

Azure para desarrolladores de Python

Puede implementar las aplicaciones de Python en Azure en varios entornos, incluidas las aplicaciones web, las funciones sin servidor, los contenedores y los modelos de aprendizaje automático. Azure ofrece opciones de hospedaje flexibles para satisfacer diferentes necesidades de arquitectura y escalabilidad. Para interactuar con los servicios de Azure mediante programación, use el SDK de Azure para Python. Este conjunto de bibliotecas simplifica tareas como la autenticación, la administración de recursos y la integración de servicios, lo que facilita la compilación de aplicaciones en la nube seguras y escalables con Python.

Bibliotecas de Azure (SDK)

INTRODUCCIÓN

[Comienza](#)

[Conozca las bibliotecas de Azure](#)

[Aprender patrones de uso de bibliotecas](#)

[Autenticación con servicios de Azure](#)

Aplicaciones web

TUTORIAL

[Creación e implementación rápida de nuevas aplicaciones de Django/ Flask/FastAPI](#)

[Implementación de una aplicación web de Django o Flask](#)

[Implementación de una aplicación web con PostgreSQL](#)

[Implementación de una aplicación web con identidad administrada](#)

[Implementación mediante Acciones de GitHub](#)

INTELIGENCIA ARTIFICIAL

INICIO RÁPIDO

[Desarrollo con los servicios de Azure AI](#)

Contenedores



TUTORIAL

[Introducción a los contenedores de Python](#)

[Implementación en App Service](#)

[Implementación en Container Apps](#)

[Implementación de un clúster de Kubernetes](#)

Datos y almacenamiento



INICIO RÁPIDO

[Bases de datos SQL](#)

[Tablas, blobs, archivos, NoSQL](#)

[Macrodatos y análisis](#)

Aprendizaje automático



GUÍA PASO A PASO

[Creación de un experimento de Machine Learning](#)

[Entrenamiento de un modelo de predicción](#)

[Creación de canalizaciones de Machine Learning](#)

[Uso de servicios de IA listos \(cara, voz, texto, imagen, etc.\)](#)

[Sin servidor, ETL en la nube](#)

Funciones sin servidor



GUÍA PASO A PASO

[Implementación mediante Visual Studio Code](#)

[Implementación mediante la línea de comandos](#)

[Conexión al almacenamiento mediante Visual Studio Code](#)

[Conexión al almacenamiento mediante la línea de comandos](#)

Herramientas para desarrolladores

INTRODUCCIÓN

[Visual Studio Code \(IDE\)](#) ↗

[Interfaz de la línea de comandos \(CLI\) de Azure](#)

[Subsistema de Windows para Linux \(WSL\)](#)

[Visual Studio \(para el desarrollo de Python/C++\)](#)

Introducción a Python en Azure

Artículo • 03/02/2025

Si no está familiarizado con el desarrollo de aplicaciones para la nube, esta breve serie de 8 artículos es el mejor lugar para empezar.

- Parte 1: [Resumen de Azure para desarrolladores](#)
- Parte 2: [Servicios de Azure clave para desarrolladores](#)
- Parte 3: [hospedaje de aplicaciones en Azure](#)
- Parte 4: [Conexión de la aplicación a los servicios de Azure](#)
- Parte 5: [Cómo crear y administrar recursos en Azure](#)
- Parte 6: [Conceptos clave para compilar aplicaciones de Azure](#)
- Parte 7: [¿Cómo se me factura?](#)
- Parte 8: [Directiva de control de versiones para servicios, SDK y herramientas de la CLI de Azure](#)

Creación de una cuenta de Azure

Para desarrollar aplicaciones de Python con Azure, necesita una cuenta de Azure. Tu cuenta de Azure son las credenciales que usas para iniciar sesión y para crear recursos de Azure.

Si usa Azure en el trabajo, hable con el administrador de la nube de su empresa para obtener sus credenciales para iniciar sesión en Azure.

De lo contrario, puede crear una cuenta de [Azure gratuitamente](#) y recibir 12 meses de servicios populares gratuitos y un crédito de 200 USD para explorar Azure durante 30 días.

[Crear una cuenta de Azure de forma gratuita](#)

Crear y administrar recursos de Azure

Para usar recursos de Azure como bases de datos, colas de mensajes, almacenamiento de archivos, etc., primero debe crear una instancia del recurso. La creación de recursos implica:

- elección de la capacidad o las opciones de informática
- adición de un recurso nuevo a un grupo de recursos
- selección de la región del mundo donde se ejecuta el servicio

- asignación de un nombre único al servicio

Hay varias herramientas que puede usar para crear y administrar recursos de Azure, en función de su escenario:

- [Azure Portal](#): si no está familiarizado con Azure y quiere que una interfaz de usuario basada en web cree y administre un par de recursos.
- [Azure CLI](#) - si te sientes más cómodo con las interfaces de línea de comandos.
- [Azure PowerShell](#): Si prefiere una sintaxis al estilo de PowerShell en la interfaz de línea de comandos (CLI).
- [CLI para desarrolladores de Azure](#) - cuando usted desea crear implementaciones repetibles que incluyen muchos recursos de Azure con dependencias complejas. Requiere el aprendizaje de plantillas de Bicep.
- [Paquete de extensiones de herramientas de Azure](#): el paquete de extensiones contiene extensiones para trabajar con algunos de los servicios de Azure más populares en un práctico paquete.

También puede usar las [bibliotecas de administración de Azure para Python](#) para crear y administrar recursos. Las bibliotecas de administración permiten usar Python para implementar la funcionalidad de administración e implementación personalizada. A continuación se ofrecen son algunos artículos que pueden ayudarle a comenzar:

- [Crear un grupo de recursos](#)
- [Enumeración de grupos y recursos](#)
- [Creación de una instancia de Azure Storage](#)
- [Creación e implementación de una aplicación web](#)
- [Crear y consultar una base de datos](#)
- [Crear una máquina virtual](#)

Escribe tu aplicación de Python

El desarrollo en Azure requiere [Python](#) 3.8 o posterior. Para comprobar la versión de Python en la estación de trabajo, en una ventana de consola, escriba el comando

`python3 --version` para macOS/Linux o `py --version` para Windows.

Use sus herramientas favoritas para escribir la aplicación de Python. Si usa Visual Studio Code, debe probar la [extensión de Python para Visual Studio Code](#).

La mayoría de las instrucciones de este conjunto de artículos usan un entorno virtual porque es un procedimiento recomendado. No dude en usar cualquier entorno virtual que desee, pero las instrucciones del artículo hacen referencia a `venv`.

Uso de bibliotecas de clientes

Cuando empieza a trabajar, los artículos le instruyen sobre qué bibliotecas de Python en Azure deben instalarse y referenciarse utilizando la utilidad `pip`.

En algún momento, es posible que quiera [instalar y crear una referencia](#) a las [bibliotecas cliente de Azure SDK para Python](#) sin seguir las instrucciones de un artículo. La [Información general de Azure SDK](#) es un excelente punto de partida.

Autenticación de la aplicación en Azure

Al usar el SDK de Azure para Python, debe agregar lógica de autenticación a la aplicación. La autenticación de la aplicación depende de si está ejecutando la aplicación localmente durante el desarrollo y las pruebas, hospedando la aplicación en sus propios servidores o hospedando la aplicación en Azure. Para más información sobre la autenticación en Azure, consulte [Autenticación de aplicaciones de Python en servicios de Azure mediante el SDK de Azure para Python](#).

También deberá configurar directivas de acceso que controlan qué identidades (entidades de servicio y/o identificadores de aplicación) pueden acceder a esos recursos. Las directivas de acceso se administran mediante el [control de acceso basado en roles \(RBAC\)](#) de Azure; algunos servicios también tienen controles de acceso más específicos. Como desarrollador en la nube que trabaja con Azure, asegúrese de familiarizarse con Azure RBAC, ya que se usa prácticamente con cualquier recurso que tenga cuestiones de seguridad.

Agregar intereses transversales

- Gestione los secretos de su aplicación mediante [Azure Key Vault](#)
- Obtención de visibilidad de la aplicación mediante el registro con [Azure Monitor](#)

Hosting de la aplicación de Python

Si quiere que el código de la aplicación se ejecute en Azure, dispone de varias opciones, como se describe en [Hosting de aplicaciones en Azure](#).

Si va a compilar aplicaciones web o API (Django, Flask, FastAPI, etc.), tenga en cuenta lo siguiente:

- [Azure App Service](#)
- [Azure App Service \(ya en contenedor\)](#)
- [Azure Container Apps](#)

- clúster de Kubernetes de Azure

Si va a compilar una aplicación web, consulte [Configuración del entorno local para implementar aplicaciones web de Python en Azure](#).

Además, si va a compilar una API web, debe considerar la posibilidad de usar [Azure API Management](#).

Si va a desarrollar procesos de back-end:

- [Azure Functions](#)
- [Trabajos web de Azure App Service](#)
- [Azure Container Apps](#)

Pasos siguientes

- [Desarrollo de una aplicación web de Python](#)
- [Desarrollo de una aplicación de contenedor](#)
- [Aprenda a usar las bibliotecas de Azure para Python](#)

Comentarios

¿Le ha resultado útil esta página?

 Sí

 No

[Proporcionar comentarios sobre el producto](#) | [Obtener ayuda en Microsoft Q&A](#)

Desarrollo de aplicaciones de IA con Python

25/06/2025

En este artículo se ofrece una lista seleccionada de los principales recursos de aprendizaje para desarrolladores de Python que no están familiarizados con la creación de aplicaciones de inteligencia artificial. Incluye vínculos a guías de inicio rápido, proyectos de ejemplo, documentación oficial, cursos de formación y otros materiales útiles.

Recursos para Azure OpenAI Service

El servicio Azure OpenAI proporciona acceso a la API REST a los modelos de lenguaje eficaces disponibles en OpenAI. Azure OpenAI le ayuda a adaptar estos modelos para realizar tareas específicas, como la generación de contenido, el resumen, la comprensión de imágenes, la búsqueda semántica y el lenguaje natural a la traducción de código. Acceda a Azure OpenAI mediante las API REST, el SDK de Azure OpenAI para .NET o la interfaz basada en web en [Azure OpenAI Studio](#).

SDK y bibliotecas

 Expandir tabla

Link	Descripción
SDK de OpenAI para Python	La versión de código fuente de GitHub de la biblioteca de Python de OpenAI, que proporciona un acceso cómodo a la API de OpenAI desde aplicaciones escritas en el lenguaje Python.
Paquete de Python de OpenAI	Versión de PyPi de la biblioteca de OpenAI Python.
Cambio de OpenAI a Azure OpenAI	Un artículo de instrucciones sobre los pequeños cambios que debe realizar en su código, para que pueda alternar fácilmente entre OpenAI y el Servicio de Azure OpenAI.
Finalizaciones de chat en streaming	Ejemplo de cuaderno que demuestra cómo hacer que las finalizaciones de chat funcionen utilizando los puntos de conexión de Azure. El ejemplo se centra en las finalizaciones de chat, pero también presenta otras operaciones disponibles con la API.
Incrustaciones de Azure	Ejemplo de cuaderno que muestra cómo usar incrustaciones con puntos de conexión de Azure. El ejemplo se centra en las incrustaciones, pero también presenta otras operaciones disponibles con la API.

Link	Descripción
Implementación del modelo y generación de texto	Un artículo con pasos detallados mínimos y sencillos para implementar un modelo que puede chatear mediante programación.
OpenAI con el control de acceso basado en roles de Microsoft Entra ID	Una mirada a la autenticación mediante Microsoft Entra ID y el control de acceso basado en roles de Azure .
OpenAI con identidades administradas por Azure AD para recursos de Azure	Un artículo con escenarios de seguridad más complejos que requieren el control de acceso basado en roles de Azure. Explore cómo autenticarse en el recurso de OpenAI con el identificador de Microsoft Entra.
Ejemplos del servicio Azure OpenAI	Una compilación de recursos útiles de Azure OpenAI Service y ejemplos de código que le ayudarán a empezar a trabajar y acelerar el recorrido de adopción de la tecnología.

Documentación

 Expandir tabla

Link	Descripción
Documentación del servicio Azure OpenAI	La página central de la documentación de Azure OpenAI Service.
Inicio rápido: Introducción a la generación de texto con el servicio Azure OpenAI	Inicio rápido que muestra cómo configurar los servicios que necesita y escribir código para solicitar un modelo mediante Python.
Inicio rápido: empezar a utilizar GPT-35-Turbo y GPT-4 con Azure OpenAI Service	Inicio rápido que muestra cómo trabajar con roles de usuario, asistente y sistema para adaptar el contenido en respuesta a determinadas preguntas.
Inicio rápido: Chatear con modelos de Azure OpenAI usando sus propios datos	Inicio rápido que le ayuda a agregar sus propios datos, como un PDF u otro documento.
Inicio rápido: Introducción al uso de Azure OpenAI Assistants (versión preliminar)	Inicio rápido que muestra cómo indicar a un modelo que use el intérprete de código de Python integrado para resolver problemas matemáticos paso a paso. En este ejemplo se proporciona un punto de partida para usar sus propios asistentes de IA a los que se accede a través de instrucciones personalizadas.
Inicio rápido: Usar imágenes en los chats de IA	Inicio rápido que muestra cómo pedir mediante programación a un modelo que describa el contenido de una imagen.

Link	Descripción
Inicio rápido: Generación de imágenes con Azure OpenAI Service	Inicio rápido que demuestra cómo generar imágenes de manera programática usando Dall-E en función de una indicación.

Recursos para otros servicios de Azure AI

Además del servicio Azure OpenAI, hay muchos otros servicios de Azure AI. Los desarrolladores y organizaciones pueden crear rápidamente aplicaciones inteligentes, listas para el mercado y responsables con api y modelos personalizables precompilados y listos para usar. Algunos ejemplos de aplicaciones son el procesamiento del lenguaje natural para conversaciones, búsqueda, supervisión, traducción, voz, visión y toma de decisiones.

Ejemplos

[\[+\] Expandir tabla](#)

Link	Descripción
Integración de voz en las aplicaciones con ejemplos del SDK de Voz de Azure AI ↗	Ejemplos para el SDK de Voz de Azure Cognitive Services. Vínculos a ejemplos de reconocimiento de voz, traducción, síntesis de voz, etc.
SDK de Documento de inteligencia de Azure AI	Documento de inteligencia de Azure AI (anteriormente Form Recognizer) es un servicio en la nube que usa el aprendizaje automático para analizar texto y datos estructurados de documentos. El kit de desarrollo de software (SDK) de Documento de inteligencia es un conjunto de bibliotecas y herramientas que le permiten integrar fácilmente en sus aplicaciones los modelos y las funcionalidades de Documento de inteligencia en sus aplicaciones.
Extracción de datos estructurados de formularios, recibos, facturas y tarjetas mediante Form Recognizer en Python ↗	Muestras para la biblioteca cliente Azure.AI.FormRecognizer.
Extracción, clasificación y comprensión del texto de los documentos con Text Analytics en Python	Biblioteca cliente para Text Analytics. Estas API forman parte del servicio lenguaje de Azure AI , que proporciona características de procesamiento de lenguaje natural (NLP) para comprender y analizar texto.
Traducción de documentos en Python	Artículo de inicio rápido que usa la traducción de documentos para traducir un documento de origen a un idioma de destino conservando la estructura y el formato del texto.

Link	Descripción
Respuesta a preguntas en Python	Artículo de inicio rápido con pasos para obtener una respuesta (y una puntuación de confianza) de un cuerpo de texto que envíe junto con su pregunta.
Reconocimiento del lenguaje conversacional en Python	La biblioteca cliente de Conversational Language Understanding (CLU). CLU es un servicio de inteligencia artificial conversacional basado en la nube que puede extraer intenciones y entidades en conversaciones. CLU actúa como un orquestador para seleccionar el mejor candidato para analizar conversaciones para obtener la mejor respuesta de aplicaciones como QnA, Luis y Conversation App.
Análisis de imágenes	Código de ejemplo y documentos de configuración para el SDK de análisis de imágenes de Microsoft Azure AI.
SDK de Seguridad del contenido de Azure AI para Python	El SDK puede ayudar a detectar contenido perjudicial generado por usuarios y por IA en aplicaciones y servicios. Content Safety incluye varias API de texto e imagen que permiten detectar todo aquel material que sea perjudicial.

Documentación

 Expandir tabla

Servicio de IA	Descripción	Referencia de API	Inicio rápido
Seguridad del contenido	Un servicio de IA que detecta contenido no deseado.	Referencia de API Content Safety	Inicio rápido
Inteligencia de documentos	Convertir los documentos en soluciones inteligentes controladas por datos.	Referencia de API de Documento de inteligencia	Inicio rápido
Lenguaje	Creación de aplicaciones con funcionalidades de reconocimiento del lenguaje natural líderes del sector.	Referencia de Text Analytics API	Inicio rápido
Search	Lleve la búsqueda en la nube con tecnología de inteligencia artificial a sus aplicaciones.	Referencia de la API de Search	Inicio rápido
Voz	Conversión de voz en texto, texto a voz, traducción y reconocimiento del hablante.	Referencia de la API de voz	Inicio rápido
Translator	Use servicios de traducción con tecnología de inteligencia artificial para traducir más de 100 idiomas y dialectos en peligro o en peligro de extinción.	Referencia de API de traducción	Inicio rápido

Servicio de IA	Descripción	Referencia de API	Inicio rápido
Visión	Análisis de contenido en imágenes y vídeos.	Referencia de API Análisis de imágenes	Inicio rápido

Cursos

Expandir tabla

Link	Descripción
Taller de inteligencia artificial generativa para principiantes ↗	Conozca los aspectos básicos de la creación de aplicaciones de inteligencia artificial generativa con nuestro curso completo de 18 lecciones por parte de los defensores de Microsoft Cloud.
Introducción a los servicios de Azure AI	Los servicios de Azure AI son bloques de creación de la funcionalidad de inteligencia artificial que puede integrar en sus aplicaciones. Complete esta ruta de aprendizaje para explorar cómo aprovisionar, proteger, supervisar e implementar recursos de servicios de Azure AI y usarlos para crear soluciones inteligentes.
Aspectos básicos de Microsoft Azure AI: IA generativa	Complete esta ruta de aprendizaje para comprender cómo los modelos de lenguaje de gran tamaño forman la base de la inteligencia artificial generativa. Explore cómo Azure OpenAI Service proporciona acceso a la última tecnología de inteligencia artificial generativa. Obtenga información sobre cómo se pueden ajustar las solicitudes y respuestas de Azure OpenAI y cómo los principios de inteligencia artificial responsable de Microsoft impulsan los avances éticos en la inteligencia artificial.
Desarrollo de soluciones de inteligencia artificial generativa con Azure OpenAI Service	Azure OpenAI Service proporciona acceso a los potentes y grandes modelos de lenguaje de OpenAI, como los modelos ChatGPT, GPT, Codex y Embeddings. Complete esta ruta de aprendizaje para desarrolladores y explore cómo generar código, imágenes y texto mediante el SDK de Azure OpenAI y otros servicios de Azure.
Creación de aplicaciones de IA con Azure Database for PostgreSQL	Complete esta ruta de aprendizaje para explorar las integraciones de Azure AI y Azure Machine Learning Services proporcionadas por la extensión de Azure AI para Azure Database for PostgreSQL: servidor flexible. Obtenga información sobre cómo estos servicios pueden permitirle crear aplicaciones con tecnología de inteligencia artificial.

Plantillas de aplicación de IA

Las plantillas de aplicación de IA proporcionan implementaciones de referencia bien mantenidas y fáciles de implementar que proporcionan un punto de partida de alta calidad

para las aplicaciones de INTELIGENCIA ARTIFICIAL.

Hay dos categorías de plantillas de aplicación de IA, **bloques de creación y soluciones de un extremo a otro**. Los bloques de creación son ejemplos a menor escala que se centran en escenarios y tareas específicos. Las soluciones de un extremo a otro son ejemplos de referencia completos que incluyen documentación, código fuente y características de implementación. Puede crear soluciones y ampliarlas para sus propios fines.

- Para revisar una lista de plantillas clave disponibles para cada lenguaje de programación, consulte [Plantillas de aplicación de IA](#).
- Para examinar todas las plantillas disponibles, consulte las plantillas de aplicación de IA en la galería de la [CLI para desarrolladores de Azure](#) ↗.

Introducción: Chatear con sus propios datos (ejemplo de Python)

25/06/2025

En este artículo se muestra cómo implementar y ejecutar el [Chat usando tu propio conjunto de datos mediante código de ejemplo para Python](#). Esta aplicación de chat de ejemplo se compila con Python, Azure OpenAI Service y [Recuperación de generación aumentada \(RAG\)](#) a través de Azure AI Search.

La aplicación proporciona respuestas a preguntas del usuario sobre las ventajas de los empleados en una empresa ficticia. Usa Retrieval-Augmented Generación (RAG) para hacer referencia al contenido de los archivos PDF proporcionados, que pueden incluir:

- Manual de un empleado
- Documento de información general sobre las ventajas
- Lista de roles y expectativas de la empresa

Al analizar estos documentos, la aplicación puede responder a consultas de lenguaje natural con respuestas precisas y relevantes contextualmente. Este enfoque muestra cómo puede usar sus propios datos para impulsar experiencias de chat inteligentes específicas del dominio con Azure OpenAI y Azure AI Search.

También aprenderá a configurar la configuración de la aplicación para modificar su comportamiento de respuesta.

Después de completar los pasos de este artículo, puede empezar a personalizar el proyecto con su propio código. Este artículo forma parte de una serie que le guía a través de la creación de una aplicación de chat con Azure OpenAI Service y Azure AI Search. Otros artículos de esta serie son:

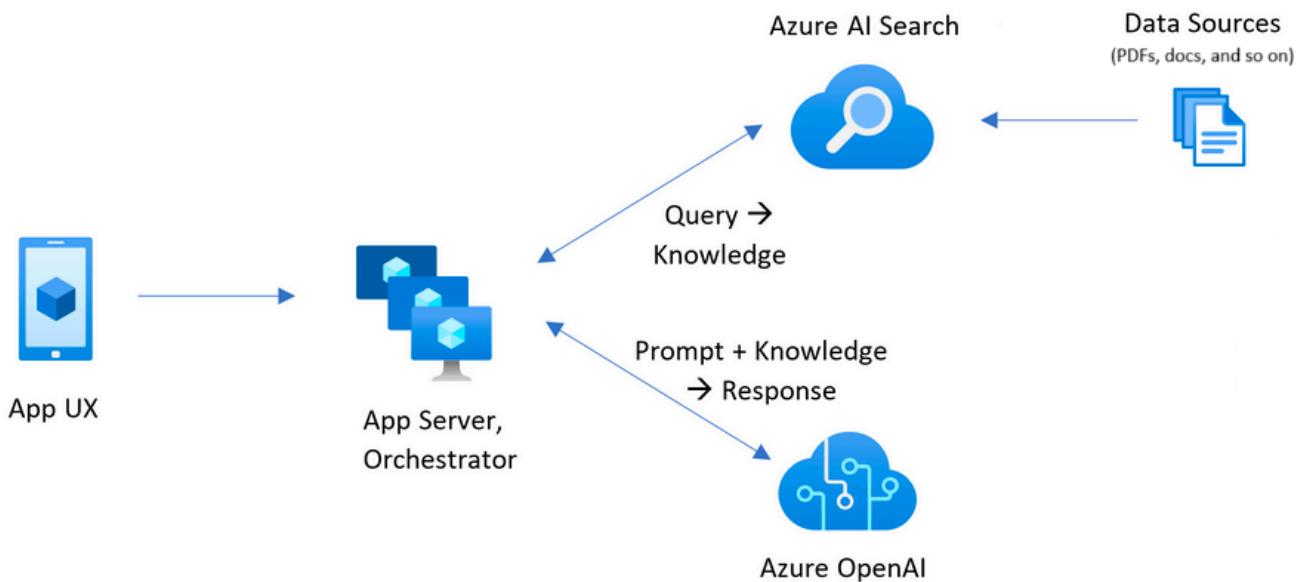
- [.NET](#)
- [Java](#)
- [JavaScript](#)
- [Front-end de JavaScript con back-end de Python](#)

Nota:

Este artículo se basa en una o varias [plantillas de aplicación de IA](#), que sirven para implementaciones de referencia bien mantenidas. Estas plantillas están diseñadas para ser fáciles de implementar y proporcionar un punto de partida confiable y de alta calidad para crear sus propias aplicaciones de inteligencia artificial.

Arquitectura de una aplicación de ejemplo

En el diagrama siguiente se muestra una arquitectura sencilla de la aplicación de chat.



Entre los componentes clave de la arquitectura se incluyen:

- Una aplicación web que hospeda la interfaz de chat interactiva (normalmente creada con Python Flask o JavaScript/React) y envía preguntas de usuario al back-end para su procesamiento.
- Un recurso de Azure AI Search que realiza búsquedas inteligentes sobre documentos indexados (ARCHIVOS PDF, archivos de Word, etc.) y devuelve extractos de documentos relevantes (fragmentos) para su uso en las respuestas.
- Una instancia del servicio OpenAI de Azure que:
 - Convierte documentos y preguntas de usuario en representaciones vectoriales para la búsqueda de similitud semántica.
 - Extrae palabras clave importantes para refinar las consultas de Azure AI Search.
 - Sintetiza las respuestas finales mediante los datos recuperados y la consulta de usuario.

El flujo típico de la aplicación de chat es el siguiente:

- **El usuario envía una pregunta:** un usuario escribe una pregunta en lenguaje natural a través de la interfaz de la aplicación web.
- **Azure OpenAI procesa la pregunta:** El back-end usa Azure OpenAI para:
 - Genere una inserción de la pregunta mediante el modelo text-embedding-ada-002.
 - Opcionalmente, extraiga palabras clave para refinar la relevancia de la búsqueda.
- **Azure AI Search recupera los datos pertinentes:** la inserción o las palabras clave se usan para realizar una búsqueda semántica sobre contenido indexado (como archivos PDF) en Azure AI Search.

- **Combinar resultados con la pregunta:** Los extractos de documentos (fragmentos) más relevantes se combinan con la pregunta original del usuario.
- **Azure OpenAI genera una respuesta:** la entrada combinada se pasa a un modelo GPT (por ejemplo, gpt-35-turbo o gpt-4), que genera una respuesta compatible con el contexto.
- **La respuesta se devuelve al usuario:** la respuesta generada se muestra en la interfaz de chat.

Requisitos previos

Está disponible un entorno contenedor de desarrollo con todas las dependencias necesarias para completar este artículo. Puede ejecutar el contenedor de desarrollo en GitHub Codespaces (en un explorador) o localmente mediante Visual Studio Code.

Para usar este artículo, necesita los siguientes requisitos previos:

Codespaces de GitHub (recomendado)

- Una suscripción de Azure. [Cree una cuenta gratuita](#) antes de comenzar.
- Permisos de cuenta de Azure. La cuenta de Azure debe tener permisos de Microsoft.Authorization/roleAssignments/write. Los roles como [administrador de acceso de usuario](#) o [propietario](#) cumplen este requisito.
- Acceso concedido a Azure OpenAI en la suscripción de Azure. En la mayoría de los casos, puede crear filtros de contenido personalizados y administrar niveles de gravedad con acceso general a los modelos de Azure OpenAI. El registro de acceso basado en aprobación no es necesario para el acceso general. Para más información, consulte [Características de acceso limitado para los servicios de Azure AI](#).
- Filtros de contenido o modificaciones relacionadas con el abuso (opcional). Para crear filtros de contenido personalizados, cambiar los niveles de gravedad o admitir la supervisión del abuso, necesita aprobación de acceso formal. Puede solicitar acceso completando los formularios de registro necesarios. Para obtener más información, consulte [Registro de filtros de contenido modificados o supervisión de abusos](#).
- Soporte técnico y solución de problemas de acceso. Para obtener acceso a la solución de problemas, abra un problema de soporte técnico en el repositorio de GitHub.

- Una cuenta de GitHub. Necesario para bifurcar el repositorio y usar GitHub Codespaces o clonarlo localmente.

Costo de uso de los recursos de ejemplo

La mayoría de los recursos usados en esta arquitectura se encuentran en los planes de tarifa básicos o basados en el consumo. Esto significa que solo paga por lo que usa y los cargos suelen ser mínimos durante el desarrollo o las pruebas.

Para completar este ejemplo, puede haber un pequeño costo en el que se incurre en el uso de servicios como Azure OpenAI, AI Search y almacenamiento. Una vez que haya terminado de evaluar o implementar la aplicación, puede eliminar todos los recursos aprovisionados para evitar cargos en curso.

Para obtener un desglose detallado de los costos esperados, consulte [estimación](#) de costos en el repositorio de GitHub del ejemplo.

Entorno de desarrollo abierto

Comience configurando un entorno de desarrollo que tenga instaladas todas las dependencias para completar este artículo.

Codespaces de GitHub (recomendado)

- Una suscripción de Azure. [Crear uno gratis](#).
- Permisos de cuenta de Azure. La cuenta de Azure debe tener permisos de Microsoft.Authorization/roleAssignments/write. Los roles como [administrador de acceso de usuario](#) o [propietario](#) cumplen este requisito.
- Una cuenta de GitHub. Necesario para bifurcar el repositorio y usar GitHub Codespaces o clonarlo localmente.

Apertura de un entorno de desarrollo

Use las instrucciones siguientes para implementar un entorno de desarrollo preconfigurado que contenga todas las dependencias necesarias para completar este artículo.

Codespaces de GitHub (recomendado)

Para la configuración más sencilla y simplificada, use [GitHub Codespaces](#). GitHub Codespaces ejecuta un contenedor de desarrollo administrado por GitHub y proporciona [Visual Studio Code para la Web](#) como interfaz de usuario (UI). Este entorno incluye todas las herramientas, SDK, extensiones y dependencias necesarias preinstaladas, por lo que puede empezar a desarrollar inmediatamente sin configuración manual.

El uso de Codespaces garantiza:

- Ya están instaladas las versiones y las herramientas de desarrollo correctas.
- No es necesario instalar Docker, VS Code ni extensiones localmente.
- Configuración rápida del entorno de incorporación y reproducible.

Importante

Todas las cuentas de GitHub pueden usar GitHub Codespaces durante hasta 60 horas gratuitas cada mes con 2 instancias principales. Si supera la cuota gratuita o usa opciones de proceso más grandes, se aplican las tarifas de facturación estándar de GitHub Codespaces. Para obtener más información, consulte [GitHub Codespaces: horas de almacenamiento y núcleo incluidas mensualmente](#).

1. Para empezar a trabajar con el proyecto de ejemplo, cree un nuevo espacio de código de GitHub en la `main` rama del [Azure-Samples/azure-search-openai-demo](#) repositorio de GitHub.

Haga clic con el botón derecho en la opción **GitHub Codespaces - Open (Abrir)** en la parte superior de la página del repositorio y seleccione **Abrir vínculo en la nueva ventana**. Esto garantiza que el contenedor de desarrollo se inicie en una pestaña de explorador dedicado de pantalla completa, lo que le proporciona acceso tanto al código fuente como a la documentación integrada.



2. En la página **Crear un nuevo espacio** de código, revise las opciones de configuración de codespace y, a continuación, seleccione **Crear espacio de código**:

Create a new codespace

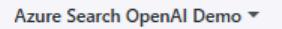
Repository
To be cloned into your codespace

 Codespace usage for this repository is paid for by developer-name

Branch
This branch will be checked out on creation

 main ▾

Dev container configuration
Your codespace will use this configuration

 Azure Search OpenAI Demo ▾

Region
Your codespace will run in the selected region

 US West ▾

Machine type
Resources for your codespace

 2-core ▾

Create codespace

Espere a que se inicie el espacio de código de GitHub. El proceso de inicio puede tardar unos minutos.

3. Una vez que se abra el espacio de código de GitHub, inicie sesión en Azure con la CLI para desarrolladores de Azure; para ello, escriba el siguiente comando en el panel Terminal del espacio de código:

Bash

```
azd auth login
```

GitHub muestra un código de seguridad en el panel Terminal.

- Copie el código de seguridad en el panel Terminal y seleccione Entrar. Se abre una ventana del explorador.
- En el aviso, pegue el código de seguridad en el campo del navegador.
- Siga las instrucciones para autenticarse con su cuenta Azure.

En este artículo se completan las tareas restantes de GitHub Codespaces en el contexto de este contenedor de desarrollo.

Implementación de la aplicación de chat en Azure

El repositorio de ejemplo incluye todo lo que necesita para implementar un chat con su propia aplicación de datos en Azure, lo que incluye:

- Código fuente de la aplicación (Python)
- Archivos de infraestructura-como-código (Bicep)
- Configuración para la integración de GitHub y CI/CD (opcional)

Utiliza los siguientes pasos para implementar la aplicación con la CLI para desarrolladores de Azure (azd).

Importante

Los recursos de Azure creados en esta sección(especialmente Azure AI Search) pueden comenzar a acumular cargos inmediatamente después del aprovisionamiento, incluso si la implementación se interrumpe antes de la finalización. Para evitar cargos inesperados, supervise el uso de Azure y elimine los recursos no utilizados inmediatamente después de las pruebas.

1. En el panel Terminal de Visual Studio Code, cree los recursos de Azure e implemente el código fuente ejecutando el comando siguiente azd :

```
Bash
azd up
```

2. El proceso le pide una o varias de las opciones siguientes en función de la configuración:

- **Nombre del entorno:** este valor se usa como parte del nombre del grupo de recursos. Introduzca un nombre corto con letras minúsculas y guiones (-), como myenv. No se admiten letras mayúsculas, números ni caracteres especiales.
- **Suscripción:** seleccione una suscripción para crear los recursos. Si no ve la suscripción deseada, use las teclas de dirección para desplazarse por la lista completa de suscripciones disponibles.
- **Ubicación:** esta ubicación de región se usa para la mayoría de los recursos, incluido el hospedaje. Seleccione una ubicación de región cercana geográficamente.
- **Ubicación para el modelo openAI o el recurso de Inteligencia de documentos:** seleccione la ubicación más cercana geográficamente. Si la región seleccionada para la **ubicación** está disponible para esta configuración, seleccione la misma región.

La aplicación puede tardar algún tiempo en implementarse. Antes de continuar, espere a que se complete la implementación.

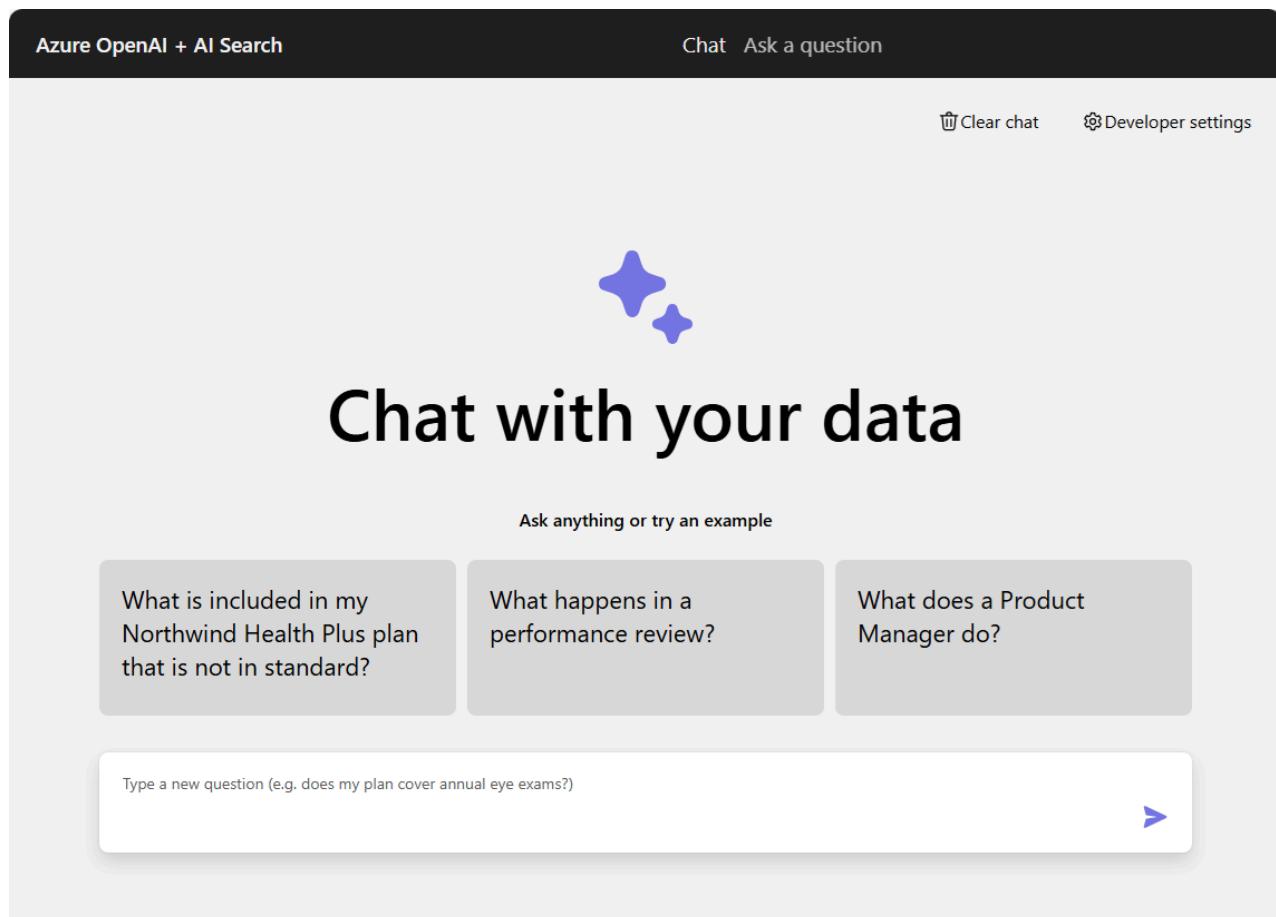
3. Una vez que la aplicación se implemente correctamente, el panel Terminal muestra una dirección URL del punto de conexión:

```
Deploying services (azd deploy)

(✓) Done: Deploying service backend
- Endpoint: https://app-backend-72xomfpzf3j4o.azurewebsites.net/

SUCCESS: Your Azure app has been deployed!
```

4. Seleccione la dirección URL del punto de conexión para abrir la aplicación de chat en un explorador:



Utilice la aplicación de chat para obtener respuestas de archivos PDF

La aplicación de chat está precargada con información sobre los beneficios de los empleados a partir de [archivos PDF](#). Puede utilizar la aplicación de chat para hacer preguntas sobre los

beneficios. Los siguientes pasos le guiarán a través del proceso de uso de la aplicación de chat. Las respuestas pueden variar a medida que se actualizan los modelos subyacentes.

1. En la aplicación de chat, seleccione la opción **¿Qué ocurre en una revisión de rendimiento?** o escriba el mismo texto en el cuadro de texto de chat. La aplicación devuelve la respuesta inicial:

The screenshot shows the Azure OpenAI + AI Search interface. At the top, it says "Azure OpenAI + AI Search" and "Chat Ask a question". Below that are "Clear chat" and "Developer settings" buttons. A question box contains the text "What happens in a performance review?". The response is displayed in a large red-bordered box:

During a performance review at Contoso Electronics, your supervisor will discuss your performance over the past year and provide feedback on areas for improvement. They will also provide you with an opportunity to discuss your goals and objectives for the upcoming year. Performance reviews are a two-way dialogue between managers and employees, and employees are encouraged to be honest and open during the review process ¹. The feedback provided during the performance review should be used as an opportunity to help employees develop and grow in their roles. Employees will receive a written summary of their performance review, which will include a rating of their performance, feedback, and goals and objectives for the upcoming year ¹.

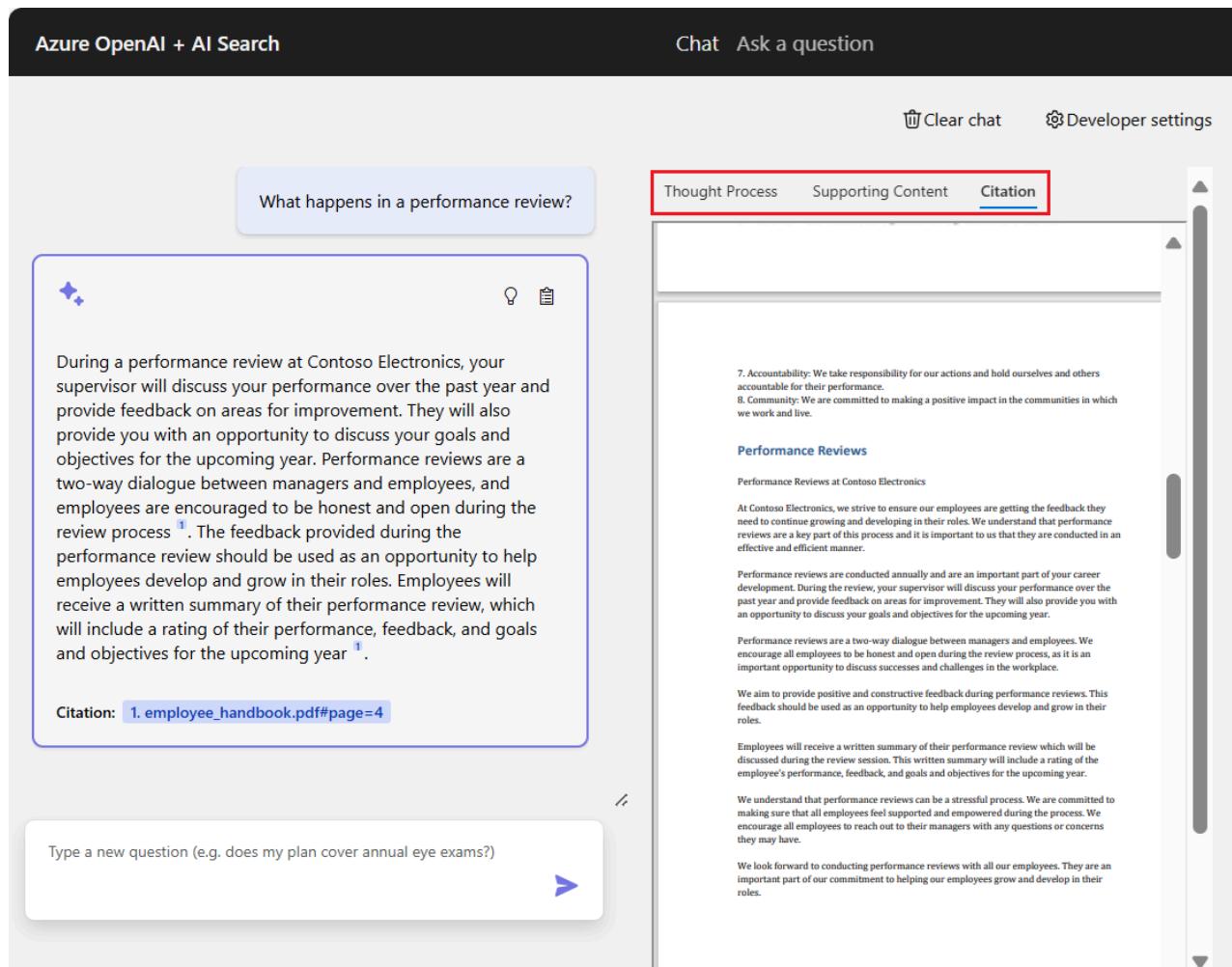
Citation: [1. employee_handbook.pdf#page=4](#)

Below the response box is a text input field with placeholder text "Type a new question (e.g. does my plan cover annual eye exams?)". To the right of the input field is a blue arrow button.

2. En el cuadro de respuesta, seleccione una cita:

The screenshot shows the Azure OpenAI + AI Search interface again. The top elements are identical: "Azure OpenAI + AI Search", "Chat Ask a question", "Clear chat", and "Developer settings". The question "What happens in a performance review?" is in the input box. The response box contains the same text as the previous screenshot, but the citation link "1. employee_handbook.pdf#page=4" is highlighted with a red border. The text input field and blue arrow button are also present at the bottom.

3. GitHub Codespaces abre el panel **Cita** derecho con tres regiones con pestañas y el foco está en la pestaña **Cita** :



GitHub Codespaces proporciona tres pestañas de información para ayudarle a comprender cómo la aplicación de chat generó la respuesta:

Pestaña	Descripción
Proceso de pensamiento	Muestra un script de las interacciones de preguntas y respuestas en el chat. Puede ver el contenido proporcionado por la aplicación de chat <code>system</code> , las preguntas introducidas por el <code>user</code> y las aclaraciones realizadas por el sistema <code>assistant</code> .
Contenido auxiliar	Enumera la información utilizada para responder a su pregunta y el material de origen. La configuración del desarrollador especifica el número de citas de material de origen. El número predeterminado de citas es 3.
Referencia bibliográfica	Muestra el origen original que contiene la cita seleccionada.

4. Cuando haya terminado, seleccione la pestaña seleccionada actualmente en el panel derecho. Se cierra el panel derecho.

Uso de la configuración para cambiar el comportamiento de la respuesta

El modelo openAI específico determina la inteligencia del chat y la configuración que se usa para interactuar con el modelo. La opción **Configuración del desarrollador** abre el panel **Configurar generación de respuestas**, donde puede cambiar la configuración de la aplicación de chat:

The screenshot shows the Azure OpenAI + AI Search interface. On the left, there's a card with text about performance reviews and a citation link. A red dashed arrow points from the "Developer settings" button in the top right of the main window to the "Configure answer generation" panel on the right. The panel contains various configuration options like prompt template, temperature, seed, search score, and retrieval mode.

Azure OpenAI + AI Search

Developer settings

During a performance review at Contoso Electronics, your supervisor will discuss your performance and provide feedback on areas for improvement. They will also provide you with an opportunity to set objectives for the upcoming year. Performance reviews are a two-way dialogue between employees and managers. It's important that both parties are encouraged to be honest and open during the review process¹. The performance review should be used as an opportunity to help employees develop a plan for the future. You should receive a written summary of their performance review, which will include a rating of your performance and objectives for the upcoming year¹.

Citation: [1. employee_handbook.pdf#page=4](#)

Type a new question (e.g. does my plan cover annual eye exams?)

Configure answer generation

Override prompt template [\(i\)](#)

Temperature [\(i\)](#)
0.3

Seed [\(i\)](#)

Minimum search score [\(i\)](#)
0

Minimum reranker score [\(i\)](#)
0

Retrieve this many search results: [\(i\)](#)
3

Include category [\(i\)](#)
All

Exclude category [\(i\)](#)

Use semantic ranker for retrieval [\(i\)](#)

Use semantic captions [\(i\)](#)

Suggest follow-up questions [\(i\)](#)

Retrieval mode [\(i\)](#)
Vectors + Text (Hybrid)

Stream chat completion responses [\(i\)](#)

Close

Configuración	Descripción
Anular plantilla de solicitud	Invalida la solicitud usada para generar la respuesta en función de la pregunta y los resultados de búsqueda.
Temperatura	Establece la temperatura de la solicitud en el modelo de lenguaje grande (LLM) que genera la respuesta. Las temperaturas más altas dan lugar a respuestas más creativas, pero podrían ser menos fundamentadas.
Valor de inicialización	Establece una semilla para mejorar la reproducibilidad de las respuestas del modelo. La inicialización puede ser cualquier entero.
puntuación de búsqueda mínima	Establece una puntuación mínima para los resultados de búsqueda devueltos desde Azure AI Search. El intervalo de puntuación depende de si se usa Híbrido (valor predeterminado) , Solo vectores o Solo texto para la configuración del modo de recuperación .
Puntuación mínima del reclasificador	Establece una puntuación mínima para los resultados de búsqueda devueltos por el reevaluador semántico. La puntuación siempre oscila entre 0 y 4. Cuanto mayor sea la puntuación, más relevante semánticamente será el resultado para la pregunta.
Recuperar esta cantidad de resultados	Establece el número de resultados de búsqueda que se van a recuperar de Azure AI Search. Más resultados pueden aumentar la probabilidad de encontrar la respuesta correcta, pero puede provocar que el modelo se pierda en el medio. Puede ver los orígenes devueltos en las pestañas Proceso de pensamiento y Contenido auxiliar del panel Cita .
Incluir categoría	Especifica las categorías que se van a incluir al generar los resultados de la búsqueda. Use la lista desplegable para realizar la selección. La acción predeterminada es incluir todas las categorías .
Excluir categoría	Especifica las categorías que se van a excluir de los resultados de la búsqueda. No se usan categorías en el conjunto de datos predeterminado.
Usar clasificador semántico para la recuperación	Habilita el clasificador semántico de Azure AI Search, un modelo que vuelve a generar resultados de búsqueda en función de la similitud semántica con la consulta del usuario.
Usar subtítulos semánticos	Envía subtítulos semánticos al LLM en lugar del resultado de búsqueda completo. Un subtítulo semántico se extrae de un resultado de búsqueda durante el proceso de clasificación semántica.
Sugerir preguntas de seguimiento	Solicita al LLM que sugiera preguntas de seguimiento basadas en la consulta del usuario.
modo de recuperación	Define el modo de recuperación de la consulta de Búsqueda de Azure AI. La acción predeterminada es Vectors + Texto (híbrido) , que usa una combinación de búsqueda vectorial y búsqueda de texto completo. La opción Vectors solo usa la búsqueda de vectores. La opción Texto solo usa la búsqueda de texto completo. El enfoque híbrido es óptimo.

Configuración	Descripción
Transmitir respuestas de finalización de chat	Transmite continuamente la respuesta a la interfaz de usuario de chat a medida que se genera el contenido.

Los siguientes pasos le guiarán a través del proceso de cambio de la configuración.

1. En el explorador, seleccione la opción **Configuración del desarrollador**.
2. Active la casilla **Sugerir preguntas de seguimiento** para habilitar la opción y seleccione **Cerrar** para aplicar el cambio de configuración.
3. En la aplicación de chat, vuelva a la pregunta, esta vez escribiendo el texto en el cuadro de pregunta:



La respuesta de la aplicación de chat ahora incluye preguntas de seguimiento sugeridas:

A screenshot of a chat application interface. At the top right, there is a message bubble containing the question 'What happens in a performance review?'. Below the message area, there is a detailed response about performance reviews at Contoso Electronics, mentioning supervisor feedback, goal setting, and honest communication. At the bottom of the response, there is a citation link: 'Citation: 1. employee_handbook.pdf#page=4'. Below the citation, there is a red-bordered box containing three follow-up questions: 'Follow-up questions: How long does a performance review typically last?', 'What should I do to prepare for a performance review?', and 'Are there any consequences if an employee receives a poor performance review?'. The entire interface is clean and modern, typical of a Microsoft Teams or similar messaging platform.

4. Vuelva a seleccionar la opción **Configuración del desarrollador** y anule la selección de **Usar clasificador semántico para la recuperación**. Cierre la configuración.
5. Vuelva a formular la misma pregunta y observe la diferencia en la respuesta de la aplicación de chat.

Con el clasificador semántico: "Durante una revisión de rendimiento en Contoso Electronics, el supervisor analizará el rendimiento durante el último año y proporcionará

comentarios sobre las áreas para mejorar. También tendrá la oportunidad de discutir sus objetivos y objetivos para el próximo año. La revisión es un diálogo bidireccional entre los gerentes y los empleados, y se anima a que los empleados sean honestos y abiertos durante el proceso (1). Los comentarios proporcionados durante la revisión deben ser positivos y constructivos, destinados a ayudar a los empleados a desarrollar y crecer en sus roles. Los empleados recibirán un resumen escrito de su revisión de rendimiento, que incluirá una clasificación de su rendimiento, comentarios y objetivos para el próximo año (1).

Sin el clasificador semántico: "Durante una revisión de rendimiento en Contoso Electronics, el supervisor analizará el rendimiento durante el último año y proporcionará comentarios sobre las áreas para mejorar. Es un diálogo bidireccional en el que se recomienda ser honesto y abierto (1). Los comentarios proporcionados durante la revisión deben ser positivos y constructivos, destinados a ayudarle a desarrollar y crecer en su rol. Recibirá un resumen escrito de la revisión, incluida una clasificación de su rendimiento, comentarios y objetivos para el próximo año (1)."

Limpieza de recursos

Después de completar el ejercicio, se recomienda quitar los recursos que ya no sean necesarios.

Limpieza de los recursos de Azure

Los recursos Azure creados en este artículo se facturan a su suscripción Azure. Si no espera necesitar estos recursos en el futuro, elimínelos para evitar incurrir en más gastos.

Elimine los recursos de Azure y quite el código fuente ejecutando el siguiente `azd` comando:

Bash

```
azd down --purge --force
```

Los modificadores de comandos incluyen:

- `purge`: Los recursos eliminados se purgan inmediatamente. Esta opción permite reutilizar el valor de la métrica de tokens por minuto (TPM) de Azure OpenAI.
- `force`: la eliminación se produce de forma silenciosa, sin necesidad de consentimiento del usuario.

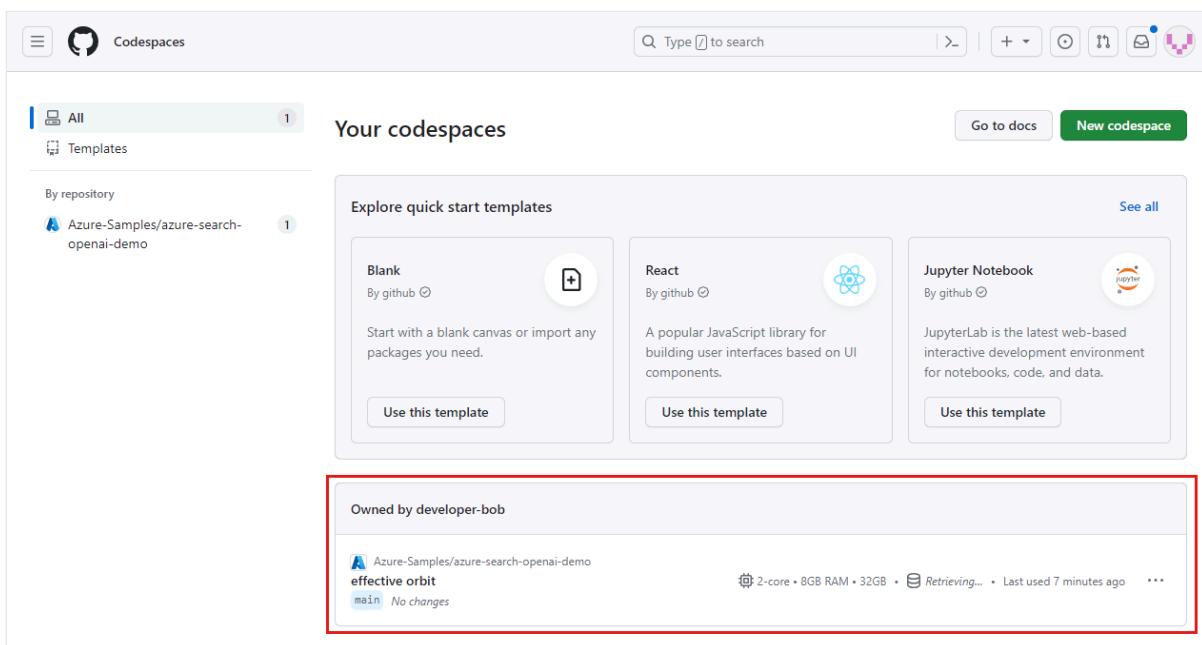
Limpieza de GitHub Codespaces

La eliminación del entorno de GitHub Codespaces garantiza que pueda maximizar la cantidad de derechos de horas gratuitas por núcleo que obtiene para su cuenta.

Importante

Para obtener más información sobre los derechos de la cuenta de GitHub, consulte [GitHub Codespaces: almacenamiento incluido mensual y horas principales](#).

1. Inicie sesión en el [panel de GitHub Codespaces](#).
2. En el panel de control, busque los espacios de código que están en ejecución actualmente procedentes del repositorio de GitHub [Azure-Samples/azure-search-openai-demo](#).



3. Abra el menú contextual del espacio de código y seleccione Eliminar:

The screenshot shows the GitHub Codespaces interface. At the top, there's a search bar and several navigation icons. Below that, a sidebar on the left lists 'All' (1), 'Templates', and a repository named 'Azure-Samples/azure-search-openai-demo'. The main area is titled 'Your codespaces' and features a section for 'Explore quick start templates' with options for 'Blank', 'React', and 'Jupyter Notebook'. Each template has a 'Use this template' button. Below this, there's a section for codespaces owned by 'developer-bob', showing one entry for 'Azure-Samples/azure-search-openai-demo' which is currently 'Retrieving...' and was last used 7 minutes ago. A context menu is open over this entry, with the 'Delete' option highlighted by a red box. The menu also includes options like 'Open in ...', 'Rename', 'Export changes to a fork', 'Change machine type', and 'Keep codespace'. At the bottom of the page, there's a footer with links to GitHub's Terms, Privacy, Security, Status, Docs, Contact GitHub, Pricing, API, Training, Blog, and About.

Obtener ayuda

Este repositorio de muestras ofrece [información para la resolución de problemas](#).

Si su problema no se resuelve, agréguelo a la página web de [problemas](#) del repositorio.

Contenido relacionado

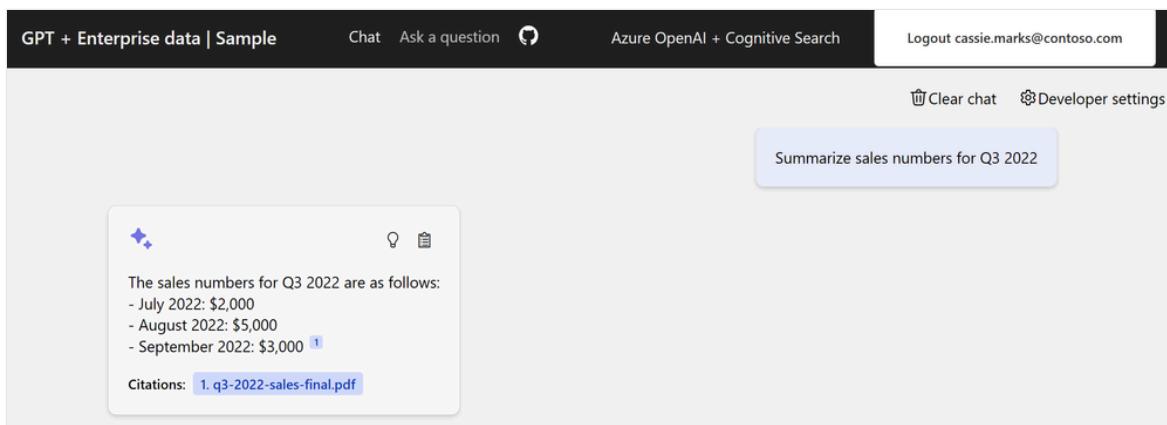
- [Obtener el código fuente para el ejemplo usado en este artículo](#)
- [Crear una aplicación de chat con la arquitectura de soluciones de mejores procedimientos de Azure OpenAI](#)
- [Control de acceso en aplicaciones de IA generativa con la búsqueda de Azure AI](#)
- [Cree una solución OpenAI preparada para la empresa con Azure API Management](#)
- [Mejorar el vector de búsqueda gracias a las capacidades híbridas de recuperación y clasificación](#)

Introducción a la seguridad de documentos de chat para Python

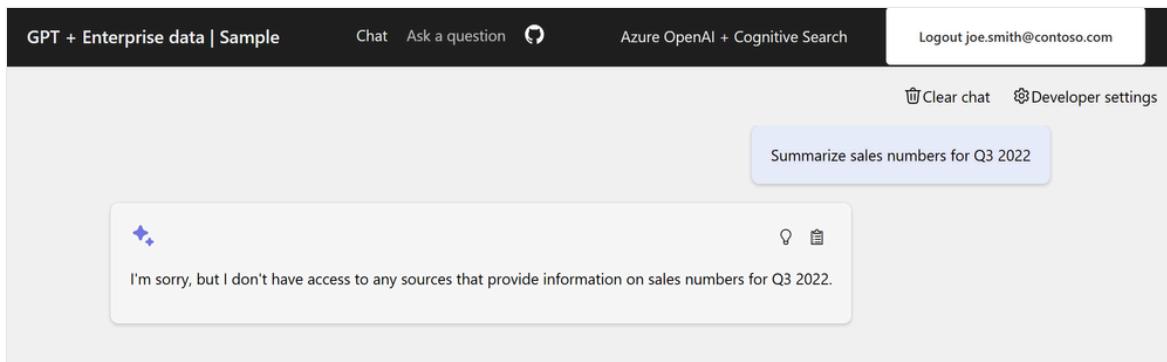
Artículo • 31/12/2024

Al compilar una aplicación de chat de [mediante el patrón de generación aumentada de recuperación \(RAG\)](#) con sus propios datos, asegúrese de que cada usuario recibe una respuesta basada en sus permisos. Siga el proceso de este artículo para agregar el control de acceso de documentos a la aplicación de chat.

