



Detección de opiniones en  
texto con Text Analytics API

---

---

# BIENVENIDA

Obtenga información sobre lo que los clientes dicen realmente sobre su producto o marca cuando envían comentarios. Se va a crear una solución que use Azure Functions y la inteligencia de Text Analytics API para detectar opiniones en mensajes de texto.

En este módulo, aprenderá lo siguiente:

- Más información acerca de Text Analytics API
- Registrarse para conseguir una clave de Text Analytics API
- Diseñar y compilar un servicio con Azure Functions que use Text Analytics API para clasificar comentarios de texto
- Leer y escribir mensajes de Azure Queue Storage en una aplicación de función con la ayuda de enlaces

2

Requisitos previos

Ninguno

Introducción 2 min

Análisis de opiniones y Text Analytics API 4 min

Ejercicio: Llamada a Text Analytics API desde la consola de pruebas en línea 10 min

Diseño de un clasificador de comentarios 6 min

Ejercicio: Creación de una aplicación de función en el portal para hospedar la lógica de negocios 6 min

Ejercicio: Llamada a Text Analytics API desde una aplicación **9 min**

Ejercicio: Ordenación de los mensajes en colas diferentes en función de la puntuación de opiniones **8 min**

Resumen **2 min**

---

# INTRODUCCIÓN

**Si le muestro** un texto escrito con TODAS LAS LETRAS EN MAYÚSCULAS, ¿qué emociones transmitiría? Si en la reseña de un libro escribo un comentario que indica que “el final era impredecible”, ¿indica esa afirmación algo positivo o negativo? ¿Cómo puedo averiguar mediante programación en qué idioma se escribió un correo electrónico? Text Analytics consiste en comprender y analizar texto no estructurado para responder a ese tipo de preguntas. Incluye el análisis de opiniones, la extracción de frases clave, la detección de idioma, etc.

En este módulo, vamos a centrar la atención en el análisis de opiniones. Vamos a conocer más información sobre Text Analytics API. Este servicio basado en la nube de Microsoft proporciona un procesamiento avanzado del lenguaje natural (PLN) sobre texto sin formato. Una vez que haya completado el módulo, habrá creado una solución con Azure Functions que permite clasificar los comentarios de texto según las opiniones.

## Objetivos de aprendizaje

En este módulo, aprenderá a:

- Más información acerca de Text Analytics API



---

# ANÁLISIS DE OPINIONES Y TEXT ANALYTICS API

**Todos queremos** saber la opinión de los clientes sobre la marca, el producto y el mensaje. ¿Cómo cambia su opinión con el tiempo? La búsqueda de opiniones en lo que escriben puede darnos algunas pistas. El análisis de opiniones ayuda a responder esta pregunta: *¿Qué quieren realmente nuestros clientes?* Se usa para analizar los tweets y cualquier otro contenido de las redes sociales, los comentarios de los clientes y los correos electrónicos.



Un enfoque habitual del análisis de opiniones consiste en entrenar modelos de Machine Learning que detecten las opiniones. Sin embargo, este proceso es complejo. Implica tener datos de entrenamiento de buena calidad que estén etiquetados, crear características a partir de esos datos, entrenar un clasificador y usarlo para predecir la opinión de nuevos fragmentos de texto. No todas las compañías tienen el dinero y los conocimientos para invertir en la compilación de soluciones de inteligencia artificial desde cero. Afortunadamente, Microsoft y otras empresas sí que pueden, y lo hacen, e invierten en las investigaciones más avanzadas en estas áreas.

Como desarrolladores, podemos beneficiarnos de sus descubrimientos a través de las API, SDK y plataformas que ellos crean. Microsoft Cognitive Services es una de esas ofertas.

## Azure Cognitive Services

Microsoft Cognitive Services consta de un conjunto de API, SDK y servicios. El objetivo es ayudar a los desarrolladores a hacer que sus aplicaciones sean más inteligentes, atractivas y detectables.

Azure Cognitive Services ofrece algoritmos inteligentes de visión, voz, idioma, conocimiento y búsqueda. Para ver las ofertas, eche un vistazo a [Directorio de Cognitive Services](#). Puede probar cada servicio de forma gratuita. Si decide integrar uno o varios de estos servicios en las aplicaciones, suscríbase a una suscripción de pago. El servicio que vamos a usar a lo largo de este módulo es Text Analytics API, así que vamos a ver más información al respecto.

## Text Analytics API

Text Analytics API es un servicio de Cognitive Services diseñado para ayudarle a extraer información a partir de texto. A través del servicio puede identificar el idioma, detectar opiniones, extraer frases clave y detectar entidades conocidas del texto.

En esta lección, vamos a obtener más información sobre el elemento de análisis de opinión de esta API. En un segundo plano, este servicio utiliza un algoritmo de clasificación de aprendizaje automático para generar una puntuación de opinión entre 0 y 1. Las puntuaciones cercanas a 1 indican una opinión positiva y las puntuaciones cercanas a 0 indican una opinión negativa. Una puntuación cercana a 0,5 indica

que no hay ninguna opinión o que se trata de una declaración neutra. No tiene que preocuparse por los detalles de implementación del algoritmo. Puede centrarse en realizar llamadas al servicio desde la aplicación. Como veremos en breve, puede estructurar una solicitud **POST**, enviarla al punto de conexión /sentiment y recibir una respuesta JSON que le indique una *puntuación de opinión*.

Primero vamos a experimentar con Text Analytics API mediante una consola de pruebas de API en línea. Una vez que se sienta cómodo con la API, vamos a usarla en un escenario para detectar opiniones en mensajes a fin de clasificarlos para su posterior procesamiento.



---

# EJERCICIO: LLAMADA A TEXT ANALYTICS API DESDE LA CONSOLA DE PRUEBAS EN LÍNEA

**Para ver a** Text Analytics API en funcionamiento, se realizarán algunas llamadas mediante la herramienta de consola de pruebas de API integrada que se encuentra en la documentación de referencia en línea. Antes de hacerlo, necesitaremos una clave de acceso para realizar esas llamadas.

## Creación de una clave de acceso

Cada llamada a Text Analytics API requiere una clave de suscripción. A menudo se la denomina clave de acceso y se usa para validar que tiene acceso para realizar la llamada. Se usará Azure Portal para obtener una clave.

1. Inicie sesión en [Azure Portal](#) con la misma cuenta que ha usado para activar el espacio aislado.
2. En el menú de Azure Portal o en la página **Inicio**, seleccione **Crear un recurso**.
3. En el cuadro de búsqueda **Buscar en Marketplace**, escriba *text analytics* y presione la tecla Entrar o Retorno.

4. Seleccione **Text Analytics** en los resultados de la búsqueda y, después, seleccione el botón **Crear** en la parte inferior derecha de la pantalla.
5. En la página **Crear** que se abre, escriba los siguientes valores en cada campo.

**TABLA 1**

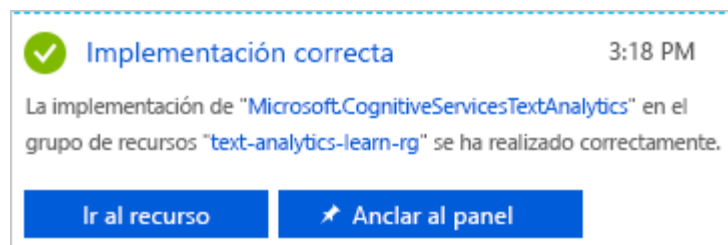
Propiedad	Valor	Descripción
Nombre	<i>Elija un nombre único</i>	El nombre de la cuenta de Cognitive Services. Es recomendable que use un nombre descriptivo. Los caracteres válidos son a-z, 0-9 y -.
Suscripción	Concierge Subscription	La suscripción en la que se crea esta cuenta de Cognitive Services API con <b>Text Analytics API</b> .
Ubicación	<i>Elija una región en la lista desplegable</i>	Seleccione una ubicación en la lista desplegable.
Plan de tarifa	<b>F0 gratis</b>	El costo de la cuenta de Cognitive Services depende del uso real y de las opciones que elija. Se recomienda seleccionar el nivel gratis para nuestros fines.

TABLA 1

Propiedad	Valor	Descripción
Grupo de recursos	Haga clic en <b>Usar existente</b> y elija [nombre del grupo de recursos del espacio aislado].	Nombre del nuevo grupo de recursos en el que se va a crear la cuenta de Text Analytics API de Cognitive Services.

## 6. Sugerencia

- Recuerde la ubicación que ha seleccionado al crear la cuenta de Cognitive Services Text Analytics. La usará para realizar llamadas de API en breve.
- Haga clic en **Crear** en la parte inferior de la página para iniciar el proceso de creación de la cuenta. Esté atento a una notificación de que la implementación está en curso. A continuación, recibirá una notificación que le avisa de que la cuenta se ha implementado correctamente en el grupo de recursos.

