comportamiento de la API a través de la configuración. Las directivas son una colección de declaraciones que se ejecutan secuencialmente en la solicitud o respuesta de una API. Se aplican en la puerta de enlace que se encuentra entre el consumidor de la API y la API administrada. La puerta de enlace recibe todas las solicitudes y normalmente las reenvía sin modificar a la API subyacente. Una directiva puede aplicar cambios a la solicitud de entrada y a la respuesta de salida.

### Directivas Avanzadas

Esta unidad proporciona una referencia para las siguientes directivas de API Management:

* **Flujo de control:** aplica condicionalmente instrucciones de directiva basadas en los resultados de la evaluación de expresiones booleanas.
* **Reenviar solicitud** : reenvía la solicitud al servicio back-end.
* **Limitar la simultaneidad**: evita que las directivas delimitadas las ejecute simultáneamente un número de solicitudes mayor que el especificado.
* **Registro en centro de eventos:** envía mensajes en el formato especificado a un centro de eventos definido por una entidad de registrador.
* **Mock response (Simular respuesta):** anula la ejecución de la canalización y devuelve la respuesta ficticia directamente al llamador.
* **Reintentar :** reintenta ejecutar las instrucciones de directiva adjuntas, si y hasta que se cumple la condición. La ejecución se repite en los intervalos de tiempo especificados y hasta el número de reintentos indicado.

# Module 09: Soluciones basadas en eventos

Application Insights es una extensión de Azure Monitor y proporciona características de Supervisión de rendimiento de aplicaciones (también conocida como "APM"). Útiles para supervisar aplicaciones de desarrollo, pruebas y producción

Instrumentar las aplicaciones para habilitar Application Insights a fin de supervisar el rendimiento y ayudar a solucionar los problemas.

Un **evento** es una situación que sucede en una aplicación o servicio que le hace llegar al mundo que algo esta sucediendo, pero no le interesa si se realiza alguna acción o no. Seria solo una notificación.

## Event Grid

Solución para la gestión de eventos. Concentrara todos los eventos desde diferentes orígenes.

Eventos discretos: se refiere al manejo de poca cantidad de eventos.

Cinco conceptos fundamentales:

* **Eventos**: información pequeña que se genera cuando algo se genera en el sistema.
* **Fuente de Eventos**: quien genera esos eventos, por ejemplo, un storage account.
* **Topics**: se utiliza para contextualizar los eventos, por ejemplo, eventos de servicios, eventos de errores de aplicación.
* **Suscripción**: cuando el event grid recibe eventos, y se desea que los suscriptores escuchen los eventos, por ejemplo una pagina web, como si fuera un Event Handler. Por lo que genera una suscripción para cada uno de esos destinos que recibirán los eventos.
* **Event Handlers**:

## Event Hub

Servicio de gestión de (**muchos**) eventos por segundo. Con un flujo de eventos constantes. Permitiendo manejar grandes volúmenes de información de eventos.  
Se utiliza para hacer analítica de Data en tiempo real.  
La velocidad de un centro de eventos de Azure está determinada por la cantidad de unidades de rendimiento que reserva para él. Puede establecer entre 1 y 20 unidades de rendimiento para Event Hub.  
**1 unidad de rendimiento** para los datos que ingresan a un centro de eventos representa 1 MB por segundo o 1000 eventos por segundo (lo que ocurra primero)

# Module 10: Soluciones basadas en mensajes

## Descubriendo colas de mensaje(Message Queues)

Se crean colas de mensajes, para permitir el procesamiento de los mensajes

Un **mensaje** ser procesado de alguna manera, y saber si se proceso correctamente o no.

Existen dos servicios:

* **Azure Service bus**:
  + Solución de tipo FIFO (First-In-First-Out)
  + Detecta duplicados
  + Mensajes de 64Kb hasta 256KB(posibilidad de extenderlo hasta 100MB en Premiun)
  + Soporta hasta 5GB de mensajes en una misma cola
* **Azure Storage Queues**:
  + Colas sencillas para soluciones rápidas
  + Soportan hasta 80 GB de mensajes en una misma cola

## Azure Service bus

Service Bus Queue es una cola de mensajes de nivel empresarial.