- **usuario autorizado:** esta persona debe tener acceso a las respuestas contenidas en los documentos de la aplicación de chat.



- **usuario no autorizado:** esta persona no debe tener acceso a respuestas de documentos protegidos que no tienen autorización para ver.

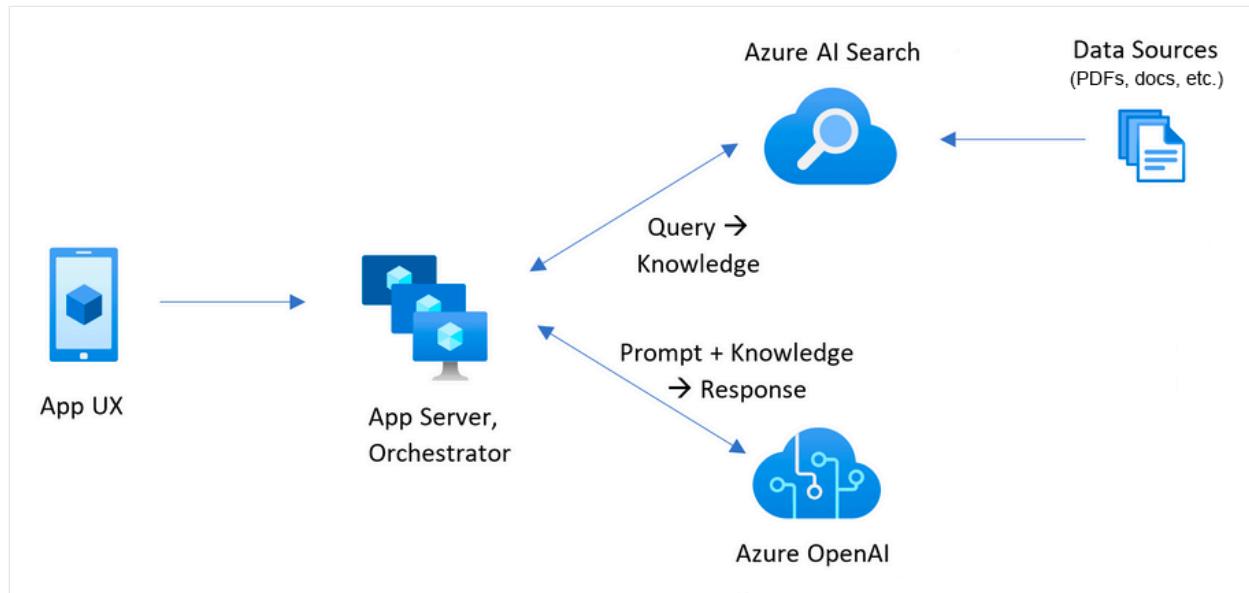


⚠ Nota

En este artículo se utilizan una o varias plantillas de aplicación de IA de como base para los ejemplos y la orientación del artículo. Las plantillas de aplicación de IA proporcionan implementaciones de referencia bien mantenidas que son fáciles de implementar. Ayudan a garantizar un punto de partida de alta calidad para las aplicaciones de inteligencia artificial.

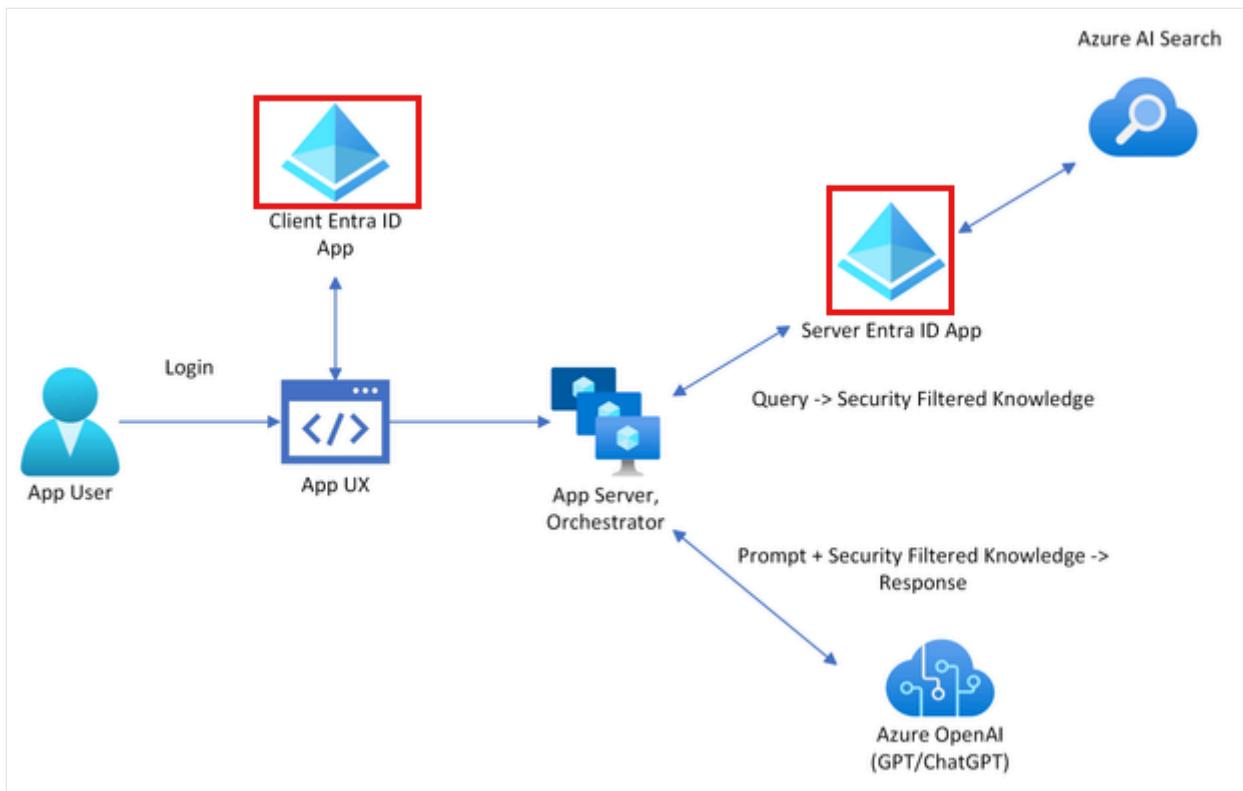
Introducción a la arquitectura

Sin una característica de seguridad de documentos, la aplicación de chat empresarial tiene una arquitectura sencilla mediante Azure AI Search y Azure OpenAI. Una respuesta se determina a partir de consultas a Azure AI Search donde se almacenan los documentos, en combinación con una respuesta de un modelo GPT de Azure OpenAI. No se usa ninguna autenticación de usuario en este flujo sencillo.

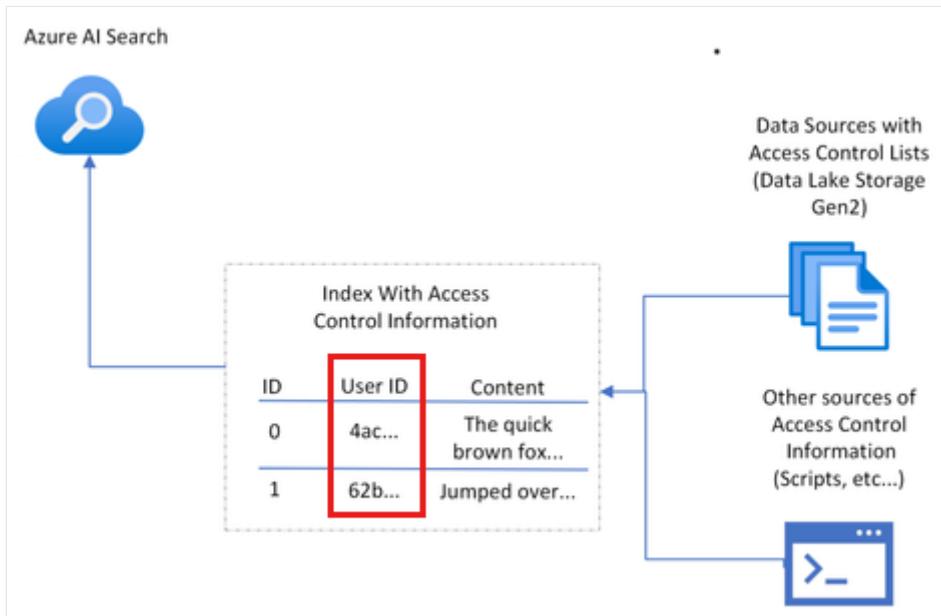


Para agregar seguridad a los documentos, debe actualizar la aplicación de chat empresarial:

- Agregue la autenticación de cliente a la aplicación de chat con Microsoft Entra.
- Agregue lógica del lado servidor para llenar un índice de búsqueda con acceso de usuario y grupo.



Búsqueda de Azure AI no proporciona permisos *nativos* a nivel de documento y no puede variar los resultados de búsqueda desde dentro de un índice por permisos de usuario. En su lugar, la aplicación puede usar filtros de búsqueda para asegurarse de que un documento sea accesible para un usuario específico o un grupo específico. Dentro del índice de búsqueda, cada documento debe tener un campo filtrable que almacene información de identidad de usuario o grupo.



Dado que la autorización no está contenida de forma nativa en Azure AI Search, debe agregar un campo para contener información de usuario o grupo y, a continuación, **filtrar** los documentos que no coincidan. Para implementar esta técnica, debe:

- Cree un campo de control de acceso a documentos en el índice dedicado a almacenar los detalles de los usuarios o grupos con acceso a documentos.
- Rellene el campo de control de acceso del documento con los detalles de usuario o grupo pertinentes.
- Actualice este campo de control de acceso siempre que haya cambios en los permisos de acceso de usuario o grupo.

Si las actualizaciones de índice se programan con un indexador, los cambios se aplican en la siguiente ejecución del indexador. Si no usa un indexador, debe volver a indexar manualmente.

En este artículo, el proceso de protección de documentos en Azure AI Search se hace posible con los scripts de ejemplo y, que usted, en su calidad de administrador de búsqueda, ejecutaría. Los scripts asocian un único documento a una sola identidad de usuario. Puede tomar estos [scripts](#) y aplicar sus propios requisitos de seguridad y producción para escalar según sus necesidades.

Determinación de la configuración de seguridad

La solución proporciona variables de entorno booleanas para activar las características necesarias para la seguridad del documento en este ejemplo.

 Expandir tabla

Parámetro	Propósito
AZURE_USE_AUTHENTICATION	Cuando se establece en <code>true</code> , habilita el inicio de sesión de usuario en la aplicación de chat y la autenticación de Azure App Service. Habilita <code>Use oid security filter</code> en la aplicación de chat Configuración del desarrollador .
AZURE_ENFORCE_ACCESS_CONTROL	Cuando se establece en <code>true</code> , requiere autenticación para cualquier acceso a documentos. La configuración de Developer para el identificador de objeto (OID) y la seguridad de grupo están activadas y deshabilitadas para que no se puedan deshabilitar desde la interfaz de usuario.
AZURE_ENABLE_GLOBAL_DOCUMENTS_ACCESS	Cuando se establece en <code>true</code> , esta opción permite a los usuarios autenticados buscar en documentos que no tienen asignados controles de acceso, incluso cuando se requiere control de acceso. Este parámetro solo se debe

Parámetro	Propósito
	usar cuando <code>AZURE_ENFORCE_ACCESS_CONTROL</code> está habilitado.
<code>AZURE_ENABLE_UNAUTHENTICATED_ACCESS</code>	Cuando se establece en <code>true</code> , esta configuración permite a los usuarios no autenticados usar la aplicación, incluso cuando se aplica el control de acceso. Este parámetro solo se debe usar cuando <code>AZURE_ENFORCE_ACCESS_CONTROL</code> está habilitado.

Use las secciones siguientes para comprender los perfiles de seguridad admitidos en este ejemplo. Este artículo configura el *perfil Enterprise*.

Empresa: cuenta requerida + filtro de documento

Cada usuario del sitio *debe* iniciar sesión. El sitio contiene contenido público para todos los usuarios. El filtro de seguridad de nivel de documento se aplica a todas las solicitudes.

Variables de entorno:

- `AZURE_USE_AUTHENTICATION=true`
- `AZURE_ENABLE_GLOBAL_DOCUMENTS_ACCESS=true`
- `AZURE_ENFORCE_ACCESS_CONTROL=true`

Uso mixto: cuenta opcional + filtro de documento

Cada usuario del sitio *puede* iniciar sesión. El sitio contiene contenido público para todos los usuarios. El filtro de seguridad de nivel de documento se aplica a todas las solicitudes.

Variables de entorno:

- `AZURE_USE_AUTHENTICATION=true`
- `AZURE_ENABLE_GLOBAL_DOCUMENTS_ACCESS=true`
- `AZURE_ENFORCE_ACCESS_CONTROL=true`
- `AZURE_ENABLE_UNAUTHENTICATED_ACCESS=true`

Prerrequisitos

Hay disponible un entorno contenedor de desarrollo con todas las dependencias necesarias para completar este artículo. Puede ejecutar el contenedor de desarrollo en

GitHub Codespaces (en un explorador) o localmente mediante Visual Studio Code.

Para usar este artículo, necesita los siguientes requisitos previos:

- Una suscripción de Azure. [Crear uno gratis](#).
- Permisos de cuenta de Azure: la cuenta de Azure debe tener:
 - Permiso para [administrar aplicaciones en Microsoft Entra ID](#).
 - Permisos `Microsoft.Authorization/roleAssignments/write`, como [Administrador de acceso de usuarios](#) o [Propietario](#).

Necesita más requisitos previos en función de su entorno de desarrollo preferido.

GitHub Codespaces (recomendado)

- [GitHub](#)

Apertura de un entorno de desarrollo

Use las instrucciones siguientes para implementar un entorno de desarrollo preconfigurado que contenga todas las dependencias necesarias para completar este artículo.

GitHub Codespaces (recomendado)

[GitHub Codespaces](#) ejecuta un contenedor de desarrollo administrado por GitHub con [Visual Studio Code para web](#) como interfaz de usuario. Para el entorno de desarrollo más sencillo, use GitHub Codespaces para que tenga las herramientas de desarrollo y las dependencias correctas preinstaladas para completar este artículo.

ⓘ Importante

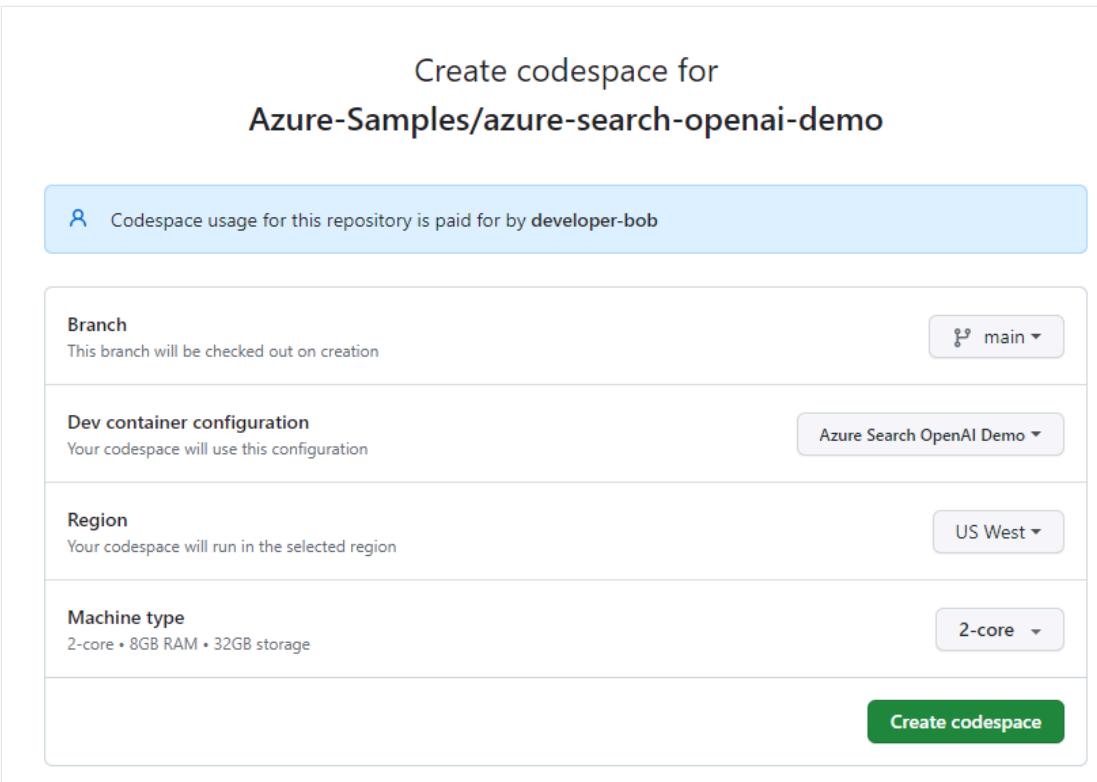
Todas las cuentas de GitHub pueden usar GitHub Codespaces durante hasta 60 horas gratuitas cada mes con dos instancias principales. Para obtener más información, consulte [Almacenamiento y horas de núcleo incluidas mensualmente en GitHub Codespaces](#).

1. Inicie el proceso para crear un nuevo espacio de código de GitHub en la rama `main` del repositorio [Azure-Samples/azure-search-openai-demo](#) GitHub.

2. Haga clic con el botón derecho en el siguiente botón y seleccione Abrir el enlace en nuevas ventanas para tener el entorno de desarrollo y la documentación disponibles al mismo tiempo.



3. En la página **Crear codespace**, revise las opciones de configuración de codespace y, después, seleccione **Crear nuevo codespace**.



4. Espere a que se inicie Codespace. Este proceso de inicio puede tardar unos minutos.
5. En el terminal de la parte inferior de la pantalla, inicie sesión en Azure con la CLI para desarrolladores de Azure.

```
Bash
azd auth login
```

6. Complete el proceso de autenticación.
7. Las tareas restantes de este artículo tienen lugar en el contexto de este contenedor de desarrollo.

Obtención de información necesaria con la CLI de Azure

Obtenga el identificador de suscripción y el identificador de inquilino con el siguiente comando de la CLI de Azure. Copie el valor para usarlo como tu valor `AZURE_TENANT_ID`.

Azure CLI

```
az account list --query "[].{subscription_id:id, name:name, tenantId:tenantId}" -o table
```

Si recibe un error sobre la política de acceso condicional de su tenant, necesita un segundo tenant sin política de acceso condicional.

- Su primer tenant, asociado a su cuenta de usuario, se utiliza para la variable de entorno `AZURE_TENANT_ID`.
- Su segundo tenant, sin acceso condicional, se utiliza para la variable de entorno `AZURE_AUTH_TENANT_ID` para acceder a Microsoft Graph. Para arrendatarios con una directiva de acceso condicional, busque el ID de un segundo arrendatario sin una directiva de acceso condicional o [cree un nuevo arrendatario](#).

Establecimiento de variables de entorno

1. Ejecute los siguientes comandos para configurar la aplicación para el perfil de Enterprise.

Consola

```
azd env set AZURE_USE_AUTHENTICATION true  
azd env set AZURE_ENABLE_GLOBAL_DOCUMENTS_ACCESS true  
azd env set AZURE_ENFORCE_ACCESS_CONTROL true
```

2. Ejecute el siguiente comando para establecer el cliente, lo que autoriza el inicio de sesión del usuario en el entorno de aplicación hospedado. Reemplace `<YOUR_TENANT_ID>` por el identificador de inquilino.

Consola

```
azd env set AZURE_TENANT_ID <YOUR_TENANT_ID>
```

 Nota

Si tiene una política de acceso condicional en su tenant de usuario, necesita [especificar un tenant de autenticación](#).

Implementación de la aplicación de chat en Azure

La implementación consta de los pasos siguientes:

- Cree los recursos de Azure.
- Cargue los documentos.
- Cree las aplicaciones de identidad de Microsoft Entra (cliente y servidor).
- Active la identidad del recurso de hospedaje.

1. Ejecute el siguiente comando de la CLI para desarrolladores de Azure para aprovisionar los recursos de Azure e implementar el código fuente.

Bash

```
azd up
```

2. Utilice la siguiente tabla para responder a las solicitudes de implementación `AZD`.

[] Expandir tabla

Pronto	Respuesta
Nombre del entorno	Use un nombre corto con información de identificación como el alias y la aplicación. Y el ejemplo es <code>tjones-secure-chat</code> .
Suscripción	Seleccione una suscripción en la que se van a crear los recursos.
Ubicación de los recursos de Azure	Seleccione una ubicación cerca de usted.
Ubicación de <code>documentIntelligentResourceGroupLocation</code>	Seleccione una ubicación cerca de usted.
Ubicación de <code>openAIResourceGroupLocation</code>	Seleccione una ubicación cerca de usted.

Espere 5 o 10 minutos después de que la aplicación se implemente para permitir que la aplicación se inicie.

3. Una vez que la aplicación se implemente correctamente, aparece una dirección URL en el terminal.
4. Seleccione la dirección URL etiquetada `(✓) Done: Deploying service webapp` para abrir la aplicación de chat en un explorador.

```
Deploying services (azd deploy)

(✓) Done: Deploying service backend
- Endpoint: https://app-backend-72xomfpzf3j4o.azurewebsites.net/

SUCCESS: Your Azure app has been deployed!
```

5. Por favor, acepte la ventana emergente de autenticación de la aplicación.
6. Cuando aparezca la aplicación de chat, observe en la esquina superior derecha que el usuario ha iniciado sesión.
7. Abra **ajustes del desarrollador** y observe que ambas de las siguientes opciones están seleccionadas e inhabilitadas para modificación:
 - Usar filtro de seguridad oid
 - Uso del filtro de seguridad de grupos
8. Seleccione la tarjeta con **¿Qué hace un administrador de productos?**.
9. Recibe una respuesta como: **Los orígenes proporcionados no contienen información específica sobre el rol de administrador de productos en Contoso Electronics.**

The screenshot shows a user interface for managing AI developer settings. At the top, there's a navigation bar with links for "GPT + Enterprise data | Sample", "Chat", "Ask a question", "Azure OpenAI + AI Search", and "Logout morgan@contoso.com". Below the navigation bar, there are buttons for "Clear chat" and "Developer settings". A search bar contains the query "What does a Product Manager do?". The main area displays a response from an AI model: "The provided sources do not contain specific information about the role of a Product Manager at Contoso Electronics." There are icons for a profile picture, a question mark, and a document. At the bottom, there's a text input field with placeholder text "Type a new question (e.g. does my plan cover annual eye exams?)" and a large blue "Ask" button with a right-pointing arrow.

Abrir el acceso a un documento para un usuario

Active sus permisos para el documento exacto para que *pueda* obtener la respuesta. Necesitas varias piezas de información:

- Azure Storage
 - Nombre de cuenta
 - Nombre del contenedor
 - Blob/documento URL para `role_library.pdf`
- Identificador del usuario en Microsoft Entra ID

Cuando se conozca esta información, actualice el campo `oids` del índice de búsqueda de Azure AI en el documento `role_library.pdf`.

Obtención de la dirección URL de un documento en el almacenamiento

1. En la carpeta `.azure` en la raíz del proyecto, busque el directorio del entorno y abra el archivo `.env` con ese directorio.
2. Busque la entrada `AZURE_STORAGE_ACCOUNT` y copie su valor.
3. Use los siguientes comandos de la CLI de Azure para obtener la dirección URL del blob `role_library.pdf` en el contenedor de `content`.

Azure CLI

```
az storage blob url \
--account-name <REPLACE_WITH_AZURE_STORAGE_ACCOUNT> \
--container-name 'content' \
--name 'role_library.pdf'
```

 Expandir tabla

Parámetro	Propósito
--nombre-de-cuenta	Nombre de la cuenta de Azure Storage.
--container-name	El nombre del contenedor de este ejemplo es <code>content</code> .
--nombre	El nombre del blob de este paso es <code>role_library.pdf</code> .

4. Copie la dirección URL del blob para usarla más adelante.

Obtención del identificador de usuario

1. En la aplicación de chat, seleccione **Configuración del desarrollador**.
2. En la sección **Notificaciones de token de ID**, copie su parámetro `objectidentifier`. Este parámetro se conoce en la sección siguiente como `USER_OBJECT_ID`.

Proporcionar acceso de usuario a un documento en Azure Search

1. Use el siguiente script para cambiar el campo `oids` de Azure AI Search para `role_library.pdf` para que tenga acceso a él.

Bash

```
./scripts/manageacl.sh \
-v \
--acl-type oids \
--acl-action add \
--acl <REPLACE_WITH_YOUR_USER_OBJECT_ID> \
--url <REPLACE_WITH_YOUR_DOCUMENT_URL>
```

 Expandir tabla

Parámetro	Propósito
<code>-v</code>	Salida detallada.
<code>--acl-type</code>	OID de grupo o usuario: <code>oids</code> .
<code>--acl-action</code>	Agregar a un campo índice de búsqueda. Otras opciones incluyen <code>remove</code> , <code>remove_all</code> y <code>list</code> .
<code>--acl</code>	Usuario o grupo <code>USER_OBJECT_ID</code> .
<code>--url</code>	Ubicación del archivo en Azure Storage, como <code>https://MYSTORAGENAME.blob.core.windows.net/content/role_library.pdf</code> . No rodear la dirección URL con comillas en el comando de la CLI.

2. La salida de la consola de este comando es similar a la siguiente:

```
console.
```

```
Loading azd .env file from current environment...
Creating Python virtual environment "app/backend/.venv"...
Installing dependencies from "requirements.txt" into virtual
environment (in quiet mode)...
Running manageacl.py. Arguments to script: -v --acl-type oids --acl-
action add --acl 00000000-0000-0000-0000-000000000000 --url
https://mystorage.blob.core.windows.net/content/role_library.pdf
Found 58 search documents with storageUrl
https://mystorage.blob.core.windows.net/content/role_library.pdf
Adding acl 00000000-0000-0000-0000-000000000000 to 58 search documents
```

3. Opcionalmente, use el siguiente comando para comprobar que el permiso aparece para el archivo en Azure AI Search.

Bash

```
./scripts/manageacl.sh \
-v \
--acl-type oids \
--acl-action list \
--acl <REPLACE_WITH_YOUR_USER_OBJECT_ID> \
--url <REPLACE_WITH_YOUR_DOCUMENT_URL>
```

 Expandir tabla

Parámetro	Propósito
-v	Salida detallada.
--acl-type	OID de grupo o usuario: <code>oids</code> .
--acl-action	Enumerar un campo de índice de búsqueda <code>oids</code> . Otras opciones incluyen <code>remove</code> , <code>remove_all</code> y <code>list</code> .
--acl	Parámetro <code>USER_OBJECT_ID</code> de grupo o usuario.
--url	La ubicación del archivo en que se muestra, como <code>https://MYSTORAGENAME.blob.core.windows.net/content/role_library.pdf</code> . No rodear la dirección URL con comillas en el comando de la CLI.

4. La salida de la consola de este comando es similar a la siguiente:

console.

```
Loading azd .env file from current environment...
Creating Python virtual environment "app/backend/.venv"...
Installing dependencies from "requirements.txt" into virtual
environment (in quiet mode)...
```

```
Running manageacl.py. Arguments to script: -v --acl-type oids --acl-action view --acl 00000000-0000-0000-0000-000000000000 --url https://mystorage.blob.core.windows.net/content/role_library.pdf Found 58 search documents with storageUrl https://mystorage.blob.core.windows.net/content/role_library.pdf [00000000-0000-0000-0000-000000000000]
```

La matriz al final de la salida incluye el parámetro `USER_OBJECT_ID` y se usa para determinar si el documento se usa en la respuesta con Azure OpenAI.

Compruebe que Azure AI Search contiene su `USER_OBJECT_ID`

1. Abra el [Azure Portal](#) y busque `AI Search`.
2. Seleccione el recurso de búsqueda en la lista.
3. Seleccione **Administración de búsqueda > Índices**.
4. Seleccione **gptkbindex**.
5. Seleccione **Ver > vista JSON**.
6. Reemplace el código JSON por el siguiente json:

```
JSON

{
  "search": "*",
  "select": "sourcefile, oids",
  "filter": "oids/any()"
}
```

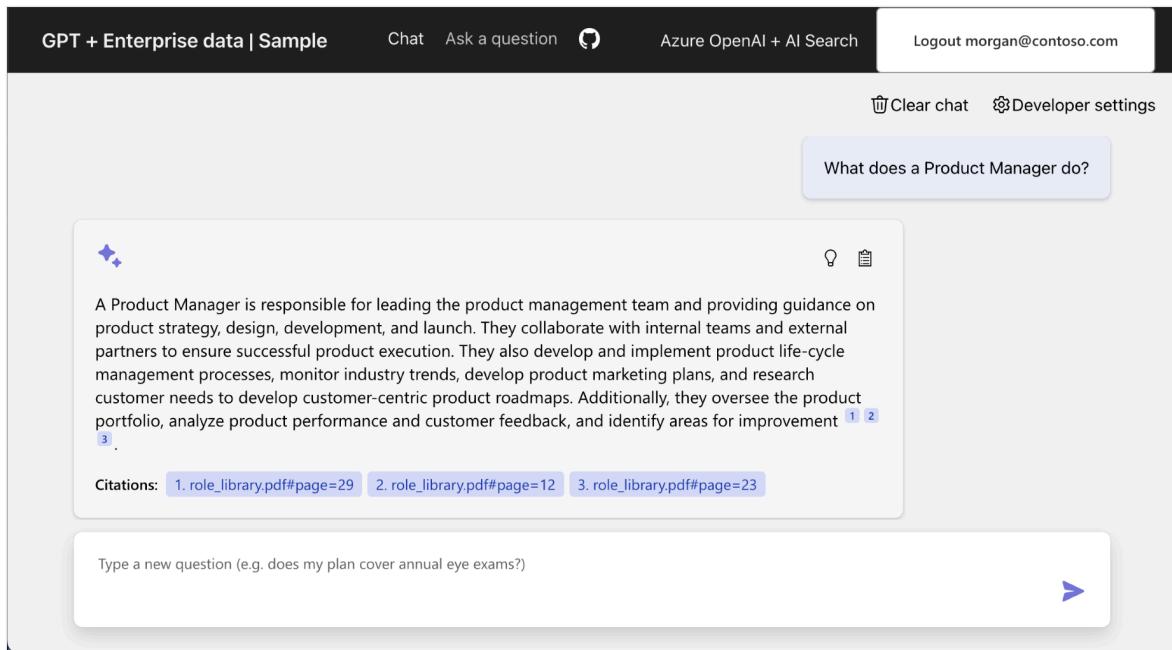
Este JSON busca en todos los documentos donde el campo `oids` tiene cualquier valor y devuelve los campos `sourcefile` y `oids`.

7. Si el `role_library.pdf` no tiene tu OID, vuelve a la sección [Proporcionar acceso de usuario a un documento en Azure Search](#) y completa los pasos.

Comprobación del acceso de usuario al documento

Si completó los pasos pero no vio la respuesta correcta, compruebe que el parámetro `USER_OBJECT_ID` se ha establecido correctamente en Azure AI Search para `role_library.pdf`.

1. Vuelva a la aplicación de chat. Es posible que tenga que volver a iniciar sesión.
2. Escriba la misma consulta para que el contenido de `role_library` se use en la respuesta de Azure OpenAI: `What does a product manager do?`.
3. Visualice el resultado, que ahora incluye la respuesta adecuada del documento de la biblioteca de roles.



Limpieza de recursos

Los pasos siguientes le guiarán por el proceso de limpieza de los recursos que usó.

Limpieza de recursos de Azure

Los recursos de Azure creados en este artículo se facturan a su suscripción de Azure. Si no espera necesitar estos recursos en el futuro, elimínelos para evitar incurrir en más cargos.

Ejecute el siguiente comando de la CLI para desarrolladores de Azure para eliminar los recursos de Azure y quitar el código fuente.

```
Bash
azd down --purge
```

Limpieza de GitHub Codespaces y Visual Studio Code

Los pasos siguientes le guiarán por el proceso de limpieza de los recursos que usó.

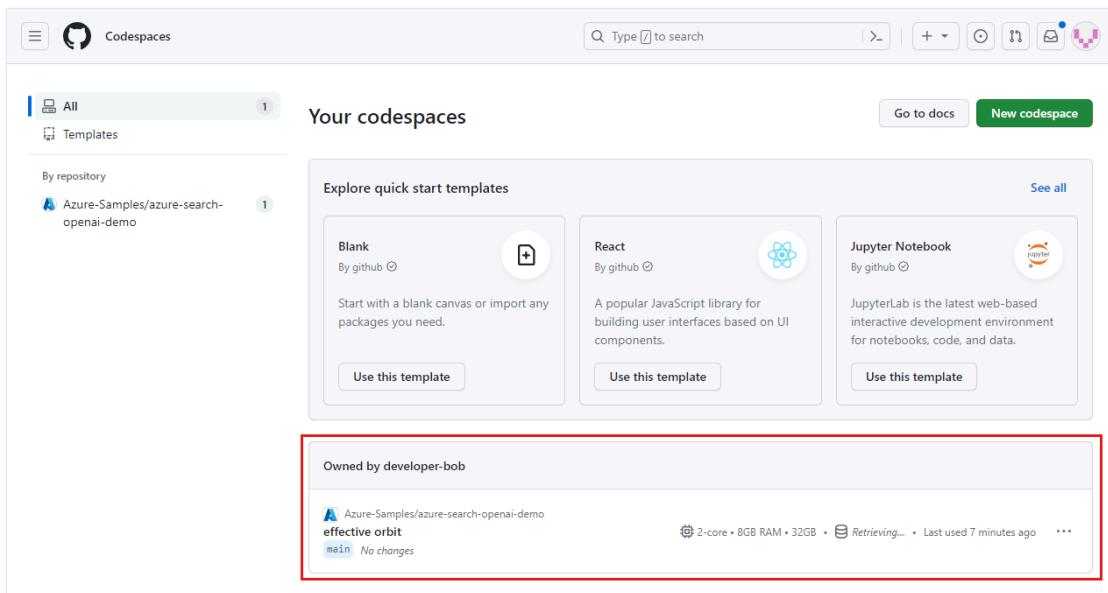
GitHub Codespaces

La eliminación del entorno de GitHub Codespaces garantiza que pueda maximizar la cantidad de derechos de horas gratuitas por núcleo que obtiene para su cuenta.

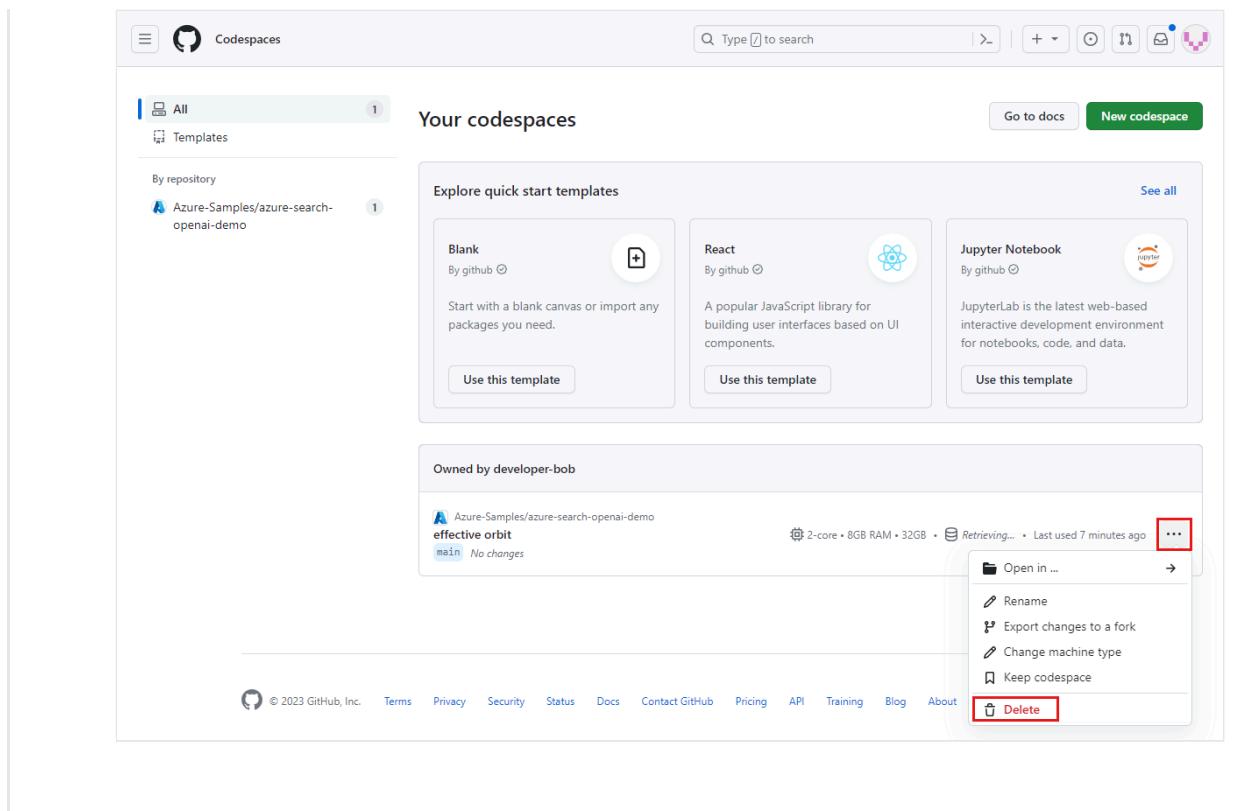
ⓘ Importante

Para obtener más información sobre los derechos de la cuenta de GitHub, consulte [Almacenamiento y horas de núcleo incluidas mensualmente en GitHub Codespaces](#).

1. Inicie sesión en el [panel de GitHub Codespaces](#).
2. Busque los codespaces que se ejecutan actualmente a partir del repositorio de GitHub [Azure-Samples/azure-search-openai-demo](#).



3. Abra el menú contextual del espacio de código y seleccione **Eliminar**.



Obtener ayuda

Este repositorio de muestras ofrece [información para la resolución de problemas](#).

Solución de problemas

En esta sección se ofrece solución de problemas específicos de este artículo.

Proporcione el tenant de autenticación

Cuando su autenticación se encuentra en un tenant separado de la aplicación de hospedaje, debe configurar ese tenant de autenticación con el siguiente proceso.

1. Ejecute el siguiente comando para configurar la muestra para utilizar una segunda entidad como la entidad de autenticación.

```
Consola  
azd env set AZURE_AUTH_TENANT_ID <REPLACE-WITH-YOUR-TENANT-ID>
```

Expandir tabla

Parámetro	Propósito
AZURE_AUTH_TENANT_ID	Si se establece AZURE_AUTH_TENANT_ID, es el arrendatario encargado de hospedar la aplicación.

2. Vuelva a implementar la solución con el siguiente comando:

Consola

```
azd up
```

Contenido relacionado

- Cree una aplicación de chat [con Azure OpenAI utilizando la arquitectura de solución de mejores prácticas](#).
- Aprenda sobre el control de acceso [en aplicaciones de inteligencia artificial generativa con Azure AI Search](#).
- Cree una solución de Azure OpenAI preparada para la empresa [con Azure API Management](#).
- Consulte [Búsqueda de Azure AI: Superar el vector de búsqueda con capacidades híbridas de recuperación y clasificación](#).

Comentarios

¿Le ha resultado útil esta página?

 Sí

 No

[Proporcionar comentarios sobre el producto](#) | [Obtener ayuda en Microsoft Q&A](#)

Introducción a los puntos de conexión privados de chat para Python

Artículo • 26/02/2025

En este artículo se muestra cómo implementar y ejecutar el ejemplo de aplicación de chat empresarial de [para Python](#) accesible por puntos de conexión privados.

En este ejemplo se implementa una aplicación de chat mediante Python, Azure OpenAI Service y [Retrieval Augmented Generation \(RAG\)](#) en Azure AI Search para obtener respuestas sobre los beneficios para empleados en una empresa ficticia. La aplicación se inicializa con archivos PDF que incluyen el manual de empleados, un documento de beneficios y una lista de roles y expectativas de la empresa.

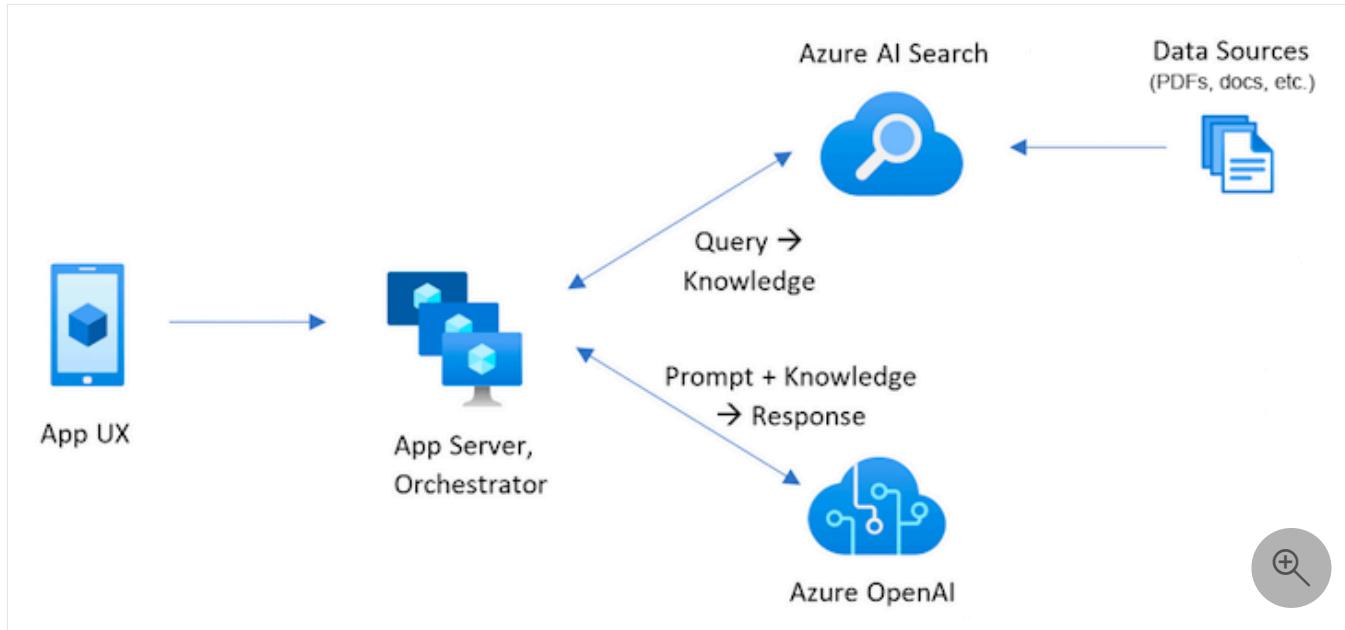
Siguiendo las instrucciones de este artículo, puede:

- Implemente una aplicación de chat en Azure para el acceso público en un explorador web.
- Vuelva a implementar una aplicación de chat con puntos de conexión privados.

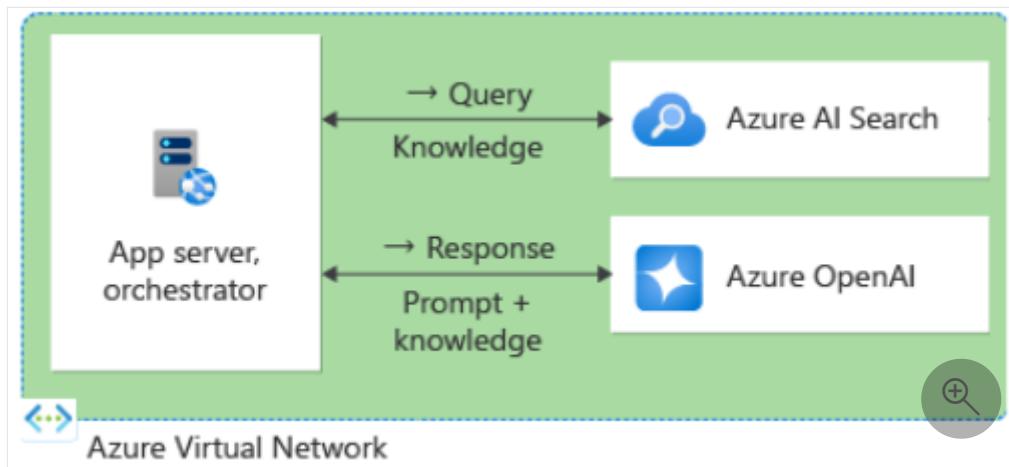
Después de finalizar este procedimiento, puede empezar a modificar el nuevo proyecto con el código personalizado y volver a implementar, sabiendo que la aplicación de chat solo es accesible a través de la red privada.

Introducción a la arquitectura

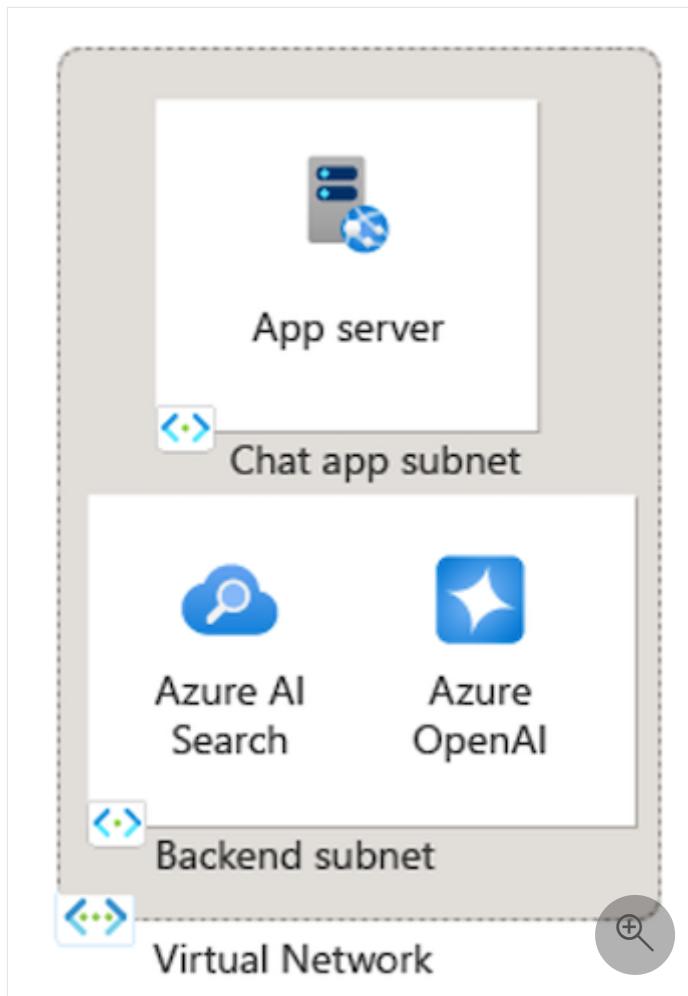
La implementación predeterminada crea una aplicación de chat con puntos de conexión públicos.



Para las aplicaciones de chat enriquecidas con datos privados, es fundamental proteger el acceso a la aplicación de chat. En este artículo se presenta una solución mediante una red virtual.

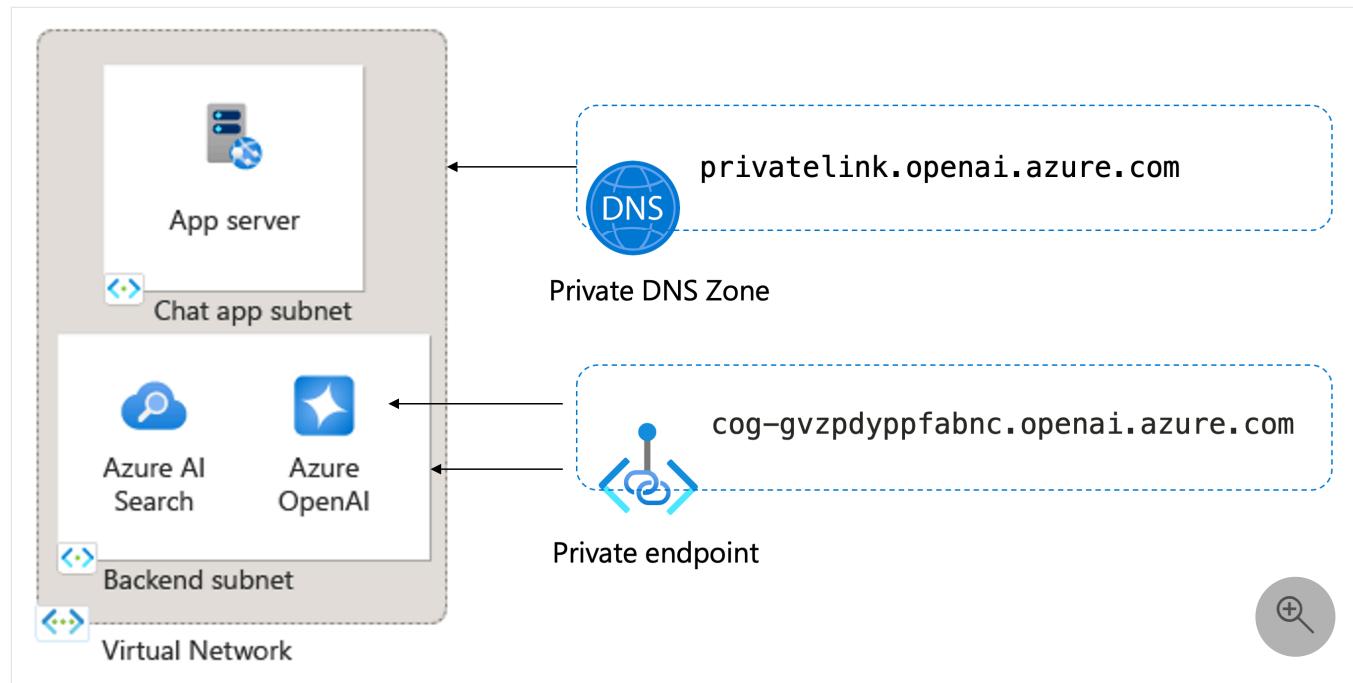


Dentro de la red virtual, hay una subred independiente para la aplicación de Azure App Service frente a los otros servicios de Azure back-end. Esta estructura facilita la aplicación de diferentes reglas de grupo de seguridad de red a cada subred.



Dentro de la red virtual, los servicios usan puntos de conexión privados para comunicarse entre sí. Cada punto de conexión privado está asociado a una zona privada del sistema de nombres

de dominio (DNS) para resolver el nombre del punto de conexión privado en una dirección IP dentro de la red virtual.



Pasos de implementación

Se recomienda implementar la solución dos veces. Implemente una vez con acceso público para validar que la aplicación de chat funciona correctamente. Vuelva a implementar con acceso privado para proteger la aplicación de chat mediante una red virtual.

Prerrequisitos

Hay disponible un entorno de [contenedor de desarrollo](#) con todas las dependencias necesarias para completar este artículo. Puede ejecutar el contenedor de desarrollo en GitHub Codespaces (en un explorador) o localmente mediante Visual Studio Code.

Para usar este artículo, necesita los siguientes requisitos previos.

Codespaces (recomendado)

- Una suscripción de Azure. [Crear uno gratis](#).
- Permisos de cuenta de Azure. Su cuenta de Azure debe tener los permisos `Microsoft.Authorization/roleAssignments/write`, como [Administrador de acceso de usuario](#) o [Propietario](#).
- Una cuenta de GitHub.

Entorno de desarrollo abierto

Comience ahora con un entorno de desarrollo que tenga instaladas todas las dependencias para completar este artículo.

GitHub Codespaces (recomendado)

[GitHub Codespaces](#) ejecuta un contenedor de desarrollo administrado por GitHub con [Visual Studio Code para web](#) como interfaz de usuario. Para el entorno de desarrollo más sencillo, use GitHub Codespaces para que tenga las herramientas de desarrollo y las dependencias correctas preinstaladas para completar este artículo.

Importante

Todas las cuentas de GitHub pueden usar GitHub Codespaces durante hasta 60 horas gratuitas cada mes con dos instancias principales. Para obtener más información, consulte [Almacenamiento y horas de núcleo incluidas mensualmente en GitHub Codespaces](#).

1. Inicie el proceso para crear un nuevo espacio de código de GitHub en la rama `main` del repositorio [Azure-Samples/azure-search-openai-demo](#) GitHub.
2. Haga clic con el botón derecho en el siguiente botón y seleccione **Abrir el enlace en una nueva ventana** para tener el entorno de desarrollo y la documentación disponibles al mismo tiempo.

 [Open in GitHub Codespaces](#) 

3. En la página **Crear codespace**, revise las opciones de configuración de codespace y, a continuación, seleccione **Crear codespace**.

Create codespace for Azure-Samples/azure-search-openai-demo

 Codespace usage for this repository is paid for by developer-bob

Branch This branch will be checked out on creation	 main ▾
Dev container configuration Your codespace will use this configuration	Azure Search OpenAI Demo ▾
Region Your codespace will run in the selected region	US West ▾
Machine type 2-core • 8GB RAM • 32GB storage	2-core ▾
Create codespace 	

4. Espere a que se inicie Codespace. Este proceso de inicio puede tardar unos minutos.
5. En el terminal de la parte inferior de la pantalla, inicie sesión en Azure con la CLI para desarrolladores de Azure:

```
Bash
azd auth login
```

6. Copie el código del terminal y péguelo en un explorador. Siga las instrucciones para autenticarse con su cuenta de Azure.

Las tareas restantes de este artículo tienen lugar en el contexto de este contenedor de desarrollo.

Configuración personalizada

Esta solución configura e implementa la infraestructura en función de las opciones personalizadas configuradas con la CLI para desarrolladores de Azure. En la tabla siguiente se explica la configuración personalizada de esta solución.

 Expandir tabla

Ajuste	Descripción
AZURE_PUBLIC_NETWORK_ACCESS	Controla el valor del acceso a la red pública en los recursos de Azure admitidos. Los valores válidos son <code>Enabled</code> o <code>Disabled</code> .
AZURE_USE_PRIVATE_ENDPOINT	Controla la implementación de puntos de conexión privados, que conectan recursos de Azure a la red virtual. El valor <code>TRUE</code> indica que los puntos de conexión privados se despliegan para garantizar la conectividad.

Implementación de la aplicación de chat

La primera implementación crea los recursos y proporciona un punto de conexión accesible públicamente.

- Ejecute el siguiente comando para configurar esta solución para el acceso público:

```
Consola
azd env set AZURE_PUBLIC_NETWORK_ACCESS Enabled
```

Cuando se le pida un nombre de entorno, recuerde que el nombre del entorno se usa para crear el grupo de recursos. Escriba un nombre descriptivo. Si está en un equipo o en una organización, incluya el nombre, como en `morgan-chat-private-endpoints`. Anote el nombre del entorno. Lo necesitará más adelante para encontrar los recursos en Azure Portal.

- Ejecute el siguiente comando para incluir el aprovisionamiento de los recursos de red virtual. Recuerde que la implementación no restringe el acceso hasta la segunda implementación.

```
Consola
azd env set AZURE_USE_PRIVATE_ENDPOINT true
```

- Implemente la solución con el siguiente comando:

```
Consola
azd up
```

Los recursos de aprovisionamiento son la parte más lenta del proceso de implementación. Espere a que finalice la implementación antes de continuar.

- Al final del proceso de implementación, aparece el punto de conexión de la aplicación. Copie ese punto de conexión en un explorador para abrir la aplicación de chat. Seleccione una de las preguntas de las tarjetas y espere la respuesta.
- Anote la dirección URL del punto de conexión porque la necesita más adelante en el artículo.

Implementación de la aplicación de chat en Azure con acceso privado

Cambie la configuración de implementación para proteger la aplicación de chat para el acceso privado.

- Ejecute el comando siguiente para desactivar el acceso público:

```
Consola  
azd env set AZURE_PUBLIC_NETWORK_ACCESS Disabled
```

- Ejecute el siguiente comando para cambiar la configuración del recurso. Este comando no vuelve a implementar el código de la aplicación porque ese código no ha cambiado.

```
Consola  
azd provision
```

- Una vez finalizado el aprovisionamiento, vuelva a abrir la aplicación de chat en un explorador. La aplicación de chat ya no es accesible porque el punto de conexión público está deshabilitado.

Acceso a la aplicación de chat

Para acceder a la aplicación de chat, use una herramienta como [azure VPN Gateway](#) o [Azure Virtual Desktop](#). Recuerde que cualquier herramienta que use para acceder a la aplicación debe ser segura y compatible con las directivas de seguridad de su organización.

Limpieza de recursos

Los pasos siguientes le guiarán por el proceso de limpieza de los recursos que usó.

La eliminación del entorno de GitHub Codespaces garantiza que pueda maximizar la cantidad de derechos de horas gratuitas por núcleo que obtiene para su cuenta.

ⓘ Importante

Para obtener más información sobre los derechos de la cuenta de GitHub, consulte [Almacenamiento y horas de núcleo incluidas mensualmente en GitHub Codespaces](#).

1. Inicie sesión en el [panel de GitHub Codespaces](#).
2. Busque los codespaces que se ejecutan actualmente a partir del repositorio de GitHub [GitHub Azure-Samples/azure-search-openai-demo](#).

The screenshot shows the GitHub Codespaces interface. At the top, there's a navigation bar with icons for search, filters, and account. Below it, a sidebar on the left shows 'All' (1) and 'Templates'. A main area titled 'Your codespaces' displays three quick start templates: 'Blank' (By github), 'React' (By github), and 'Jupyter Notebook' (By github). Each template has a 'Use this template' button. Below these, a specific codespace is listed: 'Owned by developer-bob' for repository 'Azure-Samples/azure-search-openai-demo' named 'effective orbit'. This entry includes a status bar indicating 'main No changes', a progress bar showing 'Retrieving...', and a timestamp 'Last used 7 minutes ago'. A red box highlights this specific codespace entry.