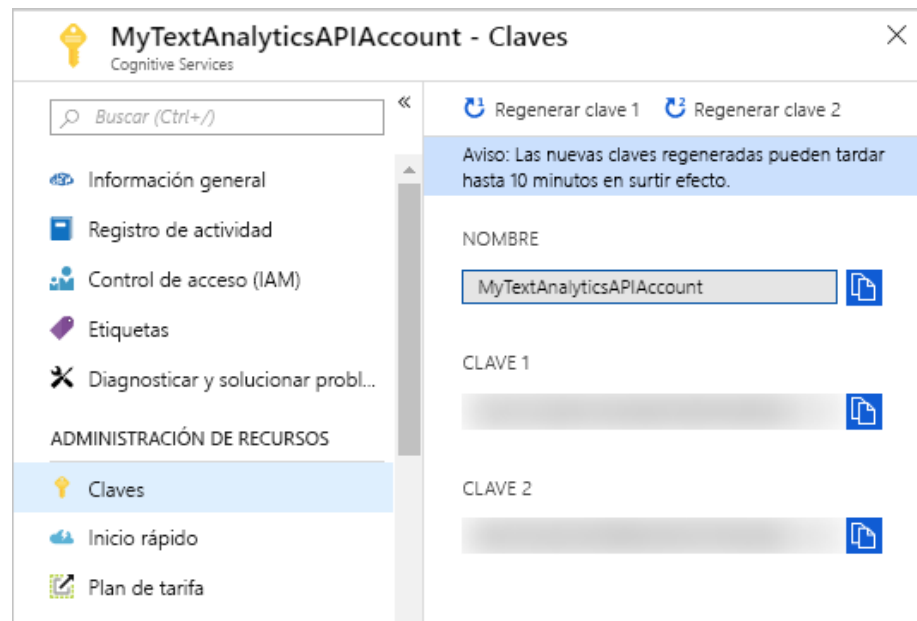


## Obtención de la clave de acceso

Ahora que tenemos nuestra cuenta de Cognitive Services, vamos a buscar la clave de acceso para que podamos iniciar una llamada a la API.

1. Haga clic en el botón **Ir al recurso** de la notificación de *implementación correcta*. Esta acción abre el inicio rápido de la cuenta.
2. Seleccione el elemento de menú **Claves** del menú de la izquierda o en la sección *Obtener las claves* del inicio rápido. Esta acción abre la página **Administrar claves**.
3. Copie una de las claves con el botón Copiar.

12



## Importante

Mantenga siempre las claves de acceso a salvo y no las comparta nunca.

4. Almacene esta clave durante el resto de este módulo. La vamos a usar en breve para realizar llamadas API desde la consola de pruebas y durante el resto del módulo.

## Llamada a la API desde la consola de pruebas

Ahora que tenemos nuestra clave, podemos ir a la consola de pruebas y elegir la API para una prueba.

1. Vaya a la siguiente URL en su explorador favorito. Reemplace [location] por la ubicación que ha seleccionado antes en esta unidad al crear la cuenta de Cognitive Services Text Analytics. Por ejemplo, si ha creado la cuenta en *eastus*, debería reemplazar [location] por *eastus* en la dirección URL.

bashCopiar

9. [https://\[location\].dev.cognitive.microsoft.com/docs/services/TextAnalytics.V2.0](https://[location].dev.cognitive.microsoft.com/docs/services/TextAnalytics.V2.0)

La página de aterrizaje muestra un menú a la izquierda y contenido a la derecha. El menú enumera los métodos POST a los que puede llamar en Text Analytics API. Estos puntos de conexión son **Detectar idioma**, **Entidades**, **Frases clave** y **Opinión**. Para llamar a alguna de estas operaciones, necesitamos hacer algunas cosas.

- Seleccione el método al que queremos llamar.

- Agregue la clave de acceso que guardamos anteriormente en esta lección para cada llamada.
- 2. En el menú de la izquierda, seleccione **Opinión**. Esta selección abre la documentación correspondiente a la derecha. Como se muestra en la documentación, se realizará una llamada REST con el formato siguiente.

`https://[location].api.cognitive.microsoft.com/text/analytics/v2.0/sentiment`

[location] se reemplaza por la ubicación que ha seleccionado al crear la cuenta Text Analytics. Pasaremos nuestra clave de suscripción, o clave de acceso, en el encabezado **ocp-Apim-Subscription-Key**.

### Realización de algunas llamadas API

1. En la sección de la **consola de pruebas de Open API** de esta página, seleccione el botón correspondiente a la ubicación o región en que se creó la cuenta de Cognitive Services. Al seleccionar este botón, se abre la consola de API activa a interactiva.
2. Pegue la clave de acceso que ha guardado en el campo con la etiqueta **Ocp-Apim-Subscription-Key**. Observe que la clave se ha escrito automáticamente en la ventana de solicitud HTTP como un valor de encabezado.
3. Desplácese a la parte inferior de la página y haga clic en **Enviar**.

Vamos a examinar las secciones de esta pantalla con más detalle.

En la sección Encabezados de la interfaz de usuario, establecemos la clave de acceso, o de suscripción, en el encabezado de nuestra solicitud.

**Headers**

Content-Type	<input type="text" value="application/json"/>	<a href="#">✕ Remove header</a>
Ocp-Apim-Subscription-Key	<input type="text" value="....."/>	

A continuación, tenemos la sección de cuerpo de la solicitud, que contiene una matriz de **documentos**. Cada documento de la matriz tiene tres propiedades. Las propiedades son *"language"*, *"id"* y *"text"*. *"id"* es un número en este ejemplo, pero puede ser cualquier cosa que desee, siempre que sea única en la matriz de documentos. En este ejemplo también vamos a pasar documentos escritos en tres idiomas diferentes. Se admiten más de 15 idiomas en la característica Opinión de Text Analytics API. Para más información, consulte [Idiomas admitidos en Text Analytics API](#). El tamaño máximo de un solo documento es de 5000 caracteres y una solicitud puede contener hasta 1000 documentos.

**Request body**

Collection of documents to analyze

```
1 {
2   "documents": [
3     {
4       "language": "en",
5       "id": "1",
6       "text": "Hello world. This is some input text that I love."
7     },
8   ]
9 }
```

La solicitud completa, incluidos los encabezados y la dirección URL de solicitud se muestran en la sección siguiente. En este ejemplo puede ver que las solicitudes se enrutan a una dirección URL que empieza

por westus. La dirección URL varía según la ubicación que haya seleccionado al crear la cuenta de Text Analytics.

#### Request URL

<https://westus.api.cognitive.microsoft.com/text/analytics/v2.0/sentiment>

#### HTTP request

```
POST https://westus.api.cognitive.microsoft.com/text/analytics/v2.0/sentiment
HTTP/1.1
Host: westus.api.cognitive.microsoft.com
Content-Type: application/json
Ocp-Apim-Subscription-Key: xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

```
{
  "documents": [
    {
      "language": "en",
      "id": "1",
      "text": "Hello world. This is some input text that I love."
    },
    {
      "language": "fr",
      "id": "2",
      "text": "Bonjour tout le monde"
    },
    {
      "language": "es",
      "id": "3",
      "text": "La carretera estaba atascada. Había mucho tráfico el día de
ayer."
    }
  ]
}
```

Después, tenemos información acerca de la respuesta. En el ejemplo, vemos que la solicitud fue correcta y que se devolvió el código 200.

#### Response status

200 OK

#### Response status

197 ms

Por último, vemos la respuesta a nuestra solicitud. La respuesta contiene las conclusiones que Text Analytics API tenía sobre nuestros



documentos. Se nos devuelve una matriz de documentos sin el texto original. Obtenemos un *"id"* y un *"score"* para cada documento. La API devuelve una puntuación numérica comprendida entre 0 y 1. Las puntuaciones cercanas a 1 indican una opinión positiva mientras que las puntuaciones cercanas a 0 indican una opinión negativa. Una puntuación de 0,5 indica la ausencia de opinión, una instrucción neutra. En este ejemplo, tenemos dos documentos bastante positivos y uno negativo.

#### Response content

```
Date: Thu, 30 Aug 2018 21:13:33 GMT
Content-Type: application/json; charset=utf-8