### Componentes

* Queues: FIFO
* Modo de recepción:
  + Recepción y borrado
  + Peek Lock
* Tópicos y suscripciones
* Reglas y acciones

## Colas de storge account(Azure Storage Queues)

Almacena gran volumen de mensajes.

### Componentes

* URL Format
* Storage
* Queue
* Message

# Module 11: Solución de problemas de soluciones mediante Application Insights

Permite obtener información para mejorar el rendimiento y la escalabilidad de las aplicaciones.

## Application Insight

Permite monitorear la infraestructura, como se comporta la VM. También podemos monitorear la aplicación, recibiendo logs.

## Telemetría

Permite tener gran flujo de información, el End To End de la aplicación. Tiempos de respuesta, stack de errores, tasas de respuestas, de fallas.

# Module 12: Implementación de almacenamiento en caché para soluciones

## Azure Cache for Redis

Redis es una empresa (tercera) que ofrece una base de datos basada en memoria.

Existe una tecnología Open Source que utiliza Microsoft.

Redis es capaz de procesar grandes volúmenes de solicitudes de aplicación al mantener los datos a los que se accede con frecuencia en la memoria del servidor, donde se pueden realizar operaciones rápidas de lectura y escritura.

El almacenamiento en caché es una técnica que tiene como objetivo mejorar el rendimiento y la escalabilidad de un sistema. se copian temporalmente los datos a los que se accede con mayor frecuencia en almacenamiento rápido ubicado cerca de la aplicación.

El objetivo de esta implementación, es mantener en memoria cierta información que no es modificada frecuentemente. Por lo que, manteniendo la en memoria, se evita consultar en base de datos, evitando consumir tiempo y recursos innecesarios.

### Escenarios principales

Azure Cache for Redis mejora el rendimiento de las aplicaciones, ya que admite patrones de arquitectura de aplicaciones comunes. Estos son algunos de los más comunes:

* **Caché de datos**: Las bases de datos suelen ser demasiado grandes para cargarlas directamente en una caché. Es habitual usar el patrón cache-aside para cargar(o actualizar) datos en la caché solo cuando es necesario.
* **Caché de contenido**: Muchas páginas web se generan a partir de plantillas que usan contenido estático como encabezados, pies de página y banners. Estos elementos estáticos no deberían cambiar a menudo. El uso de una caché en memoria proporciona acceso rápido a contenido estático en comparación con los almacenes de datos de back-end.
* **Almacén de sesión**: Este patrón se utiliza normalmente con carros de la compra y otros datos del historial de los usuarios que una aplicación web podría asociar con las cookies del usuario. El almacenamiento de demasiados datos en una cookie puede tener un efecto negativo en el rendimiento, ya que aumenta su tamaño y no hay que olvidar que se pasa y se valida con cada solicitud.
* **Almacenamiento en cola de trabajos y mensajes**: Las aplicaciones agregan a menudo tareas a una cola cuando las operaciones asociadas a la solicitud tardan tiempo en ejecutarse. Las operaciones con ejecuciones más largas se ponen en cola para procesarse en secuencia, a menudo por parte de otro servidor.
* **Distributed transactions**: A veces, las aplicaciones requieren una serie de comandos en un almacén de datos de back-end para ejecutarse como una única operación atómica. El resultado de todos los comandos debe ser satisfactorio, o todos deben revertirse al estado inicial. Azure Cache for Redis admite la ejecución de un lote de comandos como transacción única.