3. Abra el menú contextual del espacio de código y seleccione Eliminar.

The screenshot shows the GitHub Codespaces interface. At the top, there's a search bar and several navigation icons. Below that, a sidebar on the left lists 'All' (1) and 'Templates'. A section titled 'Your codespaces' shows a repository 'Azure-Samples/azure-search-openai-demo' owned by 'developer-bob'. The repository details include 'effective orbit', 'main', and 'No changes'. To the right, there are three quick start template cards: 'Blank', 'React', and 'Jupyter Notebook'. Below these, a section titled 'Owned by developer-bob' lists the repository again. A context menu is open over the repository card, with the 'Delete' option highlighted with a red box.

Obtener ayuda

Este repositorio de muestras ofrece [información para la resolución de problemas](#).

Si su problema no se resuelve, agréguelo a la página web de [problemas](#) del repositorio.

Contenido relacionado

- Consulte el repositorio de GitHub de la aplicación de chat empresarial [.](#)
- Cree una aplicación de chat [con la arquitectura de soluciones de mejores prácticas de Azure OpenAI](#).
- Obtenga información sobre el [control de acceso en aplicaciones de IA generativa con Búsqueda de Azure AI](#)

Introducción a la evaluación de respuestas en una aplicación de chat en Python

30/06/2025

Este artículo muestra cómo evaluar las respuestas de una aplicación de chat comparándolas con un conjunto de respuestas correctas o ideales (conocidas como verdad básica). Siempre que modifique su aplicación de chat de forma que afecte a las respuestas, realice una evaluación para comparar los cambios. Esta aplicación de demostración ofrece herramientas que puede usar hoy para facilitar la ejecución de evaluaciones.

Siguiendo las instrucciones de este artículo, puede:

- Use las solicitudes de ejemplo que se proporcionan adaptadas al ámbito temático. Estas solicitudes ya están en el repositorio.
- Genere preguntas de usuario de ejemplo y respuestas de verdad básica a partir de sus propios documentos.
- Ejecute evaluaciones mediante un mensaje de ejemplo con las preguntas del usuario generadas.
- Revise el análisis de las respuestas.

⚠ Nota

En este artículo se usan una o varias [plantillas de aplicaciones de IA](#) como base para los ejemplos e instrucciones del artículo. Las plantillas de aplicación de IA proporcionan implementaciones de referencia bien mantenidas que son fáciles de implementar. Ayudan a garantizar un punto de partida de alta calidad para las aplicaciones de inteligencia artificial.

Introducción a la arquitectura

Entre los componentes clave de la arquitectura se incluyen:

- **Aplicación de chat hospedada en Azure:** la aplicación de chat se ejecuta en Azure App Service.
- **Protocolo de chat de Microsoft AI:** el protocolo proporciona contratos de API estandarizados en soluciones y lenguajes de INTELIGENCIA ARTIFICIAL. La aplicación de chat se ajusta al protocolo [Microsoft AI Chat Protocol](#), que permite que la aplicación de evaluaciones se ejecute en cualquier aplicación de chat que se ajuste al protocolo.

- **Búsqueda de Azure AI:** la aplicación de chat usa Búsqueda de Azure AI para almacenar los datos de sus propios documentos.
- **Generador de preguntas de ejemplo:** la herramienta puede generar muchas preguntas para cada documento junto con la respuesta verdadera. Cuantos más preguntas haya, más largas son las evaluaciones.
- **Evaluador:** La herramienta ejecuta preguntas y solicitudes de ejemplo en la aplicación de chat y presenta los resultados.
- **Herramienta de revisión:** la herramienta revisa los resultados de las evaluaciones.
- **Herramienta de comparación:** Compara las respuestas entre evaluaciones.

Al implementar esta evaluación en Azure, el punto de conexión del servicio Azure OpenAI se crea para el `GPT-4` modelo con su propia [capacidad](#). Al evaluar las aplicaciones de chat, es importante que el evaluador tenga su propio recurso de Azure OpenAI utilizando `GPT-4` con su propia capacidad.

Requisitos previos

- Una suscripción de Azure. [cree una de forma gratuita](#).
- Complete el [procedimiento anterior de la aplicación de chat](#) para implementar la aplicación de chat en Azure. Este recurso es necesario para que funcione la aplicación de evaluaciones. No complete la sección "Limpieza de recursos" del procedimiento anterior.

Necesita la siguiente información de recursos de Azure de esa implementación, que se conoce como la *aplicación de chat* de este artículo:

- URI de api de chat. El punto de conexión del back-end del servicio que se muestra al final del proceso `azd up`.
- Azure AI Search. Se requieren los siguientes valores:
 - **Nombre del recurso:** el nombre del recurso de Azure AI Search, notificado como `Search service` durante el `azd up` proceso.
 - **Nombre del índice:** el nombre del índice de Azure AI Search donde se almacenan los documentos. Puede encontrarlo en Azure Portal para el servicio Search.

La URL de la API de chat permite que las evaluaciones realicen solicitudes a través de su aplicación de back-end. La información de Azure AI Search permite que los scripts de evaluación usen la misma implementación que su back-end, cargado con los documentos.

Después de que haya recopilado esta información, no debe tener que volver a usar el entorno de desarrollo de aplicaciones de chat. Este artículo hace referencia a él más adelante, varias veces, para indicar cómo la aplicación de evaluaciones usa la aplicación

de chat. No elimine los recursos de la aplicación de chat hasta que finalice todo el procedimiento de este artículo.

- Está disponible un entorno de contenedor de desarrollo [🔗](#) con todas las dependencias necesarias para completar este artículo. Puede ejecutar el contenedor de desarrollo en GitHub Codespaces (en un explorador) o localmente mediante Visual Studio Code.

Codespaces de GitHub (recomendado)

- GitHub

Apertura de un entorno de desarrollo

Siga estas instrucciones para configurar un entorno de desarrollo preconfigurado con todas las dependencias necesarias para completar este artículo. Organice el área de trabajo de supervisión para que pueda ver esta documentación y el entorno de desarrollo al mismo tiempo.

Este artículo se probó con la región `switzerlandnorth` para la implementación de evaluación.

Codespaces de GitHub (recomendado)

[GitHub Codespaces](#) [🔗](#) ejecuta un contenedor de desarrollo administrado por GitHub con [Visual Studio Code para la web](#) [🔗](#) como interfaz de usuario. Use GitHub Codespaces para el entorno de desarrollo más sencillo. Viene con las herramientas de desarrollo y las dependencias adecuadas preinstaladas para completar este artículo.

ⓘ Importante

Todas las cuentas de GitHub pueden usar GitHub Codespaces durante hasta 60 horas gratuitas cada mes con dos instancias principales. Para obtener más información, consulte [Almacenamiento y horas de núcleo incluidas mensualmente en GitHub Codespaces](#) [🔗](#).

1. Inicie el proceso para crear un nuevo codespace de GitHub en la rama `main` del repositorio [Azure-Samples/ai-rag-chat-evaluator](#) [🔗](#).
2. Para mostrar el entorno de desarrollo y la documentación disponible al mismo tiempo, haga clic con el botón derecho en el botón siguiente y seleccione **Abrir vínculo en la nueva ventana**.



[Open in GitHub Codespaces](#)

3. En la página **Crear codespace**, revise las opciones de configuración del codespace y seleccione **Crear nuevo codespace**.

Create codespace for
Azure-Samples/ai-rag-chat-evaluator

Branch
This branch will be checked out on creation **main**

Dev container configuration
Your codespace will use this configuration **AI RAG Chat Evaluator**

Region
Your codespace will run in the selected region **US West**

Machine type
Resources for your codespace **2-core**

Create codespace

4. Espere a que se inicie el espacio de código. Este proceso de startup puede tardar unos minutos.
5. En el terminal de la parte inferior de la pantalla, inicie sesión en Azure con la CLI para desarrolladores de Azure:

```
Bash
azd auth login --use-device-code
```

6. Copie el código del terminal y péguelo en un navegador. Siga las instrucciones para autenticarse con su cuenta Azure.
7. Aprovisione el recurso de Azure necesario, Azure OpenAI Service, para la aplicación de evaluaciones:

```
Bash
azd up
```

Este AZD comando no implementa la aplicación de evaluaciones, pero crea el recurso de Azure OpenAI con una implementación necesaria GPT-4 para ejecutar las evaluaciones en el entorno de desarrollo local.

Las tareas restantes de este artículo tienen lugar en el contexto de este contenedor de desarrollo.

El nombre del repositorio de GitHub aparece en la barra de búsqueda. Este indicador visual le ayuda a distinguir la aplicación de evaluaciones de la aplicación de chat. Este `ai-rag-chat-evaluator` repositorio se conoce como la *aplicación de evaluaciones* de este artículo.

Preparación de los valores de entorno y la información de configuración

Actualice los valores de entorno y la información de configuración con la información recopilada durante [los requisitos previos](#) de la aplicación de evaluaciones.

1. Cree un archivo `.env` basado en `.env.sample`.

```
Bash
```

```
cp .env.sample .env
```

2. Ejecute este comando para obtener los valores necesarios para

`AZURE_OPENAI_EVAL_DEPLOYMENT` y `AZURE_OPENAI_SERVICE` desde el grupo de recursos implementado. Pegue esos valores en el `.env` archivo.

```
shell
```

```
azd env get-value AZURE_OPENAI_EVAL_DEPLOYMENT  
azd env get-value AZURE_OPENAI_SERVICE
```

3. Agregue los siguientes valores desde la aplicación de chat para su instancia de Azure AI Search al `.env` archivo, que ha recopilado en la sección [Requisitos previos](#).

```
Bash
```

```
AZURE_SEARCH_SERVICE=<service-name>  
AZURE_SEARCH_INDEX=<index-name>
```

Uso del protocolo de chat de Microsoft AI para obtener información sobre la configuración

La aplicación de chat y la aplicación de evaluaciones implementan la especificación del Protocolo de Chat de Microsoft AI, un contrato de API de punto de conexión de IA que es de código abierto, independiente de la nube y no dependiente de ningún lenguaje, y que se utiliza para su consumo y evaluación. Cuando las interfaces de cliente y de nivel intermedio cumplen con esta especificación de API, puede consumir y ejecutar evaluaciones de manera coherente en sus sistemas de fondo de IA.

1. Cree un nuevo archivo llamado `my_config.json` y copie en él el siguiente contenido:

```
JSON

{
    "testdata_path": "my_input/qa.jsonl",
    "results_dir": "my_results/experiment<TIMESTAMP>",
    "target_url": "http://localhost:50505/chat",
    "target_parameters": {
        "overrides": {
            "top": 3,
            "temperature": 0.3,
            "retrieval_mode": "hybrid",
            "semantic_ranker": false,
            "prompt_template": "<READFILE>my_input/prompt_refined.txt",
            "seed": 1
        }
    }
}
```

El script de evaluación crea la carpeta `my_results`.

El `overrides` objeto contiene los valores de configuración necesarios para la aplicación. Cada aplicación define su propio conjunto de propiedades de configuración.

2. Use la tabla siguiente para comprender el significado de las propiedades de configuración que se envían a la aplicación de chat.

 Expandir tabla

Propiedad	Descripción
Configuraciones	
<code>semantic_ranker</code>	Si se debe usar el clasificador semántico , un modelo que vuelve a clasificar los resultados de búsqueda en función de la similitud semántica con la consulta del usuario. Lo deshabilitamos para este tutorial para reducir los costes.

Propiedad	Descripción
Configuraciones	
<code>retrieval_mode</code>	Modo de recuperación que se va a usar. El valor predeterminado es <code>hybrid</code> .
<code>temperature</code>	Configuración de temperatura del modelo. El valor predeterminado es <code>0.3</code> .
<code>top</code>	El número de resultados de búsqueda a devolver. El valor predeterminado es <code>3</code> .
<code>prompt_template</code>	Una invalidación de la solicitud usada para generar la respuesta en función de la pregunta y los resultados de búsqueda.
<code>seed</code>	Valor de inicialización de las llamadas a modelos GPT. Establecer un valor de inicialización permite obtener resultados más coherentes en las evaluaciones.

3. Cambie el valor de `target_url` al valor de la URI de su aplicación de chat, que has recopilado en la sección [Requisitos previos](#). La aplicación de chat debe cumplir el protocolo de chat. El URI tiene el siguiente formato: `https://CHAT-APP-URL/chat`. Asegúrese de que el protocolo y la ruta `chat` forman parte del URI.

Generación de datos de ejemplo

Para evaluar las nuevas respuestas, deben compararse con una respuesta *de verdad básica*, que es la respuesta ideal para una pregunta determinada. Genere preguntas y respuestas a partir de documentos almacenados en Azure AI Search para la aplicación de chat.

1. Copie la `example_input` carpeta en una nueva carpeta denominada `my_input`.
2. En un terminal, ejecute el siguiente comando para generar los datos de ejemplo:

Bash

```
python -m evaltools generate --output=my_input/qa.jsonl --persource=2 --numquestions=14
```

Los pares de preguntas y respuestas se generan y almacenan en (en `my_input/qa.jsonl` formato [JSONL](#)) como entrada para el evaluador que se usa en el paso siguiente. Para una evaluación de producción, se generarían más pares de preguntas y respuestas. Para este conjunto de datos se generan más de 200.

! Nota

Solo se generan algunas preguntas y respuestas por origen para que pueda completar rápidamente este procedimiento. No está pensado para ser una evaluación de producción, que debe tener más preguntas y respuestas por origen.

Ejecución de la primera evaluación con una solicitud refinada

1. Edite las propiedades del archivo de `my_config.json` configuración.

[+] Expandir tabla

Propiedad	Valor nuevo
<code>results_dir</code>	<code>my_results/experiment_refined</code>
<code>prompt_template</code>	<code><READFILE>my_input/prompt_refined.txt</code>

La solicitud refinada es específica del dominio del tema.

txt

If there isn't enough information below, say you don't know. Do not generate answers that don't use the sources below. If asking a clarifying question to the user would help, ask the question.

Use clear and concise language and write in a confident yet friendly tone. In your answers, ensure the employee understands how your response connects to the information in the sources and include all citations necessary to help the employee validate the answer provided.

For tabular information, return it as an html table. Do not return markdown format. If the question is not in English, answer in the language used in the question.

Each source has a name followed by a colon and the actual information. Always include the source name for each fact you use in the response. Use square brackets to reference the source, e.g. [info1.txt]. Don't combine sources, list each source separately, e.g. [info1.txt][info2.pdf].

2. En un terminal, ejecute el siguiente comando para ejecutar la evaluación:

Bash

```
python -m evaltools evaluate --config=my_config.json --numquestions=14
```

Este script creó una nueva carpeta de experimentos en `my_results/` con la evaluación. La carpeta contiene los resultados de la evaluación.

 Expandir tabla

Nombre del archivo	Descripción
<code>config.json</code>	Copia del archivo de configuración usado para la evaluación.
<code>evaluate_parameters.json</code>	Parámetros usados para la evaluación. Similar a <code>config.json</code> pero incluye otros metadatos, como la marca de tiempo.
<code>eval_results.jsonl</code>	Cada pregunta y respuesta, junto con las métricas GPT para cada par de pregunta y respuesta.
<code>summary.json</code>	Los resultados generales, como la media de las métricas de GPT.

Realice la segunda evaluación con una solicitud débil

1. Edite las propiedades del archivo de `my_config.json` configuración.

 Expandir tabla

Propiedad	Valor nuevo
<code>results_dir</code>	<code>my_results/experiment_weak</code>
<code>prompt_template</code>	<code><READFILE>my_input/prompt_weak.txt</code>

Esa instrucción débil no tiene contexto sobre el tema en cuestión.

txt
You are a helpful assistant.

2. En un terminal, ejecute el siguiente comando para ejecutar la evaluación:

Bash
<pre>python -m evaltools evaluate --config=my_config.json --numquestions=14</pre>

Ejecución de la tercera evaluación con una temperatura específica

Use una solicitud que permita más creatividad.

1. Edite las propiedades del archivo de `my_config.json` configuración.

 Expandir tabla

Existing	Propiedad	Valor nuevo
Existing	<code>results_dir</code>	<code>my_results/experiment_ignoreresources_temp09</code>
Existing	<code>prompt_template</code>	<code><READFILE>my_input/prompt_ignoreresources.txt</code>
Nuevo	<code>temperature</code>	<code>0.9</code>

El valor predeterminado de `temperature` es 0.7. Cuanto mayor sea la temperatura, más creativas serán las respuestas.

La solicitud `ignore` es corta.

text

Your job is to answer questions to the best of your ability. You will be given sources but you should IGNORE them. Be creative!

2. El objeto de configuración debería verse como el siguiente ejemplo, excepto que `results_dir` se ha reemplazado por su ruta de acceso.

JSON

```
{  
    "testdata_path": "my_input/qa.jsonl",  
    "results_dir": "my_results/prompt_ignoreresources_temp09",  
    "target_url": "https://YOUR-CHAT-APP/chat",  
    "target_parameters": {  
        "overrides": {  
            "temperature": 0.9,  
            "semantic_ranker": false,  
            "prompt_template": "<READFILE>my_input/prompt_ignoreresources.txt"  
        }  
    }  
}
```

3. En un terminal, ejecute el siguiente comando para ejecutar la evaluación:

Bash

```
python -m evaltools evaluate --config=my_config.json --numquestions=14
```

Revisión de los resultados de la evaluación

Ha realizado tres evaluaciones basadas en diferentes solicitudes y configuraciones de la aplicación. Los resultados se almacenan en la carpeta `my_results`. Revise cómo difieren los resultados en función de los ajustes.

1. Use la herramienta de revisión para ver los resultados de las evaluaciones.

Bash

```
python -m evaltools summary my_results
```

2. Los resultados son *algo* así:

folder	groundedness	%	relevance	%	coherence	%	citation %	length
experiment_ignoresources_temp09	5.00	1.00	4.71	0.93	4.86	0.93	0.00	1063.14
experiment_refined	5.00	1.00	5.00	1.00	5.00	1.00	1.00	1404.79
experiment_weak	5.00	1.00	5.00	1.00	5.00	1.00	0.00	1331.57

Cada valor se devuelve como un número y un porcentaje.

3. Use la siguiente tabla para comprender el significado de los valores.

[] Expandir tabla

Valor	Descripción
Base	Comprueba qué tan bien se basan las respuestas del modelo en información fáctica y verificable. Una respuesta se considera fundamentada si es objetivamente exacta y refleja la realidad.
Relevancia	Mide la estrecha alineación de las respuestas del modelo con el contexto o el mensaje. Una respuesta pertinente aborda directamente la consulta o afirmación del usuario.
Coherencia	Comprueba la coherencia lógica de las respuestas del modelo. Una respuesta coherente mantiene un flujo lógico y no se contradice.
Referencia bibliográfica	Indica si la respuesta se devolvió en el formato solicitado en la indicación.
Largura	Mide la longitud de la respuesta.

4. Los resultados deberían indicar que las tres evaluaciones tenían alta relevancia, mientras que `experiment_ignoresources_temp09` tenía la menor relevancia.
5. Seleccione la carpeta para ver la configuración de la evaluación.
6. Escriba `ctrl + c` para salir de la aplicación y volver al terminal.

Comparación de respuestas

Compare las respuestas de las evaluaciones.

1. Seleccione dos de las evaluaciones que se van a comparar y, a continuación, use la misma herramienta de revisión para comparar las respuestas.

Bash

```
python -m evaltools diff my_results/experiment_refined
my_results/experiment_ignoresources_temp09
```

2. Revise los resultados. Sus resultados pueden variar.

What should one expect when choosing an out-of-network provider or services not covered under the Northwind Standard plan?		
experiment_refined	experiment_ignoresources_temp09	
<p>When choosing an out-of-network provider or services not covered under the Northwind Standard plan, there are a few things you should expect:</p> <ol style="list-style-type: none"> 1. Limited or no coverage: The Northwind Standard plan does not provide coverage for services received from health care providers who are not contracted with Northwind Health [Northwind_Standard_Benefits_Details.pdf#page=89]. 2. Out-of-pocket expenses: If you choose to receive services from an out-of-network provider or 		
<p>To minimize your out-of-pocket costs and ensure that you are receiving the best care possible, you should</p>		
groundedness	relevance	coherence
5	5	5
groundedness	relevance	coherence
5	5	5

3. Escriba `ctrl + c` para salir de la aplicación y volver al terminal.

Sugerencias para futuras evaluaciones

- Edite las solicitudes en `my_input` para adaptar las respuestas en función del ámbito temático, la longitud y otros factores.
- Edite el archivo `my_config.json` para cambiar los parámetros como `temperature` y `semantic_ranker`, y vuelva a ejecutar los experimentos.
- Compare diferentes respuestas para comprender cómo afectan la solicitud y la pregunta a la calidad de la respuesta.

- Genere un conjunto independiente de preguntas y respuestas de verdad básicas para cada documento del índice de Azure AI Search. A continuación, vuelva a realizar las evaluaciones para ver en qué difieren las respuestas.
- Modifique las solicitudes para indicar respuestas más cortas o más largas añadiendo el requisito al final de la solicitud. Un ejemplo sería `Please answer in about 3 sentences.`

Limpieza de recursos y dependencias

Los pasos siguientes le guiarán por el proceso de limpieza de los recursos que usó.

Limpieza de los recursos de Azure

Los recursos Azure creados en este artículo se facturan a su suscripción Azure. Si no espera necesitar estos recursos en el futuro, elimínelos para evitar incurrir en más gastos.

Para eliminar los recursos de Azure y quitar el código fuente, ejecute el siguiente comando de la CLI para desarrolladores de Azure:

Bash

```
azd down --purge
```

Limpieza de GitHub Codespaces y Visual Studio Code

GitHub Codespaces

La eliminación del entorno de GitHub Codespaces garantiza que pueda maximizar la cantidad de derechos de horas gratuitas por núcleo que obtiene para su cuenta.

ⓘ **Importante**

Para obtener más información sobre los derechos de la cuenta de GitHub, consulte [Almacenamiento y horas de núcleo incluidas mensualmente en GitHub Codespaces](#).

1. Inicie sesión en el [panel de GitHub Codespaces](#).

2. Busque los codespaces que se ejecutan actualmente a partir del repositorio de GitHub [GitHub Azure-Samples/ai-rag-chat-evaluator](#).

The screenshot shows the GitHub Codespaces interface. At the top, there's a search bar and several navigation icons. Below that, a sidebar on the left lists 'All' (1) and 'Templates'. The main area is titled 'Your codespaces' and contains a section for 'Explore quick start templates' with options for 'Blank', 'React', and 'Jupyter Notebook'. A red box highlights a specific codespace card for 'effective orbit' owned by 'developer-bob'. The card shows the repository 'Azure-Samples/azure-search-openai-demo', the branch 'main', and status 'No changes'. It also indicates the machine type as '2-core + 8GB RAM + 32GB', a retrieval status of 'Retrieving...', and a last use time of '7 minutes ago'. There are three dots at the end of the card.

3. Abra el menú contextual del espacio de código y seleccione Eliminar.

This screenshot shows the same GitHub Codespaces interface as the previous one, but with a context menu open over the 'effective orbit' codespace card. The menu includes options like 'Open in ...', 'Rename', 'Export changes to a fork', 'Change machine type', 'Keep codespace', and 'Delete'. The 'Delete' button is highlighted with a red box. The rest of the interface is identical to the first screenshot.

Vuelva al artículo de la aplicación de chat para limpiar esos recursos.

- [JavaScript](#)
- [Pitón](#)

Contenido relacionado

- Consulte el [repositorio de evaluaciones ↗](#).

- Consulte el repositorio de GitHub de la aplicación de chat empresarial [↗](#).
- Cree una aplicación de chat [con la arquitectura de soluciones de mejores prácticas de Azure OpenAI↗](#).
- Conozca el control de acceso [en las aplicaciones de inteligencia artificial generativa con Azure AI Search↗](#).
- Cree una solución de Azure OpenAI preparada para la empresa [con Azure API Management↗](#).
- Consulte [Búsqueda de Azure AI: Superar el vector de búsqueda con capacidades híbridas de recuperación y clasificación↗](#).

Escalado del chat de Azure OpenAI para Python mediante RAG con Azure Container Apps

Artículo • 05/01/2025

Aprenda a agregar equilibrio de carga a la aplicación para ampliar la aplicación de chat más allá de los límites de cuota de modelo y token de Azure OpenAI. Este enfoque usa Azure Container Apps para crear tres puntos de conexión de Azure OpenAI, así como un contenedor principal para dirigir el tráfico entrante a uno de los tres puntos de conexión.

Este artículo requiere que implemente dos ejemplos independientes:

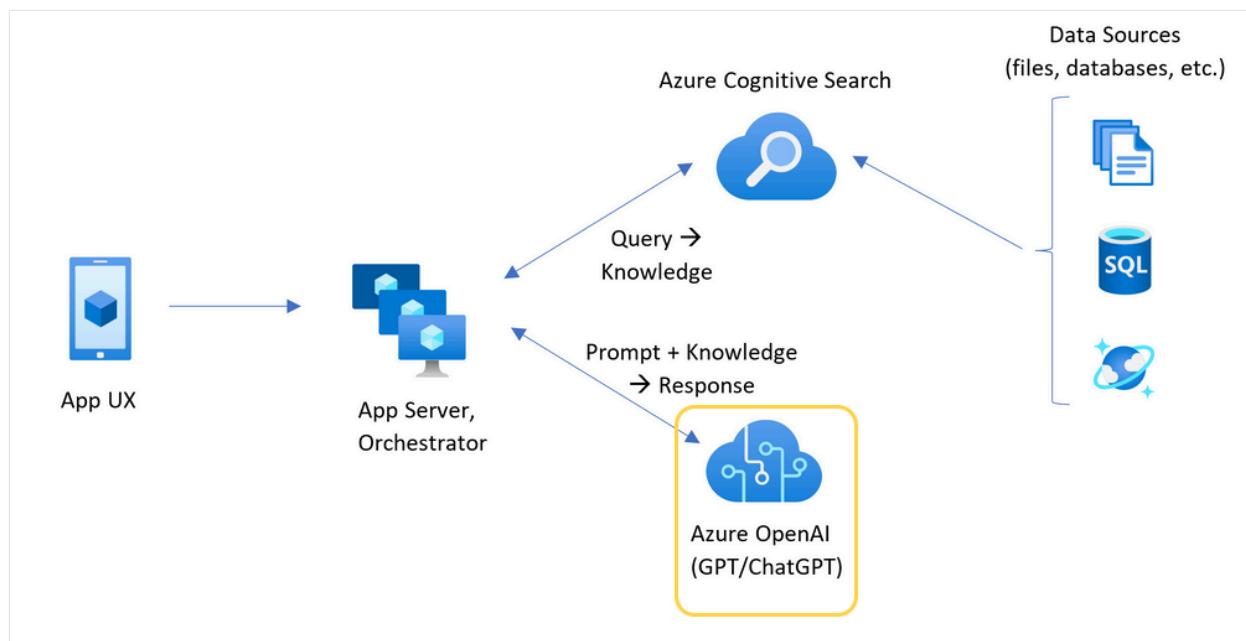
- Aplicación de chat
 - Si aún no ha implementado la aplicación de chat, espere hasta que se implemente el ejemplo del equilibrador de carga.
 - Si ya ha implementado la aplicación de chat una vez, cambiará la variable de entorno para admitir un punto de conexión personalizado para el equilibrador de carga y volver a implementarla.
 - Aplicación de chat disponible en estos idiomas:
 - [.NET](#)
 - [JavaScript](#)
 - [Python](#)
- Aplicación del equilibrador de carga

ⓘ Nota

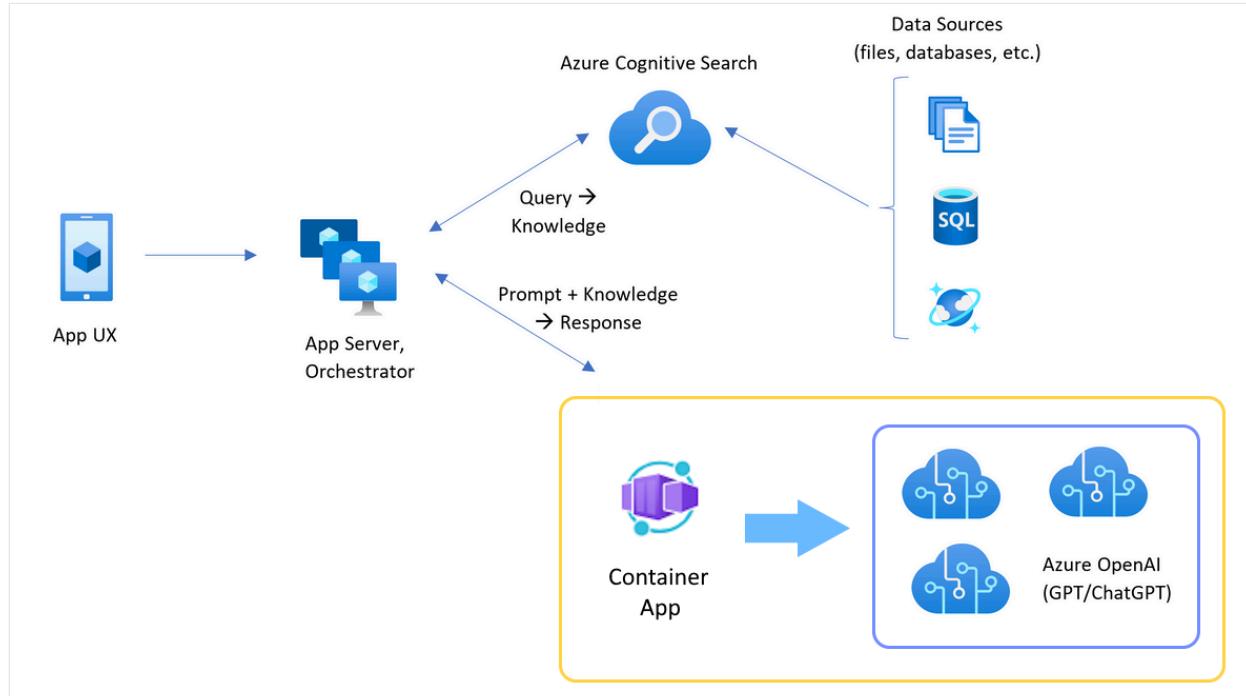
En este artículo se utilizan una o más plantillas de aplicación de IA de como base para los ejemplos y orientaciones del artículo. Las plantillas de aplicación de IA proporcionan implementaciones de referencia bien mantenidas y fáciles de implementar que ayudan a garantizar un punto de partida de alta calidad para las aplicaciones de IA.

Arquitectura para el equilibrio de carga de Azure OpenAI con Azure Container Apps

Dado que el recurso de Azure OpenAI tiene límites específicos de cuota de tokens y modelos, una aplicación de chat que usa un único recurso de Azure OpenAI es propenso a tener errores de conversación debido a esos límites.

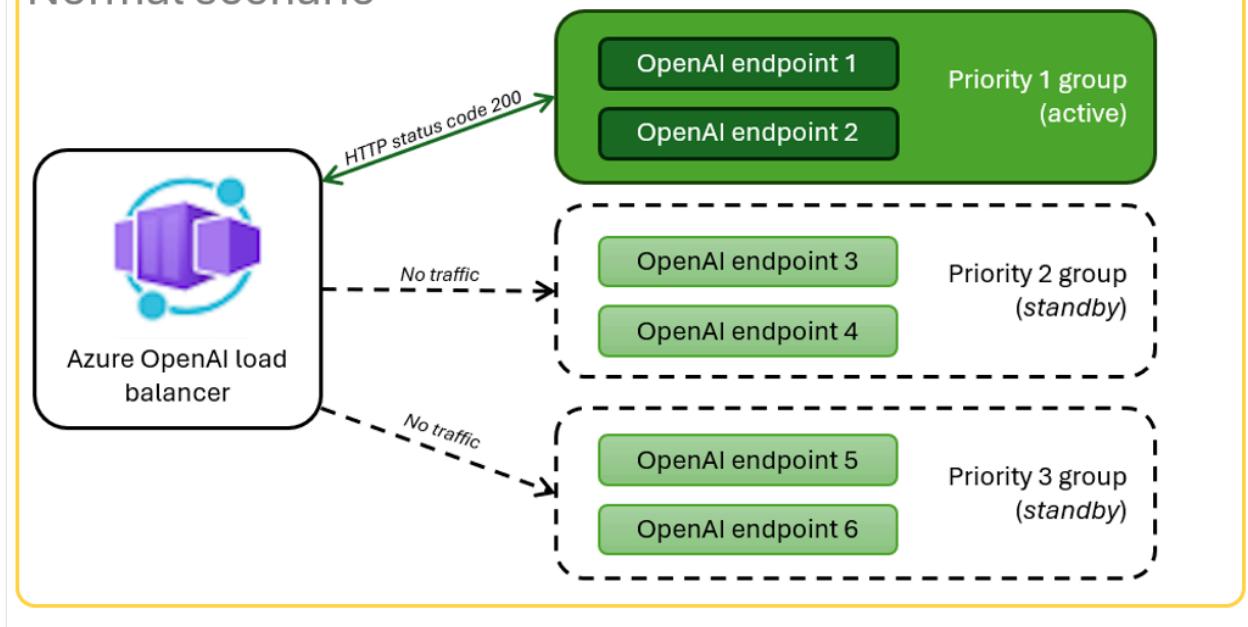


Para usar la aplicación de chat sin alcanzar esos límites, use una solución de carga equilibrada con Azure Container Apps. Esta solución expone sin problemas un único punto de conexión de Azure Container Apps al servidor de aplicaciones de chat.



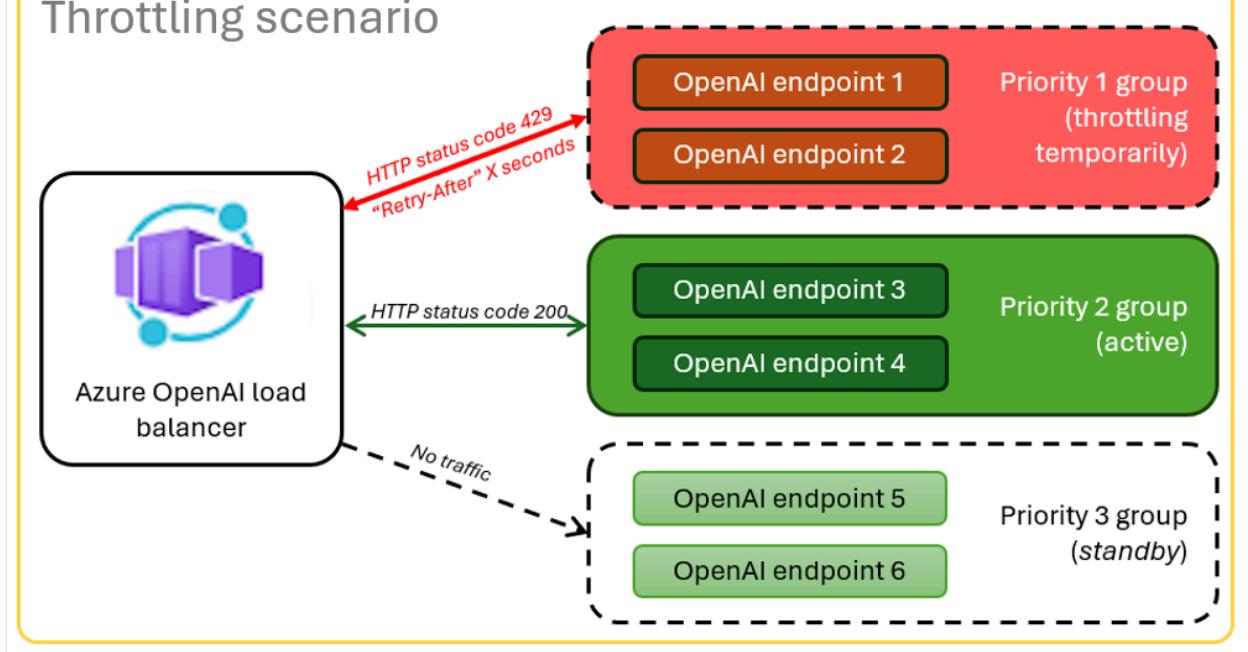
La aplicación de Azure Container se encuentra delante de un conjunto de recursos de Azure OpenAI. La aplicación de Container resuelve dos escenarios: normal y limitado. Durante un **escenario normal** donde está disponible la cuota de tokens y modelos, el recurso de Azure OpenAI devuelve un valor de 200 a través de Container App y App Server.

Normal scenario



Cuando un recurso está en un **escenario limitado**, por ejemplo, debido a los límites de cuota, la aplicación de Azure Container puede reintentar un recurso de Azure OpenAI diferente inmediatamente para completar la solicitud de aplicación de chat original.

Throttling scenario



Prerrequisitos

- Suscripción de Azure. [Crear uno gratis ↗](#)
- Acceso concedido a Azure OpenAI en la suscripción de Azure deseada.

Actualmente, la aplicación solo concede acceso a este servicio. Puede solicitar acceso a Azure OpenAI completando el formulario en <https://aka.ms/oai/access>.

- Los [contenedores de desarrollo](#) están disponibles para ambas muestras, con todas las dependencias necesarias para completar este artículo. Puede ejecutar los contenedores de desarrollo en GitHub Codespaces (en un explorador) o localmente mediante Visual Studio Code.

Codespaces (recomendado)

○ Cuenta de GitHub

Apertura de la aplicación de ejemplo del equilibrador local de aplicaciones de contenedor

Codespaces (recomendado)

[GitHub Codespaces](#) ejecuta un contenedor de desarrollo administrado por GitHub con [Visual Studio Code para web](#) como interfaz de usuario. Para el entorno de desarrollo más sencillo, use GitHub Codespaces para que tenga las herramientas de desarrollo y las dependencias correctas preinstaladas para completar este artículo.



[Open in GitHub Codespaces](#)

ⓘ Importante

Todas las cuentas de GitHub pueden usar Codespaces durante un máximo de 60 horas gratis cada mes con 2 instancias principales. Para obtener más información, consulte [Almacenamiento y horas de núcleo incluidas mensualmente en GitHub Codespaces](#).

Implementación del equilibrador de carga de Azure Container Apps

1. Inicie sesión en la CLI para desarrolladores de Azure para proporcionar autenticación a los pasos de aprovisionamiento e implementación.

```
Bash
```

```
azd auth login --use-device-code
```

2. Establezca una variable de entorno para usar la autenticación de la CLI de Azure en el paso posterior al aprovisionamiento.

```
Bash
```

```
azd config set auth.useAzCliAuth "true"
```

3. Implemente la aplicación del equilibrador de carga.

```
Bash
```

```
azd up
```

Deberá seleccionar una suscripción y una región para la implementación. No es necesario que sean la misma suscripción y región que la aplicación de chat.

4. Espere a que se complete la implementación antes de continuar.

Obtener el extremo de implementación

1. Use el comando siguiente para mostrar el punto de conexión implementado para la aplicación Contenedor de Azure.

```
Bash
```

```
azd env get-values
```

2. Copie el valor `CONTAINER_APP_URL`. Lo usarás en la siguiente sección.

Reimplementación de la aplicación de chat con el punto de chat del equilibrador de carga

Estos se completan en el ejemplo de la aplicación de chat.

Despliegue inicial

1. Abre el contenedor de desarrollo de la muestra de aplicación de chat con una de las siguientes opciones.

[Expandir tabla](#)

Idioma	Codespaces	Visual Studio Code
.NET	Open in GitHub Codespaces	Dev Containers Open
JavaScript	Open in GitHub Codespaces	Dev Containers Open
Pitón	Open in GitHub Codespaces	Dev Containers Open

2. Inicie sesión en la CLI para desarrolladores de Azure (AZD).

```
Bash  
azd auth login
```

Finalice las instrucciones de inicio de sesión.

3. Cree un entorno de AZD con un nombre como `chat-app`.

```
Bash  
azd env new <name>
```

4. Agregue la siguiente variable de entorno, que indica al back-end de la aplicación chat que use una dirección URL personalizada para las solicitudes de OpenAI.

```
Bash  
azd env set OPENAI_HOST azure_custom
```

5. Agregue la siguiente variable de entorno, sustituyendo `<CONTAINER_APP_URL>` por la dirección URL de la sección anterior. Esta acción indica al back-end de la aplicación chat cuál es el valor de la dirección URL personalizada para la solicitud openAI.

```
Bash
```

```
azd env set AZURE_OPENAI_CUSTOM_URL <CONTAINER_APP_URL>
```

6. Implemente la aplicación de chat.

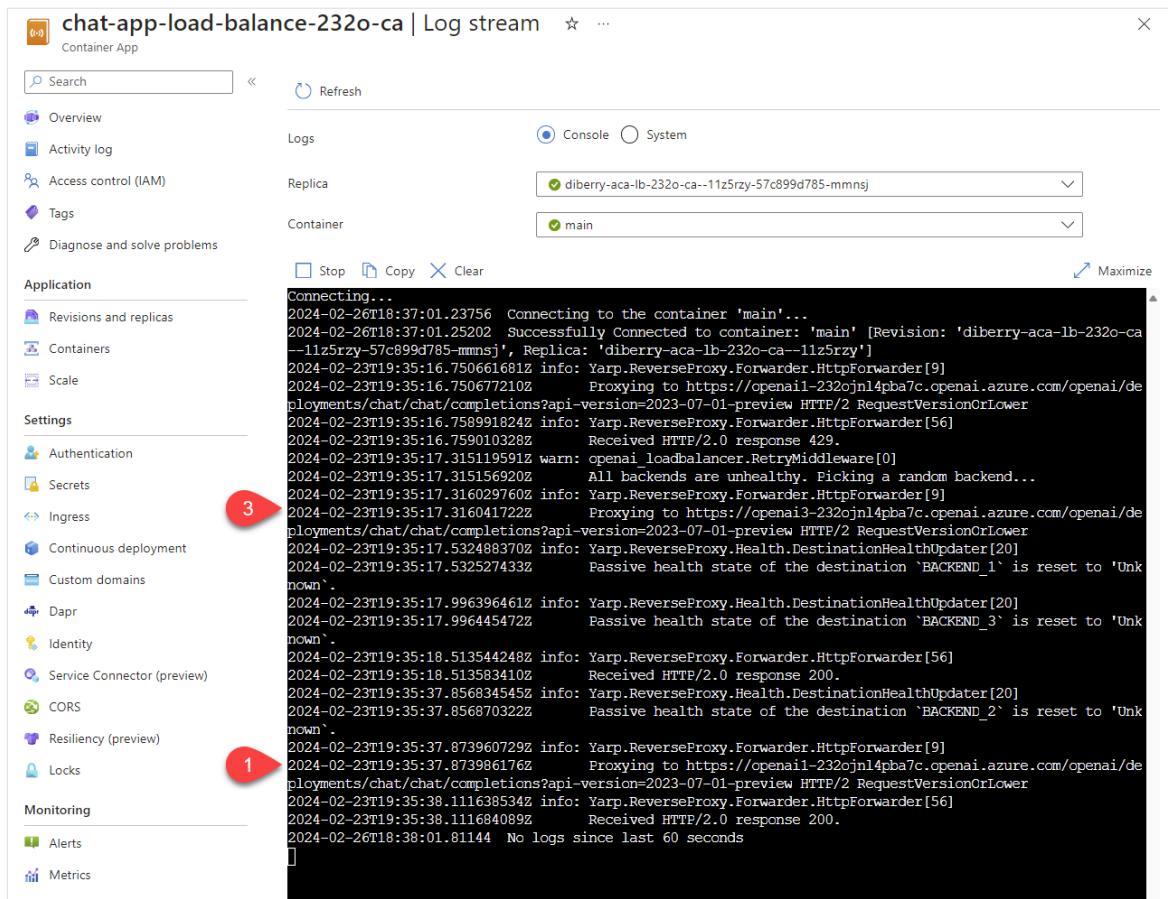
```
Bash
```

```
azd up
```

Ahora puede usar la aplicación de chat con confianza, sabiendo que está diseñada para escalar entre muchos usuarios sin agotar la cuota.

Transmitir registros para ver los resultados del equilibrador de carga

1. En [Azure Portal](#), busque el grupo de recursos.
2. En la lista de recursos del grupo, seleccione el recurso Container App.
3. Seleccione **Supervisión** -> **Flujo de registro** para ver el registro.
4. Use la aplicación de chat para generar tráfico en el registro.
5. Busque los registros, que hacen referencia a los recursos de Azure OpenAI. Cada uno de los tres recursos tiene su identidad numérica en el comentario de registro a partir de `Proxying to https://openai3`, donde `3` indica el tercer recurso de Azure OpenAI.



6. Al usar la aplicación de chat, cuando el equilibrador de carga recibe el estado de que la solicitud ha superado la cuota, el equilibrador de carga gira automáticamente a otro recurso.

Configuración de la cuota de TPM

De forma predeterminada, cada una de las instancias de Azure OpenAI del equilibrador de carga se implementa con una capacidad de 30 000 tokens por minuto (TPM). Puede usar la aplicación de chat con la confianza de que está construida para escalar entre muchos usuarios sin agotar la cuota. Cambie este valor cuando:

- Obtiene errores de capacidad de implementación: menor que ese valor.
- Necesita una capacidad más alta: aumente el valor.

1. Use el siguiente comando para cambiar el valor:

```
Bash
azd env set OPENAI_CAPACITY 50
```

2. Vuelva a implementar el equilibrador de carga:

```
Bash
```

```
azd up
```

Limpieza de recursos

Cuando haya terminado con la aplicación de chat y el equilibrador de carga, limpie los recursos. Los recursos de Azure creados en este artículo se facturan a su suscripción de Azure. Si no espera necesitar estos recursos en el futuro, elimínelos para evitar incurrir en más cargos.

Limpieza de los recursos de la aplicación de chat

Vuelva al artículo de la aplicación de chat para limpiar esos recursos.

- [.NET](#)
- [JavaScript](#)
- [Python](#)

Limpieza de los recursos del equilibrador de carga

Ejecute el siguiente comando de la CLI para desarrolladores de Azure para eliminar los recursos de Azure y quite el código fuente:

```
Bash
```

```
azd down --purge --force
```

Los interruptores proporcionan:

- `purge`: Los recursos eliminados se purgan inmediatamente. Esto le permite reutilizar el TPM de Azure OpenAI.
- `force`: la eliminación se produce de forma silenciosa, sin necesidad de consentimiento del usuario.

Limpieza de GitHub Codespaces

GitHub Codespaces

La eliminación del entorno de GitHub Codespaces garantiza que pueda maximizar la cantidad de derechos de horas gratuitas por núcleo que obtiene para su cuenta.

ⓘ Importante

Para obtener más información sobre los derechos de la cuenta de GitHub, consulte [Almacenamiento y horas de núcleo incluidas mensualmente en GitHub Codespaces](#).

1. Inicie sesión en el panel de GitHub Codespaces (<https://github.com/codespaces>).
2. Busque los espacios de código que se ejecutan actualmente procedentes del repositorio de GitHub [azure-samples/openai-aca-lb](#).

The screenshot shows the GitHub Codespaces interface. On the left, there's a sidebar with 'All' selected, 'Templates' button, and a 'By repository' section showing 'Azure-Samples/openai-aca-lb' with a '1' badge. The main area is titled 'Your codespaces' and contains a message 'Help us improve GitHub Codespaces'. Below it, a list of codespaces is shown under the heading 'Owned by Azure-Samples'. One codespace, 'Azure-Samples/openai-aca-lb' with the branch 'main', is highlighted with a red box. To its right, status indicators show '2-core', 'Retrieving...', and 'Active'.

3. Abra el menú contextual del espacio de código y seleccione Eliminar.

The screenshot shows the context menu for a specific codespace. The menu items include: Rename, Export changes to a fork, Change machine type, Stop codespace, Auto-delete codespace (with a checked checkbox), Open in Browser, Open in Visual Studio Code, Open in JetBrains Gateway (Beta), and Open in JupyterLab (Beta). The 'Delete' option at the bottom is highlighted with a red box and a red arrow pointing to it.

Obtener ayuda

Si tiene problemas para implementar el equilibrador de carga de Azure API Management, registre el problema en el repositorio [Problemas](#).

Código de ejemplo

Entre los ejemplos usados en este artículo se incluyen:

- aplicación de chat de Python con RAG ↗
- Balanceador de Carga con Aplicaciones de Contenedores de Azure ↗

Paso siguiente

- Usa [Azure Load Testing](#) para realizar pruebas de carga en tu aplicación de chat
-

Comentarios

¿Le ha resultado útil esta página?

 Sí

 No

[Proporcionar comentarios sobre el producto](#) ↗ | [Obtener ayuda en Microsoft Q&A](#)

Escalado de Azure OpenAI para Python con Azure API Management

Artículo • 16/05/2024

Aprenda a agregar equilibrio de carga de nivel empresarial a la aplicación para ampliar la aplicación de chat más allá de los límites de cuota de modelo y token de Azure OpenAI. Este enfoque usa Azure API Management para dirigir el tráfico inteligentemente entre tres recursos de Azure OpenAI.

Este artículo requiere que implemente dos ejemplos independientes:

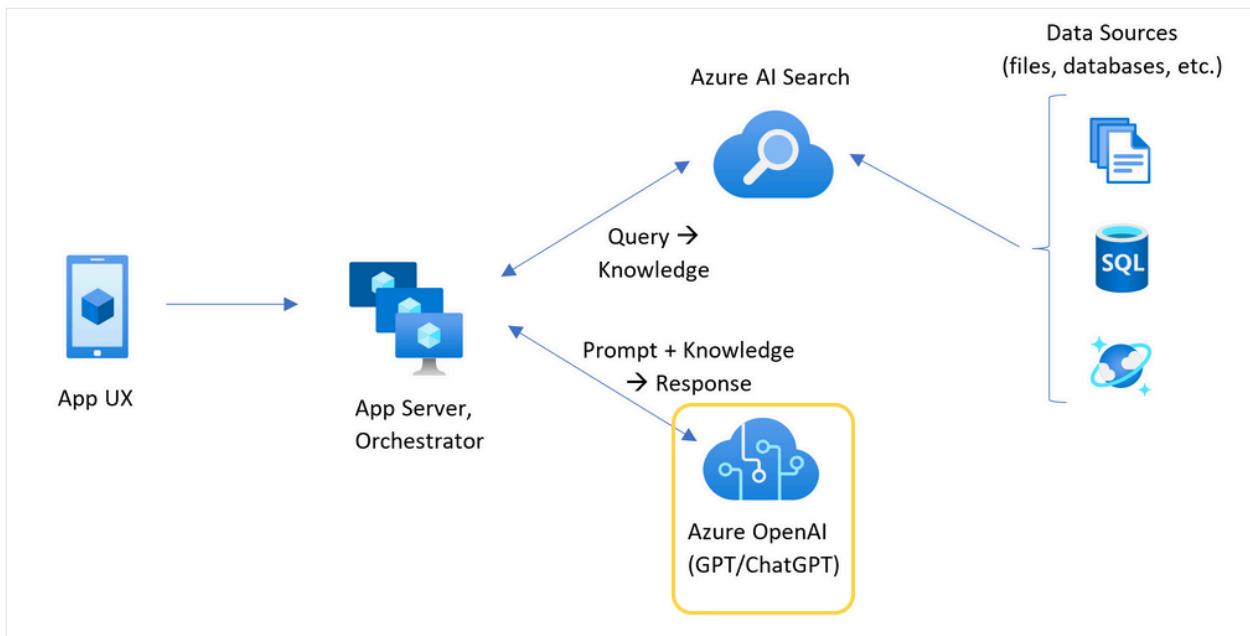
- Aplicación de chat
 - Si aún no ha implementado la aplicación de chat, espere hasta que se implemente el ejemplo del equilibrador de carga.
 - Si ya ha implementado la aplicación de chat una vez, cambiará la variable de entorno para admitir un punto de conexión personalizado para el equilibrador de carga y volver a implementarla.
- Equilibrador de carga con Azure API Management

ⓘ Nota

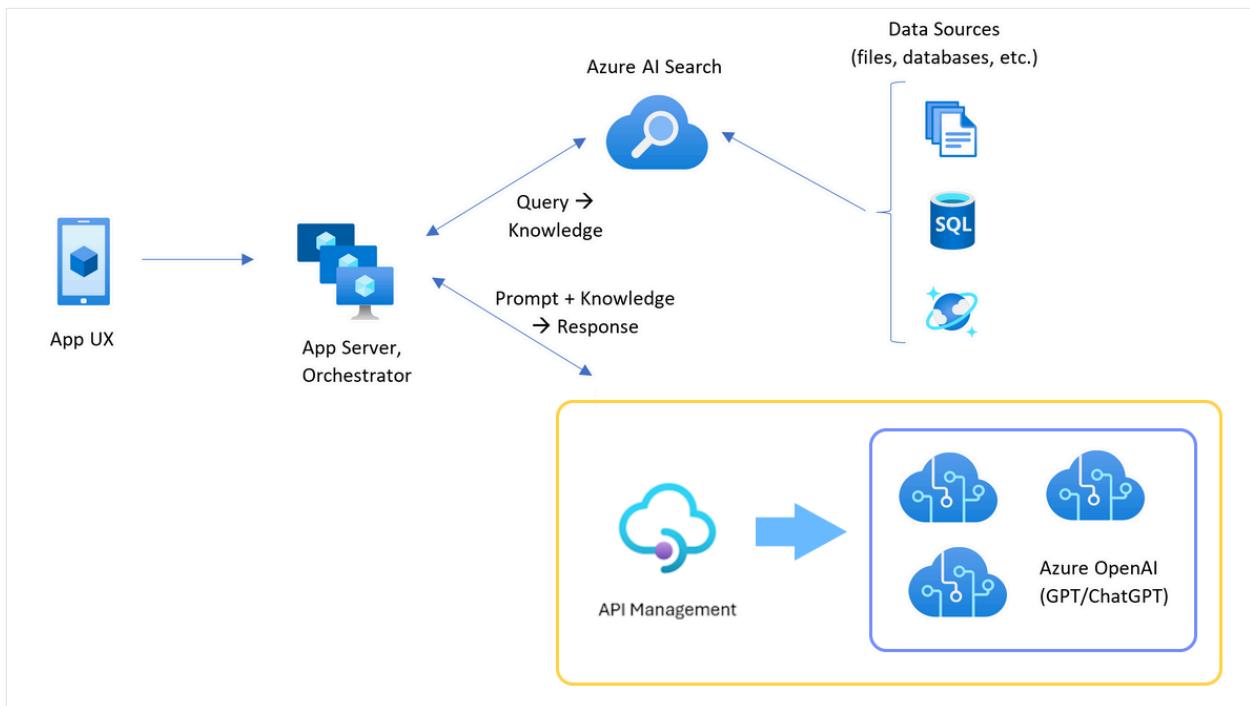
En este artículo se usan una o varias [plantillas](#) de aplicación de IA como base para los ejemplos e instrucciones del artículo. Las plantillas de aplicación de IA proporcionan implementaciones de referencia bien mantenidas y fáciles de implementar que ayudan a garantizar un punto de partida de alta calidad para las aplicaciones de IA.

Arquitectura para el equilibrio de carga de Azure OpenAI con Azure API Management

Dado que el recurso de Azure OpenAI tiene límites específicos de cuota de tokens y modelos, una aplicación de chat que usa un único recurso de Azure OpenAI es propenso a tener errores de conversación debido a esos límites.

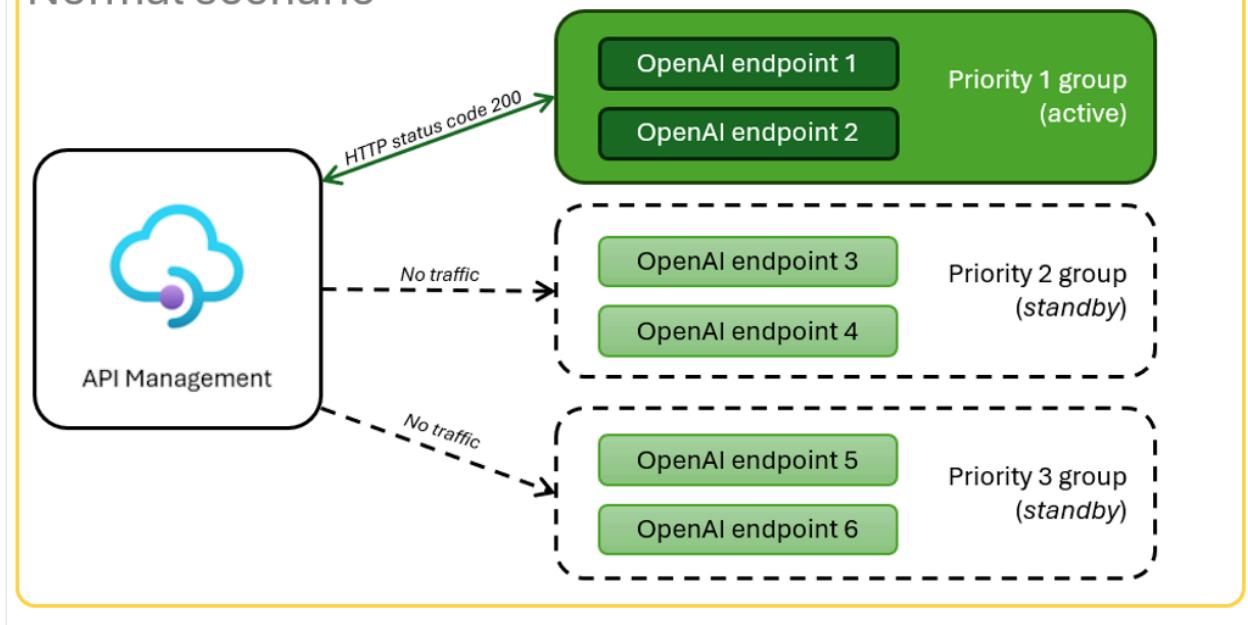


Para usar la aplicación de chat sin alcanzar esos límites, use una solución de carga equilibrada con Azure API Management. Esta solución expone sin problemas un único punto de conexión de Azure API Management al servidor de aplicaciones de chat.



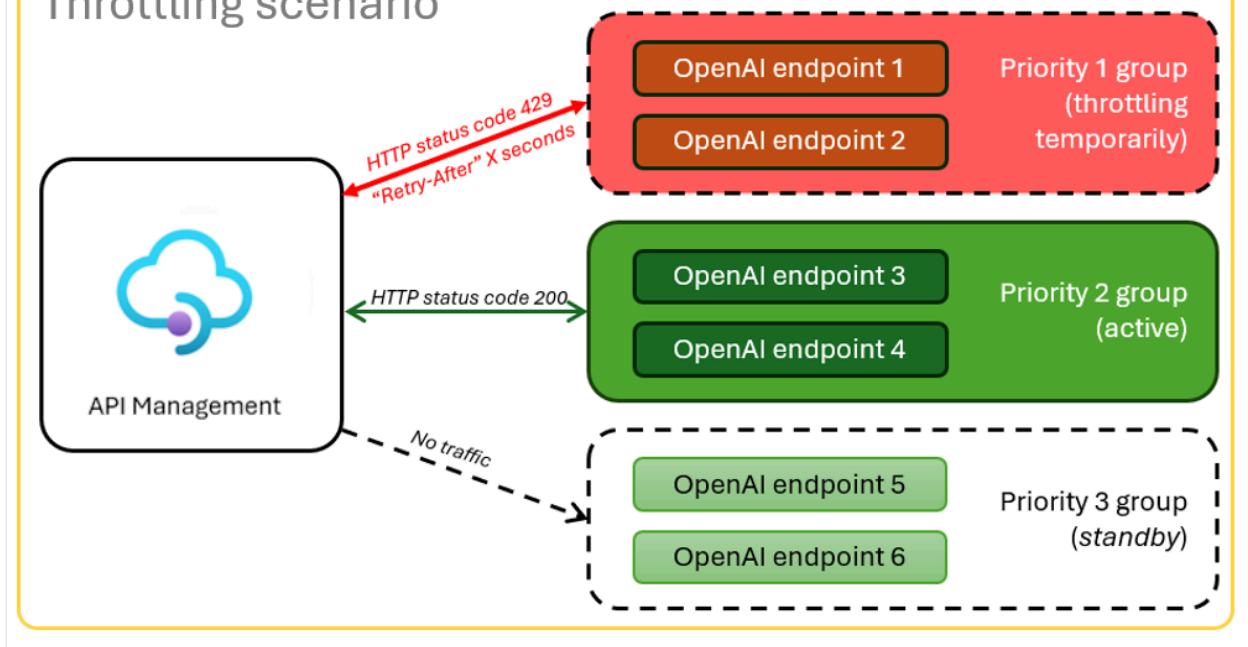
El recurso de Azure API Management, como capa de API, se encuentra delante de un conjunto de recursos de Azure OpenAI. La capa de API se aplica a dos escenarios: normales y limitados. Durante un **escenario** normal en el que el token y la cuota del modelo están disponibles, el recurso de Azure OpenAI devuelve un valor de 200 a través de la capa de API y el servidor de aplicaciones back-end.