{
  "documents": [{
    "score": 0.98690706491470337,
    "id": "1"
  }, {
```

Enhorabuena. Ha realizado su primera llamada a Text Analytics API sin escribir una sola línea de código. No dude en permanecer en la consola y probar a realizar más llamadas. Estas son algunas sugerencias:

- Cambie los documentos de la sección número 2 y vea lo que devuelve la API.
- Pruebe los otros métodos, **Detectar idioma**, **Entidades** y **Frases clave**, con la misma clave de suscripción.
- Intente realizar una llamada desde otra región con su suscripción y observe lo que sucede.

La consola de pruebas de API es una excelente manera de explorar las funcionalidades de esta API. Ahora que ha explorado por sí mismo, pasemos a poner esta inteligencia en un escenario real.

---

# DISEÑO DE UN CLASIFICADOR DE COMENTARIOS

**Vamos a aplicar** los conocimientos de Text Analytics a una solución práctica. Nuestra solución se centrará en el análisis de sentimiento de documentos de texto. Vamos a describir el problema que queremos abordar para establecer el contexto.

## **Administración más eficaz de los comentarios de clientes**

En los medios sociales se habla activamente del producto de su empresa. El alias de correo electrónico de comentarios también recibe comentarios de los clientes que desean compartir su opinión sobre su producto.

Como sucede con cualquier startup, vive con el mantra de escuchar a los clientes. Sin embargo, el éxito de su producto ha hecho que mantener esta promesa sea más fácil de decir que de hacer. Es un buen problema, pero un problema al fin y al cabo.

El equipo ya no puede con el volumen de comentarios. Necesitan ayuda para clasificar los comentarios para poder administrar los problemas de la forma más eficaz posible. Como desarrollador jefe de la organización, se le ha pedido que cree una solución.

Veamos algunos requisitos generales:

**TABLA 1**

Requisito	Detalles
Clasificar los comentarios para poder reaccionar a ellos.	<p>No todos los comentarios son iguales. Algunos son mordaces de clientes frustrados. En otros casos, los comentarios son positivos.</p> <p>Como mínimo, tener una indicación de la opinión para poder clasificarlos.</p>
La solución debe poder ampliarse o reducirse verticalmente para satisfacer la demanda.	<p>Somos una startup. Los costos fijos son difíciles de manejar. Necesitamos un modelo de negocio que sea flexible y que pueda crecer o reducirse según el tráfico de comentarios. Necesitaremos una solución que sea escalable, pero con un costo mínimo.</p> <p>En este caso, un buen candidato es una arquitectura de microservicios.</p>
Genere un producto viable mínimo (MVP), pero haga que la solución sea adaptable.	<p>Hoy queremos clasificar los comentarios en categorías limitadas a los comentarios que importan. Si alguien tiene una idea, podemos enseguida y empezar a charlar con él. En el futuro, podemos agregar más cosas. Una idea para una nueva característica es automatizar la detección de puntos problemáticos antes de que lleguen a los clientes. Es automatizar las respuestas a los clientes que se quejan con nosotros, quieren saber más sobre nosotros, etc.</p> <p>En este escenario, una buena elección es una solución basada en reglas. Por ejemplo, podríamos usar colas como una forma de manejar la carga. Escribir el resultado en una cola para que la siguiente tarea lo procese.</p>

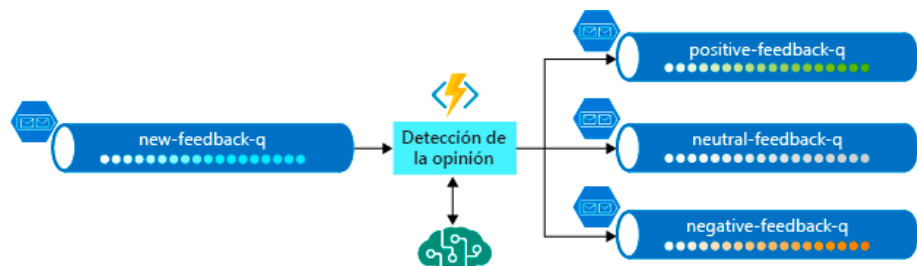
**TABLA 1**

Requisito	Detalles
Entrega rápida.	<p>Todos hemos oído esto antes. Recuerde que  rápidamente en nuestro escenario. Para lograr ve  código.</p> <p>Vamos a aprovechar las ventajas de Text Analytics  detectar opiniones. Con Azure Functions y un en  código que hay que escribir. Una solución sin  preocuparse por la administración del servidor.</p>

La solución que proponemos para cada requisito de la tabla anterior nos da una idea de cómo asignar los requisitos a las soluciones. Ahora veamos el aspecto que tendría una solución basada en Azure.

### Una solución basada en Azure Functions, Azure Queue Storage y Text Analytics API

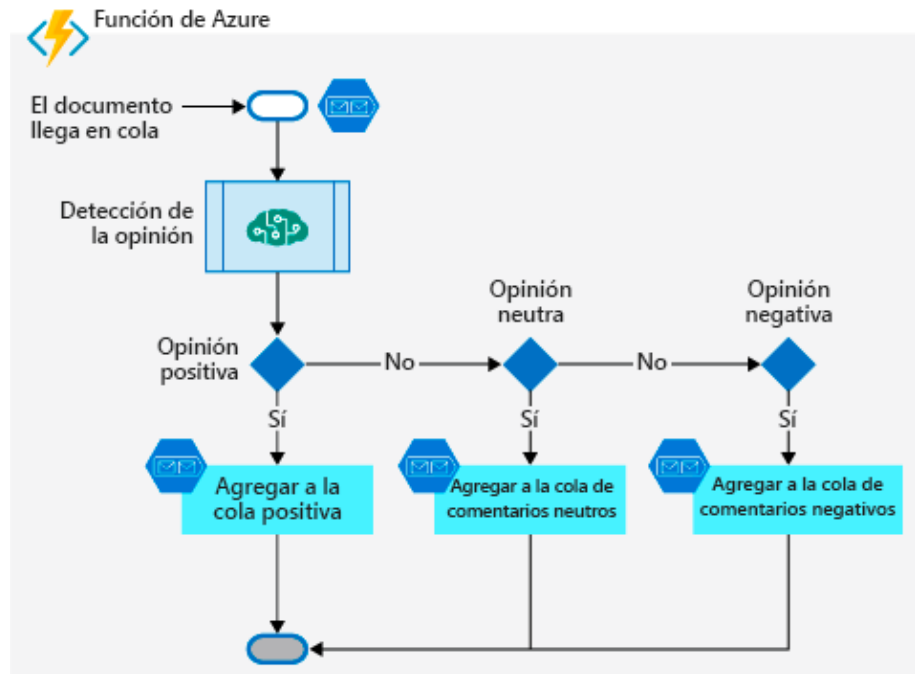
El diagrama siguiente es una propuesta de diseño de una solución. Usa tres componentes principales de Azure: Azure Queue Storage, Azure Functions y Cognitive Services de Azure.



La idea es que los documentos de texto que contienen comentarios de los usuarios se coloquen en una cola que hemos llamado *new-feedback-q* en el diagrama anterior. La llegada de un mensaje que contenga el documento de texto a la cola desencadenará o iniciará la ejecución de funciones. La función lee los mensajes que contienen documentos nuevos en la cola de entrada y los envía para su análisis a Text Analytics API. Según los resultados que la API devuelva, se coloca en una cola de salida un nuevo mensaje que contiene el documento para su posterior procesamiento.

El resultado que obtenemos para cada documento es una puntuación de la opinión. Las colas de salida se usan para almacenar los comentarios clasificados como positivos, neutros y negativos. ¡Esperamos que la cola de comentarios negativos esté siempre vacía! Una vez clasificado cada comentario entrante en una cola de salida en función de la opinión, vamos a agregar lógica para actuar sobre los mensajes de cada cola.

Echemos un vistazo al diagrama de flujo siguiente para ver lo que debe hacer la lógica de la función.



Nuestra lógica es como un enrutador. Toma la entrada de texto y lo enruta a una cola de salida según la puntuación de la opinión del texto. Tenemos una dependencia de Text Analytics API. Aunque la lógica parece trivial, esta función eliminará la necesidad de tener personas en el equipo para analizar manualmente los comentarios.

### Pasos para implementar nuestra solución

Para implementar la solución descrita en esta unidad, se deberán completar los pasos siguientes.

1. Crear una aplicación de función para hospedar nuestra solución.
2. Buscar las opiniones en los mensajes de comentarios entrantes con Text Analytics API. Usaremos la clave de acceso del ejercicio anterior y escribiremos código para enviar las solicitudes.

3. Publique los comentarios en las colas de procesamiento en función de la opinión.

Vamos a pasar a crear la aplicación de función.

---

# EJERCICIO: CREACIÓN DE UNA APLICACIÓN DE FUNCIÓN EN EL PORTAL PARA HOSPEDAR LA LÓGICA DE NEGOCIOS

**Una aplicación** de función proporciona un contexto para administrar y ejecutar las funciones. Vamos a crear una aplicación de función y, a continuación, agregarle una función.

## **Creación de una aplicación de función para hospedar nuestra función**

1. Inicie sesión en [Azure Portal](#) con la misma cuenta con la que ha activado el espacio aislado.
2. En el menú de Azure Portal o la página **Inicio**, seleccione **Crear un recurso** y luego **Proceso** > **Aplicación de funciones**.
3. Escriba la configuración de la aplicación de funciones como se especifica en la tabla siguiente.



TABLA 1

Configuración	Valor	Descripción
<b>Nombre de la aplicación</b>	Nombre único globalmente	Nombre que identifica la nueva aplicación. Solo se permiten caracteres alfabéticos, 0-9 y -.
<b>Suscripción</b>	<b>Suscripción de Concierge</b>	Suscripción en la que se creará esta aplicación.
<b>Grupo de recursos</b>	[nombre del grupo de recursos del espacio aislado]	Nombre del grupo de recursos en el que se creará esta aplicación. Asegúrese de seleccionar <b>Usar espacio aislado</b> en la configuración anterior. De esta forma, todos los recursos de este grupo conservan juntos.
<b>SO</b>	Windows	Sistema operativo que hospeda la aplicación.
<b>Plan de hospedaje</b>	Plan de consumo	Plan de hospedaje que define cómo se ejecutará la aplicación. El <b>Plan de consumo</b> predetermina los requisitos de las funciones. El tiempo durante el cual se ejecutará la aplicación.
<b>Ubicación</b>	Seleccione la misma ubicación que usó anteriormente.	Elija una región cerca de usted o de la región que usó anteriormente.  Seleccione la misma región que ha usado en el último ejercicio.

TABLA 1

Configuración	Valor	Descripción
<b>Pila en tiempo de ejecución</b>	Node.js	El código de ejemplo de este módulo
<b>Almacenamiento</b>	Nombre único global	Nombre de la nueva cuenta de almacenamiento. Los nombres de las cuentas de almacenamiento solo pueden incluir números y letras, y el campo con un nombre único de almacenamiento. Sin embargo, no dude en usar otro nombre.

- Haga clic en **Crear** para aprovisionar e implementar la aplicación de función.
- Haga clic en el icono de notificación de la esquina superior derecha del portal y observe el mensaje **Implementación en curso**.
- La implementación puede tardar un tiempo. Por tanto, permanezca en el Centro de notificaciones y esté atento para ver un mensaje **Implementación correcta**.
- Una vez implementada la aplicación de función, vaya a **Todos los recursos** en el portal. Se muestra la aplicación de función con el tipo **App Service** y el nombre que le asignó. Para abrir la aplicación de función, selecciónela en la lista.

Enhorabuena. Ha creado e implementado la aplicación de función.

## Sugerencia

¿Tiene dificultades para encontrar las aplicaciones de función en el portal? Intente [agregar aplicaciones de función a sus favoritos en Azure Portal](#).

## Creación de una función que ejecute nuestra lógica

Ahora que tenemos una aplicación de función, es el momento de crear una función. Se activa una función a través de un desencadenador. En este módulo, usaremos un desencadenador de colas. El runtime sondea una cola e inicia esta función para procesar un mensaje nuevo.

1. Seleccione el botón Agregar (+) situado junto a **Funciones**. Esta acción inicia el proceso de creación de funciones.
2. En la página **Azure Functions for JavaScript - getting started** (Azure Functions para JavaScript: introducción), seleccione **En el portal** y luego **Continuar**.
3. En el paso **Crear una función**, seleccione **Más plantillas...** y luego **Finish and view templates** (Plantillas Finalizar y ver).
4. En la lista de plantillas disponibles para esta aplicación de función, seleccione **Desencadenador de Azure Queue Storage**.
5. Si ve un mensaje que indica **Extensiones no instaladas**, seleccione **Instalar**. La instalación de dependencias puede llevar un par de minutos. Espere hasta que termine la instalación antes de continuar.

6. En el cuadro de diálogo **Nueva función** que aparece, especifique los siguientes valores.

**TABLA 2**

Propiedad	Valor
Nombre	discover-ser
Nombre de cola	new-feedba
Conexión de cuenta de Storage	AzureWebJ

7. Haga clic en **Crear** para iniciar el proceso de creación de la función.
8. Se crea una función en el lenguaje elegido mediante la plantilla de funciones del desencadenador de colas. Aunque en este módulo vamos a implementar la función en JavaScript, puede crear una función en cualquier [lenguaje compatible](#).

Una vez completado el proceso de creación, el editor de código se abre en el portal y carga la página *index.js*. Este archivo es el archivo de código en el que escribimos nuestra lógica de la función.

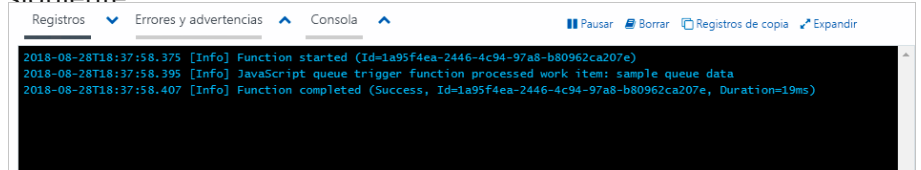
### **Pruébalo**

Vamos a probar lo que tenemos hasta ahora. Aún no hemos escrito ningún código, por lo que esta prueba es para asegurarnos de que lo que hemos configurado hasta ahora, funciona.

1. Haga clic en **Ejecutar** en la parte superior del editor de código.

2. Observe la pestaña **Registros** que se abre en la parte inferior de la pantalla. Si todo funciona según lo previsto, verá un mensaje similar al

siguiente