### Niveles de servicio

* **Básico**: Una memoria caché de OSS Redis que se ejecuta en una sola máquina virtual. Este nivel no tiene contrato de nivel de servicio (SLA) y es ideal para las cargas de trabajo de desarrollo y pruebas y no críticas.
* **Estándar**: Una memoria caché de OSS Redis que se ejecuta en dos máquinas virtuales en una configuración replicada.
* **Premium**: Memorias caché de OSS Redis de alto rendimiento. Este nivel ofrece mayor rendimiento, menor latencia, mejor disponibilidad y más características. Las memorias caché Prémium se implementan en máquinas virtuales más eficaces en comparación con las máquinas virtuales de las memorias caché de nivel Básico o Estándar.
* **Enterprise**: Memorias caché de alto rendimiento con la tecnología del software Redis Enterprise de Redis Labs. Este nivel admite módulos de Redis, incluidos RediSearch, RedisBloom y RedisTimeSeries. Además, ofrece una disponibilidad aún mayor que el nivel Prémium.
* **Enterprise Flash**: Memorias caché de gran tamaño y rentabilidad basadas en el software Redis Enterprise de Redis Labs. Este nivel amplía el almacenamiento de datos de Redis en memoria no volátil, que es más barata que DRAM, en una máquina virtual. Reduce el costo total de memoria por GB.

### Creación y Configuración

Se pueden crear desde Azure Portal, la CLI de Azure o Azure PowerShell.

Parámetros para configurar la memoria caché:

* **Nombre**: nombre único global, porque se usa para generar una dirección URL pública para conectarse y comunicarse con el servicio.
* **Ubicación**: La instancia de caché y la aplicación siempre deben colocarse en la misma región.
* **Tipo**: nivel determina el tamaño, rendimiento y características disponibles para la memoria caché.(Basic, Standard, Premium, Enterprise, Enterprise Flash)
* **Compatibilidad con la agrupación en clústeres**: niveles Premium, Enterprise y Enterprise Flash, puede implementar la agrupación en clústeres y dividir automáticamente el conjunto de datos entre varios nodos.

### Linea de comandos

Estos son algunos comandos comunes que puede usar:

| **Comando** | **Descripción** |
| --- | --- |
| ping | Hacer ping al servidor. Devuelve "PONG". |
| set [key] [value] | Establece una clave o un valor en la memoria caché. Devuelve "OK" cuando la operación se realiza correctamente. |
| get [key] | Obtiene un valor de la caché. |
| exists [key] | Devuelve "1" si la clave existe en la memoria caché o "0" si no existe. |
| type [key] | Devuelve el tipo asociado con el valor de la clave dada. |
| incr [key] | Incremente el valor dado asociado con la clave en "1". El valor debe ser un número entero o un valor doble. Devuelve el nuevo valor. |
| incrby [key] [amount] | Incremente el valor dado asociado con la clave en la cantidad especificada. El valor debe ser un número entero o un valor doble. Devuelve el nuevo valor. |
| del [key] | Elimina el valor asociado con la clave. |
| flushdb | Elimine todas las claves y valores de la base de datos. |

### Tiempo de expiración

Es preciso tener una forma de que los valores expiren cuando estén obsoletos. En Redis, para hacerlo es preciso aplicar un período de vida (TTL) a una clave.  
Cuando transcurra el TTL, la clave se elimina automáticamente, exactamente como si se hubiera emitido el comando DEL. Estas son algunas notas acerca de las expiraciones de TTL.

* Las expiraciones se pueden establecer con una precisión de segundos o mili segundos.
* La resolución de tiempo de expiración es siempre 1 mili segundo.
* La información sobre las expiraciones se replica y se conserva en el disco. El tiempo pasa virtualmente mientras el servidor de Redis permanece detenido (lo que significa que Redis guarda la fecha en que expiran las claves).

### StackExchange.Redis

Librería que permite la conexión con Redis y poder enviar comandos desde una aplicación en .NET.

Usamos la dirección del host, el número de puerto y una clave de acceso para conectarse a un servidor de Redis.