Normal scenario



Cuando se **limita** un recurso debido a los límites de cuota, la capa de API puede reintentar un recurso de Azure OpenAI diferente inmediatamente para cumplir la solicitud de aplicación de chat original.

Throttling scenario



Requisitos previos

- Suscripción de Azure. [Crear una cuenta gratuita ↗](#)
- Acceso concedido a Azure OpenAI en la suscripción de Azure que quiera.

Actualmente, solo la aplicación concede acceso a este servicio. Para solicitar acceso a Azure OpenAI, rellene el formulario en <https://aka.ms/oai/access>.

- Los [contenedores de desarrollo](#) están disponibles para ambos ejemplos, con todas las dependencias necesarias para completar este artículo. Puede ejecutar los contenedores de desarrollo en GitHub Codespaces (en un explorador) o localmente mediante Visual Studio Code.

Codespaces (recomendado)

- Solo se requiere una [cuenta](#) de GitHub para usar Codespaces.

Apertura de la aplicación de ejemplo del equilibrador local de Azure API Management

Codespaces (recomendado)

[GitHub Codespaces](#) ejecuta un contenedor de desarrollo administrado por GitHub con [Visual Studio Code para la web](#) como interfaz de usuario. Para obtener el entorno de desarrollo más sencillo, utilice Codespaces de GitHub de modo que tenga las herramientas y dependencias de desarrollador correctas preinstaladas para completar este artículo.



[Open in GitHub Codespaces](#)

ⓘ Importante

Todas las cuentas de GitHub pueden usar Codespaces durante un máximo de 60 horas gratis cada mes con 2 instancias principales. Para obtener más información, consulte [Almacenamiento y horas de núcleo incluidas mensualmente en GitHub Codespaces](#).

Implementación del equilibrador de carga de Azure API Management

1. Para implementar el equilibrador de carga en Azure, inicie sesión en Azure Developer CLI (AZD).

```
Bash
```

```
azd auth login
```

2. Finalice las instrucciones de inicio de sesión.
3. Implemente la aplicación del equilibrador de carga.

```
Bash
```

```
azd up
```

Deberá seleccionar una suscripción y una región para la implementación. No tienen por qué ser de la misma suscripción y región que la aplicación de chat.

4. Antes de continuar, espere a que se complete la implementación. Esto puede tardar hasta 30 minutos.

Obtención del punto de conexión del equilibrador de carga

Ejecute el siguiente comando de Bash para ver las variables de entorno desde la implementación. Necesitará esta información más adelante.

```
Bash
```

```
azd env get-values | grep APIM_GATEWAY_URL
```

Reimplementación de la aplicación de chat con el punto de chat del equilibrador de carga

Estos se completan en el ejemplo de la aplicación de chat.

Implementación inicial

1. Abra el contenedor de desarrollo de la aplicación de chat mediante una de las siguientes opciones.

 Expandir tabla

Lenguaje	Codespaces	Visual Studio Code
.NET	 Open in GitHub Codespaces ↗	 Dev Containers Open ↗
JavaScript	 Open in GitHub Codespaces ↗	 Dev Containers Open ↗
Python	 Open in GitHub Codespaces ↗	 Dev Containers Open ↗

2. Inicie sesión en Azure Developer CLI (AZD).

```
Bash
azd auth login
```

Finalice las instrucciones de inicio de sesión.

3. Cree un entorno de AZD con un nombre como `chat-app`.

```
Bash
azd env new <name>
```

4. Agregue la siguiente variable de entorno, que indica al back-end de la aplicación chat que usa una dirección URL personalizada para las solicitudes de OpenAI.

```
Bash
azd env set OPENAI_HOST azure_custom
```

5. Agregue la siguiente variable de entorno, que indica al back-end de la aplicación chat cuál es el valor de la dirección URL personalizada para la solicitud de OpenAI.

```
Bash
azd env set AZURE_OPENAI_CUSTOM_URL <APIM_GATEWAY_URL>
```

6. Implemente la aplicación de chat.

```
Bash
```

```
azd up
```

Configuración de los tokens por cuota de minutos (TPM)

De forma predeterminada, cada una de las instancias de OpenAI del equilibrador de carga se implementará con 30 000 TPM (tokens por minuto). Puedes usar la aplicación de chat con la confianza de que se ha creado para escalar horizontalmente entre muchos usuarios sin quedarse sin cuota. Cambie este valor cuando:

- Obtiene errores de capacidad de implementación: menor que ese valor.
- Planee una mayor capacidad, aumente el valor.

1. Use el siguiente comando para cambiar el valor.

```
Bash
```

```
azd env set OPENAI_CAPACITY 50
```

2. Vuelva a implementar el equilibrador de carga.

```
Bash
```

```
azd up
```

Limpieza de recursos

Cuando haya terminado con la aplicación de chat y el equilibrador de carga, limpie los recursos. Los recursos Azure creados en este artículo se facturan a su suscripción Azure. Si no espera necesitar estos recursos en el futuro, elimínelos para evitar incurrir en más gastos.

Limpieza de los recursos de la aplicación de chat

Vuelva al artículo de la aplicación de chat para limpiar esos recursos.

- [.NET](#)
- [JavaScript](#)

- Python

Limpieza de los recursos del equilibrador de carga

Ejecute el siguiente comando de la Azure Developer CLI para eliminar los recursos de Azure y eliminar el código de origen:

Bash

```
azd down --purge --force
```

Los modificadores proporcionan:

- `purge`: los recursos eliminados se purgan inmediatamente. Esto le permite reutilizar el TPM de Azure OpenAI.
- `force`: la eliminación se produce de forma silenciosa, sin necesidad de consentimiento del usuario.

Limpiar GitHub Codespaces

GitHub Codespaces

La eliminación del entorno de GitHub Codespaces garantiza que pueda maximizar la cantidad de derechos de horas gratuitas por núcleo que obtiene para su cuenta.

ⓘ Importante

Para obtener más información sobre los derechos de la cuenta de GitHub, consulte [Almacenamiento y horas de núcleo incluidas mensualmente en GitHub Codespaces](#).

1. Inicie sesión en el panel de GitHub Codespaces (<https://github.com/codespaces>).
2. Busque los espacios de código que se ejecutan actualmente procedentes del repositorio de GitHub [azure-samples/openai-apim-lb](https://github.com/azure-samples/openai-apim-lb).

The screenshot shows the GitHub Codespaces interface. On the left, there's a sidebar with filters: 'All', 'Templates', 'By repository', and one for 'Azure-Samples/openai-apim-lb'. The main area is titled 'Your codespaces' and contains a single item: 'didactic bassoon' (owned by 'Azure-Samples'). The card shows details: 2-core, 8GB RAM, 32GB storage, 2.74 GB used, last used 29 minutes ago, and three dots for more options. A red box highlights the entire card.

3. Abra el menú contextual del elemento Codespaces y, a continuación, seleccione **Eliminar**.

The screenshot shows the GitHub Codespaces interface with the same 'didactic bassoon' codespace selected. A context menu is open over the codespace card. The menu items are: Rename, Export changes to a fork, Change machine type, Auto-delete codespace (with a checked checkbox and a checkmark), Open in Browser, Open in Visual Studio Code, Open in JetBrains Gateway (Beta), and Open in JupyterLab (Beta). At the bottom of the menu, there is a red button labeled 'Delete' with a red arrow pointing to it.

Obtener ayuda

Si tiene problemas para implementar el equilibrador de carga de Azure API Management, registre el problema en el repositorio [Problemas ↗](#).

Código de ejemplo

Entre los ejemplos usados en este artículo se incluyen:

- [Aplicación de chat de Python con RAG ↗](#)
- [Load Balancer con Azure API Management ↗](#)

Paso siguiente

- Visualización de datos de diagnóstico de Azure API Management en Azure Monitor
 - Uso de Azure Load Testing para probar la carga de la aplicación de chat con
-

Comentarios

¿Le ha resultado útil esta página?



Sí



No

Proporcionar comentarios sobre el producto [↗](#) | Obtener ayuda en Microsoft Q&A

Prueba de carga de la aplicación de chat de Python mediante RAG con locust

Artículo • 21/05/2024

En este artículo se proporciona el proceso para realizar pruebas de carga en una aplicación de chat de Python mediante el patrón RAG con Locust, una popular herramienta de prueba de carga de código abierto. El objetivo principal de las pruebas de carga es asegurarse de que la carga esperada en la aplicación de chat no supere la cuota actual de transacciones de Azure OpenAI por minuto (TPM). Al simular el comportamiento del usuario bajo una carga pesada, puede identificar posibles cuellos de botella y problemas de escalabilidad en la aplicación. Este proceso es fundamental para asegurarse de que la aplicación de chat sigue respondiendo y confiable, incluso cuando se enfrenta a un gran volumen de solicitudes de usuario.

Vea el vídeo de demostración para obtener más información sobre las pruebas de carga de la aplicación de chat.

- [Vídeo ↗](#)

ⓘ Nota

En este artículo se usan una o varias [plantillas](#) de aplicación de IA como base para los ejemplos e instrucciones del artículo. Las plantillas de aplicación de IA proporcionan implementaciones de referencia bien mantenidas y fáciles de implementar que ayudan a garantizar un punto de partida de alta calidad para las aplicaciones de IA.

Requisitos previos

- Suscripción de Azure. [Crear una cuenta gratuita ↗](#)
- Acceso concedido a Azure OpenAI en la suscripción de Azure que quiera. Actualmente, solo la aplicación concede acceso a este servicio. Para solicitar acceso a Azure OpenAI, rellene el formulario en [https://aka.ms/oai/access ↗](https://aka.ms/oai/access).
- Los [contenedores de desarrollo ↗](#) están disponibles para ambos ejemplos, con todas las dependencias necesarias para completar este artículo. Puede ejecutar los contenedores de desarrollo en GitHub Codespaces (en un explorador) o localmente mediante Visual Studio Code.

Codespaces (recomendado)

- Solo necesita una [cuenta de GitHub](#)

- [Aplicación de chat de Python con RAG](#) : si configuró la aplicación de chat para usar una de las soluciones de equilibrio de carga, este artículo le ayudará a probar el equilibrio de carga. Las soluciones de equilibrio de carga incluyen [Azure Container Apps](#).

Abrir aplicación de ejemplo de prueba de carga

La prueba de carga se encuentra en [la solución de la aplicación](#) de chat de Python como una [prueba](#) de locust. Debe volver a ese artículo, implementar la solución y, a continuación, usar ese entorno de desarrollo de contenedores de desarrollo para completar los pasos siguientes.

Ejecutar la prueba

1. Instale las dependencias de la prueba de carga.

Bash

```
python3 -m pip install -r requirements-dev.txt
```

2. Inicie Locust, que usa el archivo de prueba locust: [locustfile.py](#) se encuentra en la raíz del repositorio.

Bash

```
locust
```

3. Abra el sitio web locust en ejecución, como `http://localhost:8089`.

4. Escriba lo siguiente en el sitio web de Locust.

 Expandir tabla

Propiedad	Valor
Número de usuarios	20

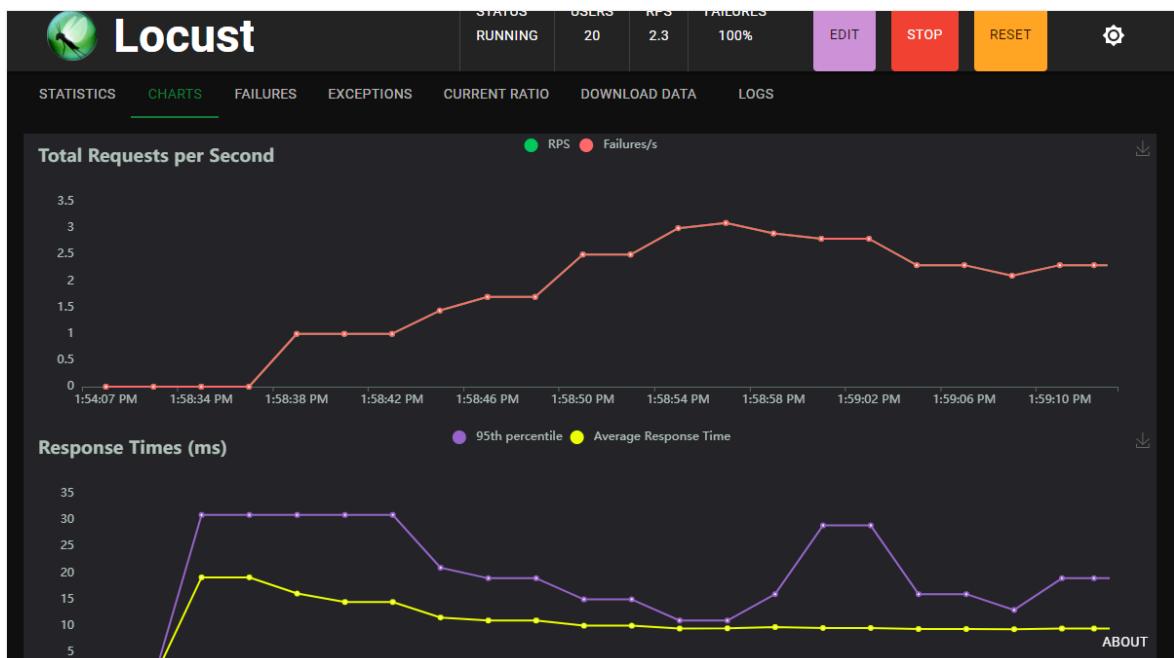
Propiedad	Valor
Ascenso	1
Host	<a href="https://<YOUR-CHAT-APP-URL>.azurewebsites.net">https://<YOUR-CHAT-APP-URL>.azurewebsites.net

The screenshot shows the Locust web interface for setting up a new load test. The configuration includes:

- Number of users (peak concurrency): 20
- Ramp Up (users started/second): 1
- Host: https://app-backend-1234.azurewebsites.net

A large green "START SWARM" button is prominently displayed at the bottom.

5. Seleccione **Iniciar swarm** para iniciar la prueba.
6. Seleccione **Gráficos** para ver el progreso de la prueba.



Limpieza de recursos

Cuando haya terminado con las pruebas de carga, limpie los recursos. Los recursos Azure creados en este artículo se facturan a su suscripción Azure. Si no espera necesitar estos recursos en el futuro, elimínelos para evitar incurrir en más gastos. Después de eliminar recursos específicos de este artículo, recuerde volver al otro tutorial de la aplicación de chat y seguir los pasos de limpieza.

Vuelva al artículo de la aplicación de chat para [limpiar](#) esos recursos.

Obtener ayuda

Si tiene problemas para usar este evaluador de carga, registre el problema en los [problemas](#) del repositorio.

Comentarios

¿Le ha resultado útil esta página?

 Sí

 No

[Proporcionar comentarios sobre el producto](#) | [Obtener ayuda en Microsoft Q&A](#)

Configuración del entorno local para implementar aplicaciones web de Python en Azure

Artículo • 04/02/2025

En este artículo se explica cómo configurar el entorno local para desarrollar aplicaciones web de Python e implementarlas en Azure. La aplicación web puede ser python pura o usar uno de los marcos web comunes basados en Python, como [django](#), [Flask](#) o [FastAPI](#).

Las aplicaciones web de Python desarrolladas localmente se pueden implementar en servicios como [Azure App Service](#), [Azure Container Apps](#) o [Azure Static Web Apps](#). Hay muchas opciones para la implementación. Por ejemplo, para la implementación de App Service, puede optar por implementar desde código, un contenedor de Docker o una aplicación web estática. Si implementa desde código, puede implementar con Visual Studio Code, con la CLI de Azure, desde un repositorio git local o con acciones de GitHub. Si implementa en un contenedor de Docker, puede hacerlo desde Azure Container Registry, Docker Hub o desde cualquier registro privado.

Antes de continuar con este artículo, se recomienda revisar el [Configuración del entorno de desarrollo](#) para obtener instrucciones sobre cómo configurar el entorno de desarrollo para Python y Azure. A continuación, analizaremos la configuración y el ajuste específicos para el desarrollo de aplicaciones web en Python.

Después de configurar el entorno local para el desarrollo de aplicaciones web de Python, estará listo para abordar estos artículos:

- Inicio rápido: [Creación de una aplicación web de Python \(Django o Flask\) en Azure App Service](#).
- Tutorial: [Implementación de una aplicación web de Python \(Django o Flask\) con PostgreSQL en Azure](#)
- [Creación e implementación de una aplicación web de Flask en Azure con una identidad administrada asignada por el sistema](#)

Trabajar con Visual Studio Code

El [Visual Studio Code](#) entorno de desarrollo integrado (IDE) es una manera fácil de desarrollar aplicaciones web de Python y trabajar con recursos de Azure que usan las aplicaciones web.

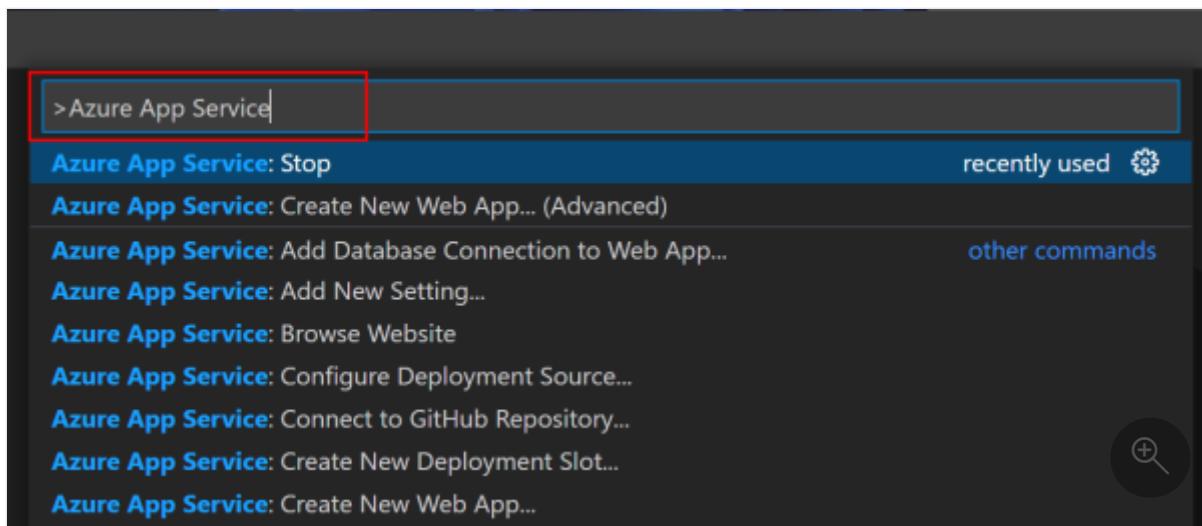
Sugerencia

Asegúrese de que tiene instalada la extensión [Python](#). Para obtener información general sobre cómo trabajar con Python en VS Code, consulte [Introducción a Python en VS Code](#).

En VS Code, trabajará con recursos de Azure a través de [extensiones de VS Code](#). Puede instalar extensiones desde la **Extensiones** Vista o la combinación de teclas Ctrl+Mayús+X. En el caso de las aplicaciones web de Python, es probable que trabaje con una o varias de las siguientes extensiones:

- La extensión [azure App Service](#) le permite interactuar con Azure App Service desde Visual Studio Code. App Service proporciona hospedaje totalmente administrado para aplicaciones web, incluidos sitios web y API web.
- La extensión [Azure Static Web Apps](#) permite crear Azure Static Web Apps directamente desde VS Code. Static Web Apps es sin servidor y es una buena opción para el hospedaje de contenido estático.
- Si planea trabajar con contenedores, instale:
 - La extensión [Docker](#) para compilar y trabajar con contenedores localmente. Por ejemplo, puede ejecutar una aplicación web de Python en contenedor en Azure App Service mediante [Web Apps for Containers](#).
 - La extensión [Azure Container Apps](#) para crear e implementar aplicaciones en contenedor directamente desde Visual Studio Code.
- Hay otras extensiones, como [Azure Storage](#), [Azure Databases](#) y [Azure Resources](#). Siempre puede agregar estas y otras extensiones según sea necesario.

Las extensiones de Visual Studio Code son accesibles como cabría esperar en una interfaz IDE típica y con compatibilidad con palabras clave enriquecidas mediante la paleta de comandos de VS Code. Para acceder a la paleta de comandos, use la combinación de teclas Ctrl+Mayús+P. La paleta de comandos es una buena manera de ver todas las posibles acciones que puede realizar en un recurso de Azure. En la captura de pantalla siguiente se muestran algunas de las acciones de App Service.



Trabajar con contenedores de desarrollo en Visual Studio Code

Los desarrolladores de Python suelen confiar en entornos virtuales para crear un entorno aislado y independiente para un proyecto específico. Los entornos virtuales permiten a los desarrolladores administrar dependencias, paquetes y versiones de Python por separado para cada proyecto, evitando conflictos entre proyectos diferentes que podrían requerir versiones de paquetes diferentes.

Aunque hay opciones populares disponibles en Python para administrar entornos como `virtualenv` o `venv`, la extensión [contenedor de desarrollo de Visual Studio Code](#) (basada en la especificación de contenedor de desarrollo abierta [abierta](#)) le permite usar un contenedor de Docker de [abierta](#) como entorno contenedorizado completo. Permite a los desarrolladores definir una cadena de herramientas coherente y fácilmente reproducible con todas las herramientas, dependencias y extensiones necesarias preconfiguradas. Esto significa que si tiene requisitos del sistema, configuraciones de shell o usa otros lenguajes por completo, puede usar un contenedor de desarrollo para configurar explícitamente todas las partes del proyecto que podrían residir fuera de un entorno básico de Python.

Por ejemplo, un desarrollador puede configurar un único contenedor de desarrollo para incluir todo lo necesario para trabajar en un proyecto, incluido un servidor de base de datos PostgreSQL junto con la base de datos del proyecto y los datos de ejemplo, un servidor de Redis, Nginx, código de front-end, bibliotecas cliente como React, etc. Además, el contenedor contendrá el código del proyecto, el entorno de ejecución de Python y todas las dependencias del proyecto de Python con las versiones correctas. Por último, el contenedor puede especificar las extensiones de Visual Studio Code que se van a instalar para que todo el equipo tenga las mismas herramientas disponibles. Por lo tanto, cuando un nuevo desarrollador se une al equipo, todo el entorno, incluidas las

herramientas, las dependencias y los datos, está listo para clonarse en su máquina local y puede empezar a trabajar inmediatamente.

Ver [Desarrollo dentro de un contenedor](#).

Trabajar con Visual Studio 2022

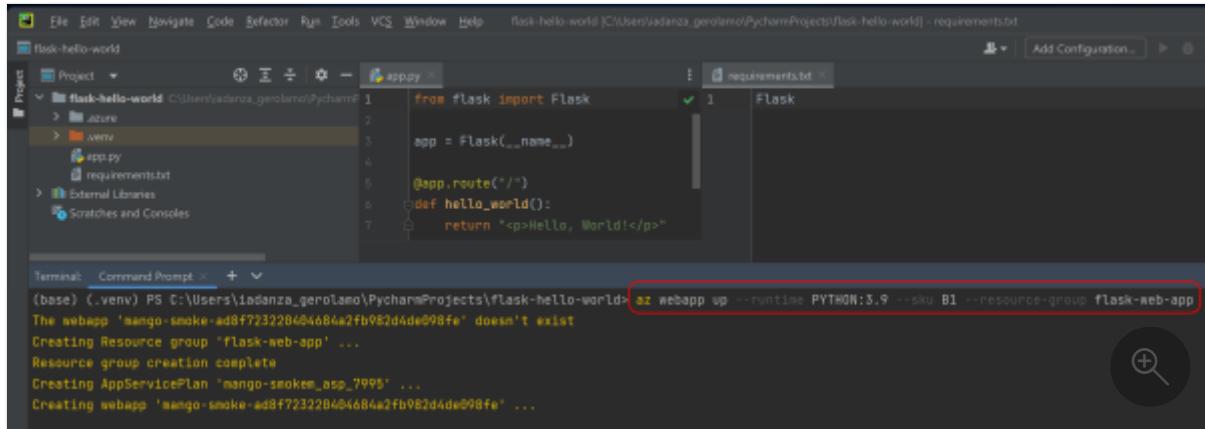
[visual Studio 2022](#) es un entorno de desarrollo integrado (IDE) completo compatible con el desarrollo de aplicaciones de Python y muchas herramientas y extensiones integradas para acceder a los recursos de Azure e implementarlos. Aunque la mayoría de la documentación para compilar aplicaciones web de Python en Azure se centra en el uso de Visual Studio Code, Visual Studio 2022 es una excelente opción si ya la tiene instalada, se siente cómodo con su uso y la usa para proyectos de .NET o C++.

- En general, consulte [Visual Studio | Documentación de Python](#) para toda la documentación relacionada con el uso de Python en Visual Studio 2022.
- Para conocer los pasos de instalación, consulte [compatibilidad con Python en Visual Studio](#) que le guía por los pasos necesarios para instalar la carga de trabajo de Python en Visual Studio 2022.
- Para obtener un flujo de trabajo general de uso de Python para el desarrollo web, consulte [Inicio rápido: Creación de la primera aplicación web de Python mediante Visual Studio](#). Este artículo es útil para comprender cómo compilar una aplicación web de Python desde cero (pero no incluye la implementación en Azure).
- Para usar Visual Studio 2022 para administrar los recursos de Azure e implementarlos en Azure, consulte [Desarrollo de Azure con Visual Studio](#). Aunque gran parte de la documentación aquí menciona específicamente .NET, las herramientas para administrar recursos de Azure e implementar en Azure funcionan igual independientemente del lenguaje de programación.
- Cuando no hay ninguna herramienta integrada disponible en Visual Studio 2022 para una tarea de implementación o administración de Azure determinada, siempre puede usar [comandos de la CLI de Azure](#).

Trabajar con otros IDE

Si trabaja en otro IDE que no tiene compatibilidad explícita con Azure, puede usar la CLI de Azure para administrar los recursos de Azure. En la captura de pantalla siguiente, se abre una aplicación web de Flask sencilla en el IDE de [PyCharm](#). La aplicación web se puede implementar en una instancia de Azure App Service mediante el comando `az`

`webapp up`. En la captura de pantalla, el comando de la CLI se ejecuta en el emulador de terminal insertado de PyCharm. Si el IDE no tiene un emulador incrustado, puede usar cualquier terminal y el mismo comando. La CLI de Azure debe estar instalada en el equipo y ser accesible en cualquier caso.



The screenshot shows a PyCharm interface with a project named "flask-hello-world". The code editor displays a file named "app.py" containing a simple Flask application. Below the code editor is a terminal window with the following output:

```
(base) (.venv) PS C:\Users\iadanza_gerolamo\PycharmProjects\flask-hello-world> az webapp up --runtime PYTHON:3.9 --sku B1 --resource-group flask-web-app
The webapp 'mango-smoke-ad8f7232204684a2fb982d4de098fe' doesn't exist
Creating Resource group 'flask-web-app' ...
Resource group creation complete
Creating AppServicePlan 'mango-smoke_asp_7995' ...
Creating webapp 'mango-smoke-ad8f7232204684a2fb982d4de098fe' ...
```

Comandos de la CLI de Azure

Al trabajar localmente con aplicaciones web mediante los comandos [CLI de Azure](#), normalmente trabajarás con los siguientes comandos:

[Expandir tabla](#)

Get-Help	Descripción
az webapp	Administra las aplicaciones web. Incluye los subcomandos crear y arriba para crear una aplicación web o para crear e implementar desde un área de trabajo local, respectivamente.
az container app	Administra Azure Container Apps.
az staticwebapp	Administra Azure Static Web Apps.
az group	Administra los grupos de recursos y las implementaciones de plantillas. Use el subcomando crear para convertir un grupo de recursos en el que colocar los recursos de Azure.
az appservice	Administra los planes de App Service.
az config	Administra la configuración de la CLI de Azure. Para guardar pulsaciones de tecla, puede definir una ubicación predeterminada o un grupo de recursos que otros comandos usen automáticamente.

Este es un comando de la CLI de Azure de ejemplo para crear una aplicación web y recursos asociados e implementarla en Azure en un comando mediante [az webapp up](#).

Ejecute el comando en el directorio raíz de la aplicación web.



```
bash
Azure CLI
az webapp up \
    --runtime PYTHON:3.9 \
    --sku B1 \
    --logs
```

Para más información sobre este ejemplo, consulte [Inicio rápido: Implementación de una aplicación web de Python \(Django o Flask\) en Azure App Service](#).

Tenga en cuenta que para algunos de los flujos de trabajo de Azure también puede usar la CLI de Azure desde una [Azure Cloud Shell](#). Azure Cloud Shell es un shell interactivo, autenticado y accesible para exploradores para administrar recursos de Azure.

Paquetes de claves del SDK de Azure

En las aplicaciones web de Python, puede hacer referencia mediante programación a los servicios de Azure mediante el SDK de Azure [para Python](#). Este SDK se describe ampliamente en la sección [Uso de las bibliotecas de Azure \(SDK\) para Python](#). En esta sección, mencionaremos brevemente algunos paquetes clave del SDK que usará en el desarrollo web. Además, mostraremos un ejemplo sobre los procedimientos recomendados para autenticar el código con recursos de Azure.

A continuación se muestran algunos de los paquetes que se usan habitualmente en el desarrollo de aplicaciones web. Puede instalar paquetes en el entorno virtual directamente con `pip`. O bien, coloque el nombre del índice de paquetes de Python (Pypi) en el archivo `requirements.txt`.

 Expandir tabla

Documentación del SDK	Instalar	Índice de paquetes de Python
Azure Identity	<code>pip install azure-identity</code>	azure-identity
Blobs de Azure Storage	<code>pip install azure-storage-blob</code>	azure-storage-blob
Azure Cosmos DB	<code>pip install azure-cosmos</code>	azure-cosmos

Documentación del SDK	Instalar	Índice de paquetes de Python
Secretos de Azure Key Vault	<code>pip install azure-keyvault-secrets</code>	azure-keyvault-secrets

El paquete [azure-identity](#) permite que la aplicación web se autentique con Microsoft Entra ID. Para la autenticación en el código de la aplicación web, se recomienda usar el `DefaultAzureCredential` de en el paquete de `azure-identity`. Este es un ejemplo de cómo acceder a Azure Storage. El patrón es similar para otros recursos de Azure.

Python

```
from azure.identity import DefaultAzureCredential
from azure.storage.blob import BlobServiceClient

azure_credential = DefaultAzureCredential()
blob_service_client = BlobServiceClient(
    account_url=account_url,
    credential=azure_credential)
```

El `DefaultAzureCredential` buscará en algunas ubicaciones predefinidas para obtener información de la cuenta, por ejemplo, en variables de entorno o a partir de la autenticación de Azure CLI. Para obtener información detallada sobre la lógica de `DefaultAzureCredential`, consulte [Autenticación de aplicaciones de Python en servicios de Azure mediante el SDK de Azure para Python](#).

Marcos web basados en Python

En el desarrollo de aplicaciones web de Python, a menudo se trabaja con marcos web basados en Python. Estos marcos proporcionan funcionalidad, como plantillas de página, administración de sesiones, acceso a bases de datos y acceso sencillo a objetos de solicitud y respuesta HTTP. Los marcos le permiten evitar tener que reinventar la rueda para funcionalidades comunes.

Tres marcos web comunes de Python son [django](#), [flask](#) o [FastAPI](#). Estos y otros marcos web se pueden usar con Azure.

A continuación se muestra un ejemplo de cómo puede empezar a trabajar rápidamente con estos marcos localmente. Al ejecutar estos comandos, terminará con una aplicación, aunque una sencilla que se pueda implementar en Azure. Ejecute estos comandos dentro de un entorno virtual de .

Paso 1: Descargar los marcos con [pip](#).

Django

```
pip install Django
```

Paso 2: Crear la aplicación Hola, Mundo.

Django

Cree un proyecto de ejemplo mediante el comando [django-admin startproject](#). El proyecto incluye un archivo *manage.py* que es el punto de entrada para ejecutar la aplicación.

```
django-admin startproject hello_world
```

paso 3: Ejecutar el código localmente.

Django

Django usa WSGI para ejecutar la aplicación.

```
python hello_world\manage.py runserver
```

Paso 4: Examinar la aplicación "Hola Mundo".

Django

```
http://127.0.0.1:8000/
```

En este momento, agregue un archivo *requirements.txt* y, a continuación, puede implementar la aplicación web en Azure o incluirla en contenedores con Docker y, a

continuación, implementarla.

Pasos siguientes

- Inicio rápido: Creación de una aplicación web de Python (Django o Flask) en Azure App Service.
 - Tutorial: Implementación de una aplicación web de Python (Django o Flask) con PostgreSQL en Azure
 - Creación e implementación de una aplicación web de Flask en Azure con una identidad administrada asignada por el sistema
-

Comentarios

¿Le ha resultado útil esta página?

 Sí

 No

Proporcionar comentarios sobre el producto  | Obtener ayuda en Microsoft Q&A

Información general sobre las plantillas azd web de Python

25/06/2025

Las plantillas de la CLI para desarrolladores de Azure web de Python (`azd`) son la manera más rápida y sencilla de compilar, configurar e implementar aplicaciones web de Python en Azure. En este artículo se proporciona información contextual en segundo plano para ayudarle a comprender los componentes implicados y cómo las plantillas simplifican la implementación.

La mejor manera de empezar es [seguir el inicio rápido](#) para crear su primera aplicación web de Python e implementarla en Azure en cuestión de minutos con `azd` plantillas. Si prefiere no configurar un entorno de desarrollo local, puede seguir el [inicio rápido mediante GitHub Codespaces](#) para obtener una experiencia totalmente basada en la nube con todas las herramientas preconfiguradas.

¿Cuáles son las plantillas de azd web de Python?

Las `azd` plantillas están diseñadas para desarrolladores web de Python experimentados que desean implementar aplicaciones escalables y listas para la nube en Azure con un tiempo de configuración mínimo.

Estas plantillas ofrecen el punto de partida más sencillo posible para compilar e implementar aplicaciones web de Python mediante:

- Configurar rápidamente un entorno de desarrollo y hospedaje local completo.
- Automatización de la creación de un entorno de implementación de Azure coincidente.
- Uso de un flujo de trabajo sencillo y memorable de la CLI.

Una vez configurados los entornos, las plantillas proporcionan la manera más rápida de empezar a compilar la aplicación web de Python. Ustedes pueden:

- Modifique los archivos de código proporcionados para que coincidan con los requisitos de la aplicación.
- Implemente actualizaciones con un esfuerzo mínimo mediante comandos azd.
- Amplíe la plantilla para que se ajuste a la arquitectura.

Estas plantillas reflejan patrones de diseño probados y procedimientos recomendados, lo que le permite:

- Construye con confianza sobre una sólida base arquitectónica.

- Siga las instrucciones desarrolladas por expertos del sector con experiencia profunda en Python y Azure.
- Asegúrese de la mantenibilidad, escalabilidad y seguridad desde el principio.

¿Qué tareas puedo hacer con las plantillas?

Al ejecutar una plantilla web `azd` de Python, se completan rápidamente varias tareas:

- **Cree una aplicación de inicio.** Crea un sitio web para una empresa ficticia denominada Relecloud. Este proyecto de inicio incluye:
 - Código bien organizado y listo para producción
 - Procedimientos recomendados para marcos web de Python (como Flask, Django).
 - Uso adecuado de las dependencias, la configuración y la estructura.

La plantilla está diseñada para ser un punto de partida: puede personalizar libremente la lógica y expandir o quitar recursos de Azure para ajustarse al proyecto.

- **Aprovisionar recursos de Azure.** Con [Bicep](#), un lenguaje moderno de infraestructura como código (IaC), la plantilla aprovisiona todos los recursos de Azure necesarios para:
 - Hospedar la aplicación web (como App Service, Container Apps)
 - Conexión a bases de datos (como PostgreSQL, Cosmos DB)

Los archivos de Bicep son totalmente editables: puede agregar o personalizar los servicios de Azure a medida que evoluciona la aplicación. De forma similar a la tarea anterior, puede [modificar las plantillas de Bicep](#) para agregar más servicios de Azure, según sea necesario.

- **Implemente la aplicación de inicio en los recursos de Azure aprovisionados.** Una vez aprovisionados los recursos, la aplicación se implementa automáticamente en el entorno de Azure. Ahora puede hacer lo siguiente:
 - Ve tu aplicación ejecutándose en la nube en tan solo unos minutos.
 - Pruebe su comportamiento.
 - Decida qué funcionalidad o configuración se va a actualizar a continuación.
- **(Opcional) Configure el repositorio de GitHub y la canalización de CI/CD.** Puede, si lo desea, inicializar un repositorio de GitHub con una [canalización de integración continua y entrega continua \(CI/CD\)](#) de GitHub Actions para:
 - Automatice las implementaciones en los cambios de código.
 - Colabore con miembros del equipo.
 - Envíe actualizaciones a Azure fusionando en la rama principal.

Esta integración le ayuda a adoptar los procedimientos recomendados de DevOps desde el principio.

¿Dónde puedo acceder a las plantillas?

Muchas azd plantillas están disponibles en la [galería de plantillas de la CLI para desarrolladores de Azure impresionantes](#). Estas plantillas proporcionan proyectos de aplicaciones web de Python listos para usar con paridad de características entre combinaciones populares de servicios de Azure y marcos web de Python.

Cada plantilla incluye lo siguiente:

- Una aplicación de ejemplo con código limpio y fácil de mantener.
- Infraestructura-como-código preconfigurada mediante Bicep.
- Flujos de trabajo de implementación sin problemas mediante la CLI para desarrolladores de Azure.
- Integración opcional de CI/CD a través de Acciones de GitHub

En las siguientes tablas se enumeran los monikers de plantilla web azd de Python que están disponibles para usar con el comando azd init. Las tablas identifican las tecnologías implementadas en cada plantilla y proporcionan un vínculo al repositorio de GitHub correspondiente, donde puede contribuir a los cambios.

Django

Las plantillas siguientes azd están disponibles para el [marco web de Django](#).

[\[+\] Expandir tabla](#)

Plantilla	Base de datos	Plataforma de hospedaje	Repositorio de GitHub
azure-django-postgres-flexible-aca	Servidor flexible de Azure Database for PostgreSQL	Azure Container Apps	https://github.com/Azure-Samples/azure-django-postgres-flexible-aca
azure-django-postgres-flexible-appservice	Servidor flexible de Azure Database for PostgreSQL	Azure App Service	https://github.com/Azure-Samples/azure-django-postgres-flexible-appservice
azure-django-cosmos-postgres-aca	Azure Cosmos DB para Azure Database for PostgreSQL	Azure Container Apps (Aplicaciones de Contenedores de Azure)	https://github.com/Azure-Samples/azure-django-cosmos-postgres-aca

Plantilla	Base de datos	Plataforma de hospedaje	Repositorio de GitHub
azure-django-cosmos-postgres-appservice	Azure Cosmos DB para Azure Database for PostgreSQL	Azure App Service	https://github.com/Azure-Samples/azure-django-cosmos-postgres-appservice ↗
azure-django-postgres-addon-aca	Azure Container Apps con Base de Datos de Azure para PostgreSQL	Azure Container Apps (Aplicaciones de Contenedores de Azure)	https://github.com/Azure-Samples/azure-django-postgres-addon-aca ↗

¿Cómo debo usar las plantillas?

Cada `azd` plantilla consta de un repositorio de GitHub que contiene el código de aplicación (código de Python que usa un marco web popular) y los archivos de infraestructura como código (es decir, [Bicep](#)) para crear los recursos de Azure. La plantilla también contiene la configuración necesaria para configurar un repositorio de GitHub con una canalización de CI/CD.

Entre los componentes clave de cada plantilla se incluyen:

- **Código de aplicación:** escrito en Python y creado con un marco web popular (como Flask, Django, FastAPI). En la aplicación de ejemplo se muestran los procedimientos recomendados de enrutamiento, acceso a datos y configuración.
- **Infraestructura como código (IaC):** se proporciona a través de archivos de Bicep para definir y aprovisionar los recursos de Azure necesarios, como:
 - Servicio de Aplicaciones o Container Apps
 - Bases de datos de Azure (como PostgreSQL, Cosmos DB)
 - Servicios de Azure AI, Almacenamiento y mucho más
- **Configuración de CI/CD (opcional):** incluye archivos para configurar un repositorio de GitHub con una canalización de CI/CD de Acciones de GitHub, lo que habilita:
 - Implementación automática en Azure en cada solicitud de inserción o extracción en la rama principal.
 - Integración sin problemas en el flujo de trabajo de DevOps

Estas plantillas son totalmente personalizables, lo que proporciona una base sólida para basarse y adaptarse a las necesidades específicas del proyecto.

Para realizar las tareas definidas por una `azd` plantilla web, use varios comandos de Python

`azd`. Para obtener descripciones detalladas de estos comandos, consulte [Inicio rápido](#):

[Implementación de una plantilla de la CLI para desarrolladores de Azure](#). El inicio rápido le guía por los pasos para usar una plantilla específica `azd`. Solo tiene que ejecutar cinco instrucciones de línea de comandos esenciales para el entorno de hospedaje de producción y el entorno de desarrollo local.

En la tabla siguiente se resumen los cinco comandos esenciales:

Expandir tabla

Comando	Descripción de la tarea
<code>azd init --template <template name></code>	Cree un nuevo proyecto a partir de una plantilla y cree una copia del código de aplicación en el equipo local. El comando le pide que proporcione un nombre de entorno (como "myapp") que se usa como prefijo en la nomenclatura de los recursos implementados.
<code>azd auth login</code>	Inicie sesión en Azure. El comando abre una ventana del explorador donde puede iniciar sesión en Azure. Después de iniciar sesión, se cierra la ventana del explorador y se completa el comando. El <code>azd auth login</code> comando solo es necesario la primera vez que use la CLI para desarrolladores de Azure (<code>azd</code>) por sesión.
<code>azd up</code>	Aprovisione los recursos en la nube e implemente la aplicación en esos recursos.
<code>azd deploy</code>	Implemente los cambios en el código fuente de la aplicación en los recursos ya aprovisionados por el <code>azd up</code> comando .
<code>azd down</code>	Elimine los recursos de Azure y la canalización de CI/CD, si fue utilizada.

Sugerencia

Cuando trabaje con comandos `azd`, esté atento a mensajes para especificar más información. Después de ejecutar el `azd up` comando, es posible que se le pida que seleccione una suscripción, si tiene más de una. También se le pedirá que especifique su región. Puede cambiar las respuestas a las solicitudes editando las variables de entorno almacenadas en la carpeta `./azure/` de la plantilla.

Después de completar las tareas esenciales proporcionadas por la `azd` plantilla, tiene una copia personal de la plantilla original donde puede modificar cualquier archivo, según sea necesario.

- **Código de aplicación:** personalice el código del proyecto de Python para implementar su propio diseño, rutas y lógica de negocios.

- **Infraestructura como código (Bicep)**: actualice los archivos de Bicep para aprovisionar servicios de Azure adicionales, cambiar configuraciones o quitar recursos innecesarios.

Este punto de partida flexible le permite crear sobre una base bien estructurada y adaptar la aplicación a su caso de uso real.

También puede [modificar la configuración de infraestructura como código](#) si necesita cambiar los recursos de Azure. Para obtener más información, consulte la sección [¿Qué puedo editar o eliminar?](#) más adelante en este artículo.

Tareas de plantillaopcionales

Además de los cinco comandos esenciales, hay tareas opcionales que puede completar con las `azd` plantillas.

Reaprovisionamiento y modificación de recursos de Azure

Después de aprovisionar recursos de Azure con una `azd` plantilla, puede modificar y volver a aprovisionar un recurso.

- Para modificar un recurso aprovisionado, [modifique los archivos de Bicep adecuados](#) en la plantilla.
- Para iniciar la tarea de reprovisionamiento, use el `azd provision` comando .

Configuración de canalizaciones de integración y entrega continuas

La CLI para desarrolladores de Azure (`azd`) proporciona una manera sencilla de configurar una canalización de CI/CD para la nueva aplicación web de Python. Al fusionar confirmaciones o solicitudes de incorporación de cambios en tu rama principal, la pipeline compila y publica automáticamente los cambios en sus recursos de Azure.

- Para configurar la canalización de CI/CD, designe el repositorio de GitHub y la configuración deseada para habilitar la canalización.
- Para crear la canalización, use el `azd pipeline config` comando .

Después de configurar la canalización, cada vez que se combinan los cambios de código en la rama *principal* del repositorio, la canalización implementa los cambios en los servicios de Azure aprovisionados.

Alternativas a las plantillas

Si prefiere no usar las plantillas web azd de Python, hay métodos alternativos para implementar aplicaciones web de Python en Azure y aprovisionar recursos de Azure.

Puede crear muchos recursos y completar los pasos de implementación mediante varias herramientas:

- [Azure Portal ↗](#)
- La [CLI de Azure](#)
- Visual Studio Code con la [extensión Azure Tools ↗](#)

También puede seguir un tutorial completo que incluya marcos de desarrollo web de Python:

- [Implementación de una aplicación web flask o FastAPI en Azure App Service](#)
- [Aplicación web de Python en contenedor en Azure con MongoDB](#)

Preguntas más frecuentes

En las secciones siguientes se resumen las respuestas a las preguntas más frecuentes sobre cómo trabajar con las plantillas web azd de Python.

¿Tengo que usar contenedores de desarrollo?

No. Las plantillas web azd de Python usan contenedores de desarrollo de [Visual Studio Code ↗](#) de forma predeterminada. Los contenedores de desarrollo proporcionan muchas ventajas, pero requieren algunos conocimientos y software previos. Si prefiere no usar contenedores de desarrollo y, en su lugar, use el entorno de desarrollo local, consulte el archivo *README.md* en el directorio raíz de la aplicación de ejemplo para obtener instrucciones de configuración del entorno.

¿Qué puedo editar o eliminar?

El contenido de cada plantilla web azd de Python puede variar según el tipo de proyecto y la pila de tecnología subyacente empleada. Las plantillas identificadas en este artículo siguen una convención común de carpeta y archivo, como se describe en la tabla siguiente.

⋮ Expandir tabla

Carpeta/archivos	Propósito	Descripción
/	Directorio raíz	La carpeta raíz de cada plantilla contiene muchos tipos diferentes de archivos y carpetas con distintos fines.

Carpeta/archivos	Propósito	Descripción
./celestes	azd archivos de configuración	La carpeta <code>.azure</code> se crea después de ejecutar el <code>azd init</code> comando. La carpeta almacena los archivos de configuración de las variables de entorno usadas por los <code>azd</code> comandos. Puede cambiar los valores de las variables de entorno para personalizar la aplicación y los recursos de Azure. Para obtener más información, consulte Archivo .env específico del entorno .
./devcontainer	Archivos de configuración del contenedor de desarrollo	Los contenedores de desarrollo permiten crear un entorno de desarrollo basado en contenedores completo con todos los recursos que necesita para el desarrollo de software dentro de Visual Studio Code. La carpeta <code>.devcontainer</code> se crea después de que Visual Studio Code genere un archivo de configuración de contenedor de desarrollo en respuesta a un comando de plantilla.
./github	Archivos de configuración de Acciones de GitHub	Esta carpeta contiene ajustes de configuración para la pipeline opcional de CI/CD de GitHub Actions, linting y pruebas. Si no desea configurar la canalización de Acciones de GitHub mediante <code>azd pipeline config</code> el comando , puede modificar o eliminar el archivo <code>azure-dev.yaml</code> .
/infra	Archivos de Bicep	La carpeta <code>infra</code> contiene los archivos de configuración de Bicep. Bicep permite declarar los recursos de Azure que desea implementar en el entorno. Solo debe modificar los archivos <code>main.bicep</code> y <code>web.bicep</code> . Para obtener más información, consulte Inicio rápido: Escalado de servicios implementados con las plantillas web de Python azd mediante Bicep .
/src	Archivos de código del proyecto de inicio	La carpeta <code>src</code> contiene varios archivos de código necesarios para preparar el proyecto de inicio. Algunos ejemplos de los archivos incluyen plantillas requeridas por el marco web, archivos estáticos, archivos de Python (.py) para la lógica de código y los modelos de datos, un archivo <code>derequirements.txt</code> , etc. Los archivos específicos dependen del marco web, del marco de acceso a datos, etc. Puede modificar estos archivos para que se adapten a los requisitos del proyecto.
./cruft.json	Archivo de generación de plantillas	El archivo JSON <code>.cruft</code> se usa internamente para generar las plantillas web <code>azd</code> de Python. Puede eliminar este archivo de forma segura, según sea necesario.
./gitattributes	Archivo con configuración de atributos para Git	Este archivo proporciona a Git opciones de configuración importantes para controlar archivos y carpetas. Puede modificar este archivo, según sea necesario.
./gitignore	Archivo con elementos	El archivo <code>.gitignore</code> informa a Git sobre los archivos y carpetas que se van a excluir (omitar) al escribir en el repositorio de

Carpeta/archivos	Propósito	Descripción
	ignorados para Git	GitHub para la plantilla. Puede modificar este archivo, según sea necesario.
/azure.yaml	azd up archivo de configuración	Este archivo de configuración contiene los valores de configuración del azd up comando . Especifica los servicios y las carpetas de proyecto que se van a implementar. Importante: Este archivo no debe eliminarse.
/*.md	Archivos de formato Markdown	Una plantilla puede incluir varios archivos de formato Markdown (.md) para diferentes propósitos. Puede eliminar archivos Markdown de forma segura.
/docker-compose.yml	Configuración de Docker Compose	Este archivo YML crea el paquete de contenedor para la aplicación web de Python antes de que la aplicación se implemente en Azure.
/pyproject.toml	Archivo de configuración de compilación de Python	El archivo TOML contiene los requisitos del sistema de compilación de proyectos de Python. Puede modificar este archivo para identificar sus preferencias de herramienta, como un linter específico o un marco de pruebas unitarias.
/requirements-dev.in	archivo de requisitos de pip	Este archivo se usa para crear una versión del entorno de desarrollo de los requisitos mediante el pip install -r comando . Puede modificar este archivo para incluir otros paquetes, según sea necesario.

Sugerencia

A medida que modifique los archivos de plantilla para el programa, asegúrese de practicar un buen control de versiones. Este enfoque puede ayudarle a restaurar el repositorio a una versión de trabajo anterior, si los nuevos cambios provocan problemas de programa.

¿Cómo puedo controlar los errores de plantilla?

Si recibe un error al usar una azd plantilla, revise las opciones descritas en el artículo Solución de problemas de la [CLI para desarrolladores de Azure](#) . También puede notificar problemas en el repositorio de GitHub asociado a la azd plantilla.

Contenido relacionado

- [Creación e implementación de aplicaciones web de Python en Azure con plantillas azd](#)
- [Creación e implementación de aplicaciones web de Python desde GitHub Codespaces en Azure con plantillas azd](#)

Inicio rápido: Creación e implementación de una aplicación web de Python en Azure mediante una plantilla azd

21/06/2025

Este inicio rápido le guía a través de la manera más sencilla y rápida de crear e implementar una solución web y de base de datos de Python en Azure. Siguiendo las instrucciones de este inicio rápido, hará lo siguiente:

- Elija una azd plantilla basada en el marco web de Python, la plataforma de base de datos de Azure y la plataforma de hospedaje web de Azure en la que quiera compilar.
- Use comandos de la CLI para ejecutar una azd plantilla para crear una base de datos y una aplicación web de ejemplo, y crear y configurar los recursos de Azure necesarios y, a continuación, implementar la aplicación web de ejemplo en Azure.
- Edite la aplicación web en el equipo local y use un azd comando para volver a implementar.
- Use un azd comando para limpiar los recursos de Azure.

Este tutorial tardará menos de 15 minutos en completarse. Al finalizar, puede empezar a modificar el nuevo proyecto con el código personalizado.

Para más información sobre estas azd plantillas para el desarrollo de aplicaciones web de Python:

- [¿Cuáles son estas plantillas?](#)
- [¿Cómo funcionan las plantillas?](#)
- [¿Por qué querría hacer esto?](#)
- [¿Cuáles son mis otras opciones?](#)

Prerrequisitos

Una suscripción de Azure: [cree una gratuitamente.](#) ↗

Debe tener instalado lo siguiente en el equipo local:

- [CLI de desarrollo de Azure](#)
- [Docker Desktop](#) ↗
- [Visual Studio Code](#) ↗
- [Extensión del contenedor de desarrollo](#) ↗

Elegir una plantilla

Elija una `azd` plantilla basada en el marco web de Python, la plataforma de hospedaje web de Azure y la plataforma de base de datos de Azure en la que quiera compilar.

1. Seleccione un nombre de plantilla (primera columna) en la siguiente lista de plantillas de las tablas siguientes. Usará el nombre de la plantilla en el paso `azd init` de la sección siguiente.

Django

[\[+\] Expandir tabla](#)

Plantilla	Marco web	Base de datos	Plataforma de hospedaje	Repositorio de GitHub
azure-django-postgres-flexible-aca	Django	Servidor flexible de PostgreSQL	Azure Container Apps (Aplicaciones de Contenedores de Azure)	Repo ↗
azure-django-postgres-flexible-appservice	Django	Servidor flexible de PostgreSQL	Azure App Service	Repo ↗
azure-django-cosmos-postgres-aca	Django	Cosmos DB (adaptador de PostgreSQL)	Azure Container Apps (Aplicaciones de Contenedores de Azure)	Repo ↗
azure-django-cosmos-postgres-appservice	Django	Cosmos DB (adaptador de PostgreSQL)	Azure App Service	Repo ↗
azure-django-postgres-addon-aca	Django	Complemento de PostgreSQL de Azure Container Apps	Azure Container Apps (Aplicaciones de Contenedores de Azure)	Repo ↗

El repositorio de GitHub (última columna) solo se proporciona con fines de referencia. Solo debe clonar el repositorio directamente si desea contribuir a los cambios en la plantilla. De lo

contrario, siga las instrucciones de este inicio rápido para usar la `azd` CLI para interactuar con la plantilla en un flujo de trabajo normal.

Ejecución de la plantilla

La ejecución de una `azd` plantilla es la misma en todos los lenguajes y marcos de trabajo. Además, los mismos pasos básicos se aplican a todas las plantillas. Los pasos son:

1. En un terminal, vaya a una carpeta en el equipo local donde normalmente almacena los repositorios git locales y, a continuación, cree una carpeta denominada `azdtest`. A continuación, cambie a ese directorio mediante el `cd` comando .

```
shell
mkdir azdtest
cd azdtest
```

No use el terminal de Visual Studio Code para este inicio rápido.

2. Para configurar el entorno de desarrollo local, escriba los siguientes comandos en el terminal y responda a las indicaciones:

```
shell
azd init --template <template name>
```

Sustituya `<template name>` por una de las plantillas de las [tablas](#) que seleccionó en un paso anterior, como `azure-django-postgres-aca`, por ejemplo.

Cuando se le solicite un nombre de entorno, use `azdtest` o cualquier otro nombre. El nombre del entorno se usa al asignar nombres a los grupos de recursos y recursos de Azure. Para obtener los mejores resultados, utilice un nombre corto, en minúsculas y sin caracteres especiales.

3. Para autenticarse `azd` en su cuenta de Azure, escriba los siguientes comandos en su terminal y siga las indicaciones:

```
shell
azd auth login
```

Siga las instrucciones cuando se le pida que "Elija una cuenta" o inicie sesión en su cuenta de Azure. Una vez que se haya autenticado correctamente, se muestra el siguiente

mensaje en una página web: "Se ha completado la autenticación. Puede volver a la aplicación. No dude en cerrar esta pestaña del navegador".

Al cerrar la pestaña, el shell muestra el mensaje:

Resultados

Logged in to Azure.

4. Asegúrese de que Docker Desktop está abierto y ejecutándose en segundo plano antes de intentar el paso siguiente.

5. Para crear los recursos de Azure necesarios, escriba los siguientes comandos en el terminal y responda a las indicaciones:

shell

azd up

Importante

Una vez `azd up` completada correctamente, la aplicación web de ejemplo estará disponible en la red pública de Internet y la suscripción de Azure comenzará a acumular cargos por todos los recursos que se crean. Los creadores de las `azd` plantillas eligieron intencionadamente niveles económicos, pero no necesariamente niveles *gratuitos*, ya que los niveles gratuitos a menudo tienen disponibilidad restringida.

Siga las instrucciones cuando se le pida que elija La suscripción de Azure que se usará para el pago y, a continuación, seleccione una ubicación de Azure que se va a usar. Elija una región cercana geográficamente.

`azd up` La ejecución puede tardar varios minutos, ya que se aprovisiona e implementa varios servicios de Azure. A medida que se muestra el progreso, observe si hay errores. Si ve errores, pruebe lo siguiente para corregir el problema:

- Elimine la carpeta `azd-quickstart` y las instrucciones de inicio rápido desde el principio.
- Cuando se le solicite, elija un nombre más sencillo para su entorno. Use solo letras minúsculas y guiones. No hay números, letras mayúsculas ni caracteres especiales.
- Elija otra ubicación.

Si sigue teniendo problemas, consulte la sección [Solución de problemas](#) en la parte inferior de este documento.

Importante

Una vez que haya terminado de trabajar con la aplicación web de ejemplo, use `azd down` para quitar todos los servicios creados por `azd up`.

6. Cuando `azd up` se completa correctamente, se muestra la siguiente salida:

```
Deploying services (azd deploy)

|== |          Deploying service web (Fetching endpoints for container a
(✓) Done: Deploying service web
- Endpoint: https://azdtest-cpz2yvly5xa-ca.delightfulflower-54a525cc.eastus2.azurecontainerapps.io/

SUCCESS: Your application was provisioned and deployed to Azure in 10 minutes 45 seconds.
You can view the resources created under the resource group azdtest-rg in Azure Portal:
https://portal.azure.com/#@/resource/subscriptions/aaaa0a0a-bb1b-cc2c-dd3d-e
eeeeee4e4e/resourceGroups/azdtest-rg/overview

c:\source\azdtest>
```

Copie la primera dirección URL después de la palabra `- Endpoint:` y péguela en la barra de ubicación de un explorador web para ver el proyecto de aplicación web de ejemplo que se ejecuta en vivo en Azure.

7. Abra una nueva pestaña en el explorador web, copie la segunda dirección URL del paso anterior y péguela en la barra de ubicación. Azure Portal muestra todos los servicios del nuevo grupo de recursos que se han implementado para hospedar el proyecto de aplicación web de ejemplo.