```
2018-08-28T18:37:58.375 [Info] Function started (Id=1a95f4ea-2446-4c94-97a8-b80962ca207e)
2018-08-28T18:37:58.395 [Info] JavaScript queue trigger function processed work item: sample queue data
2018-08-28T18:37:58.407 [Info] Function completed (Success, Id=1a95f4ea-2446-4c94-97a8-b80962ca207e, Duration=19ms)
```

El botón **Ejecutar** inició nuestra función y pasó los *datos de cola de ejemplo*, el texto predeterminado de la ventana de solicitud **Prueba** de nuestra función.

### Sugerencia

Si se agota el tiempo de espera de la función o no devuelve un valor correctamente, pruebe a reiniciar la aplicación de función. Seleccione la aplicación de función en el menú de la izquierda y, a continuación, seleccione **Reiniciar** en el panel *Introducción*. Espere a que la aplicación de función se reinicie y, a continuación, pruebe a ejecutar de nuevo la función.

Buen trabajo. Ha agregado correctamente una función desencadenada por colas a su aplicación de función y la ha probado para asegurarse de que funciona según lo previsto. Vamos a agregar más funcionalidad a la función en el ejercicio siguiente.

Echemos un vistazo rápido al otro archivo de la función, el archivo de configuración *function.json*. Los datos de configuración de este archivo se muestran en el siguiente listado JSON.

JSONCopiar

```
{  
  
  "bindings": [  
  
    {  
  
      "name": "myQueueItem",  
  
      "type": "queueTrigger",  
  
      "direction": "in",  
  
      "queueName": "new-feedback-q",  
  
      "connection": "AzureWebJobsDashboard"  
  
    }  
  
  ],  
  
  "disabled": false  
  
}
```

Como puede ver, esta función tiene un enlace de desencadenador que se llama **myQueueItem** del tipo `queueTrigger`. Si llega un mensaje nuevo a la cola que hemos denominado **new-feedback-q**, se llama a la función. Hacemos referencia al nuevo mensaje mediante el parámetro de enlace `myQueueItem`. Los enlaces se ocupan realmente de parte del trabajo pesado por nosotros.

En el paso siguiente, vamos a agregar código para llamar al servicio Text Analytics API.

### Sugerencia

Puede ver `index.js` y `function.json` expandiendo el menú **Ver archivos** en la parte derecha del panel de función de Azure Portal.

En este ejercicio se explicó como poner en funcionamiento nuestra infraestructura de Azure Functions. Tenemos una función en marcha hospedada en una aplicación de función que se ejecuta cuando llega un mensaje nuevo a la cola. La hemos llamado **new-feedback-q**. Lo divertido empieza en el siguiente ejercicio, en el que vamos a agregar código para llamar a Azure Cognitive Services a fin de realizar análisis de opiniones.

---

# EJERCICIO: LLAMADA A TEXT ANALYTICS API DESDE UNA APLICACIÓN

**Vamos a actualizar** la implementación de la función para llamar al servicio Text Analytics API y obtener una puntuación de opinión.

1. Seleccione la función **discover-sentiment-function**, en nuestra aplicación de función en el portal.
2. Expanda el menú **Ver archivos** a la derecha de la pantalla.
3. En la pestaña **Ver archivos**, seleccione **index.js** para abrir el archivo de código en el editor.
4. Reemplace todo el contenido de **index.js** por el siguiente código JavaScript y haga clic en **Guardar**.

JavaScriptCopiar

```
module.exports = function (context, myQueueItem) {  
  
    context.log('Processing queue message', myQueueItem);  
  
  
  
    let https = require('https');  
  
  
  
    // Replace the accessKey string value with your valid access key.
```



```
let accessKey = '<YOUR ACCESS KEY HERE>';
```

// Replace [region], including square brackets, in the uri variable below.

// You must use the same region in your REST API call as you used to obtain your access keys.

// For example, if you obtained your access keys from the northeurope region, replace

// "westus" in the URI below with "northeurope".