Ejemplo de conexión:

//Creating a connection

using StackExchange.Redis;

...

var redisConnection = "[cache-name].redis. cache.windows.net:6380,password=[password-here],ss1=True,abortConnect=Fa1se";

var connectionString = ConnectionMu1tip1exer.Connect(connectionString);

## Develop for Storage on CDNs

Los CDNs pueden ser aplicados en App Services y Storage Account.

red de entrega de contenido (CDN) es una red distribuida de servidores que puede proporcionar contenido web a los usuarios de manera eficaz. Las redes CDN guardan contenido almacenado en caché en servidores perimetrales en ubicaciones de **puntos de presencia o Puntos de distribución(POP)** que están cerca de los usuarios finales, para minimizar la latencia.

### Ventajas

* Mejor rendimiento y experiencia
* Gran escalado para mejorar la administración de cargas instantáneas pesadas
* Distribución de las solicitudes de usuario y entrega del contenido directamente desde los servidores perimetrales, de forma que se envía menos tráfico al servidor de origen.

### Controlar el comportamiento de almacenamiento en caché

Dos mecanismos para almacenar en caché los archivos.

* **Reglas de caché**: pueden ser globales (se aplica a todo el contenido desde un punto de conexión especificado) o personalizadas. Las reglas personalizadas se aplican a rutas de acceso y extensiones de archivo específicas.
* **Almacenamiento en caché de cadena de consulta**: permite configurar cómo responde Azure CDN en una cadena de consulta.

### Filtrado geográfico

permite autorizar o bloquear contenido en países o regiones específicos, según el código de país o región.

### Proveedores

* Microsoft
* Akamai
* Verizon

# Maquinas virtuales

El uso de **Azure Spot Virtual Machines** le permite aprovechar nuestra capacidad no utilizada con un ahorro de costos significativo. En cualquier momento en que Azure necesite recuperar la capacidad, la infraestructura de Azure expulsará las máquinas virtuales al contado de Azure. Por lo tanto, Azure Spot Virtual Machines es ideal para cargas de trabajo que pueden manejar interrupciones como trabajos de procesamiento por lotes, entornos de desarrollo/prueba, grandes cargas de trabajo de cómputo y más. Consulte el documento de Microsoft

# Plantillas de Resource Manager

Para implementar repetidamente sus soluciones en la nube y saber que su infraestructura se encuentra en un estado confiable. Puede automatizar las implementaciones y usar la práctica de infraestructura como código. En el código, debe definir la infraestructura que va a implementar. Así pues, el código de infraestructura se convierte en parte del proyecto. Al igual que el código de la aplicación, puede almacenar el código de infraestructura en un repositorio de origen y agregarle un número de versión.

Para implementar la infraestructura como código para las soluciones de Azure, use las plantillas de Azure Resource Manager (plantillas de ARM). La plantilla es un archivo de notación de objetos JavaScript (JSON) que contiene la infraestructura y la configuración del proyecto. La plantilla usa sintaxis declarativa, lo que permite establecer lo que pretende implementar sin tener que escribir la secuencia de comandos de programación para crearla. En la plantilla se especifican los recursos que se van a implementar y las propiedades de esos recursos.

## Caracteristicas:

* **Sintaxis declarativa**: permiten crear e implementar una infraestructura de Azure completa de forma declarativa.
* **Resultados repetibles**: puede implementar la misma plantilla varias veces y obtener los mismos tipos de recursos en el mismo estado.
* **Orquestación**: Resource Manager se encarga de gestionar la implementación de recursos interdependientes para que se creen en el orden correcto.

## Archivo de plantilla

Dentro de la plantilla, puede escribir expresiones de plantilla que aumentan las capacidades de JSON. Estas expresiones hacen uso de las funciones que proporciona Resource Manager.

La plantilla contiene las secciones siguientes:

* **Parámetros**: proporcione valores durante la implementación que permitan usar la misma plantilla con entornos diferentes.
* **Variables**: defina los valores que se reutilizan en las plantillas. Se pueden crear a partir de valores de parámetro.
* **Funciones definidas por el usuario**: cree funciones personalizadas que simplifiquen la plantilla.
* **Recursos**: especifique los recursos que se van a implementar.
* **Salidas**: devuelva valores de los recursos implementados.