Edición y reimplementación

El siguiente paso consiste en realizar un pequeño cambio en la aplicación web y, a continuación, volver a implementar.

1. Abra Visual Studio Code y abra la carpeta `azdtest` creada anteriormente.
2. Esta plantilla está configurada para usar opcionalmente Contenedores de desarrollo. Cuando vea que aparece la notificación de Dev Container en Visual Studio Code, seleccione el botón "Reabrir en Contenedor".

3. Use la vista explorador de Visual Studio Code para ir a *la carpeta src/templates* y abrir el archivo *index.html* . Busque la línea de código siguiente:

```
HTML
```

```
<h1 id="pagte-title">Welcome to ReleCloud</h1>
```

Cambie el texto dentro del H1:

```
HTML
```

```
<h1 id="pagte-title">Welcome to ReleCloud - UPDATED</h1>
```

Guarde los cambios.

4. Para volver a implementar la aplicación con el cambio, en el terminal, ejecute el siguiente comando:

```
Shell
```

```
azd deploy
```

Dado que está utilizando contenedores de desarrollo y se conecta de forma remota al shell del contenedor, no use la ventana Terminal de Visual Studio Code para ejecutar comandos `azd`.

5. Una vez completado el comando, actualice el explorador web para ver la actualización. Dependiendo de la plataforma de hospedaje web que se use, puede tardar varios minutos antes de que los cambios sean visibles.

Ya está listo para editar y eliminar archivos en la plantilla. Para obtener más información, consulte [¿Qué puedo editar o eliminar en la plantilla?](#)

Limpieza de recursos

1. Limpie los recursos creados por la plantilla ejecutando el `azd down` comando .

```
Shell
```

```
azd down
```

El comando `azd down` elimina los recursos de Azure y el flujo de trabajo de Acciones de GitHub. Cuando se le solicite, acepte eliminar todos los recursos asociados al grupo de

recursos.

También puede eliminar la carpeta `azdtest` o usarla como base para su propia aplicación modificando los archivos del proyecto.

Solución de problemas

Si ve errores durante `azd up`, pruebe los pasos siguientes:

- Ejecute `azd down` para quitar los recursos que se puedan haber creado. Como alternativa, puede eliminar el grupo de recursos que se creó en Azure Portal.
- Elimine la carpeta `azdtest` en el equipo local.
- En Azure Portal, busque Almacenes de claves. Seleccione *Manage deleted vaults* (Administrar almacenes eliminados), elija la suscripción, seleccione todos los almacenes de claves que contengan el nombre `azdtest` o el nombre que haya asignado a su entorno y seleccione *Purgar*.
- Repita los pasos de esta guía de inicio rápido. Esta vez, cuando se le solicite, elija un nombre más sencillo para su entorno. Pruebe un nombre corto, letras minúsculas, sin números, sin letras mayúsculas, sin caracteres especiales.
- Al reintentar los pasos de inicio rápido, elija otra ubicación.

Consulte las [preguntas más frecuentes](#) para obtener una lista más completa de posibles problemas y soluciones.

Contenido relacionado

- [Más información sobre las plantillas de azd web de Python](#)
- [Más información sobre los comandos de azd.](#)
- Obtenga información sobre qué hacen cada una de las carpetas y los archivos del proyecto y [¿qué puede editar o eliminar?](#)
- [Obtenga más información sobre Los contenedores de desarrollo](#).
- [Actualice las plantillas de Bicep para agregar o quitar servicios de Azure.](#) ¿No conoces a Bicep? Pruebe esta ruta de aprendizaje : [Aspectos básicos de Bicep](#)
- [Usa azd para configurar una pipeline de CI/CD de GitHub Actions para implementar nuevamente al fusionar con la rama principal](#)
- Configuración de la supervisión para que pueda [Supervisar la aplicación mediante la CLI para desarrolladores de Azure](#)

Inicio rápido: Creación e implementación de una aplicación web de Python desde GitHub Codespaces en Azure mediante una plantilla de la CLI para desarrolladores de Azure

20/06/2025

Este inicio rápido le guía a través de la manera más sencilla y rápida de crear e implementar una solución web y de base de datos de Python en Azure. Siguiendo las instrucciones de este inicio rápido, hará lo siguiente:

- Elija una plantilla de la [CLI para desarrolladores de Azure \(azd\)](#) basada en el marco web de Python, la plataforma de base de datos de Azure y la plataforma de hospedaje web de Azure en la que quiera compilar.
- Cree un nuevo Codespace de GitHub que contenga el código generado a partir de la plantilla `azd` que seleccionaste.
- Utiliza GitHub Codespaces y la terminal bash del Visual Studio Code en línea. El terminal permite usar comandos de la CLI para desarrolladores de Azure para ejecutar una `azd` plantilla para crear una base de datos y una aplicación web de ejemplo, y crear y configurar los recursos de Azure necesarios y, a continuación, implementar la aplicación web de ejemplo en Azure.
- Edite la aplicación web en un espacio de código de GitHub y use un `azd` comando para volver a implementar.
- Use un `azd` comando para limpiar los recursos de Azure.
- Cierre y vuelva a abrir GitHub Codespace.
- Publique el nuevo código en un repositorio de GitHub.

Este tutorial tardará menos de 25 minutos en completarse. Al finalizar, puede empezar a modificar el nuevo proyecto con el código personalizado.

Para más información sobre estas `azd` plantillas para el desarrollo de aplicaciones web de Python:

- [¿Cuáles son estas plantillas?](#)
- [¿Cómo funcionan las plantillas?](#)
- [¿Por qué querría hacer esto?](#)
- [¿Cuáles son mis otras opciones?](#)

Prerrequisitos

- Una suscripción de Azure: [cree una gratuitamente](#) ↗
- Una cuenta de GitHub: [crear una gratuitamente](#) ↗

ⓘ Importante

Tanto GitHub Codespaces como Azure son servicios basados en suscripciones de pago. Después de algunas asignaciones gratuitas, puede que se le cobre por usar estos servicios. Seguir este inicio rápido podría afectar a estas asignaciones o a la facturación. Cuando es posible, las azd plantillas se crearon con el nivel de opciones menos costoso, pero es posible que algunas no sean gratuitas. Use la [calculadora de precios de Azure](#) ↗ para comprender mejor los costos. Para más información, consulte [Precios de GitHub Codespaces](#) ↗ para obtener más información.

Elección de una plantilla y creación de un espacio de código

Elija una azd plantilla basada en el marco web de Python, la plataforma de hospedaje web de Azure y la plataforma de base de datos de Azure en la que quiera compilar.

1. En la siguiente lista de plantillas, elija una que use las tecnologías que desea usar en la nueva aplicación web.

Django

↔ Expandir tabla

Plantilla	Marco web	Base de datos	Plataforma de hospedaje	Nuevo espacio de código
azure-django-postgres-flexible-aca	Django	Servidor flexible de PostgreSQL	Azure Container Apps (Aplicaciones de Contenedores de Azure)	Nuevo espacio de código ↗
azure-django-postgres-flexible-appservice	Django	Servidor flexible de PostgreSQL	Azure App Service	Nuevo espacio de código ↗

Plantilla	Marco web	Base de datos	Plataforma de hospedaje	Nuevo espacio de código
azure-django-cosmos-postgres-aca	Django	Cosmos DB (adaptador de PostgreSQL)	Azure Container Apps (Aplicaciones de Contenedores de Azure)	Nuevo espacio de código ↗
azure-django-cosmos-postgres-appservice	Django	Cosmos DB (adaptador de PostgreSQL)	Azure App Service	Nuevo espacio de código ↗
azure-django-postgres-addon-aca	Django	Complemento de PostgreSQL de Azure Container Apps	Azure Container Apps (Aplicaciones de Contenedores de Azure)	Nuevo espacio de código ↗

2. Para mayor comodidad, la última columna de cada tabla contiene un vínculo que crea un nuevo Codespace e inicializa la plantilla en la `azd` cuenta de GitHub. Haga clic con el botón derecho y seleccione "Abrir en nueva pestaña" en el vínculo "Nuevo espacio de código" junto al nombre de plantilla que seleccionó para iniciar el proceso de instalación.

Durante este proceso, es posible que se le pida que inicie sesión en su cuenta de GitHub y se le pedirá que confirme que desea crear codespace. Seleccione el botón "Crear espacio de código" para ver la página "Configurar el espacio de código".

3. Después de unos minutos, se carga una versión basada en web de Visual Studio Code en una nueva pestaña del explorador con la plantilla web de Python cargada como un área de trabajo en la vista Explorador.

Autenticación en Azure e implementación de la plantilla azd

Ahora que tiene un espacio de código de GitHub que contiene el código recién generado, use la `azd` utilidad desde codespace para publicar el código en Azure.

1. En Visual Studio Code basado en web, el terminal debe estar abierto de forma predeterminada. Si no es así, use la tecla tilde `~` para abrir el terminal. Además, de forma predeterminada, el terminal debe ser un terminal bash. Si no es así, cambie a Bash en el área superior derecha de la ventana de terminal.

2. En el terminal de Bash, escriba el siguiente comando:

Bash

```
azd auth login
```

`azd auth login` comienza el proceso de autenticación de Codespace en su cuenta de Azure.

Resultados

```
Start by copying the next code: XXXXXXXX  
Then press enter and continue to log in from your browser...  
  
Waiting for you to complete authentication in the browser...
```

3. Siga las instrucciones, entre las que se incluyen:

- Copia de un código generado
- Selección de entrar para abrir una nueva pestaña del explorador y pegar el código en el cuadro de texto
- Elección de la cuenta de Azure en una lista
- Confirmación de que está intentando iniciar sesión en la CLI de Microsoft Azure

4. Cuando se ejecuta correctamente, el mensaje siguiente se muestra de nuevo en la pestaña Codespaces del terminal:

Resultados

```
Device code authentication completed.  
Logged in to Azure.
```

5. Implemente la nueva aplicación en Azure escribiendo el siguiente comando:

Bash

```
azd up
```

Durante este proceso, se le pide que:

- Escribir un nuevo nombre de entorno
- Seleccione una suscripción de Azure para usar [Usar flechas para mover, escriba para filtrar]

- Seleccione una ubicación de Azure que se va a usar: [Use flechas para mover, escriba para filtrar]

Una vez que responda a esas preguntas, la salida de `azd` indica que la implementación está progresando.

ⓘ Importante

Una vez `azd up` completada correctamente, la aplicación web de ejemplo estará disponible en la red pública de Internet y la suscripción de Azure comenzará a acumular cargos por todos los recursos que se crean. Los creadores de las `azd` plantillas eligieron intencionadamente niveles económicos, pero no necesariamente niveles *gratuitos*, ya que los niveles gratuitos a menudo tienen disponibilidad restringida. Una vez que haya terminado de trabajar con la aplicación web de ejemplo, use `azd down` para quitar todos los servicios creados por `azd up`.

Siga las instrucciones cuando se le pida que elija La suscripción de Azure que se usará para el pago y, a continuación, seleccione una ubicación de Azure que se va a usar. Elija una región cercana geográficamente.

`azd up` La ejecución puede tardar varios minutos, ya que se aprovisiona e implementa varios servicios de Azure. A medida que se muestra el progreso, observe si hay errores. Si ve errores, consulte la sección [Solución de problemas](#) en la parte inferior de este documento.

6. Cuando `azd up` se completa correctamente, se muestra una salida similar:

Resultados

```
(✓) Done: Deploying service web
- Endpoint: https://xxxxx-xxxxxxxxxxxx-ca.example-
xxxxxxxx.azurecontainerapps.io/
```

```
SUCCESS: Your application was provisioned and deployed to Azure in 11 minutes
44 seconds.
```

```
You can view the resources created under the resource group xxxx-rg in Azure
Portal:
```

```
https://portal.azure.com/#@/resource/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxx/resourceGroups/xxxxx-rg/overview
```

Si ve una pantalla predeterminada o una pantalla de error, es posible que la aplicación se inicie. Espere entre 5 y 10 minutos para ver si el problema se resuelve antes de solucionarlo.

Ctrl + haga clic en la primera dirección URL después de la palabra `- Endpoint:` para ver el proyecto de aplicación web de ejemplo que se ejecuta en vivo en Azure.

7. Ctrl + haga clic en la segunda dirección URL del paso anterior para ver los recursos aprovisionados en Azure Portal.

Edición y reimplementación

El siguiente paso consiste en realizar un pequeño cambio en la aplicación web y, a continuación, volver a implementar.

1. Vuelva a la pestaña del explorador que contiene Visual Studio Code y use la vista Explorador de Visual Studio Code para ir a *la carpeta src/templates* y abrir el archivo *index.html*. Busque la línea de código siguiente:

```
HTML
```

```
<h1 id="page-title">Welcome to ReleCloud</h1>
```

Cambie el texto dentro del H1:

```
HTML
```

```
<h1 id="page-title">Welcome to ReleCloud - UPDATED</h1>
```

El código se guarda mientras escribe.

2. Para volver a implementar la aplicación con el cambio, ejecute el siguiente comando en el terminal:

```
Bash
```

```
azd deploy
```

3. Una vez completado el comando, actualice la pestaña del explorador con el sitio web de ReleCloud para ver la actualización. Dependiendo de la plataforma de hospedaje web que se use, puede tardar varios minutos antes de que los cambios sean visibles.

Ya está listo para editar y eliminar archivos en la plantilla. Para obtener más información, consulte [¿Qué puedo editar o eliminar en la plantilla?](#)

Limpieza de recursos

Limpie los recursos creados por la plantilla ejecutando el comando `azd down`.

```
Bash
```

```
azd down
```

El comando `azd down` elimina los recursos de Azure y el flujo de trabajo de Acciones de GitHub. Cuando se le solicite, acepte eliminar todos los recursos asociados al grupo de recursos.

Opcional: busque el codespace.

En esta sección se muestra cómo tu código se ejecuta (temporalmente) y se conserva a corto plazo en un Codespace. Si planea seguir trabajando en el código, debe publicar el código en un nuevo repositorio.

1. Cierre todas las pestañas relacionadas con este artículo de inicio rápido o cierre completamente el explorador web.
2. Abra el explorador web y una nueva pestaña y vaya a: <https://github.com/codespaces>
3. Cerca de la parte inferior, aparece una lista con los codespaces recientes. Busca el que creaste en una sección denominada "Propiedad de Azure-Samples".
4. Seleccione los puntos suspensivos situados a la derecha de este codespace para abrir un menú contextual. Desde aquí puede cambiar el nombre del espacio de código, publicar en un nuevo repositorio, cambiar el tipo de máquina, detener el espacio de código, etc.

Opcional: Publicación de un repositorio de GitHub desde Codespaces

En este momento, tiene un codespace, que es un contenedor alojado por GitHub en el que se ejecuta su entorno de desarrollo de Visual Studio Code con el código nuevo generado a partir de una plantilla de `azd`. Sin embargo, el código no se almacena en un repositorio de GitHub. Si planea seguir trabajando en el código, debe hacer que sea una prioridad.

1. En el menú contextual del espacio de código, seleccione "Publicar en un nuevo repositorio".
2. En el cuadro de diálogo "Publicar en un nuevo repositorio", cambie el nombre del nuevo repositorio y elija si desea que sea un repositorio público o privado. Seleccione "Crear repositorio".

3. Despu s de unos instantes, se cre r el repositorio y el c digo que gener  anteriormente en este inicio r pido se insertar  en el nuevo repositorio. Seleccione el bot n "Ver repositorio" para ir al nuevo repositorio.
4. Para volver a abrir y continuar con la edici n de c digo, seleccione la lista desplegable verde "<> C digo", cambie a la pesta a Codespaces y seleccione el nombre del espacio de c digo en el que estaba trabajando anteriormente. Ahora debe volver al entorno de desarrollo de Visual Studio Codespace.
5. Utilice el panel Control de c digo fuente para crear nuevas ramas, almacenar de forma provisional los cambios en el c digo y confirmarlos.

Soluci n de problemas

Si ve errores durante `azd up`, pruebe lo siguiente:

- Ejecute `azd down` para quitar los recursos que se puedan haber creado. Como alternativa, puede eliminar el grupo de recursos que se cre  en Azure Portal.
- Vaya a la p gina Codespaces de su cuenta de GitHub, busque el Codespaces creado durante este Quickstart, haga clic en el icono de tres puntos a la derecha y elija "Eliminar" en el men  contextual.
- En Azure Portal, busque Almacenes de claves. Seleccione *Manage deleted vaults* (Administrar almacenes eliminados), elija la suscripc n, seleccione todos los almacenes de claves que contengan el nombre `azdtest` o el nombre que haya asignado a su entorno y seleccione *Purgar*.
- Repita los pasos de esta gu a de inicio r pido. Esta vez, cuando se le solicite, elija un nombre m s sencillo para su entorno. Pruebe un nombre corto, letras min sculas, sin n meros, sin letras may sculas, sin caracteres especiales.
- Al reintentar los pasos de inicio r pido, elija otra ubicaci n.

Consulte las [preguntas m s frecuentes](#) para obtener una lista m s completa de posibles problemas y soluciones.

Contenido relacionado

- [M s informaci n sobre las plantillas de azd web de Python](#)
- [M s informaci n sobre los comandos de azd.](#)
- Obtenga informaci n sobre qu  hacen cada una de las carpetas y los archivos del proyecto y [qu  puede editar o eliminar?](#)
- [M s informaci n sobre GitHub Codespaces ↗](#)
- [Actualice las plantillas de Bicep para agregar o quitar servicios de Azure.](#) ¿No conoces a Bicep? Pruebe esta ruta de aprendizaje : [Aspectos b sicos de Bicep](#)

- Usa azd para configurar una pipeline de CI/CD de GitHub Actions para implementar nuevamente al fusionar con la rama principal
- Configuración de la supervisión para que pueda [Supervisar la aplicación mediante la CLI para desarrolladores de Azure](#)

Inicio rápido: Escalado de servicios implementados con las plantillas web de Python azd mediante Bicep

Artículo • 27/03/2025

Las plantillas de `azd` web de Python permiten crear rápidamente una nueva aplicación web e implementarla en Azure. Las plantillas de `azd` se diseñaron para usar opciones de servicio de Azure de bajo costo. Sin duda, querrá ajustar los niveles de servicio (o sku) para cada uno de los servicios definidos en la plantilla para su escenario.

En este inicio rápido, actualizará los archivos de plantilla de Bicep correspondientes para escalar los servicios existentes y agregar nuevos servicios a la implementación. A continuación, ejecutará el comando `azd provision` y verá el cambio realizado en la implementación de Azure.

Prerrequisitos

Una suscripción de Azure: [Crear una gratis](#)

Debe tener instalado lo siguiente en el equipo local:

- [CLI de desarrollo de Azure](#)
- [Docker Desktop](#)
- [Visual Studio Code](#)
- [Extensión del contenedor de desarrollo](#)
- [Bicep de Visual Studio Code](#) Esta extensión le ayuda a crear la sintaxis de Bicep.

Implementación de una plantilla

Para empezar, necesita una implementación de `azd` en funcionamiento. Una vez que lo tenga en su lugar, podrá modificar los archivos de Bicep generados por la plantilla de `azd`.

1. Siga los pasos del 1 al 7 en el artículo [de la guía de inicio rápido](#). En el paso 2, use la plantilla `azure-django-postgres-flexible-appservice`. Para mayor comodidad, esta es la secuencia completa de comandos que se van a emitir desde la línea de comandos:

shell

```
mkdir azdtest
cd azdtest
azd init --template azure-django-postgres-flexible-appservice
azd auth login
azd up
```

Una vez que finalice `azd up`, abra el portal de Azure, vaya al Azure App Service que fue implementado en su nuevo grupo de recursos y anote el plan de precios del App Service (consulte la página Información general del plan de App Service, sección 'Esenciales', valor "Plan de precios").

2. En el paso 1 del artículo de inicio rápido, se le indicó que creara la carpeta `azdtest`. Abra esa carpeta en Visual Studio Code.
3. En el panel Explorador, vaya a la carpeta `infra`. Observe las subcarpetas y los archivos de la carpeta `infra`.

El archivo `main.bicep` organiza la creación de todos los servicios implementados al realizar una `azd up` o `azd provision`. Llama a otros archivos, como `db.bicep` y `web.bicep`, que a su vez llaman a archivos contenidos en la subcarpeta `\core` .

La subcarpeta `\core` es una estructura de carpetas profundamente anidada que contiene plantillas Bicep para muchos servicios de Azure. Algunos de los archivos de la subcarpeta `\core` hacen referencia a ellos los tres archivos bicep de nivel superior (`main.bicep`, `db.bicep` y `web.bicep`) y algunos no se usan en este proyecto.

Escalar un servicio modificando sus propiedades de Bicep

Puede escalar un recurso existente en la implementación cambiando su SKU. Para demostrarlo, cambiará el plan de App Service del "plan de servicio básico" (que está diseñado para aplicaciones con requisitos de tráfico más bajos y no necesita características avanzadas de escalado automático y administración de tráfico) al "plan de servicio estándar", que está diseñado para ejecutar cargas de trabajo de producción.

ⓘ Nota

No todos los cambios de SKU se pueden realizar después del hecho. Algunas investigaciones pueden ser necesarias para comprender mejor las opciones de escalado.

1. Abra el archivo `web.bicep` de y busque la definición del módulo `appService`. En concreto, busque la configuración de la propiedad:

```
Bicep

sku: {
    name: 'B1'
}
```

Cambie el valor de `B1` a `S1` de la siguiente manera:

```
Bicep

sku: {
    name: 'S1'
}
```

 **Importante**

Como resultado de este cambio, el precio por hora aumentará ligeramente. Puede encontrar detalles sobre los diferentes planes de servicio y sus costos asociados en la página de precios de [App Service](#).

2. Suponiendo que ya tiene la aplicación implementada en Azure, use el siguiente comando para implementar los cambios en la infraestructura sin volver a implementar el propio código de la aplicación.

```
shell

azd provision
```

No se le debe solicitar una ubicación ni una suscripción. Esos valores se guardan en el archivo `.azure<environment-name>.env` donde `<environment-name>` es el nombre de entorno que proporcionó durante `azd init`.

3. Cuando se complete `azd provision`, confirme que la aplicación web sigue funcionando. Busque también el plan de App Service para el grupo de recursos y confirme que el plan de precios está establecido en el plan de servicio estándar (`S1`).

Esto concluye el inicio rápido, pero hay muchos servicios de Azure que pueden ayudarle a crear aplicaciones más escalables y listas para producción. Un excelente punto de

partida sería obtener información sobre [Azure API Management](#), [Azure Front Door](#), [Azure CDN](#)y [Azure Virtual Network](#), por nombrar algunos.

Limpieza de recursos

Limpie los recursos creados por la plantilla ejecutando el comando `azd down`.

shell

`azd down`

El comando `azd down` elimina los recursos de Azure y el flujo de trabajo de Acciones de GitHub. Cuando se le solicite, acepte eliminar todos los recursos asociados al grupo de recursos.

También puedes eliminar la carpeta `azdtest`, o usarla como base para tu propia aplicación modificando los archivos del proyecto.

Contenido relacionado

- [Más información sobre las plantillas de azd web de Python](#)
- [Más información sobre los comandos de azd.](#)
- Obtenga información sobre qué hacen cada una de las carpetas y los archivos del proyecto y [¿qué puede editar o eliminar?](#)
- Actualice las plantillas de Bicep para agregar o quitar servicios de Azure. ¿No conoces a Bicep? Pruebe esta ruta de aprendizaje : [Aspectos básicos de Bicep](#)
- [Usa azd para configurar una pipeline de CI/CD de GitHub Actions para implementar nuevamente al fusionar con la rama principal](#)
- Configuración de la supervisión para que pueda [Supervisar la aplicación mediante la CLI para desarrolladores de Azure](#)

Comentarios

¿Le ha resultado útil esta página?

 Sí

 No

[Proporcionar comentarios sobre el producto ↗](#) | [Obtener ayuda en Microsoft Q&A](#)

Inicio rápido: Implementación de una aplicación web de Python (Django, Flask o FastAPI) en Azure App Service

24/04/2025

En este inicio rápido, implementará una aplicación web de Python (Django, Flask o FastAPI) para [Azure App Service](#). Azure App Service es un servicio de hospedaje web totalmente administrado, que admite aplicaciones de Python hospedadas en un entorno de servidor Linux.

Para completar este inicio rápido necesita instalar:

- Una cuenta de Azure con una suscripción activa. [Cree una cuenta gratuita ↗](#).
- [Python 3.9 o posterior ↗](#) instalado localmente.

ⓘ Nota

Este artículo contiene instrucciones actuales sobre la implementación de una aplicación web de Python mediante Azure App Service. Python en Windows ya no se admite.

Aplicación de ejemplo

Este inicio rápido se puede completar mediante Flask, Django o FastAPI. Se proporciona una aplicación de ejemplo en cada marco para ayudarle a seguir este inicio rápido. Descargue o clone la aplicación de ejemplo en la estación de trabajo local.

Flask

Console

```
git clone https://github.com/Azure-Samples/msdocs-python-flask-webapp-quickstart
```

Ejecución de la aplicación de forma local:

Flask

1. Vaya a la carpeta de la aplicación:

Console

```
cd msdocs-python-flask-webapp-quickstart
```

2. Cree un entorno virtual para la aplicación:

Windows

Console

```
py -m venv .venv  
.venv\scripts\activate
```

3. Instale las dependencias:

Console

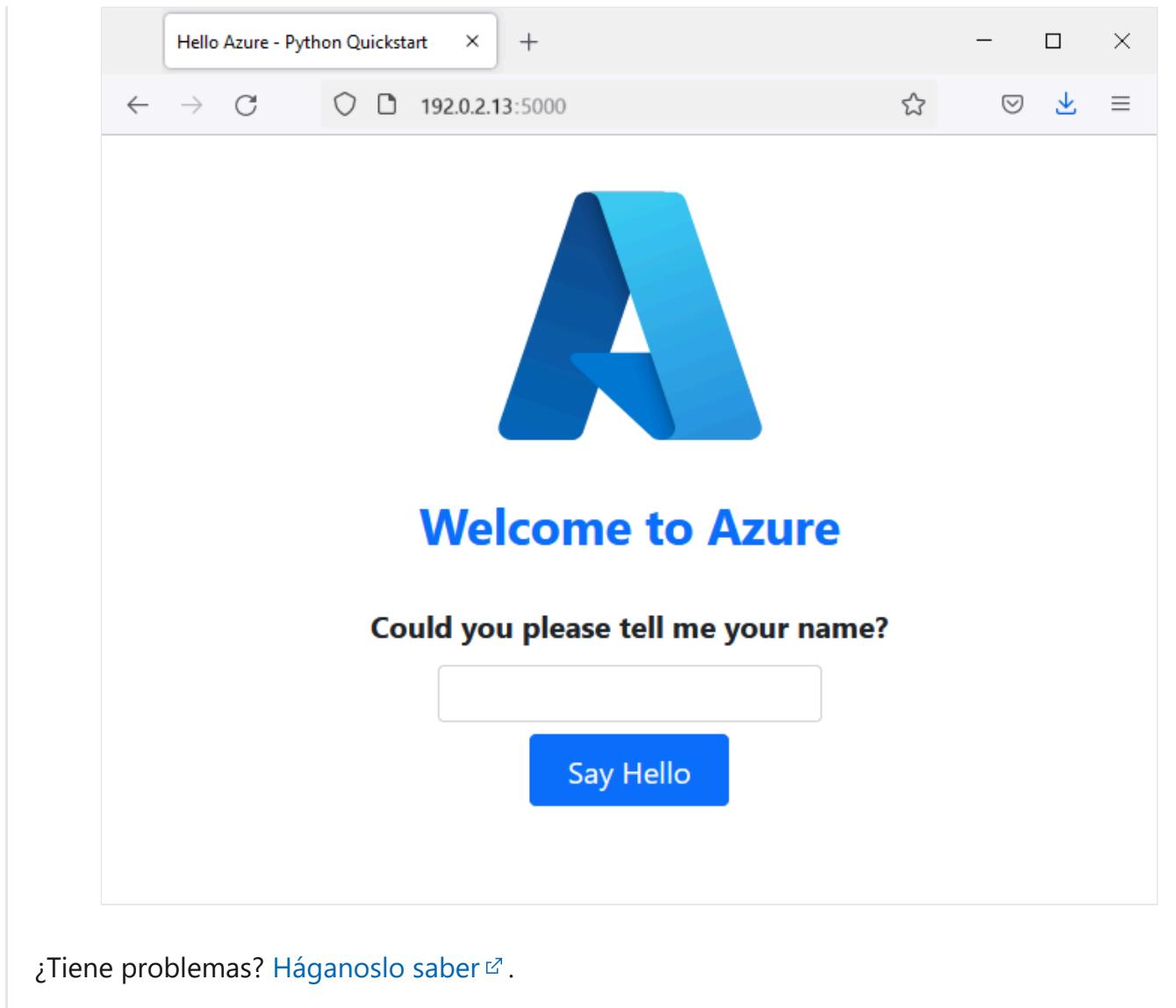
```
pip install -r requirements.txt
```

4. Ejecute la aplicación:

Console

```
flask run
```

5. Vaya a la aplicación de ejemplo en <http://localhost:5000>, en un explorador web.



Creación de una aplicación web en Azure

Para hospedar la aplicación en Azure, debe crear una aplicación web de Azure App Service en Azure. Puede crear una aplicación web mediante la CLI de Azure, [VS Code](#), el [paquete de extensiones Azure Tools](#) o [Azure Portal](#).

CLI de Azure

Los comandos de la CLI de Azure pueden ejecutarse en un equipo que tenga [instalada la CLI de Azure](#).

La CLI de Azure dispone de un comando `az webapp up` que creará los recursos necesarios e implementará la aplicación en un solo paso.

Si es necesario, inicie sesión en Azure mediante [az login](#).

Azure CLI

```
az login
```

Cree la aplicación web y otros recursos y, a continuación, implemente el código en Azure mediante [az webapp up](#).

Azure CLI

```
az webapp up --runtime PYTHON:3.13 --sku B1 --logs
```

- El parámetro `--runtime` especifica qué versión de Python está ejecutando la aplicación. En este ejemplo se usa Python 3.13. Para enumerar todos los runtimes disponibles, use el comando `az webapp list-runtimes --os linux --output table`.
- El parámetro `--sku` define el tamaño (CPU, memoria) y el costo del plan de App Service. En este ejemplo se usa el plan de servicio B1 (Básico), que incurrirá en un pequeño coste en la suscripción de Azure. Para obtener una lista completa de los planes de App Service, vea la página [Precios de App Service](#).
- La marca `--logs` configura el registro predeterminado necesario para habilitar la visualización de la secuencia de registro inmediatamente después de iniciar la aplicación web.
- Opcionalmente, puede especificar un nombre con el argumento `--name <app-name>`. Si no proporciona un nombre, se generará automáticamente.
- Opcionalmente, incluya el argumento `--location <location-name>`, donde `<location_name>` es una región de Azure disponible. Puede recuperar una lista de las regiones permitidas para su cuenta de Azure mediante la ejecución del comando [az appservice list-locations](#).

El comando puede tardar varios minutos en completarse. Mientras el comando se ejecuta, proporciona mensajes sobre la creación del grupo de recursos, el plan de App Service y el recurso de la aplicación, la configuración del registro y la implementación del archivo ZIP. A continuación, devuelve un mensaje que incluye la dirección URL de la aplicación, que es la dirección URL de la aplicación en Azure.

```
The webapp '<app-name>' doesn't exist
Creating Resource group '<group-name>' ...
Resource group creation complete
Creating AppServicePlan '<app-service-plan-name>' ...
Creating webapp '<app-name>' ...
Configuring default logging for the app, if not already enabled
Creating zip with contents of dir /home/cephas/myExpressApp ...
Getting scm site credentials for zip deployment
Starting zip deployment. This operation can take a while to complete ...
Deployment endpoint responded with status code 202
```

```
You can launch the app at <URL>
{
  "URL": "<URL>",
  "appserviceplan": "<app-service-plan-name>",
  "location": "centralus",
  "name": "<app-name>",
  "os": "<os-type>",
  "resourcegroup": "<group-name>",
  "runtime_version": "python|3.13",
  "runtime_version_detected": "0.0",
  "sku": "FREE",
  "src_path": "<your-folder-location>"
}
```

ⓘ Nota

El comando `az webapp up` realiza las acciones siguientes:

- Cree un grupo de recursos predeterminado.
- Cree un plan de App Service predeterminado.
- Cree una aplicación con el nombre especificado.
- Implemente desde un archivo ZIP todos los archivos del directorio de trabajo actual, con la automatización de compilación habilitada.
- Almacene los parámetros localmente en el archivo `.azure/config` para que no tenga que especificarlos de nuevo al implementar posteriormente con `az webapp up` u otros comandos `az webapp` desde la carpeta del proyecto. Los valores almacenados en caché se usan automáticamente de forma predeterminada.

¿Tiene problemas? [Háganoslo saber ↗](#).

Implementación del código de la aplicación en Azure

Azure App Service admite varios métodos para implementar el código de aplicación en Azure, lo que incluye Acciones de GitHub y todas las herramientas de CI/CD principales. Este artículo se centra en cómo implementar el código desde la estación de trabajo local en Azure.

Dado que el comando `az webapp up` anterior creó los recursos necesarios e implementó la aplicación en un solo paso, puede pasar al siguiente paso.

¿Tiene problemas? Consulte primero la [Guía de solución de problemas](#). Si eso no le ayuda, [infórmenos](#).

Configuración del script de inicio

En función de la presencia de determinados archivos en una implementación, App Service detecta automáticamente si una aplicación es una aplicación de Django o Flask y realiza los pasos predeterminados para ejecutar la aplicación. En el caso de las aplicaciones basadas en otros marcos web como FastAPI, debe configurar un script de inicio para que App Service ejecute la aplicación; de lo contrario, App Service ejecutará una aplicación de solo lectura predeterminada ubicada en la carpeta `opt/defaultsite`.

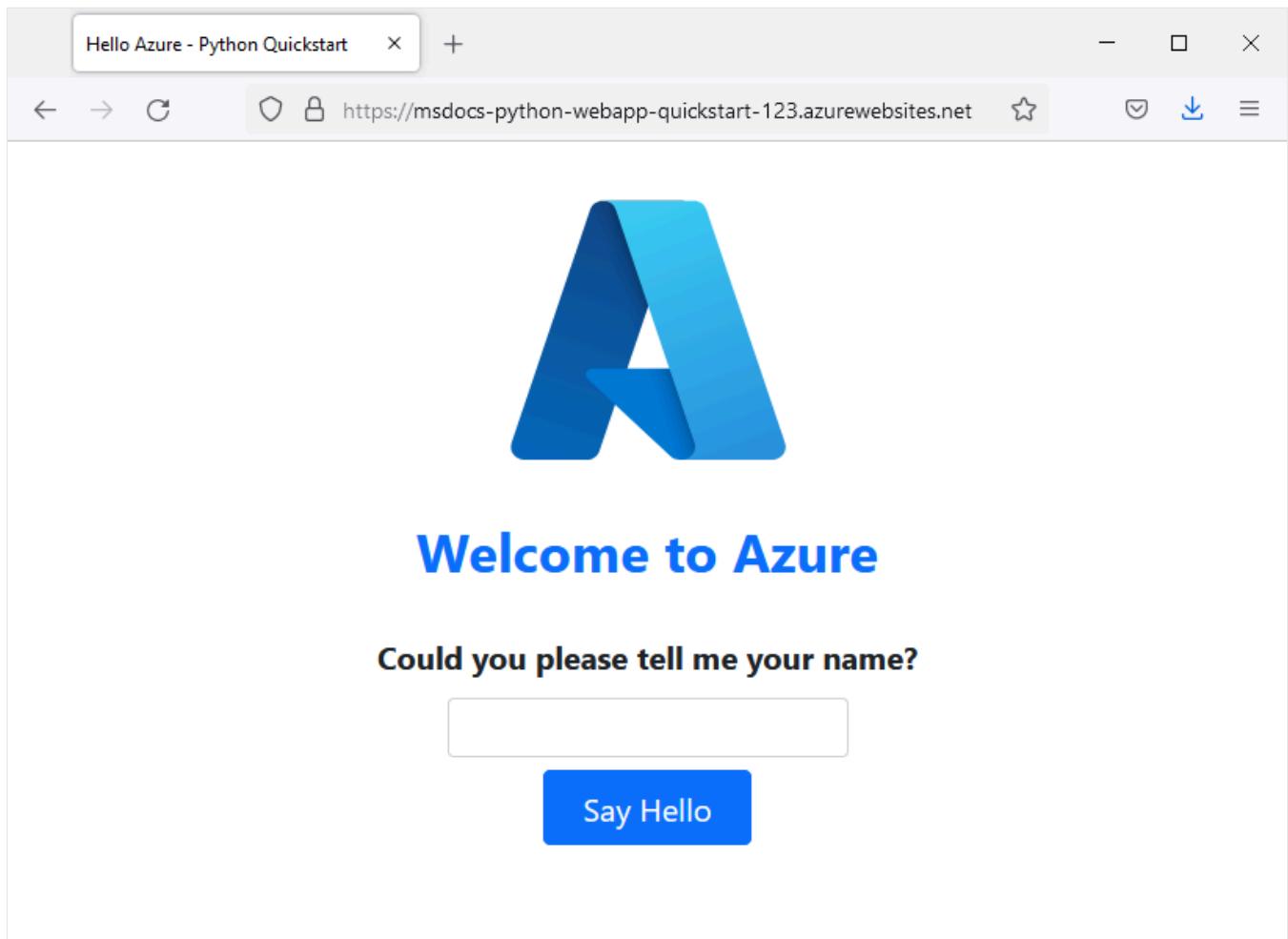
Para más información sobre cómo App Service ejecuta aplicaciones de Python y cómo puede configurar y personalizar su comportamiento con la aplicación, consulte [Configuración de una aplicación de Python de Linux para Azure App Service](#).

App Service detecta automáticamente la presencia de una aplicación de Flask. No se necesita ninguna configuración adicional para este inicio rápido.

Navegación hasta la aplicación

Vaya a la aplicación implementada en el explorador web. Puede seguir un vínculo desde Azure Portal. Vaya a la página **Información general** y seleccione **Dominio predeterminado**. Si ve una página de aplicación predeterminada, espere un momento y actualice el explorador.

El código de ejemplo de Python se ejecuta en un contenedor de Linux en App Service con una imagen integrada.



¡Enhorabuena! Ha implementado su primera aplicación Python en App Service en Linux.

¿Tiene problemas? Consulte primero la [Guía de solución de problemas](#). Si eso no le ayuda, [infórmenos](#).

Transmisión de registros

Azure App Service captura toda la salida de mensajes hacia la consola para ayudarle a diagnosticar problemas de la aplicación. Las aplicaciones de ejemplo incluyen instrucciones de `print()` para demostrar esta funcionalidad.

```
Flask

Python

@app.route('/')
def index():
    print('Request for index page received')
    return render_template('index.html')

@app.route('/favicon.ico')
def favicon():
    return send_from_directory(os.path.join(app.root_path, 'static'),
```

```
        'favicon.ico',
mimetype='image/vnd.microsoft.icon')

@app.route('/hello', methods=['POST'])
def hello():
    name = request.form.get('name')

    if name:
        print('Request for hello page received with name=%s' % name)
        return render_template('hello.html', name = name)
    else:
        print('Request for hello page received with no name or blank name --'
              'redirecting')
        return redirect(url_for('index'))
```

Puede revisar el contenido de los registros de diagnóstico de App Service mediante la CLI de Azure, VS Code o Azure Portal.

CLI de Azure

Primero debe configurar Azure App Service para generar registros en el sistema de archivos de App Service mediante el comando [az webapp log config](#).

bash

Azure CLI

```
az webapp log config \
--web-server-logging filesystem \
--name $APP_SERVICE_NAME \
--resource-group $RESOURCE_GROUP_NAME
```

Para transmitir registros, use el comando [az webapp log tail](#).

bash

Azure CLI

```
az webapp log tail \
--name $APP_SERVICE_NAME \
--resource-group $RESOURCE_GROUP_NAME
```

Actualice la página principal de la aplicación o pruebe otras solicitudes para generar algunos mensajes de registro. La salida debe tener una apariencia similar a la siguiente.

Output

Starting Live Log Stream ---

```
2021-12-23T02:15:52.740703322Z Request for index page received
2021-12-23T02:15:52.740740222Z 169.254.130.1 - - [23/Dec/2021:02:15:52 +0000]
"GET / HTTP/1.1" 200 1360 "https://msdocs-python-webapp-quickstart-
123.azurewebsites.net/hello" "Mozilla/5.0 (Windows NT 10.0; Win64; x64;
rv:95.0) Gecko/20100101 Firefox/95.0"
2021-12-23T02:15:52.841043070Z 169.254.130.1 - - [23/Dec/2021:02:15:52 +0000]
"GET /static/bootstrap/css/bootstrap.min.css HTTP/1.1" 200 0 "https://msdocs-
python-webapp-quickstart-123.azurewebsites.net/" "Mozilla/5.0 (Windows NT
10.0; Win64; x64; rv:95.0) Gecko/20100101 Firefox/95.0"
2021-12-23T02:15:52.884541951Z 169.254.130.1 - - [23/Dec/2021:02:15:52 +0000]
"GET /static/images/azure-icon.svg HTTP/1.1" 200 0 "https://msdocs-python-
webapp-quickstart-123.azurewebsites.net/" "Mozilla/5.0 (Windows NT 10.0;
Win64; x64; rv:95.0) Gecko/20100101 Firefox/95.0"
2021-12-23T02:15:53.043211176Z 169.254.130.1 - - [23/Dec/2021:02:15:53 +0000]
"GET /favicon.ico HTTP/1.1" 404 232 "https://msdocs-python-webapp-quickstart-
123.azurewebsites.net/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:95.0)
Gecko/20100101 Firefox/95.0"

2021-12-23T02:16:01.304306845Z Request for hello page received with name=David
2021-12-23T02:16:01.304335945Z 169.254.130.1 - - [23/Dec/2021:02:16:01 +0000]
"POST /hello HTTP/1.1" 200 695 "https://msdocs-python-webapp-quickstart-
123.azurewebsites.net/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:95.0)
Gecko/20100101 Firefox/95.0"
2021-12-23T02:16:01.398399251Z 169.254.130.1 - - [23/Dec/2021:02:16:01 +0000]
"GET /static/bootstrap/css/bootstrap.min.css HTTP/1.1" 304 0 "https://msdocs-
python-webapp-quickstart-123.azurewebsites.net/hello" "Mozilla/5.0 (Windows NT
10.0; Win64; x64; rv:95.0) Gecko/20100101 Firefox/95.0"
2021-12-23T02:16:01.430740060Z 169.254.130.1 - - [23/Dec/2021:02:16:01 +0000]
"GET /static/images/azure-icon.svg HTTP/1.1" 304 0 "https://msdocs-python-
webapp-quickstart-123.azurewebsites.net/hello" "Mozilla/5.0 (Windows NT 10.0;
Win64; x64; rv:95.0) Gecko/20100101 Firefox/95.0"
```

¿Tiene problemas? Consulte primero la [Guía de solución de problemas](#). Si eso no le ayuda, [infórmenos](#).

Limpieza de recursos

Cuando haya terminado con la aplicación de muestra, puede quitar todos los recursos de la aplicación de Azure. Quitar el grupo de recursos garantiza que no incurre en cargos adicionales y ayuda a mantener ordenada la suscripción de Azure. Al quitar el grupo de recursos también se quitan todos los recursos que haya dentro; es la manera más rápida de quitar todos los recursos de Azure de la aplicación.

Elimine el grupo de recursos con el comando [az group delete](#).

Azure CLI

```
az group delete \
    --name msdocs-python-webapp-quickstart \
    --no-wait
```

El argumento `--no-wait` permite la devolución del comando antes de que se complete la operación.

¿Tiene problemas? [Háganoslo saber ↗](#).

Pasos siguientes

[Tutorial: Aplicación web de Python \(Flask\) con PostgreSQL](#)

[Tutorial: Aplicación web Python \(Django\) con PostgreSQL](#)

[Configuración de una aplicación de Python](#)

[Incorporación del inicio de sesión de un usuario a una aplicación web de Python](#)

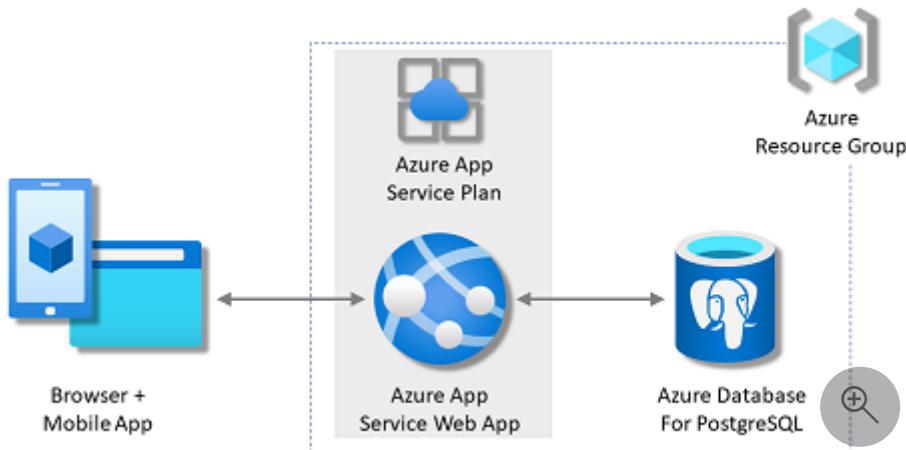
[Tutorial: Ejecución de una aplicación de Python en un contenedor personalizado](#)

[Proteger una aplicación con un dominio personalizado y un certificado](#)

Implementación de una aplicación web de Python (Flask) con PostgreSQL en Azure

17/04/2025

En este tutorial, implementará una aplicación web de Python controlada por datos ([Flask](#)) en [Azure App Service](#) con el servicio de base de datos relacional [Azure Database for PostgreSQL](#). Azure App Service admite [Python](#) en un entorno de servidor Linux. Si lo desea, consulte el [tutorial de Django](#) o el [tutorial de FastAPI](#) en su lugar.



En este tutorial, aprenderá a:

- ✓ Cree una arquitectura de App Service, PostgreSQL y Redis Cache segura de manera predeterminada.
- ✓ Asegure los secretos de conexión utilizando una identidad administrada y referencias de Key Vault.
- ✓ Implemente una aplicación de muestra de Python en App Service desde un repositorio de GitHub.
- ✓ Acceda a las cadenas de conexión de App Service y a la configuración de la aplicación en el código de la aplicación.
- ✓ Realice actualizaciones y vuelva a implementar el código de la aplicación.
- ✓ Genere el esquema de la base de datos mediante la ejecución de migraciones de bases de datos.
- ✓ Transmitir registros de diagnóstico desde Azure.
- ✓ Administrar la aplicación en Azure Portal.
- ✓ Aprovisione la misma arquitectura e impleméntela usando Azure Developer CLI.
- ✓ Optimice su flujo de trabajo de desarrollo con GitHub Codespaces y GitHub Copilot.

Requisitos previos

- Una cuenta de Azure con una suscripción activa. Si no tiene una cuenta de Azure, puede [crearla gratis](#).
- Una cuenta de GitHub. También puede [obtener una gratis](#).
- Conocimientos de Python con el desarrollo de Flask.
- (Opcional) Para probar GitHub Copilot, una [cuenta de GitHub Copilot](#). Hay disponible una evaluación gratuita de 30 días.

Ir al final

Si solo quiere ver la aplicación de ejemplo en este tutorial que se ejecuta en Azure, solo tiene que ejecutar los siguientes comandos en [Azure Cloud Shell](#) y seguir el aviso:

Bash

```
mkdir msdocs-flask-postgresql-sample-app
cd msdocs-flask-postgresql-sample-app
azd init --template msdocs-flask-postgresql-sample-app
azd up
```

1. Ejecución del ejemplo

En primer lugar, configurará una aplicación controlada por datos de ejemplo como punto inicial. Para lograr mayor comodidad, el [repositorio de ejemplo](#) incluye una configuración de [contenedor de desarrollo](#). El contenedor de desarrollo tiene todo lo que necesita para desarrollar una aplicación, incluida la base de datos, la caché y todas las variables de entorno necesarias para la aplicación de ejemplo. El contenedor de desarrollo puede ejecutarse en un [codespace de GitHub](#), lo que significa que puede ejecutar el ejemplo en cualquier equipo con un explorador web.

! Nota

Si sigue este tutorial con su propia aplicación, examine la descripción del archivo *requirements.txt* en el archivo [README.md](#) para ver los paquetes que necesitará.

Paso 1: En una nueva ventana del navegador:

1. Inicie sesión en su cuenta de GitHub.
2. Vaya a <https://github.com/Azure-Samples/msdocs-flask-postgresql-sample-app/fork>.
3. Anule la selección de **Copiar solo la rama principal**. Quieres todas las ramas.
4. Seleccione **Crear bifurcación**.

The screenshot shows the GitHub fork creation interface for the repository `msdocs-flask-postgresql-sample-app`. The top navigation bar includes links for Code, Issues, Pull requests, Actions, Projects, Security, and Insights. The 'Code' tab is selected. Below the navigation, a section titled 'Create a new fork' is displayed. A note explains that a fork is a copy of a repository, allowing experimentation without affecting the original project. It also states that required fields are marked with an asterisk (*). The 'Owner' field is set to the user's account, and the 'Repository name' field contains `msdocs-flask-postgresql-san`, with a message indicating it is available. A note says forks are typically named the same as the upstream repository but can be customized. There is a 'Description (optional)' input field, a checkbox for 'Copy the main branch only' which is unchecked, and a note about contributing back to the upstream repository. A message informs the user they are creating a fork in their personal account. At the bottom right are 'Create fork' and 'Search' buttons.

Create a new fork

A **fork** is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project. [View existing forks.](#)

Required fields are marked with an asterisk (*).

Owner * Repository name *

 / `msdocs-flask-postgresql-san`

 `msdocs-flask-postgresql-sample-app` is available.

By default, forks are named the same as their upstream repository. You can customize the name to distinguish it further.

Description (optional)

Copy the `main` branch only
Contribute back to Azure-Samples/msdocs-flask-postgresql-sample-app by adding your own branch. [Learn more.](#)

 You are creating a fork in your personal account.

Create fork 

Paso 2: En el fork de GitHub:

1. Seleccione **Principal>starter-no-infra** para la rama de inicio. Esta rama contiene solo el proyecto de ejemplo y ninguna configuración ni archivo relacionados con Azure.
2. Seleccione **Código>Crear codespace on starter-no-infra**. El espacio de código tarda unos minutos en configurarse y ejecuta `pip install -r requirements.txt` para el repositorio al final.

This branch is up to date
Azure-Samples/msdocs-1

Contribute

add copilot ext

.devcontainer

.github

azureproject

migrations

static

templates

Remove spurious line. 3 years ago

.env

convert to service connecto... 20 hours ago

starter-no-infra

Go to file

Local Codespaces

Codespaces Your workspaces in the cloud + ...

No codespaces You don't have any codespaces with this repository checked out

Create codespace on starter-no-infra

Learn more about codespaces...

Codespace usage for this repository is paid for by Icphas.

About

No description, website, or topics provided.

Readme

MIT license

Code of conduct

Activity

0 stars

0 watching

0 forks

Releases

No releases published

Create a new release

Packages

No packages published

Paso 3: En el terminal de codespace:

1. Ejecute las migraciones de la base de datos con `flask db upgrade`.
2. Ejecute la aplicación con `flask run`.
3. Cuando vea la notificación `Your application running on port 5000 is available.`, seleccione **Abrir en explorador**. Debería ver la aplicación de ejemplo en una nueva pestaña del explorador. Para detener la aplicación, escriba `Ctrl+C`.

This screenshot shows the Microsoft Visual Studio Code interface with a Python project named 'MSDOCS-FLASK-POSTGRESQL-SAMPLE-APP'. The terminal tab is active, displaying command-line output for database migrations and application startup. A tooltip provides information about the application's port status. Buttons for opening the application in a browser or making it public are also visible.

Sugerencia

Puede preguntar a [GitHub Copilot](#) sobre este repositorio. Por ejemplo:

- @workspace ¿Qué hace este proyecto?
- @workspace ¿Qué hace la carpeta .devcontainer?

¿Tiene problemas? Consulte la sección [Solución de problemas](#).

2. Creación de una instancia de App Service y PostgreSQL

En este paso, creará los recursos de Azure. Los pasos que se usan en este tutorial crean un conjunto de recursos seguros de forma predeterminada que incluyen App Service y Azure Database for PostgreSQL. En el proceso de creación, especifique lo siguiente:

- El **Nombre** de la aplicación web. Se usa como parte del nombre DNS de la aplicación.
- La **región** para ejecutar la aplicación físicamente en el mundo. También se usa como parte del nombre DNS de la aplicación.
- La **pila en tiempo de ejecución** para la aplicación. Es donde se selecciona la versión de Python que se va a usar para la aplicación.

- El **plan de hospedaje** para la aplicación. Es el plan de tarifa que incluye el conjunto de características y la capacidad de escalado de la aplicación.
- El **Grupo de recursos** de la aplicación. Un grupo de recursos permite agrupar (en un contenedor lógico) todos los recursos de Azure necesarios para la aplicación.

Inicie sesión en [Azure Portal](#) y siga estos pasos para crear los recursos de Azure App Service.

Paso 1: En el portal Azure:

1. Escriba "base de datos de aplicación web" en la barra de búsqueda de la parte superior de Azure Portal.
2. Seleccione el elemento denominado **Aplicación web + base de datos** en el encabezado **Marketplace**. También puede ir directamente al [asistente de creación](#).

The screenshot shows the Microsoft Azure portal interface. At the top, there's a search bar with the text "web app database". Below the search bar, there's a list of services under "Marketplace", with "Web App + Database" being the first item and highlighted with a red box. To the left, there's a sidebar with sections for "Azure services" (Create a resource, Resource groups, Microsoft Entra ID, DNS zones) and "Resources" (Recent, Favorite). On the right, there are some AI-related cards and a "Managed Identities" button. At the bottom right, there's a "Give feedback" link and a "Last Viewed" section.

Paso 2: In the página [Crear aplicación web + base de datos](#), rellene el formulario de la siguiente manera.

1. *Grupo de recursos*: seleccione **Crear nuevo** y use un nombre de **msdocs-flask-postgres-tutorial**.
2. *Región*: cualquier región de Azure cercana a usted.
3. *Nombre*: **msdocs-python-postgres-XYZ**.
4. *Pila en tiempo de ejecución*: **Python 3.12**.
5. *Base de datos*: **PostgreSQL: servidor flexible** está seleccionado de manera predeterminada como motor de base de datos. El nombre del servidor y el nombre de la base de datos también se establecen de manera predeterminada en los valores adecuados.
6. *¿Agregar Azure Cache for Redis?*: **no**.
7. *Plan de hospedaje*: **Básico**. Cuando esté listo, puede [escalar verticalmente](#) a un plan de tarifa de producción.
8. Seleccione **Revisar + crear**.

9. Una vez completada la validación, seleccione **Crear**.

Home >

Create Web App + Database

Basics Tags Review + create

This template will create a secure by default configuration where the only publicly accessible endpoint will be your app following the recommended security best practices. [Learn more](#)

Project Details

Select a subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription *

Resource Group * (New) msdocs-flask-postgres-tutorial [Create new](#)

Region *

Web App Details

Name -epb5abc3augnh4b2.centralus-01.azurewebsites.net

Secure unique default hostname on. [More about this update](#)

Runtime stack *

Database

i Database access will be locked down and not exposed to the public internet. This is in compliance with recommended best practices for security.

Engine *

Server name *

Database name *

Azure Cache for Redis

Add Azure Cache for Redis? No Yes

Hosting

Hosting plan * Basic - For hobby or research purposes Standard - General purpose production apps

[Review + create](#) [< Previous](#) [Next : Tags >](#)

Step 3: La implementación tarda unos minutos en completarse. Una vez completada la implementación, seleccione el botón **Ir al recurso**. Se le dirigirá directamente a la aplicación App Service, pero se crean los siguientes recursos:

- **Grupo de recursos:** contenedor para todos los recursos creados.
- **Plan de App Service:** define los recursos de proceso de App Service. Se crea un plan de Linux en el nivel *Básico*.
- **App Service:** representa su aplicación y se ejecuta en el plan de App Service.
- **Red virtual:** se integra con la aplicación App Service y aísla el tráfico de back-end.
- **Interfaces de red:** representa direcciones IP privadas, una para cada uno de los puntos de conexión privados.
- **Servidor flexible de Azure Database for PostgreSQL:** accesible solo desde dentro de la red virtual. Una base de datos y un usuario se crean automáticamente en el servidor.
- **Zonas DNS privadas:** Permite la resolución DNS del almacén de claves y del servidor de base de datos en la red virtual.

 Your deployment is complete

 Deployment name : Microsoft.Web-WebAppDatabase-Portal-afa69d9d-97bc
Subscription :
Resource group : [msdocs-python-postgres-tutorial](#)
Start time : 11/29/2023, 10:17:05 AM
Correlation ID :

> Deployment details

▽ Next steps

[Go to resource](#)

Give feedback 

 Tell us about your experience with deployment

3. Protección de secretos de conexión

El asistente para la creación ya generó automáticamente las variables de conectividad como [ajustes de la aplicación](#). Sin embargo, el procedimiento recomendado de seguridad es mantener completamente los secretos fuera de App Service. Moverá los secretos a un almacén de claves y cambiará la configuración de la aplicación a [referencias de Key Vault](#) con la ayuda de Service Connectors.

Paso 1: Recuperación de la cadena de conexión existente

1. En el menú de la izquierda de la página App Service, seleccione **Configuración > Variables de entorno**.
2. Seleccione **AZURE_POSTGRESQL_CONNECTIONSTRING**.
3. En **Agregar o editar configuración de aplicación**, en el campo **Valor**, busque la parte *password=* al final de la cadena.
4. Copie la cadena de contraseña después de *Password=* para usarla más adelante. Esta configuración de aplicación le permite conectarse a la base de datos Postgres protegida tras un punto de conexión privado. Sin embargo, el secreto se guarda directamente en la aplicación de App Service, que no es lo mejor. Cambiarás esto.

The screenshot shows the Azure portal interface for managing an App Service application named 'msdocs-python-postgres-123'. On the left, the navigation menu is visible with options like Tags, Diagnose and solve problems, Microsoft Defender for Cloud, Events (preview), Recommended services (preview), Deployment, and Settings. Under Settings, the 'Environment variables' section is highlighted with a red box. On the right, a modal window titled 'Add/Edit application setting' is open. It shows a list of environment variables: AZURE_KEYVAULT_RESOURCEENDPOINT, AZURE_KEYVAULT_SCOPE, AZURE_POSTGRESQL_CONNECTIONSTRING (which is selected and highlighted with a red box), and AZURE_REDIS_CONNECTIONSTRING. The 'AZURE_POSTGRESQL_CONNECTIONSTRING' entry has its 'Name' field set to 'AZURE_POSTGRESQL_CONNECTIONSTRING' and its 'Value' field set to a complex string starting with 'host=msdocs-python-postgres-3-server.postgres.database.azure.com port=5432 sslmode=require user=gqjcbqbitiq password=xxxxxxxxxx'. A checkbox for 'Deployment slot setting' is checked. At the bottom of the modal are 'Apply' and 'Discard' buttons, and a magnifying glass icon.