```
let uri = '[region].api.cognitive.microsoft.com';
```

```
let path = '/text/analytics/v2.0/sentiment';
```

```
let response_handler = function (response) {
```

```
    let body = '';
```

```
    response.on ('data', function (chunk) {
```

```
        body += chunk;
```

```
    });
```

```
response.on ('end', function () {  
  
    let body_ = JSON.parse (body);  
  
    let body__ = JSON.stringify (body_, null, ' ');  
  
    context.log (body__);  
  
    context.done();  
  
    return;  
  
});
```

```
response.on ('error', function (e) {  
  
    context.log ('Error: ' + e.message);  
  
    context.done();  
  
    return;  
  
});  
  
};
```

```
let get_sentiments = function (documents) {  
  
    let body = JSON.stringify (documents);
```

```
let request_params = {  
  
    method : 'POST',  
  
    hostname : uri,  
  
    path : path,  
  
    headers : {  
  
        'Ocp-Apim-Subscription-Key' : accessKey,  
  
    }  
  
};
```

```
let req = https.request (request_params, response_handler);  
  
req.write (body);  
  
req.end ();  
  
}
```

// Create a documents array with one entry.

```
let documents = { 'documents': [  
  
    {
```

```
        'id': '1',  
  
        'language': 'en',  
  
        'text': myQueueItem  
    },  
  
    ]};
```

```
get_sentiments (documents);
```

```
};
```

36

5. En el código que pegó, actualice el valor de `accessKey` con la clave de acceso de Text Analytics API que guardó anteriormente en este módulo.
6. Actualice el valor de `uri` con la región desde la que ha obtenido la clave de acceso.

Vamos a ver qué sucede en este código:

- Cada llamada al servicio Text Analytics necesita la clave de acceso, que se agrega como el encabezado `Ocp-Apim-Subscription-Key`.
- Todas las llamadas se realizan a un punto de conexión específico de la región, definido por `uri` en nuestro código.

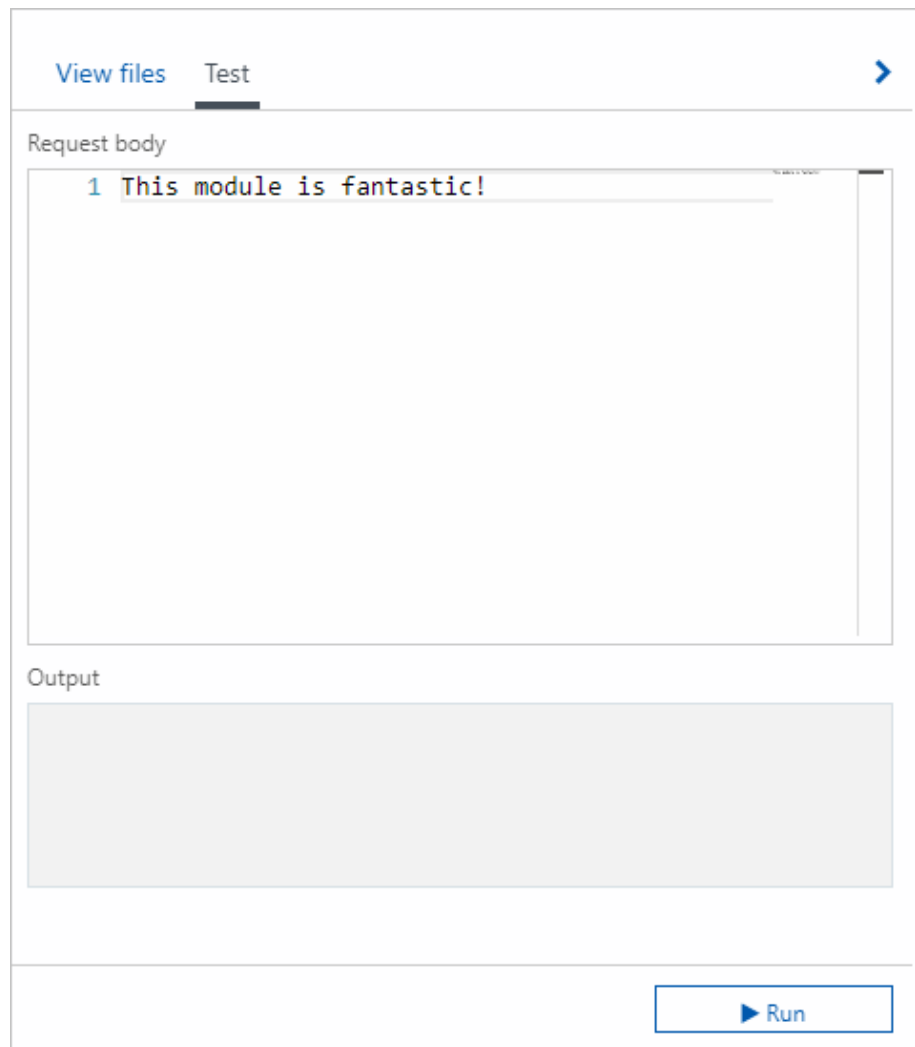
- En la parte inferior del archivo de código, se ha definido una matriz `documents`. Esta matriz es la carga que enviamos al servicio Text Analytics API.
- La matriz `documents` tiene una sola entrada en este caso, que es el mensaje de cola que ha desencadenado la función. Aunque solo tenemos un documento en la matriz, esto no significa que nuestra solución solo pueda controlar un mensaje a la vez. El entorno en tiempo de ejecución de Azure Functions recupera y procesa mensajes en lotes, llamando a varias instancias de nuestra función *en paralelo*. Actualmente, el tamaño de lote predeterminado es 16 y el tamaño máximo es 32.
- `id` debe ser único dentro de la matriz. La propiedad `language` especifica el idioma del texto del documento.
- A continuación, llamamos al método `get_sentiments`, que usa el módulo `HTTPS` para realizar la llamada a Text Analytics API. Observe que la clave de suscripción, o de acceso, se pasa en el encabezado de cada solicitud.
- Cuando el servicio vuelve, se llama a `response_handler` y se registra la respuesta en la consola mediante `context.log`

### Pruébalo

Antes de adentrarnos en la ordenación en colas, vamos a realizar una serie de pruebas con lo que ya tenemos.

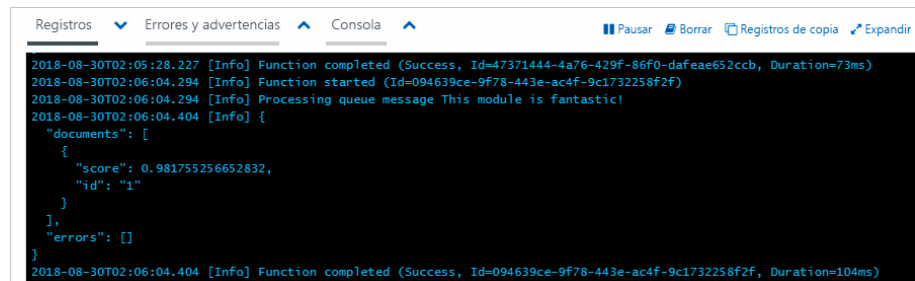
1. Con la función **discover-sentiment-function** seleccionada en el área Function App del portal, haga clic en el elemento de menú **Probar**, en el extremo derecho.

2. Haga clic en el elemento de menú **Probar** y compruebe que tiene abierto el panel de prueba.
3. Agregue una cadena de texto en el cuerpo de la solicitud tal como se indica en la captura de pantalla.



4. Haga clic en **Ejecutar** en la parte inferior del panel de prueba.
5. Asegúrese de que la pestaña **Registros** está expandida en la parte inferior izquierda de la pantalla principal, en el editor de código.

6. Compruebe que la pestaña **Registros** muestra la información de registro que ha completado la función. La ventana también mostrará la respuesta de la llamada a Text Analytics API.



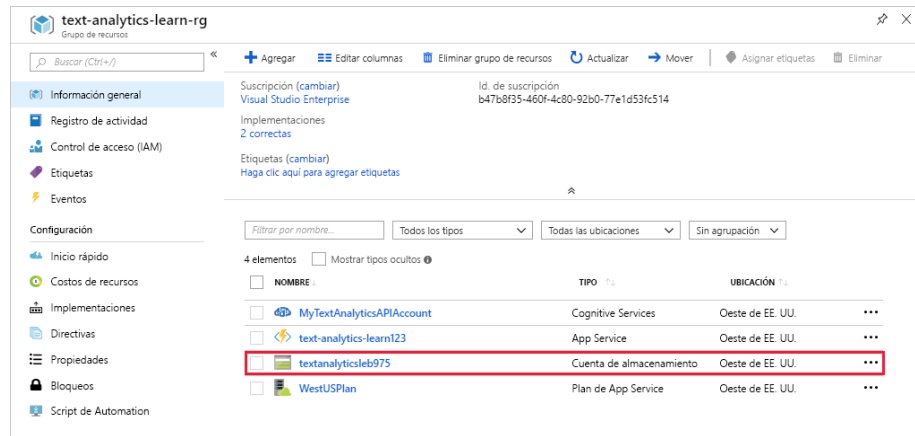
```
2018-08-30T02:05:28.227 [Info] Function completed (Success, Id=47371444-4a76-429f-86f0-dafeae652ccb, Duration=73ms)
2018-08-30T02:06:04.294 [Info] Function started (Id=094639ce-9f78-443e-ac4f-9c1732258f2f)
2018-08-30T02:06:04.294 [Info] Processing queue message This module is fantastic!
2018-08-30T02:06:04.404 [Info] {
  "documents": [
    {
      "score": 0.981755256652832,
      "id": "1"
    }
  ],
  "errors": []
}
2018-08-30T02:06:04.404 [Info] Function completed (Success, Id=094639ce-9f78-443e-ac4f-9c1732258f2f, Duration=104ms)
```

Enhorabuena. **discover-sentiment-function** funciona según lo previsto. En este ejemplo, se pasa un mensaje muy optimista y se recibe una puntuación superior a 0,98. Pruebe a cambiar el mensaje a algo menos optimista, vuelva a ejecutar la prueba y observe la respuesta.

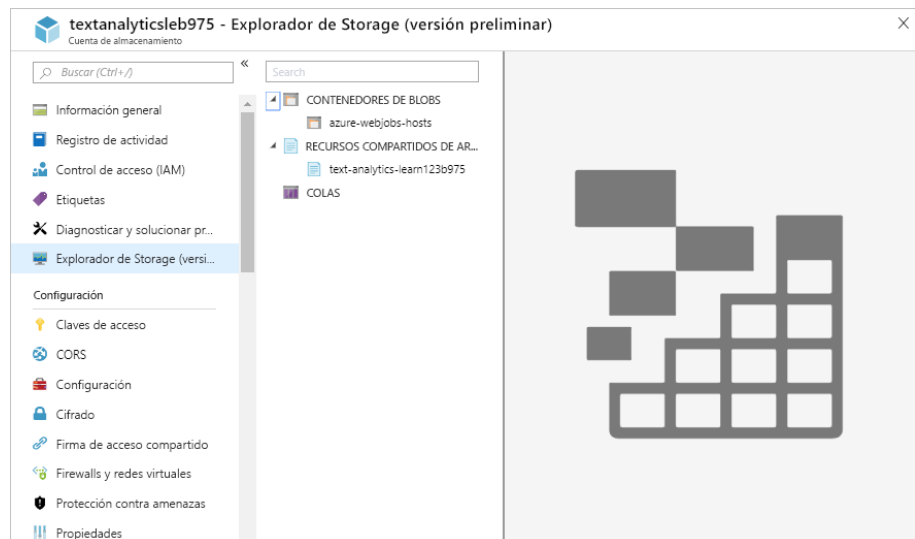
## Agregar un mensaje a la cola

Vamos a repetir la prueba. Esta vez, en lugar de usar la ventana de prueba del portal, vamos a colocar realmente un mensaje en la cola de entrada y ver qué sucede.

1. Vaya al grupo de recursos de la sección **Grupos de recursos** del portal.