Paso 2: Creación de un almacén de claves para la administración segura de secretos

1. En la barra de búsqueda superior, escriba "*almacén de claves*" y, a continuación, seleccione **Marketplace>Key Vault**.
2. En **Grupo de recursos**, seleccione **msdocs-python-postgres-tutorial**.
3. En **Nombre del almacén de claves**, escriba un nombre que consta de solo letras y números.
4. En **Región**, establézcalo en la misma ubicación que el grupo de recursos.

Azure Key Vault is a cloud service used to manage keys, secrets, and certificates. Key Vault eliminates the need for developers to store security information in their code. It allows you to centralize the storage of your application secrets which greatly reduces the chances that secrets may be leaked. Key Vault also allows you to securely store secrets and keys backed by Hardware Security Modules or HSMs. The HSMs used are Federal Information Processing Standards (FIPS) 140-2 Level 2 validated. In addition, key vault provides logs of all access and usage attempts of your secrets so you have a complete audit trail for compliance.

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription *	Dev Compute and Platform - FY25Q3
Resource group *	msdocs-python-postgres-123_group
	Create new

Instance details

Key vault name *	vault091979
Region *	Canada Central
Pricing tier *	Standard

Recovery options

Soft delete protection will automatically be enabled on this key vault. This feature allows you to recover or permanently delete a key vault and secrets for the duration of the retention period. This protection applies to the key vault and the secrets stored within the key vault.

To enforce a mandatory retention period and prevent the permanent deletion of key vaults or secrets prior to the retention period elapsing, you can turn on purge protection. When purge protection is enabled, secrets cannot be purged by users or

[Previous](#)

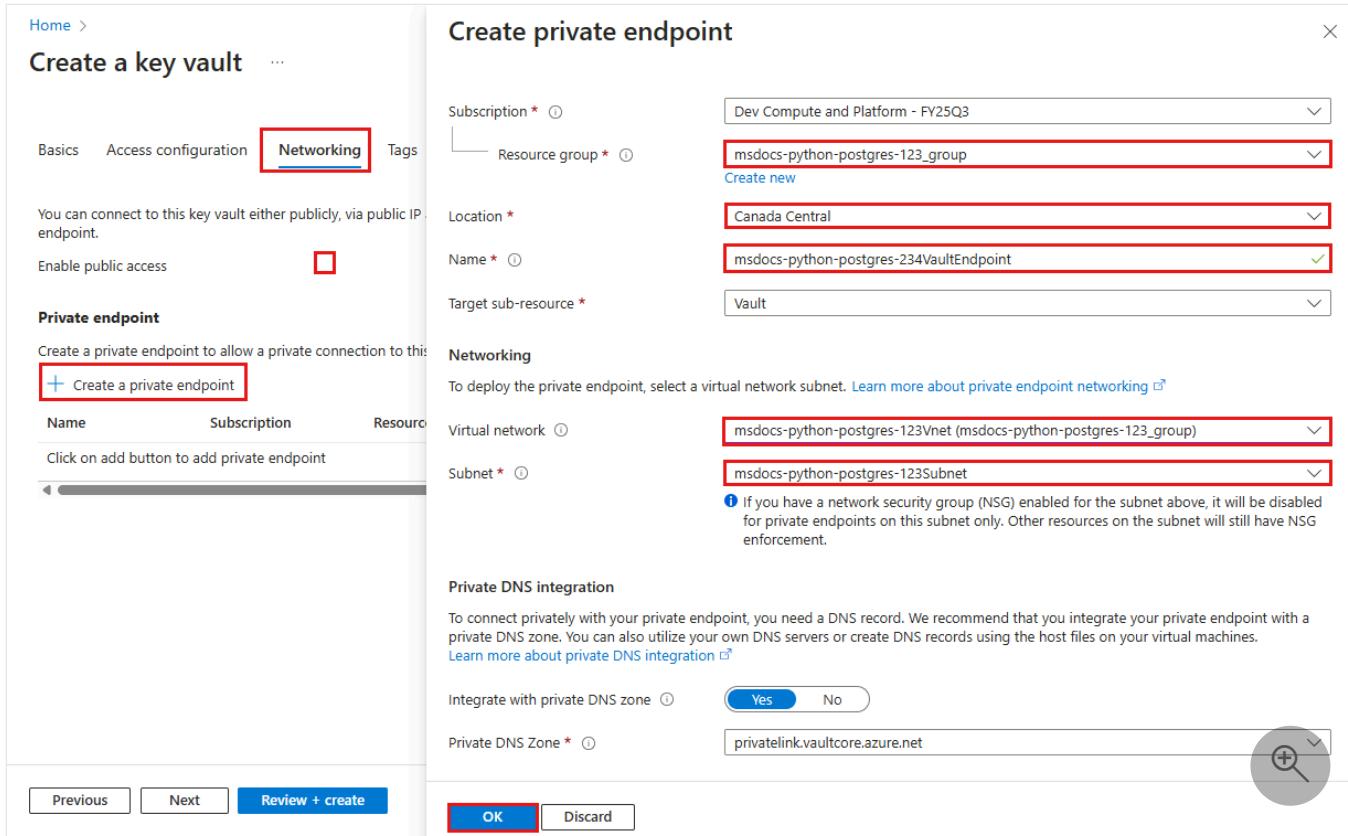
[Next](#)

[Review + create](#)



Paso 3: Protección del almacén de claves con un punto de conexión privado

1. Seleccione la pestaña **Redes**.
2. Anule la selección de **Habilitar acceso público**.
3. Seleccione **Crear un punto de conexión privado**.
4. En **Grupo de recursos**, seleccione **msdocs-python-postgres-tutorial**.
5. En el cuadro de diálogo, en **Ubicación**, seleccione la misma ubicación que la aplicación de App Service.
6. En **Nombre**, escriba **msdocs-python-postgres-XYZVaultEndpoint**.
7. En **Red virtual**, seleccione **msdocs-python-postgres-XYZVnet**.
8. En **Subred**, **msdocs-python-postgres-XYZSubnet**.
9. Seleccione **Aceptar**.
10. Seleccione **Revisar y crear** y, luego, **Crear**. Espere a que finalice la implementación del almacén de claves. Deberías ver "Tu implementación está completa".



Paso 4: Configuración del conector de PostgreSQL

1. En la barra de búsqueda superior, escriba *msdocs-python-postgres* y, a continuación, seleccione el recurso de App Service denominado **msdocs-python-postgres-XYZ**.
2. En la página App Service, en el menú de la izquierda, seleccione **Configuración > Conector de servicio**. Ya hay un conector, que el asistente para la creación de aplicaciones ha creado automáticamente.
3. Active la casilla situada junto al conector PostgreSQL y seleccione **Editar**.
4. En **Tipo de cliente**, seleccione **Django**. Aunque tiene una aplicación de Flask, el [tipo de cliente Django en el conector del servicio PostgreSQL](#) proporciona variables de base de datos en una configuración independiente en lugar de una cadena de conexión. Las variables independientes son más fáciles de usar en el código de aplicación, que usa [SQLAlchemy](#) para conectarse a la base de datos.
5. Seleccione la pestaña **Autenticación**.
6. En **Contraseña**, pegue la contraseña que copió anteriormente.
7. Seleccione **Almacenar secreto en Key Vault**.
8. En **Conexión de Key Vault**, seleccione **Crear nuevo**. Se abre un cuadro de diálogo **Crear conexión** encima del cuadro de diálogo de edición.

The screenshot shows the Azure portal interface for creating a Service Connector. On the left, the sidebar is open with various options like Environment variables, Configuration, Authentication, Identity, Backups, Custom domains, Certificates, Networking, Scale up (App Service plan), Scale out (App Service plan), WebJobs (preview), and Service Connector. The 'Service Connector' option is highlighted with a red box. The main area shows a list of existing connectors: 'DB for PostgreSQL flexible server' (selected), 'Cache for Redis', and 'Key Vault'. The 'Edit' button in the top navigation bar is also highlighted with a red box. The right side of the screen displays the 'defaultConnector' configuration page under the 'Authentication' tab. It asks to select an authentication type between compute service and target service. The 'Connection string' option is selected. Below it, there's a 'Continue with...' section with 'Database credentials' and 'Key Vault' options. The 'Key Vault' section contains fields for 'Username' (gcibcbtiq) and 'Password' (redacted). A checkbox for 'Store Secret In Key Vault' is checked, and the 'Key Vault Connection' dropdown is set to 'mangesh-key-vault-python (keyvault_362d5)'. A red box highlights the 'Create new' button in this dropdown. At the bottom, there are 'Next : Networking', 'Previous', and 'Cancel' buttons.

Paso 5: Establecimiento de la conexión de Key Vault

1. En el cuadro de diálogo **Crear conexión** para la conexión de Key Vault, en **Key Vault**, seleccione el almacén de claves que ha creado antes.
2. Seleccione **Revisar + crear**.
3. Cuando se complete la validación, seleccione **Crear**.

Create connection

Basics Networking Review + Create

Select the service instance and client type.

Service type * ⓘ

Key Vault

Connection name * ⓘ

keyvault_cc22c

Subscription * ⓘ

Dev Compute and Platform - FY25Q3

Key vault * ⓘ

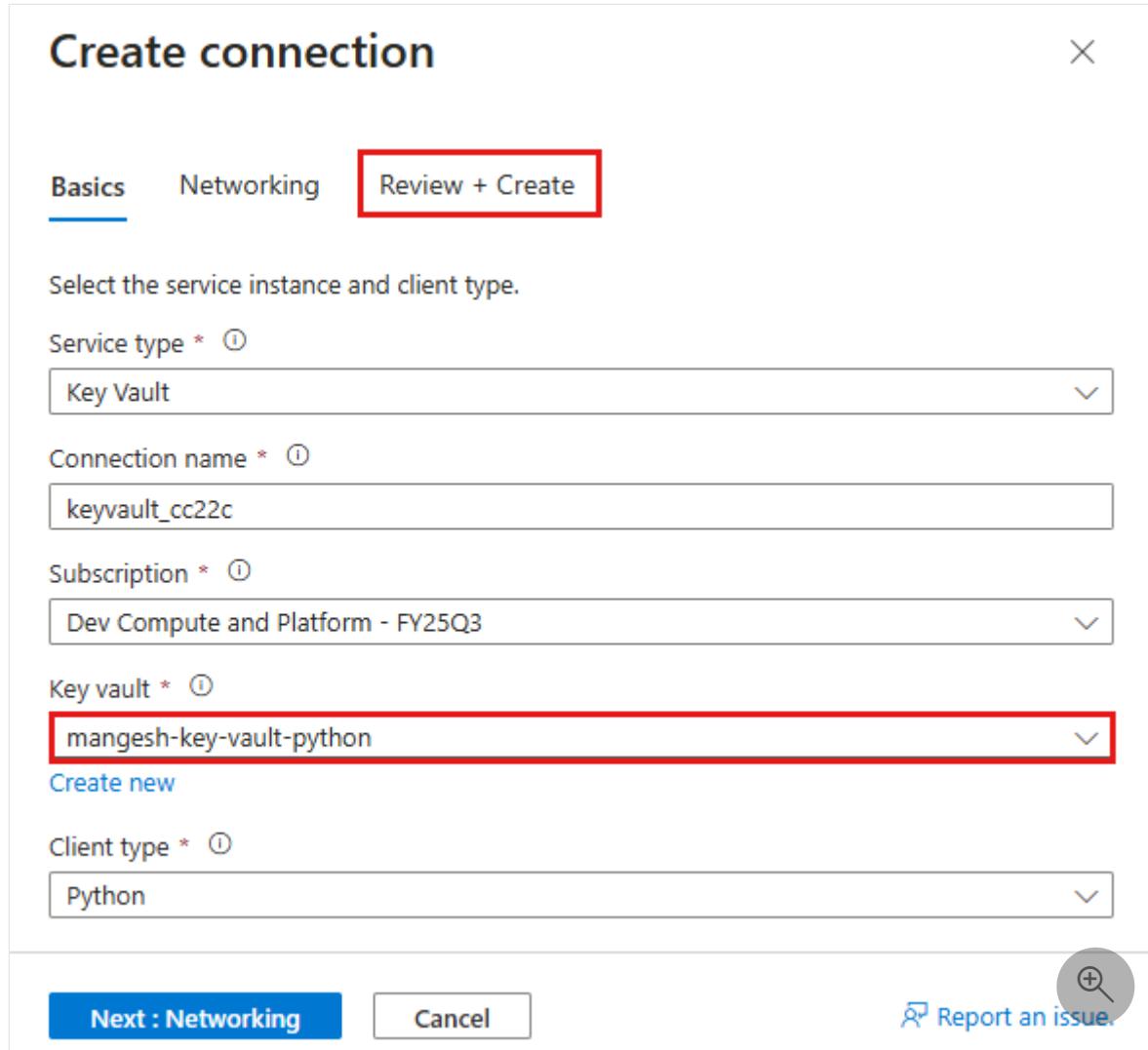
mangesh-key-vault-python

Create new

Client type * ⓘ

Python

Next : Networking Cancel Report an issue.



Paso 6: Finalización de la configuración del conector de PostgreSQL

1. Vuelve al cuadro de diálogo de edición de **defaultConnector**. En la pestaña **Autenticación**, espere a que se cree el conector del almacén de claves. Cuando haya terminado, la lista desplegable **Conexión de Key Vault** la selecciona automáticamente.
2. Seleccione **Siguiente: Redes**.
3. Seleccione **Guardar**. Espere hasta que aparezca la **Actualización correcta** notificación.

defaultConnector

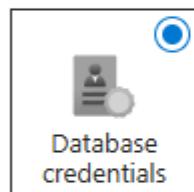
X

Basics **Authentication** Networking

Select the authentication type you'd like to use between your compute service and target service. [Learn more about authentication.](#)

- System assigned managed identity (Supported via Azure CLI. [Learn more](#)) ⓘ
- User assigned managed identity (Supported via Azure CLI. [Learn more](#)) ⓘ
- Connection string ⓘ
- Service principal (Supported via Azure CLI. [Learn more](#)) ⓘ

Continue with...



Database
credentials

Username *

gcibcbtiq

Password *

.....

[Forgot password?](#)

Store Secret In Key Vault ⓘ

Key Vault Connection * ⓘ

mangesh-key-vault-python (keyvault_362d5)

[Create new](#)

Store Configuration in App Configuration ⓘ

Next : Networking

Previous

Cancel



Paso 7: Comprobación de la integración de Key Vault

1. En el menú de la izquierda, vuelva a seleccionar Configuración > Variables de entorno.
2. Junto a AZURE_POSTGRESQL_PASSWORD, seleccione Mostrar valor. El valor debe ser `@Microsoft.KeyVault(...)`, lo que significa que es una [referencia del almacén de claves](#) porque el secreto ahora se administra en el almacén de claves.

The screenshot shows the 'App settings' section of the Azure App Service configuration. On the left, a sidebar lists 'Recommended services (preview)', 'Deployment' (with 'Deployment slots' and 'Deployment Center'), and 'Settings' (with 'Environment variables' selected). The main area displays a table of environment variables:

Name	Value	Deployment slot
AZURE_KEYVAULT_RESOURCE...	Show value	✓
AZURE_KEYVAULT_SCOPE	Show value	✓
AZURE_POSTGRESQL_CONNE...	Show value	✓
AZURE_REDIS_CONNECTIONS...	Show value	✓

At the bottom right of the main area are 'Apply' and 'Discard' buttons, and a 'Send us your feedback' link.

En resumen, el proceso para proteger los secretos de conexión implica:

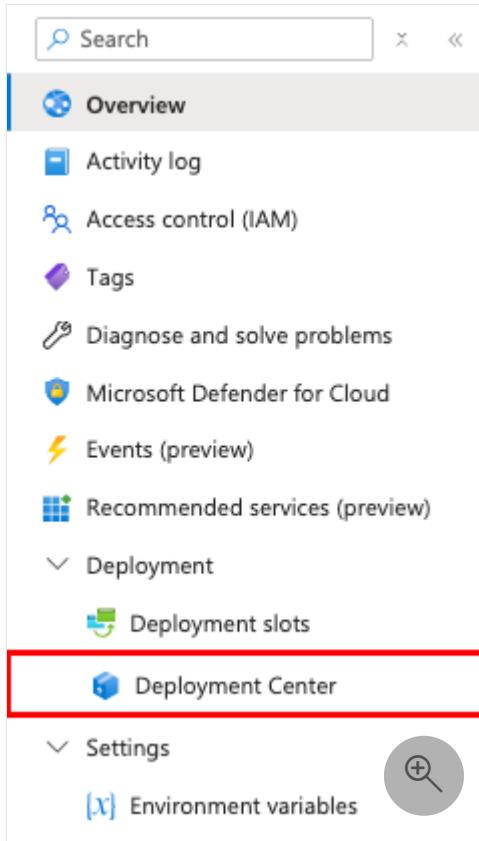
- Recuperar los secretos de conexión de las variables de entorno de la aplicación de App Service.
- Creación de un almacén de claves.
- Crear una conexión de Key Vault con la identidad administrada asignada por el sistema.
- Actualizar los conectores de servicio para almacenar los secretos en el almacén de claves.

¿Tiene problemas? Consulte la sección [Solución de problemas](#).

4. Implementación del código de ejemplo

En este paso, configurará la implementación de GitHub mediante Acciones de GitHub. Es solo una de las muchas maneras de implementar en App Service, pero también una excelente manera de disponer de integración continua en el proceso de implementación. De forma predeterminada, cada uno de los repositorios `git push` de GitHub inicia la acción de compilación e implementación.

Paso 1: En el menú de la izquierda, seleccione **Implementación>Centro de implementación**.

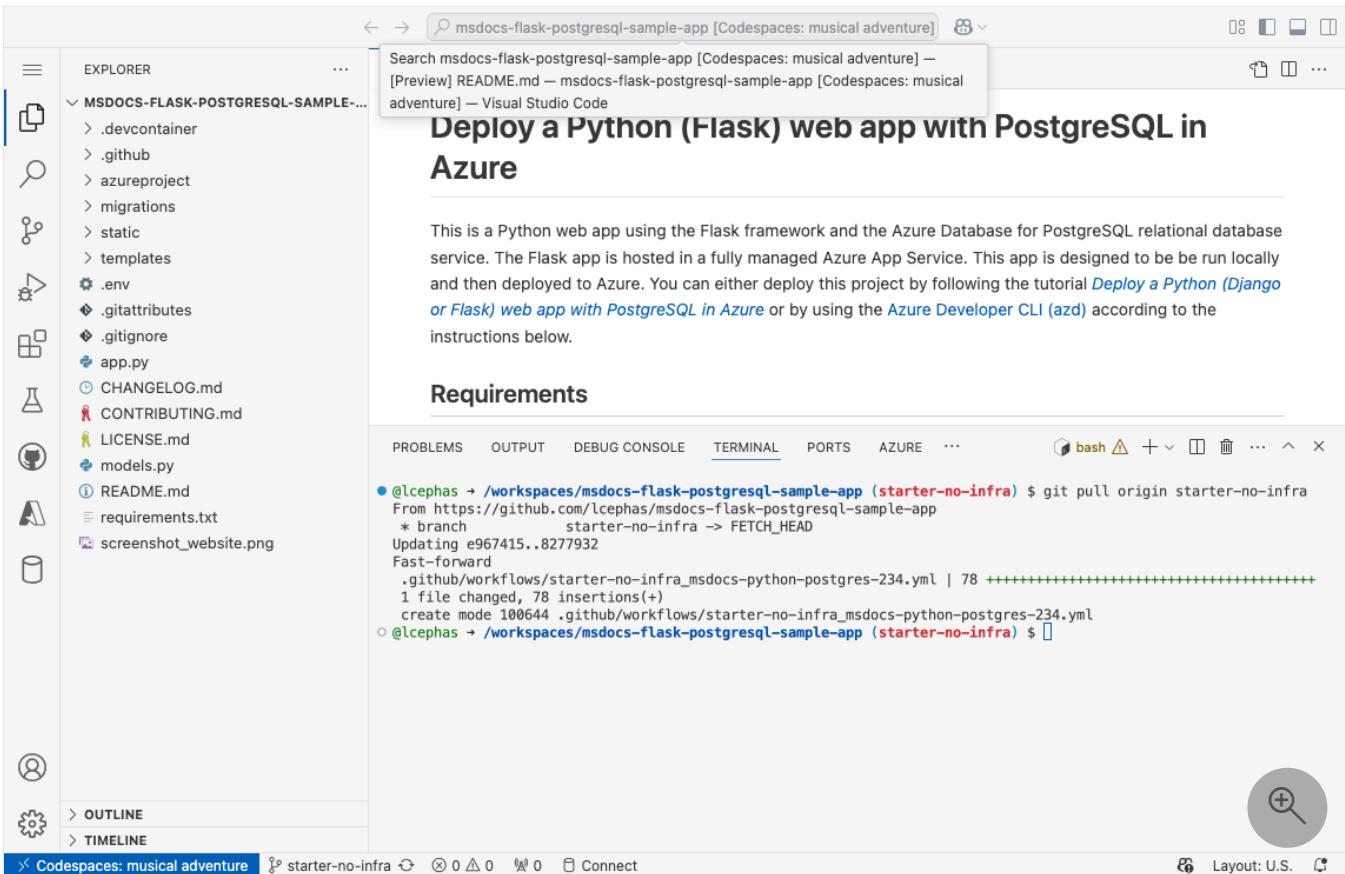


Paso 2: En la página Centro de implementación:

1. En **Origen**, seleccione **GitHub**. De forma predeterminada, **Acciones de GitHub** está seleccionado como proveedor de compilación.
2. Inicie sesión en su cuenta de GitHub y siga las indicaciones para autorizar a Azure.
3. En **Organización**, seleccione su cuenta.
4. En **Repositorio**, seleccione **msdocs-flask-postgresql-sample-app**.
5. En **Rama**, seleccione **starter-no-infra**. Esta es la misma rama en la que ha trabajado con la aplicación de ejemplo, sin archivos ni configuraciones relacionados con Azure.
6. En **Tipo de autenticación**, seleccione **Identidad asignada por el usuario**.
7. En el menú superior, elija **Guardar**. App Service confirma un archivo de flujo de trabajo en el repositorio de GitHub escogido, en el directorio `.github/workflows`. De manera predeterminada, el centro de implementación [crea una identidad asignada por el usuario](#) para que el flujo de trabajo se autentique mediante Microsoft Entra (autenticación OIDC). Para obtener opciones de autenticación alternativas, consulte [Implementación en App Service mediante Acciones de GitHub](#).

Screenshot of the Azure App Service configuration interface for GitHub Actions. The 'Source' dropdown is set to 'GitHub'. The 'Organization' dropdown is empty. The 'Repository' dropdown shows 'msdocs-flask-postgresql-sample-app'. The 'Branch' dropdown shows 'starter-no-infra'. Under 'Workflow Option', the 'Overwrite the workflow' option is selected. In the 'Build' section, 'Runtime stack' is set to 'Python' and 'Version' is 'Python 3.12'. In the 'Authentication settings' section, 'User-assigned identity' is selected. A search icon is visible in the top right corner.

Paso 3: De nuevo en el entorno de programación de GitHub de la bifurcación de ejemplo, ejecute `git pull origin starter-no-infra`. Esto extrae el archivo de flujo de trabajo recién confirmado en el codespace.



Paso 4 (Opción 1: con GitHub Copilot):

1. Inicie una nueva sesión de chat al seleccionar la vista **Chat** y, después, seleccionar +.
2. Pregunte: "*@workspace ¿Cómo se conecta la aplicación a la base de datos?*". Copilot puede proporcionar una explicación sobre `SQLAlchemy` cómo se configura su URI de conexión en `azureproject/development.py` y `azureproject/production.py`.
3. Pregunte: "*@workspace En modo de producción, mi aplicación se ejecuta en una aplicación web de App Service, que usa Azure Service Connector para conectarse a un servidor flexible de PostgreSQL mediante el tipo de cliente Django. ¿Cuáles son los nombres de las variables de entorno que necesito usar?*" Copilot puede proporcionarle una sugerencia de código similar a la de los pasos de la **Opción 2: sin GitHub Copilot** que se indican a continuación e incluso indicarle que realice el cambio en el archivo `azureproject/production.py`.
4. Abra `azureproject/production.py` en el explorador y agregue la sugerencia de código. GitHub Copilot no proporciona la misma respuesta cada vez y no siempre esta es correcta. Es posible que tenga que formular más preguntas para ajustar su respuesta. Para obtener sugerencias, consulte [¿Qué puedo hacer con GitHub Copilot en mi espacio de código?](#).



Ask Copilot

Copilot is powered by AI, so mistakes are possible.

Review output carefully before use.

As an internal user, additional telemetry is collected. If you work on a project that contains customer content, you must [disable telemetry](#).

⌚ or type # to attach context

@ to chat with extensions

Type / to use commands

[/help What can you do?](#)

@workspace How does the app connect to the database?

@ ⌚

GPT 4o ▾ ➤ ▾



Paso 4 (Opción 2: sin GitHub Copilot):

1. Abra `Program.cs` en el explorador.
2. Busque el código comentado (líneas 3-8) y quite la marca de comentario. Esto crea una cadena de conexión para SQLAlchemy mediante `AZURE_POSTGRESQL_USER`, `AZURE_POSTGRESQL_PASSWORD`, `AZURE_POSTGRESQL_HOST` y `AZURE_POSTGRESQL_NAME`.

```

1 import os
2
3 DATABASE_URI = 'postgresql+psycopg2://{}dbuser:{}@{}dbhost/{}dbname'.
4     dbuser=os.getenv('AZURE_POSTGRESQL_USER'),
5     dbpass=os.getenv('AZURE_POSTGRESQL_PASSWORD'),
6     dbhost=os.getenv('AZURE_POSTGRESQL_HOST'),
7     dbname=os.getenv('AZURE_POSTGRESQL_NAME')
8 )

```

The screenshot shows the Microsoft CodeSpace interface with the following details:

- EXPLORER** sidebar: Shows the project structure for "MSDOCS-FLASK-POSTGRESQL-SAMPLE". The "production.py" file is selected and highlighted with a red border.
- CODEVIEW**: The "production.py" file is open, displaying Python code for setting up a PostgreSQL database connection using environment variables.
- TERMINAL**: A terminal window is open, showing the command `git pull origin starter-no-infra` being run, followed by the output of the pull request.
- STATUS BAR**: Shows the current workspace ("Codespaces: musical adventure"), branch ("starter-no-infra"), and other system information like battery level and connectivity.

Paso 5:

1. Seleccione la extensión **Control de código fuente**.
2. En el cuadro de texto, escriba un mensaje de confirmación, por ejemplo, `Configure Azure database connection`. O bien, seleccione y deje que GitHub Copilot genere un mensaje de confirmación de manera automática.
3. Seleccione **Confirmar** y, a continuación, confirme con **Sí**.
4. Seleccione **Sincronización de cambios 1** y confirme con **Aceptar**.

Save Discard Browse Manage publish profile Sync Leave Feedback

Settings Logs FTPS credentials

Refresh Delete

Time	Commit ID	Logs	Commit Author	Status	Message
Wednesday, January 22, 2025 (4)					
01/22/2025, 3:26:04..	Ea55fb8	Build/Deploy Lo...	lcephas	In Progress....	Configure Azure database connection
01/22/2025, 3:08:13..	80c37f1	App Logs	N/A	Failed	OneDeploy
01/22/2025, 3:08:00..	temp-9d	App Logs	N/A	Failed	OneDeploy
01/22/2025, 3:05:24..	8277932	Build/Deploy Lo...	lcephas	Failed	Add or update the Azure App Service build and deployment workflow config

+
🔍

Paso 6: De vuelta en la página Centro de implementación en Azure Portal:

1. Seleccione la pestaña **Registros** y, a continuación, seleccione **Actualizar** para ver la nueva ejecución de implementación.
2. En el elemento de registro de la ejecución de implementación, seleccione la entrada **Registros de compilación/implementación** con la marca de tiempo más reciente.

Save Discard Browse Manage publish profile Sync Leave Feedback

Settings Logs FTPS credentials

Refresh Delete

Time	Commit ID	Logs	Commit Author	Status	Message
Wednesday, January 22, 2025 (4)					
01/22/2025, 3:26:04..	Ea55fb8	Build/Deploy Lo...	lcephas	In Progress....	Configure Azure database connection
01/22/2025, 3:08:13..	80c37f1	App Logs	N/A	Failed	OneDeploy
01/22/2025, 3:08:00..	temp-9d	App Logs	N/A	Failed	OneDeploy
01/22/2025, 3:05:24..	8277932	Build/Deploy Lo...	lcephas	Failed	Add or update the Azure App Service build and deployment workflow config

+
🔍

Paso 7: Se le llevará al repositorio de GitHub, donde ve que la acción de GitHub se está ejecutando. El archivo de flujo de trabajo define dos fases independientes: compilación e implementación. Espere a que la ejecución en GitHub muestre el estado **Éxito**. Tardará unos 5 minutos.

The screenshot shows the GitHub Actions interface for a repository named 'msdocs-flask-postgresql-sample-app'. The 'Actions' tab is selected. A workflow run titled 'Configure Azure database connecton #2' is shown, triggered via push 5 minutes ago. The status is 'Success' with a total duration of '4m 12s' and 1 artifact. The workflow file is 'starter-no-infra_msdocs-python-postgres-234.yml' and it includes a 'build' step (25s) and a 'deploy' step (3m 27s) which resulted in the URL <http://msdocs-python-postgres-234-e...>. On the left sidebar, 'Summary' is selected, along with 'build' and 'deploy' under 'Jobs'. Other options like 'Run details', 'Usage', and 'Workflow file' are also visible.

¿Tiene problemas? Consulte la [Guía de solución de problemas](#).

5. Generar esquema de base de datos

Con la base de datos PostgreSQL protegida por la red virtual, la manera más fácil de ejecutar [migraciones de base de datos de Flask](#) es mediante una sesión SSH con el contenedor de Linux en App Service.

Paso 1: De vuelta en la página App Service, en el menú de la izquierda,

1. Seleccione **Herramientas de desarrollo>SSH**.
2. Seleccione **Continuar**.

Microsoft Azure (Preview)  Search resources, services, and docs (G+/)

Home > msdocs-python-postgres-234

msdocs-python-postgres-234 | SSH

App Service

Search

Quotas

Change App Service plan

Development Tools

Clone App

SSH 

Advanced Tools

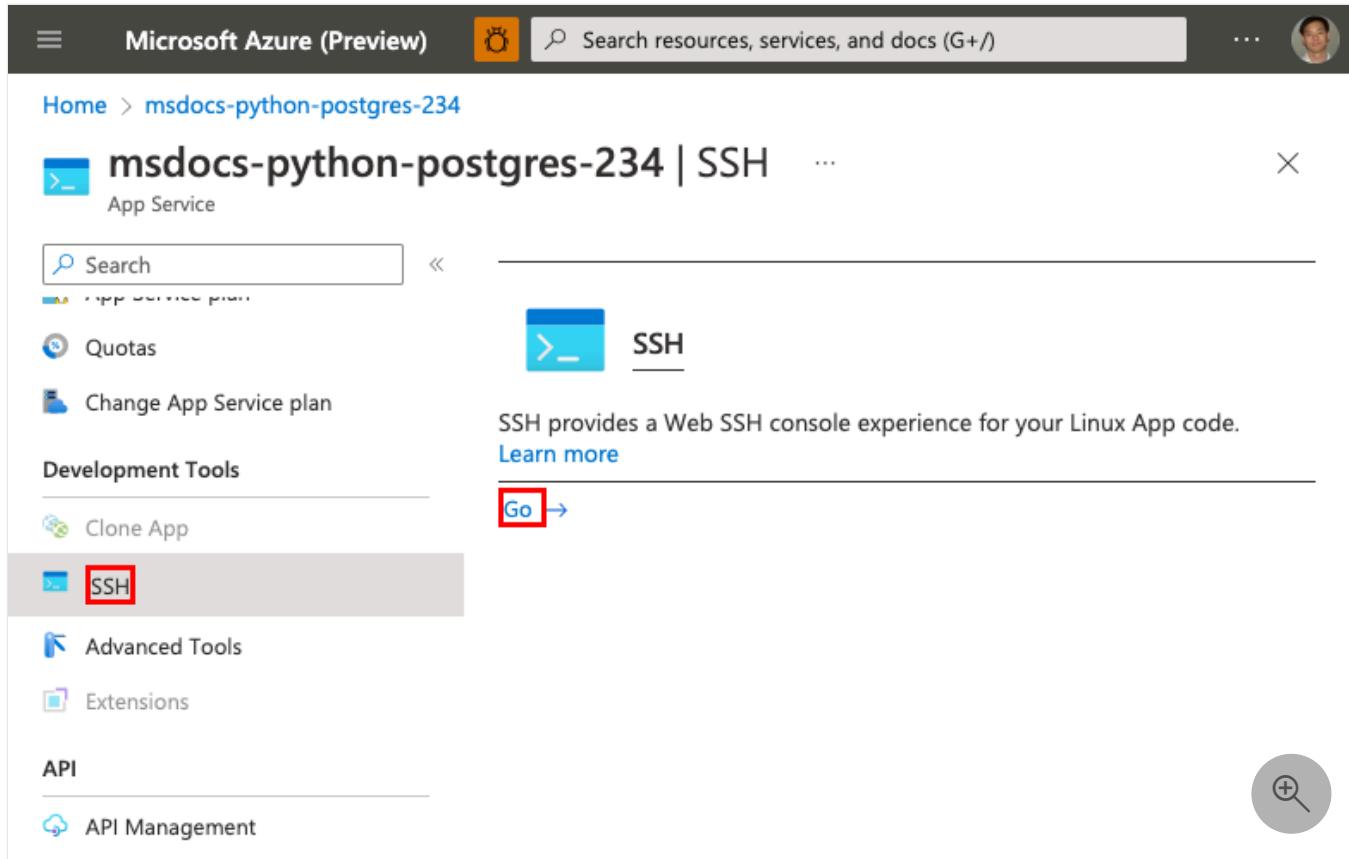
Extensions

API

API Management

SSH provides a Web SSH console experience for your Linux App code.
[Learn more](#)

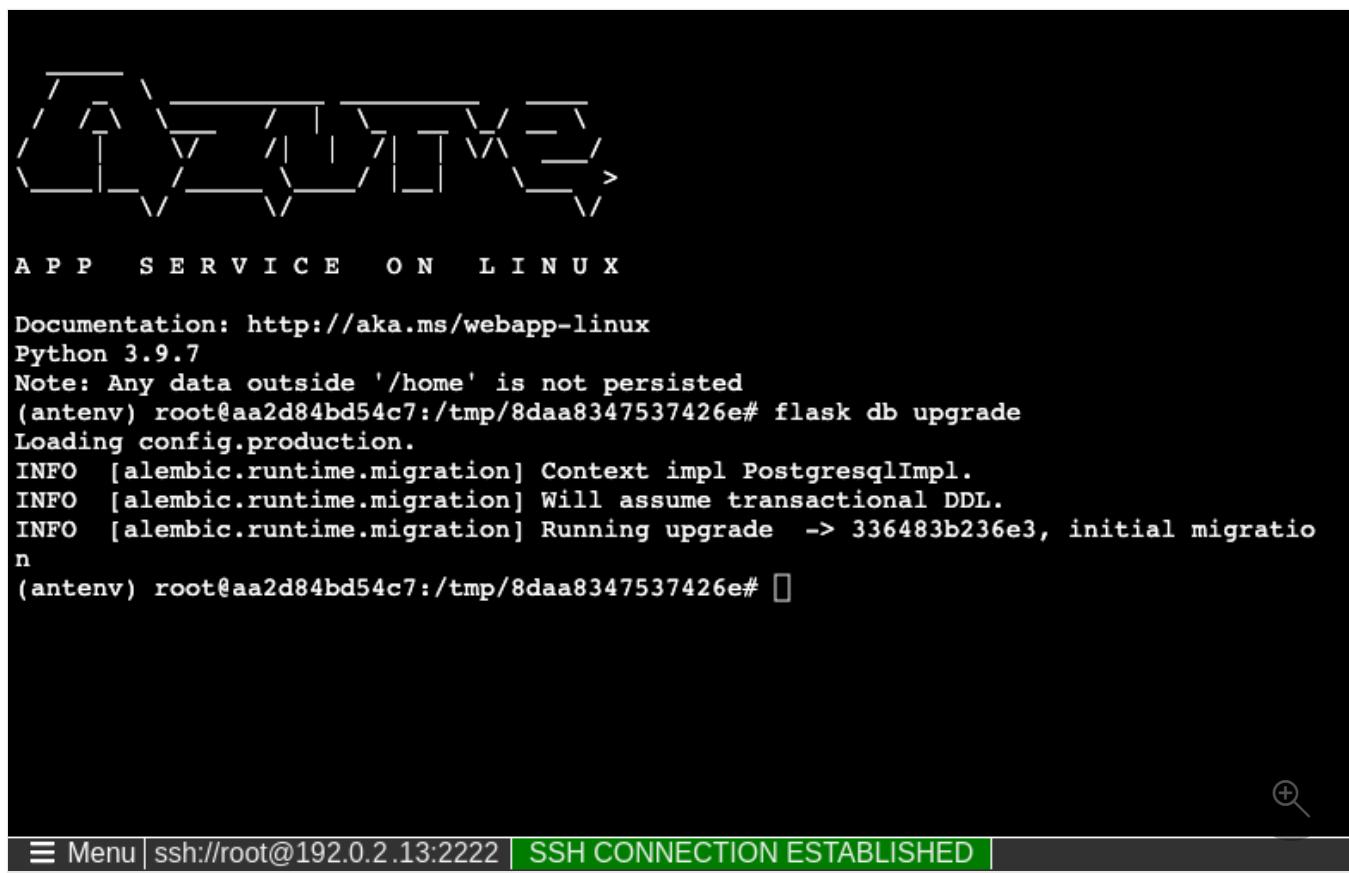
Go → 



Paso 2: En la sesión SSH, ejecute `flask db upgrade`. Si se realiza correctamente, App Service **se conecta a la base de datos**.

```
APP SERVICE ON LINUX

Documentation: http://aka.ms/webapp-linux
Python 3.9.7
Note: Any data outside '/home' is not persisted
(antenv) root@aa2d84bd54c7:/tmp/8daa8347537426e# flask db upgrade
Loading config.production.
INFO [alembic.runtime.migration] Context impl PostgresqlImpl.
INFO [alembic.runtime.migration] Will assume transactional DDL.
INFO [alembic.runtime.migration] Running upgrade  -> 336483b236e3, initial migration
(antenv) root@aa2d84bd54c7:/tmp/8daa8347537426e# 
```



 Sugerencia

En la sesión SSH, solo los cambios en los archivos de `/home` pueden persistir más allá de los reinicios de la aplicación. Los cambios efectuados fuera de `/home` no se conservan.

¿Tiene problemas? Consulte la sección [Solución de problemas](#).

6. Navegación hasta la aplicación

Paso 1. En la página App Service:

1. En el menú de la izquierda, seleccione **Información general**.
2. Seleccione la dirección URL de la aplicación.

The screenshot shows the Azure App Service Overview page. On the left, there's a sidebar with links like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Microsoft Defender for Cloud, Events (preview), Deployment slots, and Deployment Center. The main area has a search bar at the top and several actions: Browse, Stop, Swap, Restart, Delete, and three dots. Below that is a 'Essentials' section with JSON View. The details listed are:

Resource group (move)	: msdocs-python-postgres-tutorial
Status	: Running
Location (move)	: East US
Subscription (move)	:
Subscription ID	:
Default domain	: msdocs-python-postgres-125.azurewebsites...
App Service Plan	: ASP-msdocspythonpostrestutorial-b709 (B1)
Operating System	: Linux
Health Check	: Error fetching health check data. Please try again later.

Paso 2: agregue algunos restaurantes a la lista. Enhorabuena, ya está ejecutando una aplicación web en Azure App Service, con conectividad protegida a Azure Database for PostgreSQL.

Restaurants

Name	Rating	Details
Fourth Coffee	★★	2.0 (2 reviews) Details
Contoso Restaurant	★★★★	4.0 (3 reviews) Details

[Add new restaurant](#)

7. Transmisión de registros de diagnóstico

Azure App Service captura todos los registros de consola para ayudarle a diagnosticar problemas con la aplicación. La aplicación de ejemplo incluye instrucciones `print()` para demostrar esta funcionalidad, como se muestra a continuación.

Python

```
@app.route('/', methods=['GET'])
def index():
    print('Request for index page received')
    restaurants = Restaurant.query.all()
    return render_template('index.html', restaurants=restaurants)
```

Paso 1. En la página App Service:

1. En el menú de la izquierda, seleccione **Supervisión>Registros de App Service**.
2. En **Registro de aplicaciones**, seleccione **Sistema de archivos**.
3. En el menú superior, elija **Guardar**.

Microsoft Azure (Preview) Search resources, services, and docs (G+/)

Home > Microsoft.Web-WebAppDatabase-Portal-d27b8120-a643 | Overview > msdocs-python-postgres-234

msdocs-python-postgres-234 | App Service logs

App Service

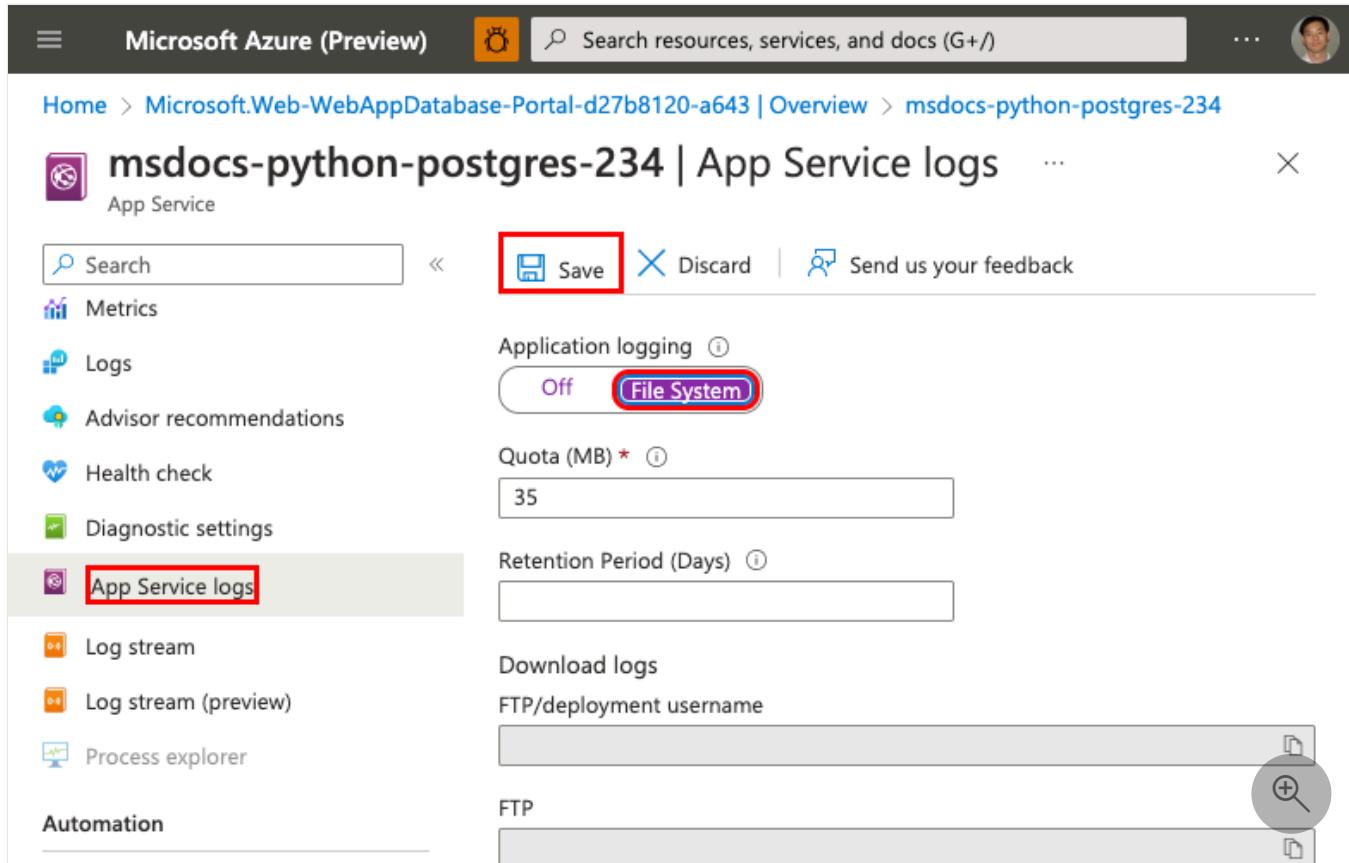
Search Save Discard Send us your feedback

Metrics Logs Advisor recommendations Health check Diagnostic settings App Service logs Log stream Log stream (preview) Process explorer Automation

Application logging Off File System Quota (MB) * 35 Retention Period (Days)

Download logs FTP/deployment username

FTP



Paso 2: En el menú de la izquierda, seleccione Flujo de registro. Verá los registros de la aplicación, incluidos los registros de plataforma y los registros de dentro del contenedor.

Microsoft Azure (Preview) Search resources, services, and docs (G+/)

Home > Microsoft.Web-WebAppDatabase-Portal-d27b8120-a643 | Overview > msdocs-python-postgres-234

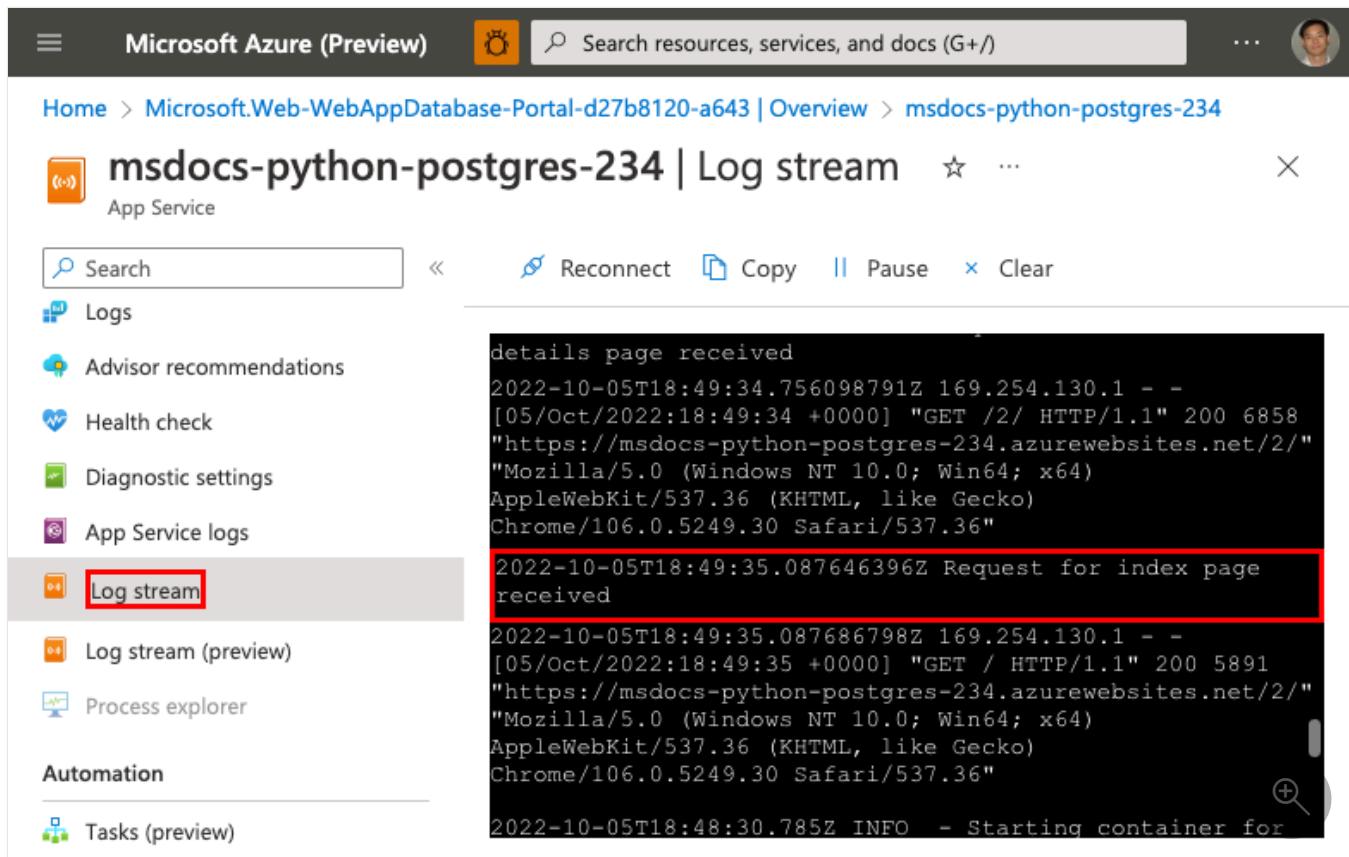
msdocs-python-postgres-234 | Log stream

App Service

Search Reconnect Copy Pause Clear

Logs Advisor recommendations Health check Diagnostic settings App Service logs Log stream Log stream (preview) Process explorer Automation Tasks (preview)

```
details page received
2022-10-05T18:49:34.756098791Z 169.254.130.1 - -
[05/Oct/2022:18:49:34 +0000] "GET /2/ HTTP/1.1" 200 6858
"https://msdocs-python-postgres-234.azurewebsites.net/2/"
Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/106.0.5249.30 Safari/537.36"
2022-10-05T18:49:35.087646396Z Request for index page
received
2022-10-05T18:49:35.087686798Z 169.254.130.1 - -
[05/Oct/2022:18:49:35 +0000] "GET / HTTP/1.1" 200 5891
"https://msdocs-python-postgres-234.azurewebsites.net/2/"
Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/106.0.5249.30 Safari/537.36"
2022-10-05T18:48:30.785Z INFO - Starting container for
```



Obtenga más información sobre cómo iniciar sesión en aplicaciones de Python en la serie sobre [cómo configurar Azure Monitor para la aplicación Python](#).

8. Limpieza de recursos

Cuando acabe, puede eliminar todos los recursos de la suscripción de Azure mediante la eliminación del grupo de recursos.

Paso 1: En la barra de búsqueda de la parte superior de Azure Portal:

1. Escriba el nombre del grupo de recursos.
2. Seleccione el grupo de recursos.

The screenshot shows the Microsoft Azure portal interface. At the top, there is a search bar with the text "msdocs-flask-postgres-tutorial". Below the search bar, the "Resource Groups" section displays a single item: "msdocs-flask-postgres-tutorial", which is highlighted with a red box. The "Azure services" sidebar on the left lists "Create a resource", "Key vaults", and "Virtual networks". The main content area shows documentation links related to Flask and PostgreSQL.

Paso 2: En la página del grupo de recursos, seleccione **Borrar grupo de recursos**.

[Create](#) [Manage view](#) [Delete resource group](#) ...

[View Cost](#) | [JSON View](#)

Essentials

Subscription ([move](#)) : [Antares-Demo](#)

Subscription ID :

Deployments : [7 Succeeded](#)

Location : West Europe

Tags ([edit](#)) : [Click here to add tags](#)

Resources Recommendations (1)

Filter for any field... [Add filter](#) [More \(2\)](#)

Showing 1 to 5 of 5 records. Show hidden types [?](#)

No grouping [List view](#)

Paso 3:

1. Escriba el nombre del grupo de recursos para confirmar la eliminación.
2. Seleccione Eliminar.
3. Confirme con Eliminar de nuevo.

 Are you sure you want to delete "msdoc..." [X](#)

Warning! Deleting the "msdocs-python-postgres-tutorial" resource group is irreversible. The action you're about to take can't be undone. Going further will delete this resource group and all the resources in it permanently.

TYPE THE RESOURCE GROUP NAME:
 

AFFECTED RESOURCES
There are 6 resources in this resource group that will be deleted.

Name	Type	Location
 ASP-msdocspythonpostgres-tut...	App Service plan	West Europe
 msdocs-python-postgres-234	App Service	West Europe
 msdocs-python-postgres-234-s...	Azure Database for ...	West Europe

[Delete](#) [Cancel](#) 

Solución de problemas

A continuación se indican los problemas que pueden surgir al intentar trabajar con este tutorial y los pasos para resolverlos.

No puedo conectarme a la sesión de SSH

Si no puede conectarse a la sesión SSH, significa que no se pudo iniciar la propia aplicación. Compruebe los [registros de diagnóstico](#) para obtener más información. Por ejemplo, si ve un error como `KeyError: 'AZURE_POSTGRESQL_HOST'`, puede significar que falta la variable de entorno (es posible que haya quitado el valor de configuración de la aplicación).

Recibo un error al ejecutar migraciones de base de datos

Si encuentra algún error relacionado con la conexión a la base de datos, compruebe si se han cambiado o eliminado los valores de configuración de la aplicación (`AZURE_POSTGRESQL_USER`, `AZURE_POSTGRESQL_PASSWORD`, `AZURE_POSTGRESQL_HOST` y `AZURE_POSTGRESQL_NAME`). Sin esa cadena de conexión, el comando de migración no se puede comunicar con la base de datos.

Preguntas más frecuentes

- [¿Cuánto cuesta esta configuración?](#)
- [¿Cómo me conecto al servidor de PostgreSQL protegido tras la red virtual con otras herramientas?](#)
- [¿Cómo funciona el desarrollo de aplicaciones locales con Acciones de GitHub?](#)
- [¿Cómo se depuran los errores durante la implementación de Acciones de GitHub?](#)
- [No tengo permisos para crear una identidad asignada por el usuario](#)
- [¿Qué puedo hacer con GitHub Copilot en mi codespace?](#)

¿Cuánto cuesta esta configuración?

Los precios por los recursos creados son los siguientes:

- El plan de App Service se crea en el nivel **Básico** y puede escalar o reducirse verticalmente. Consulte [Precios de App Service](#).
- El servidor flexible de PostgreSQL se crea en el nivel ampliable más bajo (**Standard_B1ms**), con el tamaño de almacenamiento mínimo, que se puede escalar verticalmente. Consulte los [precios de Azure Database for PostgreSQL](#).
- La red virtual no incurre en ningún cargo, a menos que se configure una funcionalidad extra, como el emparejamiento de redes. Vea [Precios de Azure Virtual Network](#).

- La zona DNS privada conlleva un pequeño cargo. Vea [Precios de Azure DNS](#).

¿Cómo me conecto al servidor de PostgreSQL protegido tras la red virtual con otras herramientas?

- Para tener un acceso básico desde una herramienta de línea de comandos, puede ejecutar `psql` desde el terminal de SSH de la aplicación.
- Para conectarse desde una herramienta de escritorio, la máquina debe estar en la red virtual. Por ejemplo, podría ser una máquina virtual de Azure conectada a una de las subredes, o una máquina de una red local que tenga una conexión [VPN de sitio a sitio](#) a la red virtual de Azure.
- También puede [integrar Azure Cloud Shell](#) en la red virtual.

¿Cómo funciona el desarrollo de aplicaciones locales con Acciones de GitHub?

Tomando como ejemplo el archivo de flujo de trabajo generado automáticamente de App Service: cada `git push` inicia una nueva ejecución de compilación e implementación. Desde un clon local del repositorio de GitHub, realiza las actualizaciones deseadas y las inserta en GitHub. Por ejemplo:

terminal

```
git add .
git commit -m "<some-message>"
git push origin main
```

¿Cómo se depuran los errores durante la implementación de Acciones de GitHub?

Si un paso falla en el archivo de flujo de trabajo de GitHub generado automáticamente, intente modificar el comando que falló para generar una salida más detallada. Por ejemplo, puede obtener más resultados del comando `python` agregando la opción `-d`. Confirme e inserte los cambios para desencadenar otra implementación en App Service.

No tengo permisos para crear una identidad asignada por el usuario

Consulte [Configuración de la implementación de Acciones de GitHub desde el centro de implementación](#).

¿Qué puedo hacer con GitHub Copilot en mi codespace?

Es posible que haya observado que la vista de chat de GitHub Copilot ya estaba allí cuando creó el codespace. Para mayor comodidad, incluimos la extensión de chat de GitHub Copilot en la definición del contenedor (consulte [.devcontainer/devcontainer.json](#)). Sin embargo, necesita una [cuenta de GitHub Copilot](#) (versión de prueba gratuita de 30 días disponible).

Algunas sugerencias para usted al hablar con GitHub Copilot:

- En una sola sesión de chat, las preguntas y respuestas se basan entre sí y puede ajustar sus preguntas para ajustar la respuesta que obtenga.
- De forma predeterminada, GitHub Copilot no tiene acceso a ningún archivo del repositorio. Para formular preguntas sobre un archivo, abra primero el archivo en el editor.
- Para permitir que GitHub Copilot tenga acceso a todos los archivos del repositorio al preparar sus respuestas, comience la pregunta con `@workspace`. Para obtener más información, vea [Use the @workspace agent](#).
- En la sesión de chat, GitHub Copilot puede sugerir cambios y (con `@workspace`) incluso donde realizar los cambios, pero no está permitido que haga los cambios por ti. Es necesario agregar los cambios sugeridos y probarlos.

Pasos siguientes

Pase al tutorial siguiente para aprender a proteger la aplicación con un dominio personalizado y un certificado.

[Protección con un certificado y dominio personalizado](#)

Obtenga información sobre cómo App Service ejecuta una aplicación de Python:

[Configuración de una aplicación de Python](#)

Creación e implementación de una aplicación web de Python de Flask en Azure con una identidad administrada asignada por el sistema

Artículo • 30/09/2024

En este tutorial, se implementa código Python [Flask](#) para crear e implementar una aplicación web que se ejecuta en Azure App Service. La aplicación web utiliza su **identidad administrada** asignada por el sistema (conexiones sin contraseña) con el control de acceso basado en roles de Azure para acceder a los recursos [Azure Storage](#) y [Azure Database for PostgreSQL - Servidor Flexible](#). El código utiliza la clase `DefaultAzureCredential` de la [biblioteca de clientes Azure Identity](#) para Python. La clase `DefaultAzureCredential` detecta automáticamente que existe una identidad administrada para el App Service y la utiliza para acceder a otros recursos de Azure.

Puede configurar conexiones sin contraseña a servicios Azure usando Service Connector o puede configurarlas manualmente. Este tutorial muestra cómo se usa Service Connector. Para obtener más información sobre las conexiones sin contraseña, consulte [Conexiones sin contraseña para servicios de Azure](#). Para obtener información sobre Service Connector, consulte la [documentación de Service Connector](#).

Este tutorial muestra cómo crear e implementar una aplicación web de Python utilizando la CLI de Azure. Los comandos de este tutorial están escritos para ejecutarse en un shell Bash. Puede ejecutar los comandos del tutorial en cualquier entorno Bash con la CLI instalada, como su entorno local o [Azure Cloud Shell](#). Con algunas modificaciones —por ejemplo, estableciendo y usando variables de entorno—, puede ejecutar estos comandos en otros entornos como el shell de comandos de Windows. Para ver ejemplos de uso de una identidad gestionada asignada por el usuario, consulte [Creación e implementación una aplicación web Django en Azure con una identidad administrada asignada por el usuario](#).

Obtención de la aplicación de ejemplo

Hay disponible una aplicación Python de ejemplo que utiliza el framework Flask que le ayudará a seguir este tutorial. Descargue o clone una de las aplicaciones de ejemplo en la estación de trabajo local.

1. Clone el ejemplo en una sesión de Azure Cloud Shell.

Consola

```
git clone https://github.com/Azure-Samples/msdocs-flask-web-app-managed-identity.git
```

2. Vaya a la carpeta de la aplicación.

Consola

```
cd msdocs-flask-web-app-managed-identity
```

Examine el código de autenticación

La aplicación web de ejemplo necesita autenticarse en dos almacenes de datos diferentes:

- El servidor de almacenamiento blob de Azure, donde almacena y recupera las fotos enviadas por los revisores.
- Una base de datos Azure Database for PostgreSQL - Servidor Flexible donde almacena los restaurantes y las reseñas.

Usa [DefaultAzureCredential](#) para autenticarse en ambos almacenes de datos. Con `DefaultAzureCredential`, la aplicación puede configurarse para ejecutarse bajo la identidad de diferentes principales de servicio, dependiendo del entorno en el que se esté ejecutando, sin realizar cambios en el código. Por ejemplo, en un entorno de desarrollo local, la aplicación puede ejecutarse con la identidad del desarrollador que ha iniciado sesión en la CLI de Azure, mientras que en Azure, como en este tutorial, puede ejecutarse con la identidad gestionada asignada por el sistema.

En cualquier caso, el principal de seguridad bajo el que se ejecuta la aplicación debe tener un rol en cada recurso Azure que la aplicación utiliza que le permita realizar las acciones en el recurso que la aplicación requiere. En este tutorial, utilizará conectores de servicio para habilitar automáticamente la identidad administrada asignada por el sistema en su aplicación en Azure y para asignar a esa identidad los roles adecuados en su cuenta de almacenamiento de Azure y en el servidor Azure Database for PostgreSQL.

Una vez habilitada la identidad administrada asignada por el sistema y asignados los roles adecuados en los almacenes de datos, puede utilizar `DefaultAzureCredential` para autenticarse con los recursos de Azure necesarios.

El siguiente código se utiliza para crear un cliente de almacenamiento blob para subir fotos en `app.py`. Se proporciona una instancia de `DefaultAzureCredential` al cliente, que

usa para adquirir tokens de acceso para realizar operaciones en el almacenamiento de Azure.

Python

```
from azure.identity import DefaultAzureCredential
from azure.storage.blob import BlobServiceClient

azure_credential = DefaultAzureCredential()
blob_service_client = BlobServiceClient(
    account_url=account_url,
    credential=azure_credential)
```

También se utiliza una instancia de `DefaultAzureCredential` para obtener un token de acceso para Azure Database for PostgreSQL en `./azureproject/get_conn.py`. En este caso, el token se adquiere directamente llamando a `get_token` en la instancia de credenciales y pasándole el valor `scope` apropiado. A continuación, se usa el token en lugar de la contraseña en el URI de conexión PostgreSQL devuelto al autor de la llamada.

Python

```
azure_credential = DefaultAzureCredential()
token = azure_credential.get_token("https://osssrdbms-
aad.database.windows.net")
conn =
str(current_app.config.get('DATABASE_URI')).replace('PASSWORDORTOKEN',
token.token)
```

Para obtener más información sobre la autenticación de sus aplicaciones con los servicios Azure, consulte [Autenticación de aplicaciones Python en los servicios Azure mediante el SDK de Azure para Python](#). Para obtener más información sobre `DefaultAzureCredential`, incluido cómo personalizar la cadena de credenciales que evalúa para su entorno, consulte [Introducción a DefaultAzureCredential](#).

Crear un servidor Azure PostgreSQL

1. Configure las variables de entorno necesarias para el tutorial.

Bash

```
LOCATION="eastus"
RAND_ID=$RANDOM
RESOURCE_GROUP_NAME="msdocs-mi-web-app"
APP_SERVICE_NAME="msdocs-mi-web-$RAND_ID"
```

```
DB_SERVER_NAME="msdocs-mi-postgres-$RAND_ID"
ADMIN_USER="demoadmin"
ADMIN_PW="ChAnG33#ThsPssWd$RAND_ID"
```

ⓘ Importante

La `ADMIN_PW` debe contener entre 8 y 128 caracteres de tres de las siguientes categorías: Letras del alfabeto inglés mayúsculas y minúsculas, números y caracteres no alfanuméricos. Al crear nombres de usuario o contraseñas, **no** use el carácter `$`. Más adelante creará variables de entorno con estos valores, donde el carácter `$` tiene un significado especial dentro del contenedor de Linux que se usa para ejecutar aplicaciones de Python.

2. Para crear un grupo de recursos, use el comando [az group create](#).

Azure CLI

```
az group create --location $LOCATION --name $RESOURCE_GROUP_NAME
```

3. Cree un servidor PostgreSQL con el comando [az postgres flexible-server create](#). (Este comando y los siguientes usan el carácter de continuación de línea de Bash Shell ('\'). Cambie el carácter de continuación de línea para su shell si es necesario).

Azure CLI

```
az postgres flexible-server create \
--resource-group $RESOURCE_GROUP_NAME \
--name $DB_SERVER_NAME \
--location $LOCATION \
--admin-user $ADMIN_USER \
--admin-password $ADMIN_PW \
--sku-name Standard_D2ds_v4
```

El *sku-name* es el nombre del nivel de precios y de la configuración informática. Para más información, consulte los [precios de Azure Database for PostgreSQL](#). Para enumerar las SKU disponibles, use `az postgres flexible-server list-skus --location $LOCATION`.

4. Cree una base de datos denominada `restaurant` mediante el comando [az postgres flexible-server execute](#).

Azure CLI

```
az postgres flexible-server execute \
  --name $DB_SERVER_NAME \
  --admin-user $ADMIN_USER \
  --admin-password $ADMIN_PW \
  --database-name postgres \
  --querytext 'create database restaurant;'
```

Creación de un Azure App Service e implementación del código

1. Cree un servicio de aplicación con el comando [az webapp up](#).

Azure CLI

```
az webapp up \
  --resource-group $RESOURCE_GROUP_NAME \
  --name $APP_SERVICE_NAME \
  --runtime PYTHON:3.9 \
  --sku B1
```

El *sku* define el tamaño (CPU, memoria) y el coste del plan de App Service. El plan de servicio B1 (Básico) incurre en un pequeño coste en su suscripción Azure. Para obtener una lista completa de los planes de App Service, vea la página [Precios de App Service](#) ↗.

2. Configura App Service para usar el *start.sh* en el repo con el comando [az webapp config set](#).

Azure CLI

```
az webapp config set \
  --resource-group $RESOURCE_GROUP_NAME \
  --name $APP_SERVICE_NAME \
  --startup-file "start.sh"
```

Creación de conectores sin contraseña a los recursos de Azure

Los comandos Service Connector configuran los recursos Azure Storage y Azure Database for PostgreSQL para usar la identidad administrada y el control de acceso basado en roles de Azure. Los comandos crean configuraciones de aplicación en App Service que conectan su aplicación web a estos recursos. La salida de los comandos

enumera las acciones del conector de servicio realizadas para habilitar la capacidad sin contraseña.

1. Agregue un conector de servicio PostgreSQL con el comando [az webapp connection create postgres-flexible](#). La identidad administrada asignada por el sistema se usa para autenticar la aplicación web en el recurso de destino, PostgreSQL, en este caso.

Azure CLI

```
az webapp connection create postgres-flexible \
--resource-group $RESOURCE_GROUP_NAME \
--name $APP_SERVICE_NAME \
--target-resource-group $RESOURCE_GROUP_NAME \
--server $DB_SERVER_NAME \
--database restaurant \
--client-type python \
--system-identity
```

2. Agregue un conector de servicio de almacenamiento con el comando [az webapp connection create storage-blob](#).

Este comando también agrega una cuenta de almacenamiento y añade la aplicación web con el rol *Colaborador de datos de blobs de almacenamiento* a la cuenta de almacenamiento.

Azure CLI

```
STORAGE_ACCOUNT_URL=$(az webapp connection create storage-blob \
--new true \
--resource-group $RESOURCE_GROUP_NAME \
--name $APP_SERVICE_NAME \
--target-resource-group $RESOURCE_GROUP_NAME \
--client-type python \
--system-identity \
--query configurations[].value \
--output tsv)
STORAGE_ACCOUNT_NAME=$(cut -d . -f1 <<< $(cut -d / -f3 <<<
$STORAGE_ACCOUNT_URL))
```

Creación de un contenedor en la cuenta de almacenamiento

La aplicación Python de ejemplo almacena las fotos enviadas por los revisores como blobs en un contenedor de su cuenta de almacenamiento.

- Cuando un usuario envía una foto con su revisión, la aplicación de ejemplo escribe la imagen en el contenedor utilizando su identidad administrada asignada por el sistema para la autenticación y autorización. Esta funcionalidad se configuró en la última sección.
- Cuando un usuario consulta las reseñas de un restaurante, la aplicación devuelve un enlace a la foto en el Blob Storage para cada reseña que tenga una asociada. Para que el navegador muestre la foto, debe poder acceder a ella en su cuenta de almacenamiento. Los datos blob deben estar disponibles para su lectura pública a través de un acceso anónimo (no autenticado).

Para mejorar la seguridad, las cuentas de almacenamiento se crean con el acceso anónimo a los datos blob desactivado por defecto. En esta sección, habilitará el acceso anónimo de lectura en su cuenta de almacenamiento y luego creará un contenedor llamado *fotos* que proveerá acceso público (anónimo) a sus blobs.

1. Actualice la cuenta de almacenamiento para permitir el acceso anónimo de lectura a los blobs con el comando [az storage account update](#).

Azure CLI

```
az storage account update \
    --name $STORAGE_ACCOUNT_NAME \
    --resource-group $RESOURCE_GROUP_NAME \
    --allow-blob-public-access true
```

La habilitación del acceso anónimo en la cuenta de almacenamiento no afecta al acceso a blobs individuales. Debe habilitar explícitamente el acceso público a los blobs a nivel de contenedor.

2. Cree un contenedor llamado *fotos* en la cuenta de almacenamiento con el comando [az storage container create](#). Permita el acceso anónimo de lectura (público) a los blobs en el contenedor recién creado.

Azure CLI

```
az storage container create \
    --account-name $STORAGE_ACCOUNT_NAME \
    --name photos \
    --public-access blob \
    --account-key $(az storage account keys list --account-name
$STORAGE_ACCOUNT_NAME \
    --query [0].value --output tsv)
```

Nota

Para abreviar, este comando utiliza la clave de cuenta de almacenamiento para autorizar con la cuenta de almacenamiento. Para la mayoría de los escenarios, el enfoque recomendado por Microsoft es usar Microsoft Entra ID y roles Azure (RBAC). Para obtener un conjunto rápido de instrucciones, consulte [Inicio rápido: Crear, descargar y listar blobs con Azure CLI](#). Tenga en cuenta que varios roles de Azure le permiten crear contenedores en una cuenta de almacenamiento, incluidos "Propietario", "Colaborador", "Propietario de datos de bloques de almacenamiento" y "Colaborador de datos de bloques de almacenamiento".

Para obtener más información sobre el acceso de lectura anónimo a los datos de blob, consulte [Configuración del acceso de lectura anónimo para contenedores y blobs](#).

Prueba de la aplicación web Python en Azure

La aplicación Python de ejemplo utiliza el paquete [azure.identity](#) y su clase `DefaultAzureCredential`. Cuando la app se ejecuta en Azure, `DefaultAzureCredential` detecta automáticamente si existe una identidad administrada para el App Service y, si es así, la usa para acceder a otros recursos de Azure (almacenamiento y PostgreSQL en este caso). No es necesario proporcionar claves de almacenamiento, certificados o credenciales al App Service para acceder a estos recursos.

1. Vaya a la aplicación implementada en la dirección URL

`http://$APP_SERVICE_NAME.azurewebsites.net.`

La aplicación puede tardar uno o dos minutos en iniciarse. Si ve una página de aplicación predeterminada que no es la página de aplicación de muestra predeterminada, espere un minuto y actualice el navegador.

2. Pruebe la funcionalidad de la aplicación de ejemplo y añada un restaurante y algunas reseñas con fotos del restaurante.

La información del restaurante y las reseñas se almacenan en Azure Database for PostgreSQL y las fotos se almacenan en Azure Storage. He aquí una captura de pantalla de ejemplo:

Contoso Café

Street address: 1 Main Street
Description: Nice coffee house.
Rating: ★★★★ 4.5 (2 reviews)

Reviews

Add new review

Date	User	Rating	Review	Photo
May 20, 2022, 10:12 a.m.	Davide Sagese	5	Friendly staff, good coffee.	
May 23, 2022, 5:24 p.m.	Francesca Lombo	4	Good breakfast choice.	

Limpiar

En este tutorial, todos los recursos Azure se crearon en el mismo grupo de recursos. Eliminar el grupo de recursos con el comando `az group delete` elimina todos los recursos del grupo de recursos y es la forma más rápida de eliminar todos los recursos de Azure utilizados para su app.

Azure CLI

```
az group delete --name $RESOURCE_GROUP_NAME
```

Opcionalmente, puede agregar el argumento `--no-wait` para permitir que el comando devuelva antes de que se complete la operación.

Pasos siguientes

- Creación e implementación de una aplicación web de Django en Azure con una identidad administrada asignada por el usuario
- Implementación de una aplicación web Python (Django o Flask) con PostgreSQL en Azure App Service

Comentarios

¿Le ha resultado útil esta página?

👍 Sí

👎 No

Creación e implementación de una aplicación web de Django en Azure con una identidad administrada asignada por el usuario

Artículo • 30/09/2024

En este tutorial se implementa una aplicación web [Django](#) en Azure App Service. La aplicación web utiliza una **identidad administrada** asignada por el usuario (conexiones sin contraseña) con control de acceso basado en roles de Azure para acceder a los recursos [Azure Storage](#) y [Azure Database for PostgreSQL - Flexible Server](#). El código utiliza la clase `DefaultAzureCredential` de la biblioteca de clientes [Azure Identity](#) para Python. La clase `DefaultAzureCredential` detecta automáticamente que existe una identidad administrada para el App Service y la utiliza para acceder a otros recursos de Azure.

En este tutorial, creará una identidad administrada asignada por el usuario y se asigna al App Service para que pueda acceder a la base de datos y a los recursos de la cuenta de almacenamiento. Para ver un ejemplo de uso de una identidad administrada asignada por el sistema, consulte [Creación e implementación de una aplicación web Flask Python en Azure con una identidad administrada asignada por el sistema](#). Las identidades administradas asignadas por el usuario se recomiendan porque pueden ser usadas por varios recursos y sus ciclos de vida están desacoplados de los ciclos de vida de los recursos a los que están asociadas. Para obtener más información sobre las prácticas recomendadas para el uso de identidades administradas, consulte [Recomendaciones de prácticas recomendadas para identidades administradas](#).

En este tutorial se muestra cómo implementar la aplicación web Python y crear recursos Azure utilizando la [CLI de Azure](#). Los comandos de este tutorial están escritos para ejecutarse en un shell Bash. Puede ejecutar los comandos del tutorial en cualquier entorno Bash con la CLI instalada, como su entorno local o [Azure Cloud Shell](#). Con algunas modificaciones —por ejemplo, estableciendo y usando variables de entorno—, puede ejecutar estos comandos en otros entornos como el shell de comandos de Windows.

Obtención de la aplicación de ejemplo

Use la aplicación de ejemplo de Django para seguir este tutorial. Descargue o clone la aplicación de ejemplo en su entorno de desarrollo.

1. Clonación del ejemplo.

Consola

```
git clone https://github.com/Azure-Samples/msdocs-django-web-app-managed-identity.git
```

2. Vaya a la carpeta de la aplicación.

Consola

```
cd msdocs-django-web-app-managed-identity
```

Examine el código de autenticación

La aplicación web de ejemplo necesita autenticarse en dos almacenes de datos diferentes:

- El servidor de almacenamiento blob de Azure, donde almacena y recupera las fotos enviadas por los revisores.
- Una base de datos Azure Database for PostgreSQL - Servidor Flexible donde almacena los restaurantes y las reseñas.

Usa `DefaultAzureCredential` para autenticarse en ambos almacenes de datos. Con `DefaultAzureCredential`, la aplicación puede configurarse para ejecutarse bajo la identidad de diferentes principales de servicio, dependiendo del entorno en el que se esté ejecutando, sin realizar cambios en el código. Por ejemplo, en un entorno de desarrollo local, la aplicación puede ejecutarse con la identidad del desarrollador que ha iniciado sesión en la CLI de Azure, mientras que en Azure, como en este tutorial, puede ejecutarse con una identidad administrada asignada por el usuario.

En cualquier caso, el principal de seguridad bajo el que se ejecuta la aplicación debe tener un rol en cada recurso Azure que la aplicación utiliza que le permita realizar las acciones en el recurso que la aplicación requiere. En este tutorial, usará los comandos de la CLI de Azure para crear una identidad administrada asignada por el usuario y asignarla a su aplicación en Azure. A continuación, asigne manualmente a esa identidad las funciones adecuadas en su cuenta de almacenamiento Azure y en el servidor Azure Database for PostgreSQL. Por último, se establece la variable de entorno `AZURE_CLIENT_ID` para su aplicación en Azure para configurar `DefaultAzureCredential` y usar la identidad administrada.

Una vez configurada la identidad administrada asignada al usuario en su aplicación y su entorno de ejecución, y asignados los roles adecuados en los almacenes de datos, puede usar `DefaultAzureCredential` para autenticarse con los recursos Azure necesarios.

El siguiente código se utiliza para crear un cliente de almacenamiento blob para subir fotos en `./restaurant_review/views.py`. Se proporciona una instancia de `DefaultAzureCredential` al cliente, que usa para adquirir tokens de acceso para realizar operaciones en el almacenamiento de Azure.

Python

```
from azure.identity import DefaultAzureCredential
from azure.storage.blob import BlobServiceClient

azure_credential = DefaultAzureCredential()
blob_service_client = BlobServiceClient(
    account_url=account_url,
    credential=azure_credential)
```

También se utiliza una instancia de `DefaultAzureCredential` para obtener un token de acceso para Azure Database for PostgreSQL en `./azureproject/get_conn.py`. En este caso, el token se adquiere directamente llamando a `get_token` en la instancia de credenciales y pasándole el valor `scope` apropiado. A continuación, el token se usa para establecer la contraseña en el URI de conexión PostgreSQL.

Python

```
azure_credential = DefaultAzureCredential()
token = azure_credential.get_token("https://osssrdbms-
aad.database.windows.net")
conf.settings.DATABASES['default']['PASSWORD'] = token.token
```

Para obtener más información sobre la autenticación de sus aplicaciones con los servicios Azure, consulte [Autenticación de aplicaciones Python en los servicios Azure mediante el SDK de Azure para Python](#). Para obtener más información sobre `DefaultAzureCredential`, incluido cómo personalizar la cadena de credenciales que evalúa para su entorno, consulte [Introducción a DefaultAzureCredential](#).

Crear un servidor flexible Azure PostgreSQL

1. Configure las variables de entorno necesarias para el tutorial.

Bash

```
LOCATION="eastus"
RAND_ID=$RANDOM
RESOURCE_GROUP_NAME="msdocs-mi-web-app"
APP_SERVICE_NAME="msdocs-mi-web-$RAND_ID"
DB_SERVER_NAME="msdocs-mi-postgres-$RAND_ID"
ADMIN_USER="demoadmin"
ADMIN_PW="ChAnG33#ThsPssWD$RAND_ID"
UA_NAME="UAManagedIdentityPythonTest$RAND_ID"
```

Importante

La `ADMIN_PW` debe contener entre 8 y 128 caracteres de tres de las siguientes categorías: Letras del alfabeto inglés mayúsculas y minúsculas, números y caracteres no alfanuméricos. Al crear nombres de usuario o contraseñas, **no** use el carácter `$`. Más adelante creará variables de entorno con estos valores, donde el carácter `$` tiene un significado especial dentro del contenedor de Linux que se usa para ejecutar aplicaciones de Python.

2. Para crear un grupo de recursos, use el comando [az group create](#).

Azure CLI

```
az group create --location $LOCATION --name $RESOURCE_GROUP_NAME
```

3. Cree un servidor flexible PostgreSQL con el comando [az postgres flexible-server create](#). (Este comando y los siguientes usan el carácter de continuación de línea de Bash Shell ('\\'). Cambie el carácter de continuación de línea para otros shells).

Azure CLI

```
az postgres flexible-server create \
--resource-group $RESOURCE_GROUP_NAME \
--name $DB_SERVER_NAME \
--location $LOCATION \
--admin-user $ADMIN_USER \
--admin-password $ADMIN_PW \
--sku-name Standard_D2ds_v4 \
--active-directory-auth Enabled \
--public-access 0.0.0.0
```

El *sku-name* es el nombre del nivel de precios y de la configuración informática. Para más información, consulte los [precios de Azure Database for PostgreSQL](#).

Para enumerar las SKU disponibles, use `az postgres flexible-server list-skus --location $LOCATION`.

4. Agregue su cuenta Azure como administrador de Microsoft Entra para el servidor con el comando `az postgres flexible-server ad-admin create`.

```
Azure CLI

ACCOUNT_EMAIL=$(az ad signed-in-user show --query userPrincipalName --output tsv)
ACCOUNT_ID=$(az ad signed-in-user show --query id --output tsv)
echo $ACCOUNT_EMAIL, $ACCOUNT_ID
az postgres flexible-server ad-admin create \
--resource-group $RESOURCE_GROUP_NAME \
--server-name $DB_SERVER_NAME \
--display-name $ACCOUNT_EMAIL \
--object-id $ACCOUNT_ID \
--type User
```

5. Configure una regla de cortafuegos en su servidor con el comando `az postgres flexible-server firewall-rule create`. Esta regla permite el acceso de su entorno local para conectarse al servidor. (Si usa Azure Cloud Shell, puede omitir este paso).

```
Azure CLI

IP_ADDRESS=<your IP>
az postgres flexible-server firewall-rule create \
--resource-group $RESOURCE_GROUP_NAME \
--name $DB_SERVER_NAME \
--rule-name AllowMyIP \
--start-ip-address $IP_ADDRESS \
--end-ip-address $IP_ADDRESS
```

Use cualquier herramienta o sitio web que muestre su dirección IP para sustituir `<your IP>` en el comando. Por ejemplo, puede utilizar el sitio web [¿Cuál es mi dirección IP?](#).

6. Cree una base de datos denominada `restaurant` mediante el comando `az postgres flexible-server execute`.

```
Azure CLI

az postgres flexible-server execute \
--name $DB_SERVER_NAME \
--admin-user $ADMIN_USER \
--admin-password $ADMIN_PW \
```

```
--database-name postgres \
--querytext 'create database restaurant;'
```

Creación de un Azure App Service e implementación del código

Ejecute estos comandos en la carpeta raíz de la aplicación de ejemplo para crear un servicio de aplicación e implementar el código en él.

1. Cree un servicio de aplicación con el comando [az webapp up](#).

Azure CLI

```
az webapp up \
--resource-group $RESOURCE_GROUP_NAME \
--location $LOCATION \
--name $APP_SERVICE_NAME \
--runtime PYTHON:3.9 \
--sku B1
```

El *sku* define el tamaño (CPU, memoria) y el coste del plan de App Service. El plan de servicio B1 (Básico) incurre en un pequeño coste en su suscripción Azure. Para obtener una lista completa de los planes de App Service, vea la página [Precios de App Service](#).

2. Configure App Service para usar *start.sh* en el repositorio de ejemplo con el comando [az webapp config set](#).

Azure CLI

```
az webapp config set \
--resource-group $RESOURCE_GROUP_NAME \
--name $APP_SERVICE_NAME \
--startup-file "start.sh"
```

Creación de una cuenta de almacenamiento y un contenedor

La aplicación de ejemplo almacena las fotos enviadas por los revisores como blobs en Azure Storage.

- Cuando un usuario envía una foto con su reseña, la app de ejemplo escribe la imagen en el contenedor mediante la identidad administrada y `DefaultAzureCredential` para acceder a la cuenta de almacenamiento.
- Cuando un usuario consulta las reseñas de un restaurante, la aplicación devuelve un enlace a la foto en el Blob Storage para cada reseña que tenga una asociada. Para que el navegador muestre la foto, debe poder acceder a ella en su cuenta de almacenamiento. Los datos blob deben estar disponibles para su lectura pública a través de un acceso anónimo (no autenticado).

En esta sección, creará una cuenta de almacenamiento y un contenedor que permita el acceso público de lectura a los blobs del contenedor. En secciones posteriores, se crea una identidad administrada asignada por el usuario y se configura para escribir blobs en la cuenta de almacenamiento.

1. Use el comando `az storage create` para crear una cuenta de almacenamiento.

Azure CLI

```
STORAGE_ACCOUNT_NAME="msdocsstorage$RAND_ID"
az storage account create \
    --name $STORAGE_ACCOUNT_NAME \
    --resource-group $RESOURCE_GROUP_NAME \
    --location $LOCATION \
    --sku Standard_LRS \
    --allow-blob-public-access true
```

2. Cree un contenedor llamado *fotos* en la cuenta de almacenamiento con el comando `az storage container create`.

Azure CLI

```
az storage container create \
    --account-name $STORAGE_ACCOUNT_NAME \
    --name photos \
    --public-access blob \
    --auth-mode login
```

⚠ Nota

Si el comando falla, por ejemplo, si obtiene un error que indica que la solicitud puede estar bloqueada por las reglas de red de la cuenta de almacenamiento, introduzca el siguiente comando para asegurarse de que su cuenta de usuario Azure tiene asignado un rol Azure con permiso para crear un contenedor.

Azure CLI

```
az role assignment create --role "Storage Blob Data Contributor" --  
assignee $ACCOUNT_EMAIL --scope  
"/subscriptions/$SUBSCRIPTION_ID/resourceGroups/$RESOURCE_GROUP_NAM  
E/providers/Microsoft.Storage/storageAccounts/$STORAGE_ACCOUNT_NAME  
"
```

Para obtener más información, consulte [Inicio rápido: Creación, descarga y listados de blobs con Azure CLI](#). Tenga en cuenta que varios roles de Azure le permiten crear contenedores en una cuenta de almacenamiento, incluidos "Propietario", "Colaborador", "Propietario de datos de bloques de almacenamiento" y "Colaborador de datos de bloques de almacenamiento".

Creación de una identidad administrada asignada por el usuario

Cree una identidad administrada asignada por el usuario y asígnela al App Service. La identidad administrada se usa para acceder a la base de datos y a la cuenta de almacenamiento.

1. Use el comando `az identity create` para crear una identidad administrada asignada por el usuario y enviar el ID de cliente a una variable para su uso posterior.

Azure CLI

```
UA_CLIENT_ID=$(az identity create --name $UA_NAME --resource-group  
$RESOURCE_GROUP_NAME --query clientId --output tsv)  
echo $UA_CLIENT_ID
```

2. Use el comando `az account show` para obtener su ID de suscripción y enviarlo a una variable que pueda utilizarse para construir el ID de recurso de la identidad administrada.

Azure CLI

```
SUBSCRIPTION_ID=$(az account show --query id --output tsv)  
RESOURCE_ID="/subscriptions/$SUBSCRIPTION_ID/resourceGroups/$RESOURCE_G  
ROUP_NAME/providers/Microsoft.ManagedIdentity/userAssignedIdentities/$U  
A_NAME"  
echo $RESOURCE_ID
```

3. Asigne la identidad administrada al App Service con el comando [az webapp identity assign](#).

```
Azure CLI
```

```
export MSYS_NO_PATHCONV=1
az webapp identity assign \
--resource-group $RESOURCE_GROUP_NAME \
--name $APP_SERVICE_NAME \
--identities $RESOURCE_ID
```

4. Cree la configuración de la aplicación App Service que contiene el ID de cliente de la identidad administrada y otra información de configuración con el comando [az webapp config appsettings set](#).

```
Azure CLI
```

```
az webapp config appsettings set \
--resource-group $RESOURCE_GROUP_NAME \
--name $APP_SERVICE_NAME \
--settings AZURE_CLIENT_ID=$UA_CLIENT_ID \
STORAGE_ACCOUNT_NAME=$STORAGE_ACCOUNT_NAME \
STORAGE_CONTAINER_NAME=photos \
DBHOST=$DB_SERVER_NAME \
DBNAME=restaurant \
DBUSER=$UA_NAME
```

La aplicación de ejemplo usa variables de entorno (configuración de la aplicación) para definir la información de conexión para la base de datos y la cuenta de almacenamiento, pero estas variables no incluyen contraseñas. En su lugar, la autenticación se realiza sin contraseña con `DefaultAzureCredential`.

El código de la aplicación de ejemplo utiliza el constructor de clase `DefaultAzureCredential` sin pasar el ID de cliente de identidad administrada asignado por el usuario al constructor. En este escenario, la alternativa es comprobar la variable de entorno `AZURE_CLIENT_ID`, que se establece como una configuración de la aplicación.

Si la variable de entorno `AZURE_CLIENT_ID` no existe, se usa la identidad administrada asignada por el sistema si está configurada. Para más información, consulte [Introducing DefaultAzureCredential](#).

Creación de roles para la identidad administrada

En esta sección, se crean asignaciones de funciones para la identidad administrada con el fin de permitir el acceso a la cuenta de almacenamiento y a la base de datos.

1. Cree una asignación de roles para la identidad administrada para permitir el acceso a la cuenta de almacenamiento con el comando [az role assignment create](#).

```
Azure CLI

export MSYS_NO_PATHCONV=1
az role assignment create \
--assignee $UA_CLIENT_ID \
--role "Storage Blob Data Contributor" \
--scope
"/subscriptions/$SUBSCRIPTION_ID/resourcegroups/$RESOURCE_GROUP_NAME"
```

El comando especifica el ámbito de la asignación de funciones al grupo de recursos. Para más información, consulte [Comprender la asignación de roles](#).

2. Use el comando [az postgres flexible-server execute](#) para conectarse a la base de datos Postgres y ejecutar los mismos comandos para asignar funciones a la identidad administrada.

```
Azure CLI

ACCOUNT_EMAIL_TOKEN=$(az account get-access-token --resource-type oss-rdbms --output tsv --query accessToken)
az postgres flexible-server execute \
--name $DB_SERVER_NAME \
--admin-user $ACCOUNT_EMAIL \
--admin-password $ACCOUNT_EMAIL_TOKEN \
--database-name postgres \
--querytext "select * from pgaadauth_create_principal(''$UA_NAME'', false, false);select * from pgaadauth_list_principals(false);"
```

Si tiene problemas para ejecutar el comando, asegúrese de haber añadido su cuenta de usuario como administrador de Microsoft Entra para el servidor PostgreSQL y de haber permitido el acceso a su dirección IP en las reglas del cortafuegos. Para obtener más información, consulte la sección [Crear un servidor Azure PostgreSQL flexible](#).

Prueba de la aplicación web Python en Azure

La aplicación Python de ejemplo utiliza el paquete [azure.identity](#) y su clase `DefaultAzureCredential`. Cuando la app se ejecuta en Azure, `DefaultAzureCredential` detecta automáticamente si existe una identidad administrada para el App Service y, si

es así, la usa para acceder a otros recursos de Azure (almacenamiento y PostgreSQL en este caso). No es necesario proporcionar claves de almacenamiento, certificados o credenciales al App Service para acceder a estos recursos.

1. Vaya a la aplicación implementada en la dirección URL

```
http://$APP_SERVICE_NAME.azurewebsites.net.
```

La aplicación puede tardar uno o dos minutos en iniciarse. Si ve una página de aplicación predeterminada que no es la página de aplicación de muestra predeterminada, espere un minuto y actualice el navegador.

2. Pruebe la funcionalidad de la aplicación de ejemplo y añada un restaurante y algunas reseñas con fotos del restaurante.

La información del restaurante y las reseñas se almacenan en Azure Database for PostgreSQL y las fotos se almacenan en Azure Storage. He aquí una captura de pantalla de ejemplo:

The screenshot shows a web application titled "Azure Restaurant Review". At the top, there's a logo and a link to "Azure Docs". Below the title, the restaurant "Contoso Café" is displayed with its address ("1 Main Street"), description ("Nice coffee house."), and rating ("★★★★★ 1 4.5 (2 reviews)"). A "Reviews" section follows, featuring a "Add new review" button. Two reviews are listed: one from "Davide Sagese" on May 20, 2022, at 10:12 a.m., and another from "Francesca Lombo" on May 23, 2022, at 5:24 p.m. Each review includes a photo thumbnail.

Date	User	Rating	Review	Photo
May 20, 2022, 10:12 a.m.	Davide Sagese	5	Friendly staff, good coffee.	
May 23, 2022, 5:24 p.m.	Francesca Lombo	4	Good breakfast choice.	

Limpiar

En este tutorial, todos los recursos Azure se crearon en el mismo grupo de recursos. Eliminar el grupo de recursos con el comando `az group delete` elimina todos los recursos del grupo de recursos y es la forma más rápida de eliminar todos los recursos de Azure utilizados para su app.

```
Azure CLI
```

```
az group delete --name $RESOURCE_GROUP_NAME
```

Opcionalmente, puede agregar el argumento `--no-wait` para permitir que el comando devuelva antes de que se complete la operación.

Pasos siguientes

- Creación e implementación de una aplicación web de Flask en Azure con una identidad administrada asignada por el sistema
 - Implementación de una aplicación web Python (Django o Flask) con PostgreSQL en Azure App Service
-

Comentarios

¿Le ha resultado útil esta página?

 Sí

 No

[Proporcionar comentarios sobre el producto](#) | [Obtener ayuda en Microsoft Q&A](#)

Hospedaje de sitios web estáticos en Azure Storage

Artículo • 22/04/2025

Azure Blob Storage es ideal para almacenar grandes cantidades de datos no estructurados, como texto, imágenes y vídeos. Dado que Blob Storage también proporciona compatibilidad con hospedaje de sitios web estáticos, es una excelente opción en los casos en los que no se requiere que un servidor web represente contenido. Aunque está limitado a hospedar contenido estático, como HTML, CSS, JavaScript y archivos de imagen, puede usar arquitecturas sin servidor, incluidos [Azure Functions](#) y otros servicios de plataforma como servicio (PaaS).

Los sitios web estáticos tienen algunas limitaciones. Por ejemplo, si desea configurar encabezados, tendrá que usar Azure Content Delivery Network (Azure CDN). No hay forma de configurar encabezados como parte de la propia característica de sitio web estático. Además, no se admiten AuthN ni AuthZ.

Si estas características son importantes para su escenario, considere la posibilidad de usar [Azure Static Web Apps](#). Es una excelente alternativa a los sitios web estáticos y también es adecuada en los casos en los que no se requiere un servidor web para representar contenido. Puede configurar encabezados y AuthN/AuthZ es totalmente compatible. Azure Static Web Apps también proporciona un flujo de trabajo de integración continua y entrega continua (CI/CD) totalmente administrado desde el origen de GitHub a la implementación global.

Si necesita un servidor web para representar contenido, puede usar [Azure App Service](#).

Configuración de un sitio web estático

La funcionalidad de hospedaje de sitios web estáticos está configurada dentro de una cuenta de almacenamiento y no está habilitada de forma predeterminada. Para habilitar el hospedaje del sitio web estático, seleccione una cuenta de almacenamiento. En el panel de navegación izquierdo, seleccione **Sitio web estático** en el grupo **Administración de datos** y, después, seleccione **Habilitado**. Proporcione un nombre para el *nombre del documento de índice*. Como alternativa, puede proporcionar una ruta de acceso a una página 404 personalizada. Por último, seleccione **Guardar** para guardar los cambios de configuración.

The screenshot shows the Azure Storage portal for the 'contosostaticsite' storage account. In the left sidebar, under 'Data storage', 'Containers' is selected. On the main page, the 'Static website' section is highlighted with a red box. Inside this section, the 'Enabled' switch is set to 'Enabled'. Below the switch, there are two input fields: 'Index document name' containing 'default.htm' and 'Error document path' containing 'error.htm'. A magnifying glass icon is positioned over the 'error.htm' field.

Si aún no existe, se crea un contenedor de blobs denominado **\$web** dentro de la cuenta de almacenamiento. Agregue los archivos del sitio web al contenedor **\$web** para que sean accesibles a través del punto de conexión principal del sitio web estático.

The screenshot shows the Azure Storage portal for the 'contosostaticsite' storage account. In the left sidebar, under 'Data storage', 'Containers' is selected. On the main page, a table lists containers. The '\$web' container is highlighted with a red box. The table has columns for 'Name', 'Last modified', 'Anonymous access', and 'Lease state'. It shows two entries: 'Slogs' and '\$web'. A magnifying glass icon is positioned over the '\$web' entry.

Los archivos del contenedor **\$web** distinguen mayúsculas de minúsculas, se proporcionan mediante solicitudes de acceso anónimo y solo están disponibles a través operaciones de lectura.

Para obtener instrucciones detalladas, consulte [Hospedaje de sitios web estáticos en Azure Storage](#).

Carga de contenido

Puede usar cualquiera de estas herramientas para cargar contenido en el contenedor **\$web**:

- ✓ [CLI de Azure](#)
- ✓ [Módulo de Azure PowerShell](#)
- ✓ [AzCopy](#)
- ✓ [Explorador de Azure Storage ↗](#)
- ✓ [Azure Portal](#)
- ✓ [Azure Pipelines ↗](#)
- ✓ [Extensión de Visual Studio Code ↗](#) y [Demostración de vídeo de Channel 9](#)

Visualización de contenido

Los usuarios pueden ver contenido del sitio desde un explorador usando la dirección URL pública del sitio web. Puede buscar la dirección URL usando Azure Portal, CLI de Azure o PowerShell. Consulte cómo [Búsqueda de la dirección URL del sitio web](#).

El documento del índice que especifique al habilitar el hospedaje de sitios web estáticos aparece cuando los usuarios abren el sitio y no especifican un archivo concreto (por ejemplo, <https://contosostaticsite.z22.web.core.windows.net>).

Si el servidor devuelve un error 404 y no se ha especificado un documento de error al habilitar el sitio web, se devuelve una página 404 predeterminada al usuario.

! Nota

El [uso compartido de recursos entre orígenes \(CORS\) para Azure Storage](#) no se admite para los sitios web estáticos.

Puntos de conexión secundarios

Si configura la [redundancia en una región secundaria](#), también puede tener acceso al contenido del sitio web mediante un punto de conexión secundario. Los datos se replican en las regiones secundarias de forma asíncrona. Por lo tanto, los archivos que están disponibles en el punto de conexión secundario no están siempre sincronizados con los archivos que están disponibles en el punto de conexión principal.

Impacto de la configuración del nivel de acceso en el contenedor web

Puede modificar el nivel de acceso anónimo del contenedor **\$web**, pero esto no afecta al punto de conexión principal del sitio web estático porque estos archivos se proporcionan a través de solicitudes de acceso anónimo. Eso significa acceso público (de solo lectura) a todos los archivos.

Aunque el punto de conexión principal del sitio web estático no se ve afectado, un cambio en el nivel de acceso anónimo afecta al punto de conexión del servicio blob principal.

Por ejemplo, si cambia el nivel de acceso anónimo del contenedor **\$web** de **Privado (sin acceso anónimo)** a **Blob (acceso anónimo de lectura solo para blobs)**, el nivel de acceso

anónimo al punto de conexión principal del sitio web estático

`https://contosostaticsite.z22.web.core.windows.net/index.html` no cambia.

Sin embargo, el acceso anónimo al punto de conexión de servicio blob principal

`https://contosostaticsite.blob.core.windows.net/$web/index.html` cambia, lo que permite a los usuarios abrir ese archivo mediante cualquiera de estos dos puntos de conexión.

Deshabilitar el acceso público en una cuenta de almacenamiento mediante la [configuración de acceso público](#) de la cuenta de almacenamiento no afecta a los sitios web estáticos hospedados en esa cuenta de almacenamiento. Para más información, consulte [Corrección del acceso de lectura anónimo a datos de blobs \(implementaciones de Azure Resource Manager\)](#).

Asignación de un dominio personalizado a una dirección URL de un sitio web estático

Puede hacer que el sitio web estático esté disponible a través de un dominio personalizado.

Es más fácil habilitar el acceso HTTP para un dominio personalizado, ya que Azure Storage lo admite de forma nativa. Para habilitar HTTPS, tendrá que usar Azure CDN porque Azure Storage no admite de forma nativa HTTPS con dominios personalizados. Para obtener instrucciones pormenorizadas, consulte [Asignación de un dominio personalizado a un punto de conexión de Azure Blob Storage](#).

Si la cuenta de almacenamiento está configurada para [requerir la transferencia segura](#) a través de HTTPS, los usuarios deben utilizar el punto de conexión HTTPS.

Sugerencia

Consider la posibilidad de hospedar el dominio en Azure. Para más información, consulte [Hospedaje de un dominio en Azure DNS](#).

Adición de encabezados HTTP

No hay forma de configurar encabezados como parte de la característica de sitio web estático. Sin embargo, puede usar Azure CDN para agregar encabezados y anexar (o sobrescribir) sus valores. Consulte [Referencia del motor de reglas estándar de Azure CDN](#).

Si quiere usar encabezados para controlar el almacenamiento en caché, consulte [Control del comportamiento del almacenamiento en caché de Azure CDN con reglas de almacenamiento en caché](#).

Hospedaje de sitios web de varias regiones

Si planea hospedar un sitio web en varias zonas geográficas, se recomienda usar [Content Delivery Network](#) para el almacenamiento en caché regional. Use [Azure Front Door](#) para proporcionar contenido diferente en cada región. También proporciona funcionalidad de conmutación por error. No se recomienda [Azure Traffic Manager](#) si tiene previsto usar un dominio personalizado. Pueden surgir problemas debido a cómo Azure Storage comprueba los nombres de dominio personalizados.

Permisos

El permiso para habilitar el sitio web estático es Microsoft.Storage/storageAccounts/blobServices/write o clave compartida. Los roles integrados que proporcionan este acceso incluyen Colaborador de la cuenta de almacenamiento.

Precios

Puede habilitar el hospedaje de sitios web estáticos de forma gratuita. Se le factura solo por el almacenamiento de blobs que el sitio usa y por los costes de las operaciones. Para obtener más información acerca de los precios de Azure Blob Storage, consulte la [página de precios de Azure Blob Storage](#).

Métricas

Puede habilitar métricas en páginas de sitios web estáticos. Después de habilitar las métricas, las estadísticas de tráfico de los archivos en el contenedor \$web aparecen en el panel de métricas.

Para habilitar métricas en las páginas de su sitio web estático, consulte [Habilitación de métricas en páginas de sitios web estáticos](#).

Compatibilidad de características

La compatibilidad con esta característica puede verse afectada al habilitar Data Lake Storage Gen2, el protocolo Network File System (NFS) 3.0 o el Protocolo de transferencia de archivos SSH (SFTP). Si ha habilitado cualquiera de estas funcionalidades, consulte [Compatibilidad con características de Blob Storage en cuentas de Azure Storage](#) para evaluar la compatibilidad con esta característica.

Preguntas más frecuentes

¿Funciona el firewall de Azure Storage con un sitio web estático?

Sí. Se admiten [reglas de seguridad de red](#) de la cuenta de almacenamiento, como firewalls basados en IP y VNET, para el punto de conexión del sitio web estático, y se pueden usar para proteger el sitio web.

¿Los sitios web estáticos admiten Microsoft Entra ID?

No. Un sitio web estático solo admite acceso de lectura anónimo para los archivos del contenedor \$web.

¿Cómo uso un dominio personalizado con un sitio web estático?

Puede configurar un [dominio personalizado](#) con un sitio web estático mediante [Azure Content Delivery Network \(Azure CDN\)](#). Azure CDN proporciona latencias bajas coherentes al sitio web desde cualquier lugar del mundo.

¿Cómo se usa el certificado de la Capa de sockets seguros (SSL) personalizado con un sitio web estático?

Puede configurar un certificado [SSL personalizado](#) con un sitio web estático mediante [Azure CDN](#). Azure CDN proporciona latencias bajas coherentes al sitio web desde cualquier lugar del mundo.

¿Cómo se agregan encabezados y reglas personalizados con un sitio web estático?

Puede configurar el encabezado de host para un sitio web estático mediante un [motor de reglas de Azure CDN](#). Nos interesa conocer sus comentarios [aquí ↗](#).

¿Por qué obtengo un error HTTP 404 de un sitio web estático?

Puede ocurrir un error 404 si hace referencia a un nombre de archivo con mayúsculas y minúsculas incorrectas. Por ejemplo, `Index.html` en lugar de `index.html`. Los nombres de archivo y las extensiones en la URL de un sitio web estático distinguen mayúsculas de minúsculas, aunque entreguen a través de HTTP. Esto también puede ocurrir si el punto de conexión de Azure CDN aún no está aprovisionado. Espere hasta 90 minutos después de aprovisionar una nueva red de Azure CDN para que se complete la propagación.

¿Por qué el directorio raíz del sitio web no redirige a la página de índice predeterminada?

En Azure Portal, abra la página de configuración del sitio web estático de su cuenta y busque el nombre y la extensión que se establecieron en el campo **Nombre del documento de índice**. Asegúrese de que este nombre sea exactamente el mismo que el nombre del archivo ubicado en el contenedor \$web de la cuenta de almacenamiento. Los nombres de archivo y las extensiones en la URL de un sitio web estático distinguen mayúsculas de minúsculas, aunque entreguen a través de HTTP.

¿Por qué no puedo acceder a sitios web estáticos en una cuenta de almacenamiento cuando un punto de conexión privado está habilitado para el blob de la cuenta de almacenamiento?

La habilitación de un punto de conexión privado para blobs en una cuenta de almacenamiento restringe el acceso a esa cuenta de almacenamiento solo a los recursos de la misma red virtual. Por lo tanto, esta restricción impide el acceso externo al sitio web estático hospedado en la cuenta de almacenamiento, lo que hace que el contenido del sitio web estático sea inaccesible. La configuración del punto de conexión privado limita el acceso a todos los recursos de la cuenta de almacenamiento, incluido el contenido del sitio web estático, a los recursos de la misma red virtual donde está habilitado el punto de conexión privado. La resolución sería crear un punto de conexión privado específicamente para la web. El sitio web estático necesita un punto de conexión privado dedicado para el dominio de \$web.

Pasos siguientes

- [Hospedaje de sitios web estáticos en Azure Storage](#)
- [Asignación de un dominio personalizado a un punto de conexión de Azure Blob Storage](#)
- [Funciones de Azure](#)
- [Azure App Service](#)
- [Crear la primera aplicación web sin servidor](#)
- [Tutorial: Hospedaje del dominio en Azure DNS](#)

Inicio rápido: Creación del primer sitio estático con Azure Static Web Apps

Artículo • 09/04/2024

Azure Static Web Apps publica un sitio web mediante la creación de una aplicación desde un repositorio de código. En este inicio rápido se implementa una aplicación en Azure Static Web Apps mediante la extensión de Visual Studio Code.

Si no tiene ninguna suscripción a Azure, [cree una cuenta de evaluación gratuita](#).

Requisitos previos

- [GitHub](#)
- Cuenta de [Azure](#)
- [Visual Studio Code](#)
- [Extensión de Azure Static Web Apps para Visual Studio Code](#)
- [Instalación de Git](#)

Creación de un repositorio

En este artículo se usa un repositorio de plantillas de GitHub para facilitar los primeros pasos. La plantilla incluye una aplicación de inicio que se implementa en Azure Static Web Apps.

Ningún marco

1. Vaya a la siguiente ubicación para crear un repositorio:
 - a. <https://github.com/staticwebdev/vanilla-basic/generate>
2. Asigne el nombre **my-first-static-web-app** al repositorio.

ⓘ Nota

Azure Static Web Apps requiere al menos un archivo HTML para crear una aplicación web. El repositorio que se crea en este paso incluye un solo archivo *index.html*.

Seleccione **Create repository** (Crear repositorio).

Create repository

Clonación del repositorio

Con el repositorio creado en su cuenta de GitHub, clone el proyecto en la máquina local mediante el siguiente comando.

Bash

```
git clone https://github.com/<YOUR_GITHUB_ACCOUNT_NAME>/my-first-static-web-app.git
```

Asegúrese de reemplazar `<YOUR_GITHUB_ACCOUNT_NAME>` por el nombre de usuario de GitHub.

Después, abra Visual Studio Code y vaya a **Archivo > Abrir carpeta** para abrir el repositorio clonado en el editor.

Instalación de la extensión de Azure Static Web Apps

Si aún no tiene la [extensión Azure Static Web Apps para Visual Studio Code](#), puede instalarla en Visual Studio Code.

1. Seleccione **Ver>Extensiones**.
2. En **Buscar extensiones en Marketplace**, escriba **Azure Static Web Apps**.
3. Seleccione **Instalar** para **Azure Static Web Apps**.

Creación de una aplicación web estática

1. En Visual Studio Code, seleccione el logotipo de Azure en la barra de actividades para abrir la ventana extensiones de Azure.



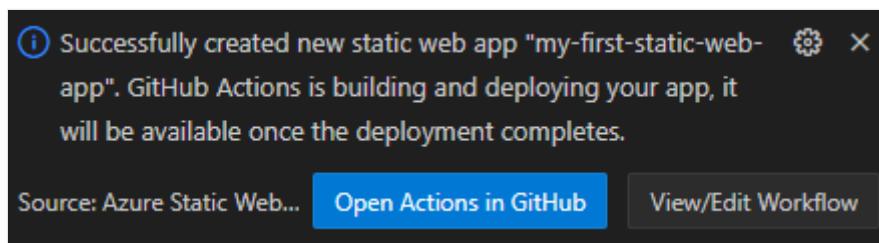
ⓘ Nota

Tendrá que iniciar sesión en Azure y GitHub en Visual Studio Code para continuar. Si todavía no está autenticado, la extensión le solicita que inicie sesión en los dos servicios durante el proceso de creación.

2. En Visual Studio Code, seleccione **F1** para abrir la paleta de comandos.
3. Escriba **Crear aplicación web estática** en el cuadro de comandos.
4. Seleccione *Azure Static Web Apps: Crear aplicación web estática....*
5. Seleccione su suscripción a Azure.
6. Escriba **my-first-static-web-app** como nombre de la aplicación.
7. Seleccione la región más cercana a la suya.
8. Escriba los valores de configuración que coincidan con la opción del marco.

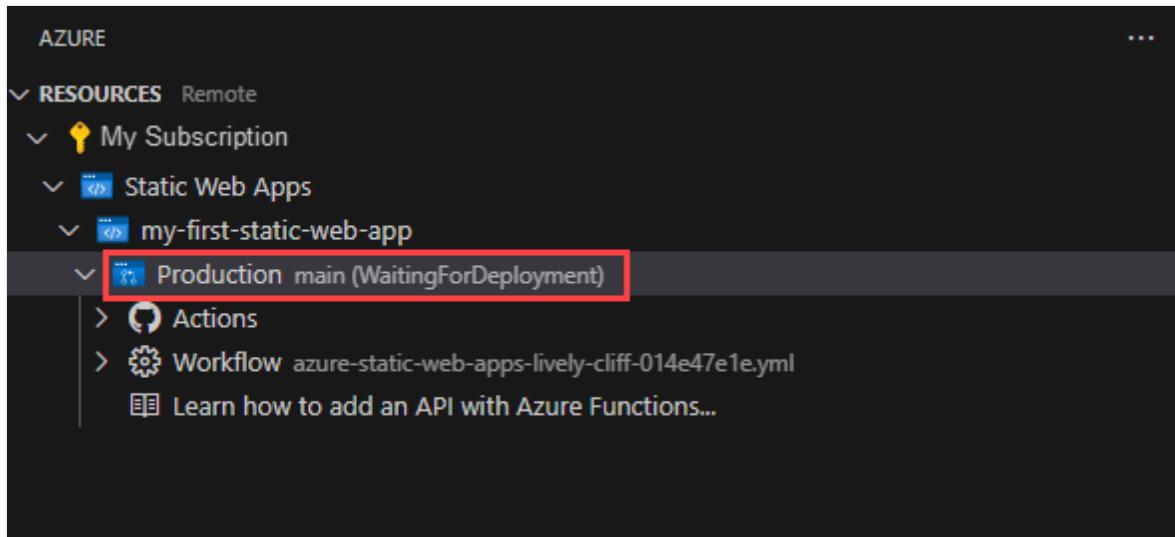
Configuración	Valor
marco	Seleccionar Personalizado
Ubicación del código de la aplicación	Escriba /src.
Ubicación de la compilación	Escriba /src.

9. Una vez creada la aplicación, se muestra una notificación de confirmación en Visual Studio Code.



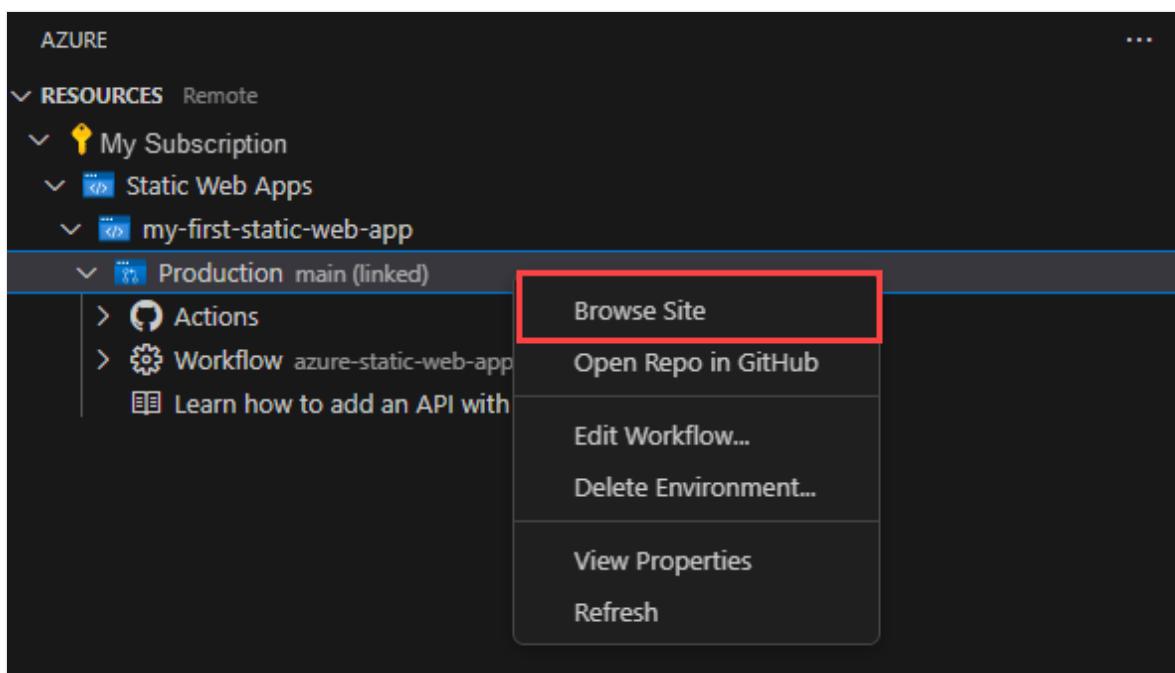
Si GitHub muestra un botón con la etiqueta **Habilitar acciones en este repositorio**, selecciónelo para permitir que la acción de compilación se ejecute en el repositorio.

A medida que la implementación está en curso, la extensión Visual Studio Code le informa del estado de compilación.



Cuando se complete la implementación, puede navegar directamente al sitio web.

10. Para ver el sitio web en el explorador, haga clic con el botón derecho en el proyecto en la extensión de Static Web Apps y seleccione **Examinar sitio**.



Limpieza de recursos

Si no va a seguir usando esta aplicación, puede eliminar la instancia de Azure Static Web Apps mediante la extensión.

En la ventana de Azure de Visual Studio Code, vuelva a la sección *Recursos* y en *Static Web Apps*, haga clic con el botón derecho en **my-first-static-web-app** y seleccione **Eliminar**.

Pasos siguientes

Adición de una API

Creación de un entorno de desarrollo de GitHub Codespaces con FastAPI y Postgres

22/06/2025

En este artículo se muestra cómo ejecutar FastAPI y Postgres juntos en un entorno [de GitHub Codespaces](#). Codespaces es un entorno de desarrollo hospedado en la nube que permite crear entornos de desarrollo configurables y repetibles.

Puede abrir el repositorio de ejemplo en un [explorador](#) o en un entorno de desarrollo integrado (IDE), como [Visual Studio Code](#) con la [extensión GitHub Codespaces](#).

Como alternativa, puede clonar el repositorio de ejemplo localmente. Al abrir el proyecto en Visual Studio Code, puede usar Contenedores de desarrollo para ejecutarlo mediante [Contenedores de desarrollo](#). Los contenedores de desarrollo requieren que [Docker Desktop](#) se instale localmente. Si Docker no está instalado, puede ejecutar el proyecto mediante GitHub Codespaces como entorno de desarrollo.

Al usar GitHub Codespaces, tenga en cuenta que tiene un número fijo de horas básicas gratuitas al mes. Se requiere menos de una hora básica para completar este tutorial. Para más información, consulte [Acerca de la facturación de GitHub Codespaces](#).

También puede usar esta configuración como punto de partida y modificar el ejemplo para ejecutar otros marcos web de Python, como Django o Flask.

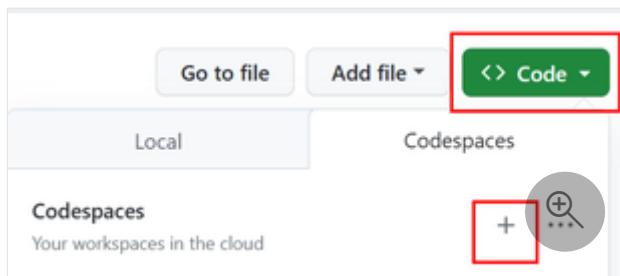
Inicio del entorno de desarrollo en Codespaces

En este tutorial se presenta una de muchas formas posibles de crear y trabajar con GitHub Codespaces.

1. Vaya al repositorio <https://github.com/Azure-Samples/msdocs-fastapi-postgres-codespace> de aplicaciones de ejemplo .

El repositorio de ejemplo tiene toda la configuración necesaria para crear un entorno con una aplicación fastAPI mediante una base de datos postgres. Puede crear un proyecto similar siguiendo los pasos descritos en [Configuración de un proyecto de Python para GitHub Codespaces](#).

2. Seleccione la pestaña **Código, Codespaces** y + para crear un nuevo espacio de código.



3. Cuando el contenedor termine de compilarse, confirme que ve **Codespaces** en la esquina inferior izquierda del explorador y vea el repositorio de ejemplo.

Los archivos clave de configuración de codespace son *devcontainer.json*, *Dockerfile* y *docker-compose.yml*. Para más información, consulte [Introducción a GitHub Codespaces](#).

Sugerencia

También puede ejecutar el codespace en Visual Studio Code. Seleccione **Codespaces** en la esquina inferior izquierda del explorador o (`Ctrl + Shift + P` / `Ctrl + Command + P`) y escriba "Codespaces". A continuación, seleccione **Abrir en VS Code**. Además, si detiene el espacio de código y vuelve al repositorio y vuelve a abrirlo en GitHub Codespaces, tiene la opción de abrirlo en VS Code o en un explorador.

4. Seleccione el archivo *.env.devcontainer* y cree una copia denominada *.env* con el mismo contenido.

El *.env* contiene variables de entorno que se usan en el código para conectarse a la base de datos.

5. Si aún no está abierta una ventana de terminal, abra una; para ello, abra la paleta de comandos (`Ctrl + Shift + P` / `Ctrl + Command + P`), escriba "Terminal: Crear nuevo terminal" y selecciónela para crear un nuevo terminal.