2. Seleccione [nombre del grupo de recursos del espacio aislado], el grupo de recursos usado en esta lección.
3. En el panel **Grupo de recursos** que aparece, busque la entrada Cuenta de almacenamiento y selecciónela.



4. Seleccione **Explorador de Storage (versión preliminar)** en el menú izquierdo de la ventana principal de Cuenta de almacenamiento. Esta acción abre el Explorador de Azure Storage dentro del portal.



Como puede ver, todavía no tenemos ninguna cola en esta cuenta de almacenamiento, así que vamos a agregar una.

5. Si recuerda lo que se ha visto anteriormente en esta lección, la cola asociada al desencadenador se ha denominado **new-feedback-q**. Haga clic con el botón derecho en el



elemento **Colas** del Explorador de Storage y seleccione *Crear cola*.

6. En el cuadro de diálogo que se abre, escriba **new-feedback-q** y haga clic en **Aceptar**. Ahora ya tenemos nuestra cola de entrada.
7. Seleccione la nueva cola en el menú izquierdo para ver el explorador de datos de esta cola. Como era de esperar, la cola está vacía. Vamos a agregar un mensaje a la cola mediante el comando **Agregar mensaje** de la parte superior de la ventana.
8. En el cuadro de diálogo **Agregar mensaje**, escriba "Este mensaje procede de la cola de entrada new-feedback-q" en el campo **Texto del mensaje** y haga clic en **Aceptar** en la parte inferior del cuadro de diálogo.
9. Observe el mensaje, parecido al mensaje de la siguiente captura de pantalla, en el explorador de datos.

<a href="#">View Message</a> <a href="#">+ Add Message</a> <a href="#">- Dequeue Message</a> <a href="#">Clear Queue</a> <a href="#">Refresh</a>			
ID	MESSAGE TEXT	INSERTION TIME	EXPIRATION TIME
cc264b19-cde9-41e3-8f04-461cd49fcd349	This message came from our input queue, new-feedback-q	8/29/2018, 9:04:58 PM	9/5/2018, 9:04:58 PM

10. Después de unos segundos, haga clic en **Actualizar** para actualizar la vista de la cola. Observe que la cola vuelve a estar **vacía**. Algo debe haber leído el mensaje de la cola.
11. Vuelva a nuestra función en el portal y abra la pestaña **Supervisor**. Seleccione el mensaje más reciente de la lista. Observe que la función ha procesado el mensaje de cola

que se había publicado en new-feedback-q. Los resultados pueden retrasarse en este registro, por lo que es posible que deba esperar unos minutos y presionar *Actualizar*.

Refresh

Application Insights instance

test-analytics-learn

Success count in last 30 days

23

Error count in last 30 days

0

Query returned 20 items

[Run in Application Insights](#)

DATE (UTC) ▾

SUCCESS ▾

RESULT CODE ▾

DURATION (MS) ▾

2018-08-30 04:05:08.012

✓

0

629.5522

2018-08-30 02:06:04.294

✓

0

105.6421

2018-08-30 02:05:28.149

✓

0

75.3189

2018-08-30 02:01:49.338

✓

0

406.8484

2018-08-30 00:25:12.591

✓

0

39.403

2018-08-30 00:25:08.716

✓

0

28.004

2018-08-30 00:25:01.524

✓

0

34.6246

2018-08-30 00:25:00.303

✓

0

32.6631

2018-08-30 00:24:46.484

✓

0

83.4321

2018-08-30 00:23:53.237

✓

0

386.4977

2018-08-29 23:26:34.514

✓

0

109.3147

2018-08-29 23:21:38.613

✓

0

105.2029

Invocation Details

Run in Application Insights

MESSAGE

LOG LEVEL

Function started (Id=96d519b-5a5d-4d11-b125-832a7774e77)

Information

Processing queue message. This message came from our input queue, new-fe...

Information

{ "documents": [ { "score": 0.7731908825378418, "id": "1" } ], "errors": [] }

Information

Function completed (Success, Id=96d519b-5a5d-4d11-b125-832a7774e77, D...

Information

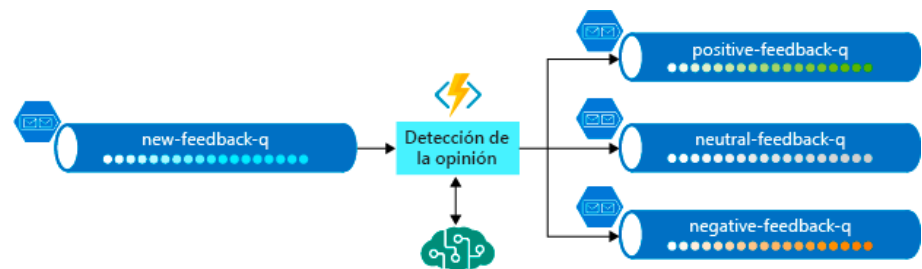
Processing queue message. This message came from our input queue, new-feedback-q

En esta prueba, hicimos un recorrido de ida y vuelta completo para publicar algo en nuestra cola y, posteriormente, ver cómo la función lo procesa.

Estamos progresando con nuestra solución. Nuestra función ahora está haciendo algo útil. Está recibiendo texto de la cola de entrada y, a continuación, llama al servicio Text Analytics API para obtener una puntuación de opinión. También hemos aprendido a probar nuestra función mediante Azure Portal y el Explorador de Storage. En el ejercicio siguiente, se verá lo fácil que es escribir en colas mediante enlaces de salida.

# EJERCICIO: ORDENACIÓN DE LOS MENSAJES EN COLAS DIFERENTES EN FUNCIÓN DE LA PUNTUACIÓN DE OPINIONES

**Veamos de** nuevo la arquitectura de la solución.



Como puede ver en el lado derecho de este diagrama, queremos enviar mensajes a tres colas. Para ello, vamos a definir esas conexiones como enlaces de salida en la función. Se pueden crear esos enlaces mediante la interfaz de usuario de **Enlace de salida**. Pero, para ahorrar tiempo, vamos a editar directamente el archivo de configuración.

## Adición de enlaces de salida a function.json

1. Seleccione la función **discover-sentiment-function** en el portal de Function App.

2. Expanda el menú **Ver archivos** a la derecha de la pantalla.
3. En la pestaña **Ver archivos**, seleccione **function.json** para abrir el archivo de configuración en el editor.
4. Reemplace todo el contenido de **function.json** por el siguiente JSON y haga clic en **Guardar**.

JSONCopiar

```
{  
  
  "bindings": [  
  
    {  
  
      "name": "myQueueItem",  
  
      "type": "queueTrigger",  
  
      "direction": "in",  
  
      "queueName": "new-feedback-q",  
  
      "connection": "AzureWebJobsStorage"  
  
    },  
  
    {  
  
      "type": "queue",  
  
      "name": "positiveFeedbackQueueItem",  
  
      "queueName": "positive-feedback-q",
```

```
        "connection": "AzureWebJobsStorage",  
  
        "direction": "out"  
    },  
  
    {  
  
        "type": "queue",  
  
        "name": "neutralFeedbackQueueItem",  
  
        "queueName": "neutral-feedback-q",  
  
        "connection": "AzureWebJobsStorage",  
  
        "direction": "out"  
    },  
  
    {  
  
        "type": "queue",  
  
        "name": "negativeFeedbackQueueItem",  
  
        "queueName": "negative-feedback-q",  
  
        "connection": "AzureWebJobsStorage",  
  
        "direction": "out"  
    }  
],
```

```
"disabled": false  
  
}
```

Hemos agregado tres nuevos enlaces a la configuración.

- Cada nuevo enlace es del tipo queue. Estos enlaces son para las tres colas que rellenaremos con nuestros mensajes de comentarios una vez que conozcamos la opinión de estos.
- Cada enlace tiene una dirección definida como out, ya que vamos a publicar los mensajes en estas colas.
- Cada enlace usa la misma conexión a nuestra cuenta de almacenamiento.
- Cada enlace tiene un único queueName y name.

46

Publicar un mensaje en una cola es tan sencillo como decir, por ejemplo, `context.bindings.negativeFeedbackQueueItem` = "`<message>`".

### **Actualización de la implementación de la función para ordenar los comentarios en colas según la puntuación de opinión**

El objetivo de nuestro clasificador de comentarios es ordenar los comentarios en tres cubos: positivos, negativos y neutrales. Hasta el momento tenemos la cola de entrada, el código para llamar a Text Analytics API y hemos definido las colas de salida. En esta sección, vamos a agregar la lógica para trasladar los mensajes a esas colas según las opiniones.

1. Vaya a nuestra función, **discover-sentiment-function** y vuelva a abrir `index.js` en el editor de código.

2. Reemplace la implementación por el código siguiente.

JavaScriptCopiar

```
module.exports = function (context, myQueueItem) {

    context.log('Processing queue message', myQueueItem);


    let https = require ('https');


    // Replace the accessKey string value with your valid access key.

    let accessKey = '<YOUR ACCESS CODE HERE>';


    // Replace [region], including square brackets, in the uri variable
    below.

    // You must use the same region in your REST API call as you used
    to obtain your access keys.

    // For example, if you obtained your access keys from the
    northeurope region, replace

    // "westus" in the URI below with "northeurope".

    let uri = '[region].api.cognitive.microsoft.com';

    let path = '/text/analytics/v2.0/sentiment';
```

```
let response_handler = function (response) {  
  
    let body = "";  
  
    response.on ('data', function (chunk) {  
  
        body += chunk;  
  
    });  
  
    response.on ('end', function () {  
  
        let body_ = JSON.parse (body);  
  
        // Even though we send and receive a documents array from  
the Text Analytics API,  
  
        // we only ever pass one document in the array.  
  
        if (body_.documents && body_.documents.length == 1) {  
  
            let score = body_.documents[0].score;  
  
            // Create a message that contains the original message we  
received and
```



```
// the sentiment score returned by Text Analytics API.

let messageWithScore = JSON.stringify({

    originalMessage: myQueueItem,

    score: score

});


// Place message into appropriate output queue based on
sentiment score.

if (score > 0.8) {

    context.log ("Positive message arrived");

    context.bindings.positiveFeedbackQueueItem      =
messageWithScore;

} else if (score < 0.3) {

    context.log ("Negative message arrived");

    context.bindings.negativeFeedbackQueueItem      =
messageWithScore;

} else {

    context.log ("Neutral message arrived");

    context.bindings.neutralFeedbackQueueItem      =
messageWithScore;
```

```
    }  
  
  }  
  
  let body__ = JSON.stringify (body_, null, ' ');  
  
  context.log (body__);  
  
  context.done();  
  
  return;  
  
});
```

```
response.on ('error', function (e) {  
  
  context.log ('Error: ' + e.message);  
  
  context.done();  
  
  return;  
  
});  
  
};
```

```
let get_sentiments = function (documents) {  
  
  let body = JSON.stringify (documents);
```

```
let request_params = {  
  
    method : 'POST',  
  
    hostname : uri,  
  
    path : path,  
  
    headers : {  
  
        'Ocp-Apim-Subscription-Key' : accessKey,  
  
    }  
  
};
```

```
let req = https.request (request_params, response_handler);  
  
req.write (body);  
  
req.end ();  
  
}
```

// Create a documents array with one entry.

```
let documents = { 'documents': [  
  
    {
```

```

        'id': '1',

        'language': 'en',

        'text': myQueueItem

    },

    ]];

    get_sentiments (documents);

};

```

52

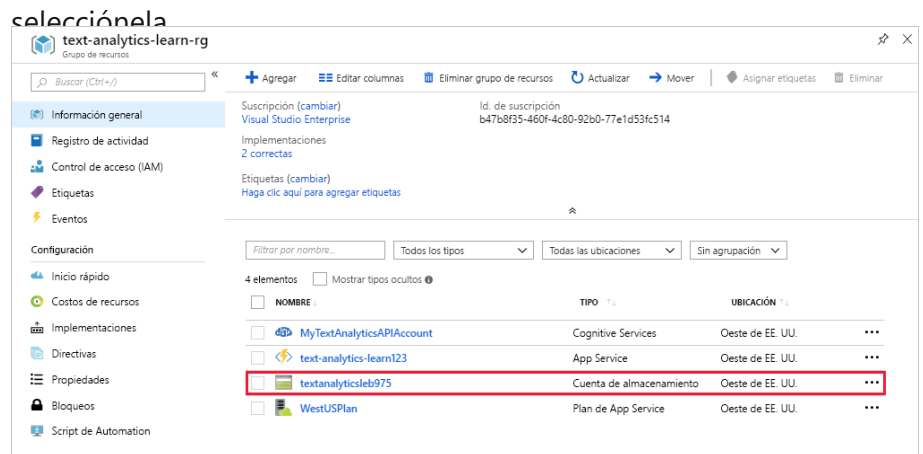
Hemos agregado el código resaltado a la implementación. El código analiza la respuesta de Text Analytics API de Cognitive Services. Según la puntuación de la opinión, se reenvía el mensaje a una de las tres colas de salida. El código para publicar el mensaje simplemente establece el parámetro de enlace correcto.

### Pruébalo

Para probar la implementación actualizada, vamos a volver al Explorador de Storage.

1. Vaya al grupo de recursos de la sección **Grupos de recursos** del portal.
2. Seleccione [nombre del grupo de recursos del espacio aislado], el grupo de recursos usado en esta lección.

3. En el panel **Grupo de recursos** que se abre, localice la entrada **Cuenta de almacenamiento** y

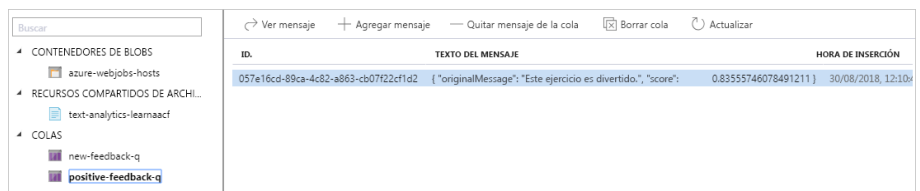


4. Seleccione **Explorador de Storage (versión preliminar)** en el menú izquierdo de la ventana principal de Cuenta de almacenamiento. Esta acción abre el Explorador de Azure Storage dentro del portal.



Tenemos una cola que aparece en la colección **Colas**. Esta cola es **new-feedback-q**, la cola de entrada que hemos definido en la sección anterior de prueba del módulo.

5. Seleccione **new-feedback-q** en el menú izquierdo para ver el explorador de datos de esta cola. Como era de esperar, la cola no contenía datos. Vamos a agregar un mensaje a la cola mediante el comando **Agregar mensaje** de la parte superior de la ventana.
6. En el cuadro de diálogo **Agregar mensaje**, escriba "Me estoy divirtiendo con este ejercicio". en el campo **Texto del mensaje** y haga clic en **Aceptar** en la parte inferior del cuadro de diálogo.
7. El mensaje aparece en la ventana de datos de **new-feedback-q**. Después de unos segundos, haga clic en **Actualizar** en la parte superior de la vista de datos para actualizar la vista de la cola. Observe que el mensaje desaparece después de unos minutos. ¿Dónde se fue?
8. Haga clic con el botón derecho en la colección **COLAS** en el menú de la izquierda. Observe que ha aparecido una *nueva* cola.

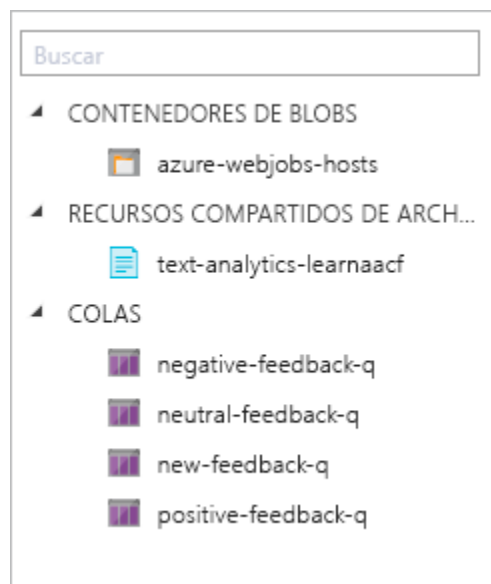


The screenshot shows the Azure Functions portal interface. On the left, a sidebar menu lists various resources under 'COLAS', with 'positive-feedback-q' highlighted. The main panel displays the 'new-feedback-q' queue. At the top of the main panel, there are buttons: 'Ver mensaje', 'Agregar mensaje', 'Quitar mensaje de la cola', 'Borrar cola', and 'Actualizar'. Below these buttons is a table with the following data:

ID	TEXTO DEL MENSAJE	HORA DE INSERCIÓN
057e16cd-89ca-4c82-a863-cb07f22cf1d2	{ "originalMessage": "Este ejercicio es divertido.", "score": 0.83555746078491211 }	30/08/2018, 12:10h

La cola **positive-feedback-queue** se creó automáticamente cuando se publicó un mensaje en ella por primera vez. Con los enlaces de salida de la cola de Azure Functions, no es necesario crear manualmente la cola de salida antes de publicar en ella. Ahora que vemos que nuestra función ha clasificado un mensaje entrante en **positive-feedback-queue**, vamos a ver dónde van los siguientes mensajes.

9. Con los mismos pasos de arriba, agregue los siguientes mensajes a **new-feedback-q**.
- "Me gusta el brócoli"
  - "Microsoft es una empresa"
10. Haga clic en **Actualizar** hasta que **new-feedback-q** esté nuevamente vacío. Este proceso podría tardar unos minutos y se requieren varias actualizaciones.
11. Haga clic con el botón derecho en la colección **COLAS** y observe que aparecen dos colas más. Las colas se denominan **neutral-feedback-queue** y **negative-feedback-queue**. Esto puede tardar unos segundos, por ello, continúe actualizando la colección **COLAS** hasta que aparezcan nuevas colas. Cuando termine, la lista de colas debería tener el siguiente aspecto.



12. Haga clic en cada cola de la lista para ver si tiene mensajes. Si ha agregado los valores sugeridos, debería ver uno en **positive-feedback-queue**, **neutral-feedback-queue** y **negative-feedback-queue**.

Enhorabuena. Ahora ya tenemos un clasificador de comentarios operativo. Conforme llegan los mensajes a la cola de entrada, nuestra función usa el servicio Text Analytics API para obtener una puntuación de opinión. En función de esa puntuación, la función reenvía los mensajes a la cola adecuada. Aunque parece que la función procesa solo un elemento de la cola cada vez, el runtime de Azure Functions realmente lee lotes de elementos de la cola y pone en marcha otras instancias de la función para procesarlas en paralelo.



---

# RESUMEN

**Azure Cognitive Services** es un completo conjunto de servicios inteligentes que se puede usar para enriquecer las aplicaciones. Text Analytics API tiene varias operaciones para convertir texto en conclusiones significativas. Hemos usado el servicio para detectar opiniones en los comentarios de texto de los clientes. Hemos creado una solución hospedada en Azure Functions para clasificar estos mensajes de texto en cubos distintos, o colas, para su posterior procesamiento.

Una vez que sepa cómo llamar a una API REST, puede integrar fácilmente estos servicios inteligentes en sus soluciones. Todos ellos siguen un patrón similar:

- Regístrese para obtener una clave de acceso.
- Explore en la consola de pruebas de la API.
- Formule solicitudes mediante la clave de acceso y la región adecuada.
- PUBLIQUE las solicitudes desde la solución y analice las respuestas para sacar conclusiones.

Hemos agregado esta inteligencia a la lógica sin servidor que creó en Azure Functions. Puede llamar fácilmente a estos servicios desde otros tipos de aplicaciones. Hay muchas bibliotecas de cliente, tutoriales y guías de inicio rápido para comenzar.

### Sugerencias para una mejora aún mayor de la solución

Aquí le presentamos algunas ideas a tener en cuenta si desea ampliar lo que ya hemos hecho.

- Pruebe la solución con más ejemplos de texto. Decida si los umbrales que establecemos para clasificar las puntuaciones de opinión en positivas, negativas y neutrales son adecuados.
- Considere la posibilidad de agregar otra función a la aplicación de función que lea los mensajes de la cola **negative-feedback-queue** y llame a Text Analytics API para buscar frases clave en el texto.
- Nuestra cola de entrada contiene comentarios de texto sin formato. En el mundo real, asociaríamos los comentarios con algún tipo de identificador de usuario, como la dirección de correo electrónico, el número de cuenta, etc. Mejore los elementos de la cola de entrada para que sean documentos JSON que contengan un campo de identificador y el texto. Después, use ese identificador cuando trabaje con el mensaje de texto.
- En la actualidad, nuestra aplicación está "codificada de forma rígida" en inglés. Piense en los cambios que implementaría para

que pudiera controlar texto en todos los idiomas compatibles mediante Text Analytics API.

- Si está familiarizado con Logic Apps, visite el vínculo para el conector integrado para el análisis de texto en la sección Lecturas adicionales.

Ahora que sabe cómo llamar a una de estas Cognitive Services APIs, explore algunos de los otros servicios y piense cómo puede usarlos en sus soluciones.

### **Lecturas adicionales**

- [Introducción a Text Analytics](#)
- [Demostración de Text Analytics](#)
- [Cómo detectar opiniones en Text Analytics](#)
- [Documentación de Cognitive Services](#)
- [Conector de Text Analytics para Logic Apps](#)

### **Limpieza**

El espacio aislado limpia los recursos automáticamente cuando haya terminado con este módulo.

Al trabajar en una suscripción propia, se recomienda identificar al final de un proyecto si aún necesita los recursos creados. Dejar recursos en funcionamiento puede costarle dinero. Puede eliminar los recursos de forma individual o eliminar el grupo de recursos para eliminar todo el conjunto de recursos.