6. Seleccione la pestaña **PUERTOS** en la ventana del terminal para confirmar que PostgreSQL se está ejecutando en el puerto 5432.

7. En la ventana del terminal, ejecute la aplicación FastAPI.

```
Bash
uvicorn main:app --reload
```

8. Seleccione la notificación **Abrir en el explorador**.

Si no ve o no ha perdido la notificación, vaya a **PUERTOS** y busque la **dirección local** del puerto 8000. Use la dirección URL que aparece allí.

9. Agregue `/docs` al final de la dirección URL de vista previa para ver la [interfaz de usuario de Swagger](#), lo que le permite probar los métodos de API.

Los métodos de API se generan a partir de la interfaz openAPI que FastAPI crea a partir del código.

The screenshot shows the FastAPI Swagger UI interface. At the top, it displays "FastAPI 0.1.0 OAS3" and a link to "/openapi.json". Below this, the word "default" is centered above a list of API endpoints. The endpoints are listed in a table-like structure:

Method	Path	Action
GET	/	Root
GET	/restaurant/{id}	Get Restaurant
POST	/restaurant	Set Restaurant
GET	/all	Get All Restaurants

A small circular icon with a plus sign and a minus sign is located to the right of the bottom-most row. Above the table, there is a collapse/expand arrow icon.

10. En la página Swagger, ejecute el método POST para agregar un restaurante.

- a. Expanda el método **POST**.
- b. Haga clic en **Probar**.
- c. Rellene el cuerpo de la solicitud.

The screenshot shows the "JSON" tab of the Swagger UI for the POST /restaurant endpoint. The JSON code is displayed in a text area:

```
{  
  "name": "Restaurant 1",  
  "address": "Restaurant 1 address"  
}
```

- d. Seleccione **Ejecutar** para confirmar el cambio.

Conexión a la base de datos y visualización de los datos

1. Vuelva a GitHub Codespace para el proyecto, seleccione la extensión SQLTools y, a continuación, seleccione **Base de datos local** para conectarse.

La extensión SQLTools debe instalarse cuando se crea el contenedor. Si la extensión SQLTools no aparece en la barra de actividad, cierre el espacio de código y vuelva a abrirlo.

2. Expanda el nodo **Base de datos local** hasta que encuentre la tabla *restaurants*, luego seleccione **Mostrar registros de la tabla**.

Deberías ver el restaurante que agregaste.

The screenshot shows the GitHub Codespace interface with the SQLTools extension installed. On the left, there's a sidebar with icons for file operations, search, and database connections. Under 'CONNECTIONS', it lists 'Azure database' and 'Local database'. The 'Local database' connection is expanded, showing 'postgres database', 'Schemas' (with 'public' selected), and 'Tables'. The 'restaurants' table is selected and highlighted with a red box. A context menu is open over the table, also with a red box around its title 'Show Table Records'. The menu items are: Show Table Records (Ctrl+E Ctrl+S), Describe Table (Ctrl+E Ctrl+D), Generate Insert Query, Add Name(s) To Cursor, and Copy Value(s) (Ctrl+C). The main pane displays the table structure and a single record:

id	name	address
1	Restaurant 1	Restaurant 1 address

Limpieza

Para dejar de usar el espacio de código, cierre el explorador. (O bien cierre VS Code si lo abrió de esa manera).

Si planea volver a usar el espacio de código, puede mantenerlo. Solo los codespaces en ejecución incurren en cargos de CPU. Un espacio de código detenido solo conlleva costes de almacenamiento.

Si desea quitar el espacio de código, vaya a <https://github.com/codespaces> para administrar los espacios de código.

Pasos siguientes

- [Desarrollo de una aplicación web de Python](#)
- [Desarrollo de una aplicación de contenedor](#)
- [Aprender a usar las bibliotecas de Azure para Python](#)

Configuración de un archivo de inicio personalizado para aplicaciones de Python en Azure App Service

Artículo • 24/04/2025

En este artículo, aprenderá cuándo y cómo configurar un archivo de inicio personalizado para una aplicación web de Python hospedada en Azure App Service. Aunque no se requiere un archivo de inicio para el desarrollo local, Azure App Service ejecuta la aplicación web implementada dentro de un contenedor de Docker que puede usar comandos de inicio si se proporcionan.

Necesita un archivo de inicio personalizado en las situaciones siguientes:

- **Argumentos de Gunicorn personalizados:** quiere iniciar el servidor web predeterminado [gunicorn](#) con argumentos adicionales más allá de sus valores predeterminados, que son `--bind=0.0.0.0 --timeout 600`.
- **Plataformas o servidores alternativos:** la aplicación se compila con un marco distinto de Flask o Django, o quiere usar un servidor web diferente además de Gunicorn.
- **Estructura de aplicación de Flask no estándar:** tiene una aplicación de Flask cuyo archivo de código principal se denomina algo distinto de `app.py` o `application.py`*, o el objeto de aplicación se denomina algo distinto de `app`.

Es decir, se requiere un comando de inicio personalizado a menos que el proyecto tenga un archivo `app.py` o `application.py` en la carpeta raíz con un objeto de aplicación flask denominado `app`.

Para más información, consulte [Configuración de aplicaciones de Python: proceso de inicio de contenedor](#).

Creación de un archivo de inicio

Cuando necesite un archivo de inicio personalizado, siga estos pasos:

1. Cree un archivo en el proyecto denominado `startup.txt`, `startup.sh` u otro nombre de su elección que contenga los comandos de inicio. Consulte las secciones posteriores de este artículo para obtener información específica sobre Django, Flask y otros marcos.

Un archivo de inicio puede incluir varios comandos si es necesario.

2. Confirme el archivo en el repositorio de código para que se pueda implementar con el resto de la aplicación.
3. En Visual Studio Code, seleccione el icono de Azure en la barra de actividades, expanda **RECURSOS**, busque y expanda la suscripción, expanda **App Services** y haga clic con el botón derecho en App Service y seleccione **Abrir en el portal**.
4. En el [portal de Azure](#), en la página de **Configuración del App Service**, seleccione **Configuración general**, escriba el nombre de su archivo de inicio (como *startup.txt* o *startup.sh*) en la sección **Configuración de la pila>Comando de inicio**, y luego seleccione **Guardar**.

 **Nota**

En lugar de usar un archivo de comandos de inicio, puede colocar el comando de inicio directamente en el campo **Comando** de inicio en Azure Portal. Se recomienda usar un archivo de comandos de inicio porque almacena la configuración en el repositorio. Esto permite que el control de versiones realice un seguimiento de los cambios y simplifica la reimplementación en otras instancias de Azure App Service.

5. Seleccione **Continuar** cuando se le pida que reinicie App Service.

Si accede al sitio de Azure App Service antes de implementar el código de la aplicación, aparece un "Error de aplicación" porque no hay código disponible para procesar la solicitud.

Comandos de inicio de Django

De forma predeterminada, Azure App Service busca la carpeta que contiene el archivo `wsgi.py` e inicia Gunicorn con el siguiente comando:

```
sh  
  
# <module> is the folder that contains wsgi.py. If you need to use a subfolder,  
# specify the parent of <module> using --chdir.  
gunicorn --bind=0.0.0.0 --timeout 600 <module>.wsgi
```

Si desea modificar cualquier argumento gunicorn, como aumentar el valor de tiempo de espera a 1200 segundos(`--timeout 1200`), cree un archivo de comandos de inicio personalizado. Esto le permite invalidar la configuración predeterminada con sus requisitos específicos. Para obtener más información, consulte [Proceso de inicio de contenedor: aplicación Django](#).

Comandos de inicio de Flask

De forma predeterminada, App Service en Linux supone que la aplicación flask cumple los siguientes requisitos:

- El WSGI al que se puede llamar se denomina `app`.
- El código de la aplicación se encuentra en un archivo denominado `application.py` o `app.py`.
- El archivo de aplicación se encuentra en la carpeta raíz de la aplicación.

Si el proyecto difiere de esta estructura, el comando de inicio personalizado debe identificar la ubicación del objeto de aplicación en el archivo de formato `:app_object`:

- **Nombre de archivo o nombre de objeto de aplicación diferentes:** si el archivo de código principal de la aplicación es `hello.py` y el objeto de aplicación se denomina `myapp`, el comando de inicio es el siguiente:

```
text
gunicorn --bind=0.0.0.0 --timeout 600 hello:myapp
```

- **El archivo de inicio está en una subcarpeta:** si el archivo de inicio es `myapp/website.py` y el objeto de aplicación es `app`, use el argumento de `--chdir` Gunicorn para especificar la carpeta y, a continuación, asigne el nombre al archivo de inicio y al objeto de aplicación como de costumbre:

```
text
gunicorn --bind=0.0.0.0 --timeout 600 --chdir myapp website:app
```

- **El archivo de inicio está dentro de un módulo:** en el código [python-sample-vscode-flask-tutorial](#) , el archivo de inicio `de webapp.py` se encuentra dentro de la carpeta `hello_app`, que es un módulo con un archivo `__init__.py` . El objeto de aplicación se denomina `app` y se define en `__init__.py` y `webapp.py` usa una importación relativa.

Debido a esta organización, cuando Gunicorn apunta a `webapp:app` , se genera el error "Attempted relative import in non-package" (Intento de importación relativa en no paquete) y la aplicación no se inicia.

En esta situación, cree un archivo shim que importe el objeto de la aplicación desde el módulo y, a continuación, haga que Gunicorn inicie la aplicación mediante el shim. El código [python-sample-vscode-flask-tutorial](#) , por ejemplo, contiene `startup.py` con el siguiente contenido:

Python

```
from hello_app.webapp import app
```

A continuación, el comando de inicio es:

txt

```
gunicorn --bind=0.0.0.0 --workers=4 startup:app
```

Para más información, consulte [Proceso de inicio del contenedor: aplicación Flask](#).

Otros marcos y servidores web

El contenedor de App Service que ejecuta aplicaciones de Python tiene Django y Flask instalados de forma predeterminada, junto con el servidor web Gunicorn.

Para usar un marco distinto de Django o Flask (como [Falcon](#), [FastAPI](#), etc.) o para usar un servidor web diferente:

- Incluya el marco o el servidor web en el archivo *requirements.txt* .
- En el comando de inicio, identifique el WSGI al que se puede llamar como se describe en la [sección anterior de Flask](#).
- Para iniciar un servidor web distinto de Gunicorn, use un `python -m` comando en lugar de invocar el servidor directamente. Por ejemplo, el siguiente comando inicia el servidor [uvicorn](#), suponiendo que el WSGI se llame `app` y que se encuentre en *application.py*:

sh

```
python -m uvicorn application:app --host 0.0.0.0
```

Se usa `python -m` porque los servidores web instalados a través de *requirements.txt* no se agregan al entorno global de Python y, por tanto, no se pueden invocar directamente. El `python -m` comando invoca el servidor desde el entorno virtual actual.

Implementación de aplicaciones web de Python en App Service mediante Acciones de GitHub (Linux)

25/06/2025

En este artículo se describe cómo usar la plataforma de integración continua y entrega continua (CI/CD) en Acciones de GitHub para implementar una aplicación web de Python en Azure App Service en Linux. El flujo de trabajo de Acciones de GitHub compila automáticamente el código e lo implementa en la instancia de App Service siempre que haya una confirmación en el repositorio. Puede agregar otra automatización en el flujo de trabajo de Acciones de GitHub, como scripts de prueba, comprobaciones de seguridad y implementación de varias fases.

Creación de un repositorio para el código de la aplicación

Para completar los procedimientos de este artículo, necesita una aplicación web de Python confirmada en un repositorio de GitHub.

- **Aplicación existente:** para usar una aplicación web de Python existente, asegúrese de que la aplicación está confirmada en un repositorio de GitHub.
- **Nueva aplicación:** si necesita una nueva aplicación web de Python, puede bifurcar y clonar el <https://github.com/Microsoft/python-sample-vscode-flask-tutorial> repositorio de GitHub. El código de ejemplo admite el tutorial [de Flask en Visual Studio Code](#) y proporciona una aplicación de Python en funcionamiento.

ⓘ Nota

Si la aplicación usa [Django](#) y una base de datos [de SQLite](#), no funcionará para estos procedimientos. SQLite no se admite en la mayoría de los entornos hospedados en la nube debido a sus limitaciones de almacenamiento local basadas en archivos. Considere la posibilidad de cambiar a una base de datos compatible con la nube, como PostgreSQL o Azure Cosmos DB. Para obtener más información, consulte [Revisar las consideraciones de Django](#) más adelante en este artículo.

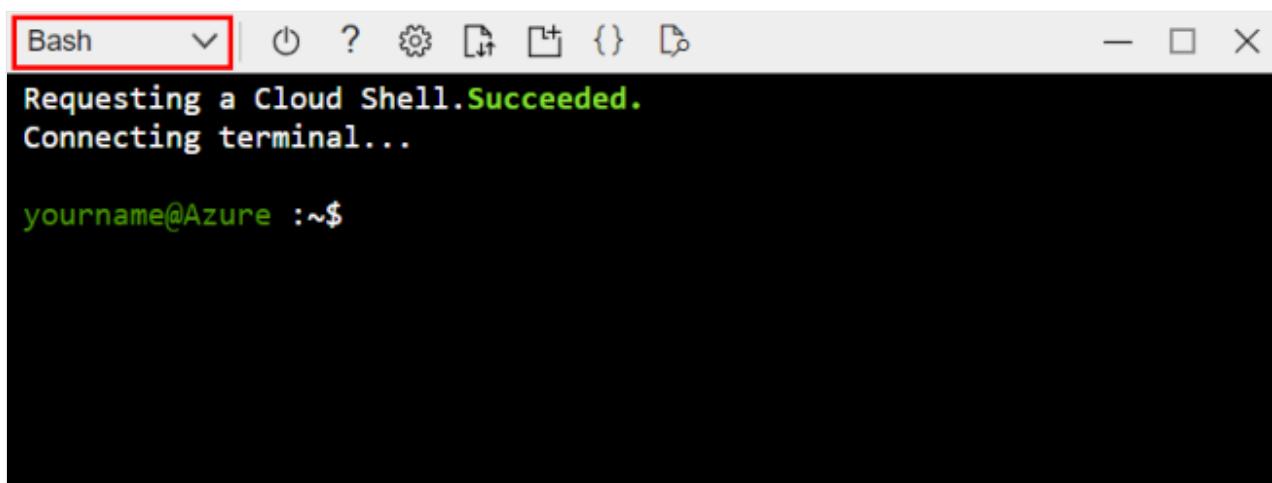
Creación de una instancia de App Service de destino

La forma más rápida de crear una instancia de App Service es usar la [interfaz de la línea de comandos](#) (CLI) de Azure a través de [Azure Cloud Shell](#) interactivo. Cloud Shell incluye [Git](#) y la CLI de Azure. En el procedimiento siguiente, usará el comando `az webapp up` para crear la instancia de App Service y realizar la implementación inicial de la aplicación.

1. Inicie sesión en Azure Portal en <https://portal.azure.com>.
2. Para abrir la CLI de Azure, seleccione la opción Cloud Shell en la barra de herramientas del portal:



3. En Cloud Shell, seleccione la opción **Bash** en el menú desplegable:



4. En Cloud Shell, clone el repositorio mediante el comando [git clone](#).

Sugerencia

Para pegar comandos o texto en Cloud Shell, use el método abreviado de teclado **Ctrl+Mayús+V** o haga clic con el botón derecho y seleccione **Pegar** en el menú contextual.

- Para la aplicación de ejemplo de Flask, puede usar el siguiente comando. Reemplaza la parte `<github-user>` por el nombre de la cuenta de GitHub donde bifurcaste el repositorio.

Bash

```
git clone https://github.com/<github-user>/python-sample-vscode-flask-tutorial.git
```

- Si la aplicación está en un repositorio diferente, configure Acciones de GitHub para el repositorio determinado. Reemplace la cuenta de GitHub `<github-user>` sección por el nombre de la donde bifurcó el repositorio y proporcione el nombre real del repositorio en el marcador de posición `<repo-name>`:

Bash

```
git clone https://github.com/<github-user>/<repo-name>.git
```

ⓘ Nota

Cloud Shell está respaldado por una cuenta de Azure Storage en un grupo de recursos denominado `cloud-shell-storage-your-region<>`. Esa cuenta de almacenamiento contiene una imagen del sistema de archivos de Cloud Shell, que almacena el repositorio clonado. Este almacenamiento tiene un pequeño costo. Puede eliminar la cuenta de almacenamiento después de completar este artículo, junto con otros recursos que cree.

5. En Cloud Shell, cambie el directorio a la carpeta del repositorio de la aplicación de Python, por lo que el comando `az webapp up` reconoce la aplicación como Python. Para la aplicación de ejemplo de Flask, use el siguiente comando:

Bash

```
cd python-sample-vscode-flask-tutorial
```

6. En Cloud Shell, use el comando `az webapp up` para crear una instancia de App Service y realizar la implementación inicial de la aplicación:

Bash

```
az webapp up --name <app-service-name> --runtime "PYTHON:3.9"
```

- Para el marcador de posición `<app-service-name>`, especifique un nombre de *App Service* que sea único en Azure. El nombre debe tener entre 3 y 60 caracteres, y solo puede contener letras, números y guiones. El nombre debe comenzar con una letra y terminar con una letra o un número.

- Para obtener una lista de los entornos de ejecución disponibles en el sistema, use el comando `az webapp list-runtimes`.
- Cuando escriba el valor en tiempo de ejecución en el comando, use el formato `PYTHON:X.Y`, donde `X.Y` es la versión principal y secundaria de Python.
- También puede especificar la ubicación de región de la instancia de App Service mediante el `--location` parámetro . Para obtener una lista de ubicaciones disponibles, use el `az account list-locations --output table` comando .

7. Si la aplicación tiene un script de inicio personalizado, use el comando `az webapp config` para iniciar el script.

- Si la aplicación no tiene un script de inicio personalizado, continúe con el paso siguiente.
- Para la aplicación de ejemplo de Flask, debe acceder al script de inicio en el archivo `startup.txt` mediante la ejecución del comando siguiente:

Bash

```
az webapp config set \
    --resource-group <resource-group-name> \
    --name <app-service-name> \
    --startup-file startup.txt
```

Proporcione el nombre del grupo de recursos y el nombre de la instancia de App Service en los `<resource-group-name>` y `<app-service-name>` placeholders. Para buscar el nombre del grupo de recursos, compruebe la salida del comando anterior `az webapp up` . El nombre del grupo de recursos incluye el nombre de la cuenta de Azure seguido del sufijo `_rg` , como en `<azure-account-name>_rg_`.

8. Para ver la aplicación en ejecución, abra un explorador y vaya al punto de conexión de implementación de la instancia de App Service. En la siguiente dirección URL, reemplace el marcador de posición `<app-service-name>` con el nombre de su instancia de App Service.

URL

`http://<app-service-name>.azurewebsites.net`

Si ve una página genérica, espere unos segundos para que se inicie la instancia de App Service y actualice la página.

- Si sigue viendo una página genérica, confirme que implementó desde la carpeta correcta.
- Para la aplicación de ejemplo de Flask, confirme que implementó desde la carpeta *python-sample-vscode-flask-tutorial*. Compruebe también que ha establecido el comando de inicio correctamente.

Configura la implementación continua en el App Service

En el siguiente procedimiento, configurará la entrega continua (CD), lo que significa que se produce una nueva implementación de código cada vez que se desencadena un flujo de trabajo. El desencadenador del ejemplo de artículo es cualquier cambio en la rama *principal* del repositorio, como con una solicitud de incorporación de cambios (PR).

1. En Cloud Shell, confirme que está en el directorio raíz del sistema (~) y no en una subcarpeta de aplicación, como *python-sample-vscode-flask-tutorial*.
2. Agregue Acciones de GitHub con el comando [az webapp deployment github-actions add](#). Reemplace los marcadores de posición por sus valores específicos:

Bash

```
az webapp deployment github-actions add \
--repo "<github-user>/<github-repo>" \
--resource-group <resource-group-name> \
--branch <branch-name> \
--name <app-service-name> \
--login-with-github
```

- El `--login-with-github` parámetro usa un método interactivo para recuperar un token de acceso personal. Siga las indicaciones y complete la autenticación.
- Si el sistema encuentra un archivo de flujo de trabajo existente con el mismo nombre de instancia de App Service, siga las indicaciones para elegir si se sobrescribe el flujo de trabajo. Puede usar el `--force` parámetro con el comando para sobrescribir automáticamente los flujos de trabajo en conflicto.

El `add` comando completa las siguientes tareas:

- Crea un nuevo archivo de flujo de trabajo en la ruta *.github/workflows/<workflow-name>.yml* de tu repositorio. El nombre de archivo contiene el nombre de la instancia de App Service.

- Captura un perfil de publicación con secretos para la instancia de App Service y lo agrega como un secreto de acción de GitHub. El nombre del secreto comienza por `AZUREAPPSERVICE_PUBLISHPROFILE_`. Se hace referencia a este secreto en el archivo de flujo de trabajo.

3. Obtenga los detalles de una configuración de implementación de control de código fuente con el comando `az webapp deployment source show` . Reemplace los parámetros de marcador de posición por sus valores específicos:

Bash

```
az webapp deployment source show \
--name <app-service-name> \
--resource-group <resource-group-name>
```

4. En la salida del comando, confirme los valores de las propiedades `repoUrl` y `branch` . Estos valores deben coincidir con los valores especificados con el `add` comando .

Examen del flujo de trabajo y las acciones de GitHub

Se especifica una definición de flujo de trabajo en un archivo YAML (`.yml`) en la ruta *de acceso* `./github/workflows/` del repositorio. Este archivo YAML contiene los distintos pasos y parámetros que componen el flujo de trabajo, un proceso automatizado asociado a un repositorio de GitHub. Puede compilar, probar, empaquetar, liberar e implementar cualquier proyecto en GitHub con un flujo de trabajo.

Cada flujo de trabajo se compone de uno o varios trabajos, y cada trabajo es un conjunto de pasos. Cada paso es un script de shell o una acción. Cada trabajo tiene una sección **Acción** en el archivo de flujo de trabajo.

En términos del flujo de trabajo configurado con el código de Python para la implementación en Azure App Service, el flujo de trabajo tiene las siguientes acciones:

 Expandir tabla

Acción	Descripción
<code>checkout</code> ↗	Consulte el repositorio en un <i>ejecutor</i> , un agente de Acciones de GitHub.
<code>setup-python</code> ↗	Instale Python en el ejecutor.
<code>appservice-</code>	Cree la aplicación web.

Acción	Descripción
build ↗	
webapps-deploy ↗	Implemente la aplicación web mediante una credencial de perfil de publicación para autenticarse en Azure. La credencial se almacena en un secreto de GitHub ↗ .

La plantilla de flujo de trabajo que se usa para crear el flujo de trabajo es [Azure/actions-workflow-samples ↗](#).

El workflow se desencadena por eventos de envío sobre la rama especificada. El evento y la rama se definen al principio del archivo de flujo de trabajo. Por ejemplo, el siguiente fragmento de código muestra que el flujo de trabajo se desencadena en eventos push de la rama *main*:

```
YAML

on:
  push:
    branches:
      - main
```

Aplicaciones autorizadas de OAuth

Al configurar la implementación continua, autoriza Azure App Service como una aplicación de OAuth autorizada para su cuenta de GitHub. App Service utiliza el acceso autorizado para crear un archivo YAML de acción de GitHub en la ruta `.github/workflows/<workflow-name>.yml` de tu repositorio.

Para ver las aplicaciones autorizadas y revocar permisos en las cuentas de GitHub, vaya a **Configuración>Integrations/Applications:**

The screenshot shows the GitHub 'Integrations/Applications' settings page. On the left, there's a sidebar with options like 'Public profile', 'Account', 'Appearance', 'Accessibility', and 'Notifications'. Below that is an 'Access' section with 'Billing and plans', 'Emails', 'Password and authentication', and 'Sessions'. The main area has a title 'Applications' with tabs for 'Installed GitHub Apps', 'Authorized GitHub Apps', and 'Authorized OAuth Apps' (which is highlighted with a red box). A message says 'You have granted 5 applications access to your account.' Below this, there are two entries: 'Azure CLI' (last used within the last week, owned by 'AzureAppServiceCLI') and 'Git Credential Manager' (last used within the last week, owned by 'git-ecosystem'). Each entry has a 'Revoke all' button and a three-dot menu icon.

Secreto del perfil de publicación del workflow

En el archivo de workflow `.github/workflows/<workflow-name>.yml` agregado a tu repositorio, hay un marcador de posición para las credenciales del perfil de publicación necesarias para la tarea de implementación del workflow. La información del perfil de publicación se almacena cifrada en el repositorio.

Para ver el secreto, vaya a **Configuración > Seguridad > Secretos y variables > Acciones**:



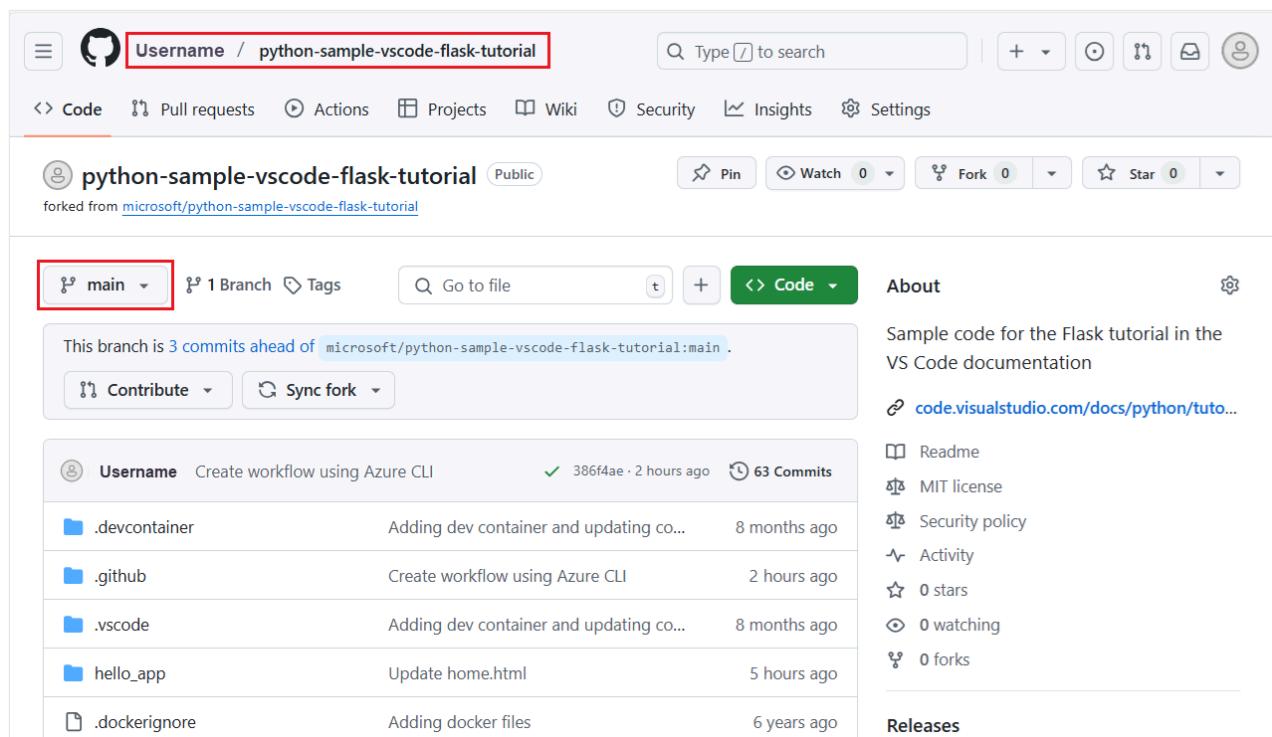
The screenshot shows the GitHub 'Repository secrets' page. On the left, there's a sidebar with 'Actions' highlighted by a red box. The main area lists a single secret named 'AZUREAPPSERVICE_PUBLISHPROFILE_0000AAAA1111BBBB2222CCCC3333DDDD' with a last update time of '24 minutes ago'. There are edit and delete icons next to it. A green button at the top right says 'New repository secret'.

En este artículo, la acción de GitHub se autentica con una credencial de perfil de publicación. Hay otras maneras de autenticarse, como con una entidad de servicio o OpenID Connect. Para obtener más información, consulte [Implementación en App Service mediante Acciones de GitHub](#).

Ejecución y prueba del flujo de trabajo

El último paso es probar el flujo de trabajo realizando un cambio en el repositorio.

1. En un explorador, vaya a la bifurcación del repositorio de ejemplo (o el repositorio que usó) y seleccione la rama que estableció como parte del desencadenador:



The screenshot shows the GitHub repository page for 'python-sample-vscode-flask-tutorial'. The 'main' branch is selected, indicated by a red box around the dropdown menu. The repository is public and was forked from 'microsoft/python-sample-vscode-flask-tutorial'. The page displays the repository's activity, including 63 commits, and provides links to the README, license, and security policy. The commit history shows several updates related to Azure DevOps and Docker.

Commit	Message	Date
.devcontainer	Adding dev container and updating co...	8 months ago
.github	Create workflow using Azure CLI	2 hours ago
.vscode	Adding dev container and updating co...	8 months ago
hello_app	Update home.html	5 hours ago
.dockerignore	Adding docker files	6 years ago

2. Realice un pequeño cambio en la aplicación web de Python.

Para el tutorial de Flask, este es un cambio sencillo:

- a. Vaya al archivo `/hello-app/templates/home.html` de la rama del desencadenador.
- b. Seleccione **Editar** (lápiz).
- c. En el Editor, busque la instrucción print `<p>` y agregue el texto "Redeployed!"

3. Confirme el cambio directamente en la rama en la que trabaja.

- a. En el Editor, seleccione **Confirmar cambios** en la parte superior derecha. Se abre la ventana **Confirmar cambios**.
- b. En la ventana **Confirmar cambios**, modifique el mensaje de confirmación según sea necesario y seleccione **Confirmar cambios**.

El proceso de confirmación desencadena el flujo de trabajo de Acciones de GitHub.

También puede desencadenar manualmente el flujo de trabajo:

1. Vaya a la pestaña **Acciones** del repositorio configurado para la implementación continua.
2. Seleccione el flujo de trabajo en la lista de flujos de trabajo y, a continuación, seleccione **Ejecutar flujo de trabajo**.

Solución de problemas de flujo de trabajo fallido

Puede comprobar el estado de un flujo de trabajo en la pestaña **Acciones** del repositorio de aplicaciones. Al examinar el archivo de flujo de trabajo creado en este artículo, verá dos trabajos: **compilar e implementar**. Como recordatorio, el flujo de trabajo se basa en la plantilla [Azure/actions-workflow-samples](#).

Para un trabajo fallido, examine los resultados de las tareas del trabajo para obtener una indicación del error.

Estos son algunos problemas comunes para investigar:

- Si se produce un error en la aplicación debido a una dependencia que falta, el archivo `requirements.txt` no se procesó durante la implementación. Este comportamiento se produce si creó la aplicación web directamente en el portal en lugar de usar el `az webapp up` comando como se muestra en este artículo.
- Si aprovisionaste el servicio de aplicaciones a través del portal, es posible que la acción de compilación `SCM_DO_BUILD_DURING_DEPLOYMENT` no esté configurada. Esta configuración debe establecerse en `true`. El `az webapp up` comando establece automáticamente la acción de compilación.

- Si ve un mensaje de error relacionado con el tiempo de espera del protocolo de enlace TLS, ejecute el flujo de trabajo manualmente seleccionando **Desencadenar implementación automática** en la pestaña **Acciones** del repositorio de aplicaciones. Puede determinar si el tiempo de espera (timeout) es un problema temporal.
- Si configura la implementación continua para la aplicación contenedora como se muestra en este artículo, el archivo de flujo de trabajo inicial `.github/workflows/<workflow-name>.yml` se crea automáticamente. Si ha modificado el archivo, quite las modificaciones para ver si están causando el error.

Ejecución del script posterior a la implementación

Un script posterior a la implementación puede completar varias tareas, como definir variables de entorno esperadas por el código de la aplicación. Agregue el script como parte del código de la aplicación y ejecute el script mediante el comando de inicio.

Para evitar valores de variables de codificación rígida en el archivo YAML de flujo de trabajo, considere la posibilidad de configurar las variables en GitHub y hacer referencia a los nombres de variable en el script. Puede crear secretos cifrados para un repositorio o para un entorno (repositorio de cuentas). Para obtener más información, consulte [Uso de secretos en Acciones de GitHub](#).

Revisión de las consideraciones de Django

Como se indicó anteriormente en este artículo, puede usar Acciones de GitHub para implementar aplicaciones de Django en Azure App Service en Linux, si usa una base de datos independiente. No se puede usar una base de datos de SQLite porque App Service bloquea el archivo `db.sqlite3`, lo que impide tanto lecturas como escrituras. Este comportamiento no afecta a una base de datos externa.

En el artículo [Configuración de la aplicación Python en App Service: proceso de inicio de contenedor](#) se describe cómo App Service busca automáticamente un archivo `wsgi.py` dentro del código de la aplicación, que normalmente contiene el objeto de aplicación. Cuando usó el `webapp config set` comando para establecer el comando de inicio, usó el `--startup-file` parámetro para especificar el archivo que contiene el objeto de aplicación. El `webapp config set` comando no está disponible en la acción `webapps-deploy`. En su lugar, puede usar el `startup-command` parámetro para especificar el comando de inicio. Por ejemplo, el código siguiente muestra cómo especificar el comando de inicio en el archivo de flujo de trabajo:

YAML

`startup-command: startup.txt`

Al usar Django, normalmente quiere migrar los modelos de datos mediante el `python manage.py migrate` comando después de implementar el código de la aplicación. Puede ejecutar el comando `migrate` en un script posterior a la implementación.

Desconectar acciones de GitHub

La desconexión de Acciones de GitHub de la instancia de App Service permite volver a configurar la implementación de la aplicación. Puede elegir lo que sucede con el archivo de flujo de trabajo después de desconectarse y si desea guardar o eliminar el archivo.

Azure CLI

Desconecte GitHub Actions con el siguiente comando de Azure CLI [az webapp deployment github-actions remove](#). Reemplace los marcadores de posición por sus valores específicos:

Bash

```
az webapp deployment github-actions remove \
--repo "<github-username>/<github-repo>" \
--resource-group <resource-group-name> \
--branch <branch-name> \
--name <app-service-name> \
--login-with-github
```

Limpieza de recursos

Para evitar incurrir en cargos en los recursos de Azure creados en este artículo, elimine el grupo de recursos que contiene la instancia de App Service y el plan de App Service.

Azure CLI

En cualquier lugar donde esté instalada la CLI de Azure, incluido Azure Cloud Shell, puede usar el comando [az group delete](#) para eliminar un grupo de recursos:

Bash

```
az group delete --name <resource-group-name>
```

Eliminar cuenta de almacenamiento

Para eliminar la cuenta de almacenamiento que mantiene el sistema de archivos de Cloud Shell, que incurre en un pequeño cargo mensual, elimine el grupo de recursos que comienza con *cloud-shell-storage-*. Si es el único usuario del grupo, es seguro eliminar el grupo de recursos. Si hay otros usuarios, puede eliminar una cuenta de almacenamiento en el grupo de recursos.

Actualización de la cuenta y el repositorio de GitHub

Si elimina el grupo de recursos de Azure, considere la posibilidad de realizar las siguientes modificaciones en la cuenta de GitHub y el repositorio que se conectó para la implementación continua:

- En el repositorio de aplicaciones, quite el archivo *.github/workflows/<workflow-name>.yml* .
- En la configuración del repositorio de aplicaciones, quite la clave secreta *AZUREAPPSERVICE_PUBLISHPROFILE_* creada para el flujo de trabajo.
- En la configuración de la cuenta de GitHub, quite Azure App Service como una aplicación de OAuth autorizada para su cuenta de GitHub.

Contenido relacionado

- [Repositorios y flujos de trabajo comunes para Acciones de GitHub ↗](#)
- [Referencia del comando: az webapp deployment github-actions](#)

Uso de Azure Pipelines para crear e implementar una aplicación web de Python en Azure App Service

07/04/2024

Azure DevOps Services

Use Azure Pipelines para la integración y entrega continuas (CI/CD) para crear e implementar una aplicación web de Python en Azure App Service en Linux. La canalización crea e implementa automáticamente la aplicación web de Python en App Service cada vez que hay una confirmación en el repositorio.

En este artículo aprenderá a:

- ✓ Crear una aplicación web en Azure App Service.
- ✓ Creación de un proyecto en Azure DevOps.
- ✓ Conectar su proyecto de DevOps con Azure.
- ✓ Crear una canalización específica de Python.
- ✓ Ejecutar la canalización para crear e implementar la aplicación en la aplicación web en App Service.

Requisitos previos

[] Expandir tabla

Producto	Requisitos
Azure DevOps	<ul style="list-style-type: none">- Un proyecto de Azure DevOps .- Tener la capacidad de ejecutar canalizaciones en agentes hospedados por Microsoft. Puede comprar un trabajo paralelo o solicitar un nivel gratuito.- Conocimientos básicos de YAML y Azure Pipelines. Para obtener más información, consulte Creación de su primera canalización.- Permisos:<ul style="list-style-type: none">- Para crear una canalización: debe estar en el grupo Colaboradores y este debe tener el permiso <i>Crear canalización de compilación</i> establecido en Permitir. Los miembros del grupo Administradores de proyectos pueden administrar canalizaciones.- Para crear conexiones de servicio: debe tener el rol <i>Administrador</i> o <i>Creador</i> para las conexiones de servicio.
GitHub	<ul style="list-style-type: none">- Una cuenta de GitHub .- Una conexión de servicio de GitHub para autorizar Azure Pipelines.

Producto	Requisitos
Celeste	Una suscripción de Azure .

Creación de un repositorio para almacenar el código de la aplicación

Bifurque el repositorio de muestra de <https://github.com/Microsoft/python-sample-vscode-flask-tutorial> a su cuenta de GitHub.

En el host local, clone el repositorio de GitHub. Use el comando siguiente, reemplazando `<repository-url>` por la dirección URL del repositorio bifurcado.

```
git  
git clone <repository-url>
```

Prueba local de la aplicación

Cree y ejecute la aplicación de forma local para asegurarse de que funciona.

1. Cambie a la carpeta del repositorio clonado.

```
Bash  
cd python-sample-vscode-flask-tutorial
```

2. Compilar y ejecutar la aplicación

```
Linux  
Bash  
  
python -m venv .env  
source .env/bin/activate  
pip install --upgrade pip  
pip install -r ./requirements.txt  
export FLASK_APP=hello_app.webapp  
python3 -m flask run
```

3. Para ver la aplicación, abra una ventana del navegador y vaya a <http://localhost:5000>.

Compruebe que ve el título `Visual Studio Flask Tutorial`.

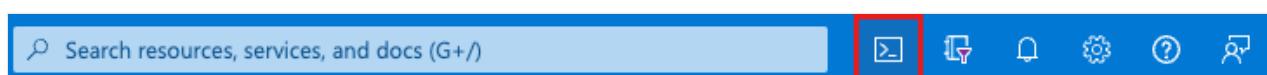
4. Cuando haya terminado, cierre la ventana del navegador y detenga el servidor Flask con

`ctrl + C`.

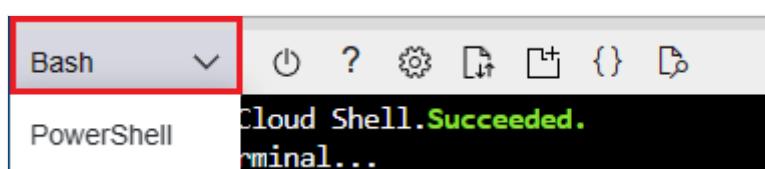
Abrir una ventana de Cloud Shell

1. Inicie sesión en Azure Portal en <https://portal.azure.com>.

2. Para abrir la CLI de Azure, seleccione el botón Cloud Shell en la barra de herramientas del portal.



3. Cloud Shell aparece en la parte inferior del explorador. Seleccione **Bash** en el menú desplegable.



4. Para disponer de más espacio para trabajar, seleccione el botón maximizar.

Creación de una aplicación web de Azure App Service

Cree la aplicación web de Azure App Service desde Cloud Shell en Azure Portal.

Sugerencia

Para pegar en Cloud Shell, use `Ctrl + Mayús + V` o haga clic con el botón derecho y seleccione **Pegar** en el menú contextual.

1. Clone el repositorio con el siguiente comando y reemplace `<repository-url>` por la dirección URL del repositorio bifurcado.

```
Bash
git clone <repository-url>
```

2. Cambie el directorio a la carpeta del repositorio clonado para que el comando `az webapp up` reconozca la aplicación como una aplicación de Python.

Bash

```
cd python-sample-vscode-flask-tutorial
```

3. Use el comando `az webapp up` para aprovisionar App Service y realizar la primera implementación de la aplicación. Reemplace `<your-web-app-name>` por un nombre que sea único en Azure. Normalmente, se usa un nombre personal o de empresa junto con un identificador de aplicación, como `<your-name>-flaskpipelines`. La dirección URL de la aplicación se convierte en `<your-appservice.azurewebsites.net>`.

Azure CLI

```
az webapp up --name <your-web-app-name>
```

La salida JSON del comando `az webapp up` muestra:

JSON

```
{
  "URL": <your-web-app-url>,
  "appserviceplan": <your-app-service-plan-name>,
  "location": <your-azure-location>,
  "name": <your-web-app-name>,
  "os": "Linux",
  "resourcegroup": <your-resource-group>,
  "runtime_version": "python|3.11",
  "runtime_version_detected": "-",
  "sku": <sku>,
  "src_path": <repository-source-path>
}
```

Anote los valores `URL` y `runtime_version`. Use `runtime_version` en el archivo YAML de la canalización. La `URL` es la dirección URL de la aplicación web. Puede usarla para comprobar que la aplicación se está ejecutando.

ⓘ Nota

El comando `az webapp up` realiza las acciones siguientes:

- Cree un grupo de recursos predeterminado.

- Cree un [plan de App Service](#) predeterminado.
- [Cree una aplicación](#) con el nombre especificado.
- [Implantar usando un archivo ZIP](#) todos los archivos del directorio de trabajo actual, con la automatización de construcción habilitada.
- Almacene los parámetros localmente en el archivo `.azure/config` para que no tenga que especificarlos de nuevo al implementar posteriormente con `az webapp up` u otros comandos `az webapp` desde la carpeta del proyecto. Los valores almacenados en caché se usan automáticamente de forma predeterminada.

Puede sustituir la acción predeterminada por sus propios valores mediante los parámetros de comando. Para más información, consulte [az webapp up](#).

4. La aplicación `python-sample-vscode-flask-tutorial` tiene un archivo `startup.txt` que contiene el comando de inicio específico de la aplicación web. Establezca la propiedad de configuración `startup-file` de la aplicación web en `startup.txt`.
 - a. Copie el valor de `az webapp up` a partir de la salida del comando `resourcegroup`.
 - b. Escriba el siguiente comando, con el grupo de recursos y el nombre de la aplicación.

Azure CLI

```
az webapp config set --resource-group <your-resource-group> --name <your-web-app-name> --startup-file startup.txt
```

Cuando el comando finaliza, muestra la salida JSON que contiene todas las opciones de configuración de la aplicación web.

5. Para ver la aplicación en ejecución, abra un explorador y vaya a la `URL` que se muestra en la salida del comando `az webapp up`. Si ve una página genérica, espere unos segundos a que se inicie App Service y, a continuación, actualice la página. Compruebe que ve el título `Visual Studio Flask Tutorial`.

Crear un proyecto de Azure DevOps.

Cree un proyecto de Azure DevOps nuevo.

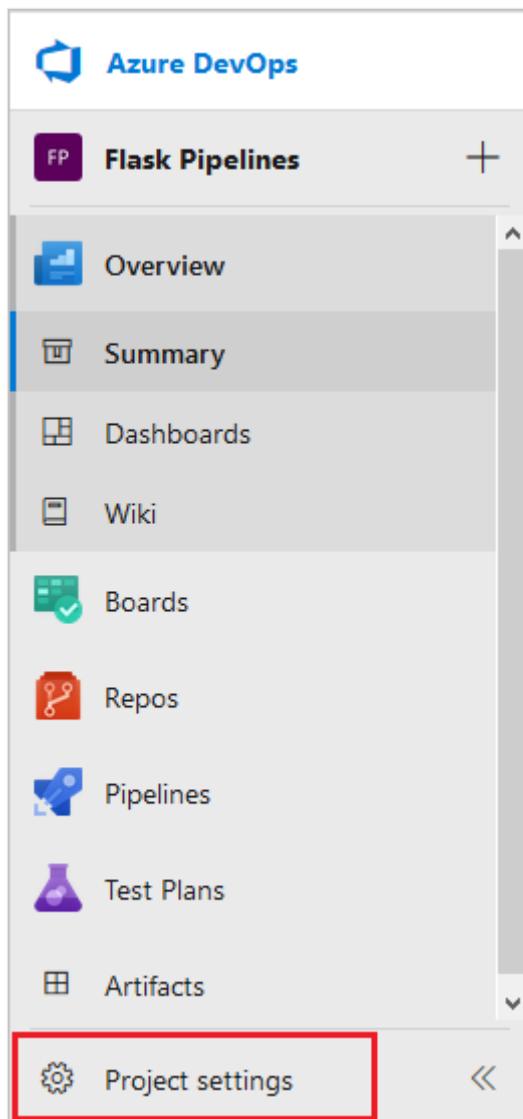
1. En un explorador, vaya a dev.azure.com e inicie sesión.

2. Seleccione su organización.
3. Para crear un nuevo proyecto, seleccione **Nuevo proyecto** o **Crear proyecto** si va a crear el primer proyecto de la organización.
4. Escriba un **nombre de proyecto**.
5. Seleccione la **visibilidad** del proyecto.
6. Seleccione **Crear**.

Creación de una conexión de servicio

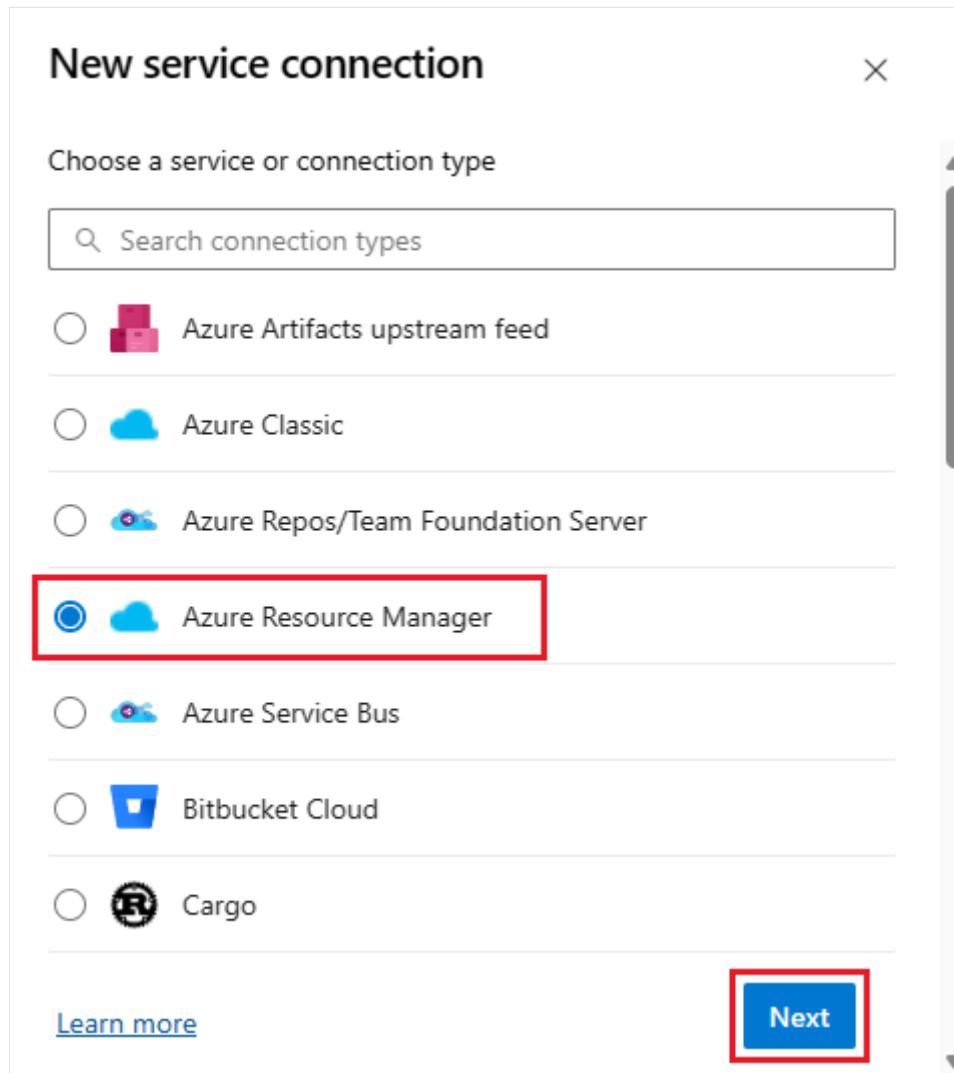
Una conexión de servicio permite crear una conexión para proporcionar acceso autenticado desde Azure Pipelines a servicios externos y remotos. Para desplegar en su aplicación web de Azure App Service, cree una conexión de servicio al grupo de recursos que contiene la aplicación web.

1. En la página del proyecto, seleccione **Configuración del proyecto**.



2. Seleccione **Conexiones de servicio** en la sección **Canalizaciones** del menú.
3. Seleccione **Crear conexión del servicio**.

4. Seleccione Azure Resource Manager y, a continuación, seleccione **Siguiente**.



5. Seleccione el método de autenticación y seleccione **Siguiente**.

6. En el cuadro de diálogo **Nueva conexión del servicio de Azure**, escriba la información específica del método de autenticación seleccionado. Para obtener más información sobre los métodos de autenticación, consulte [Conexión con Azure mediante una conexión de servicio de Azure Resource Manager](#).

Por ejemplo, si usa un método de autenticación de **Federación de identidades de carga de trabajo (automática)** o **Entidad de servicio (automática)**, escriba la información necesaria.

New Azure service connection

X

Azure Resource Manager using Workload Identity federation
with OpenID Connect (automatic)

Scope level

- Subscription
- Management Group
- Machine Learning Workspace

Subscription

<your Azure subscription>



Resource group

PythonWebApp



Details

Service connection name

Azure resource manager connection

Description (optional)

Security

- Grant access permission to all pipelines

[Learn more](#)

[Troubleshoot](#)

Back

Save

[+] Expandir tabla

Campo	Descripción
Nivel de ámbito	Seleccione Suscripción .
Suscripción	Nombre de tu suscripción de Azure.
Grupo de recursos	Nombre del grupo de recursos que contiene la aplicación web.
Nombre de conexión de servicio	Un nombre descriptivo para la conexión.
Conceder permisos de acceso a todas las canalizaciones	Seleccione esta opción para conceder acceso a todas las canalizaciones.

7. Seleccione Guardar.

La nueva conexión aparece en la lista **Conexiones de servicio** y está lista para su uso en Azure Pipeline.

Crear una canalización

Cree una canalización para crear e implementar su aplicación web de Python en Azure App Service. Para comprender los conceptos de canalización, vea:

<https://learn-video.azurefd.net/vod/player?id=20e737aa-cadc-4603-9685-3816085087e9&locale=es-es&embedUrl=%2Fazure%2Fdevops%2Fpipelines%2Fecosystems%2Fpython-webapp>

1. En el menú de navegación izquierdo, seleccione **Canalizaciones**.

The screenshot shows the Azure DevOps interface for the 'Flask Pipelines' project. The left sidebar has a red box around the 'Pipelines' item. The main area features a cartoon illustration of a person working at a desk with a laptop, accompanied by a dog. Below the illustration, the text 'Welcome to the project!' is displayed, followed by 'What service would you like to start with?'. At the bottom of the main area are four buttons: Boards, Repos, Pipelines (which is highlighted in blue), and Test Plans.

2. Seleccione Crear canalización.

The screenshot shows the 'Create your first Pipeline' wizard. It features a cartoon illustration of a robot and a person working on a laptop. Below the illustration, the text 'Create your first Pipeline' is prominently displayed in large, bold, black font. A subtitle below it reads: 'Automate your build and release processes using our wizard, and go from code to cloud-hosted within minutes.' At the bottom of the screen is a blue button with white text that says 'Create Pipeline', which is also outlined with a red box.

3. En el cuadro de diálogo Dónde está el código, seleccione GitHub. Es posible que se le pida que inicie sesión en GitHub.

Connect Select Configure Review

New pipeline

Where is your code?

 Azure Repos Git YAML Free private Git repositories, pull requests, and code search
 Bitbucket Cloud YAML Hosted by Atlassian
 GitHub YAML Home to the world's largest community of developers
 GitHub Enterprise Server YAML The self-hosted version of GitHub Enterprise
 Other Git Any generic Git repository
 Subversion Centralized version control by Apache

4. En la pantalla **Seleccionar un repositorio**, seleccione el repositorio de ejemplo bifurcado.

✓ Connect **Select** Configure Review

New pipeline

Select a repository

Filter by keywords	My repositories	X
 Microsoft/vscode-docs	2h ago	
 Microsoft/vscode-website private	Yesterday	
 Mycode/python-sample-vscode-flask-tutorial fork	Yesterday	

5. Es posible que se le pida que vuelva a escribir la contraseña de GitHub como confirmación.

6. Si la extensión de Azure Pipelines no está instalada en GitHub, GitHub le pedirá que instale la extensión de **Azure Pipelines**.

The screenshot shows the GitHub user interface with the Azure Pipelines extension installed. The left sidebar has a 'Personal settings' section with 'Profile', 'Account', 'Emails', and 'Notifications'. The main area displays the Azure Pipelines extension details: 'Installed 7 days ago', 'Developed by AzurePipelines', and a link to the Azure Pipelines website. Below this, a descriptive text reads: 'Continuously build, test, and deploy to any platform and cloud' and 'Azure Pipelines offers cloud-hosted pipelines for Linux, macOS, and Windows with 10 free parallel jobs and unlimited minutes for open source projects.'

En esta página, desplácese hacia abajo hasta la sección **Acceso al repositorio**, elija si desea instalar la extensión en todos los repositorios o solo en los seleccionados y, a continuación, seleccione **Aprobar e instalar**.

The screenshot shows the 'Repository access' dialog box. It contains two radio button options: 'All repositories' (selected) and 'Only select repositories'. Below the first option is a note: 'This applies to all current *and* future repositories.' A 'Select repositories' button is also present. At the bottom are 'Approve and install' and 'Cancel' buttons. The 'Approve and install' button is highlighted with a red box.

7. En el cuadro de diálogo **Configuración de la canalización**, seleccione **Aplicación web de Python a Linux en Azure**.
8. Seleccione la suscripción de Azure y seleccione **Continuar**.
9. Si usa su nombre de usuario y la contraseña para autenticarse, se abre un explorador para que inicie sesión en su cuenta Microsoft.
10. Seleccione el nombre de la aplicación web en la lista desplegable y seleccione **Validar y configurar**.

Azure Pipelines crea un archivo `azure-pipelines.yml` y lo muestra en el editor de canalizaciones de YAML. El archivo de canalización define la canalización de CI/CD como una serie de *fases*, *trabajos* y *pasos*, donde cada paso contiene los detalles de diferentes *tareas* y *scripts*. Eche un vistazo a la canalización para ver lo que hace. Asegúrese de que todas las entradas predeterminadas sean adecuadas para el código.

Archivo de canalización YAML

En la explicación siguiente se describe el archivo de canalización YAML. Para obtener información sobre el esquema del archivo YAML de canalización, consulte [Referencia de esquema de YAML](#).

A continuación se muestra el archivo completo de ejemplo YAML del pipeline:

```
yml

trigger:
- main

variables:
# Azure Resource Manager connection created during pipeline creation
azureServiceConnectionId: '<GUID>'

# Web app name
webAppName: '<your-webapp-name>'

# Agent VM image name
vmImageName: 'ubuntu-latest'

# Environment name
environmentName: '<your-webapp-name>'

# Project root folder. Point to the folder containing manage.py file.
projectRoot: $(System.DefaultWorkingDirectory)

pythonVersion: '3.11'

stages:
- stage: Build
  displayName: Build stage
  jobs:
  - job: BuildJob
    pool:
      vmImage: $(vmImageName)
    steps:
    - task: UsePythonVersion@0
      inputs:
        versionSpec: '$(pythonVersion)'
      displayName: 'Use Python $(pythonVersion)'

    - script: |
        python -m venv antenv
        source antenv/bin/activate
        python -m pip install --upgrade pip
        pip install setuptools
        pip install -r requirements.txt
      workingDirectory: $(projectRoot)
      displayName: "Install requirements"
```

```

- task: ArchiveFiles@2
  displayName: 'Archive files'
  inputs:
    rootFolderOrFile: '$(projectRoot)'
    includeRootFolder: false
    archiveType: zip
    archiveFile: $(Build.ArtifactStagingDirectory)/$(Build.BuildId).zip
    replaceExistingArchive: true

- upload: $(Build.ArtifactStagingDirectory)/$(Build.BuildId).zip
  displayName: 'Upload package'
  artifact: drop

- stage: Deploy
  displayName: 'Deploy Web App'
  dependsOn: Build
  condition: succeeded()
  jobs:
    - deployment: DeploymentJob
      pool:
        vmImage: $(vmImageName)
      environment: $(environmentName)
      strategy:
        runOnce:
          deploy:
            steps:
              - task: UsePythonVersion@0
                inputs:
                  versionSpec: '$(pythonVersion)'
                displayName: 'Use Python version'

              - task: AzureWebApp@1
                displayName: 'Deploy Azure Web App : $(webAppName)'
                inputs:
                  azureSubscription: $(azureServiceConnectionId)
                  appName: $(webAppName)
                  package: $(Pipeline.Workspace)/drop/$(Build.BuildId).zip

```

variables

La sección `variables` contiene las siguientes variables:

yml

```

variables:
# Azure Resource Manager connection created during pipeline creation
azureServiceConnectionId: '<GUID>'

# Web app name
webAppName: '<your-webapp-name>'
```

```

# Agent VM image name
vmImageName: 'ubuntu-latest'

# Environment name
environmentName: '<your-webapp-name>'

# Project root folder.
projectRoot: $(System.DefaultWorkingDirectory)

# Python version: 3.11. Change this to match the Python runtime version running on
# your web app.
pythonVersion: '3.11'

```

[+] Expandir tabla

Variable	Descripción
azureServiceConnectionId	El identificador o nombre de la conexión de servicio de Azure Resource Manager.
webAppName	Nombre de la aplicación web de Azure App Service.
vmImageName	Nombre del sistema operativo que se va a usar para el agente de creación.
environmentName	Nombre del entorno usado en la fase de implementación. El entorno se crea automáticamente cuando se ejecuta el trabajo de fase.
projectRoot	Carpeta raíz que contiene el código de la aplicación.
pythonVersion	Versión de Python que se va a usar en los agentes de compilación e implementación.

Fase de construcción

La fase de compilación contiene un único trabajo que se ejecuta en el sistema operativo definido en la variable `vmImageName`.

```

yml
- job: BuildJob
  pool:
    vmImage: $(vmImageName)

```

El trabajo contiene varios pasos:

1. La tarea [UsePythonVersion](#) selecciona la versión de Python que se va a usar. La versión se define en la variable `pythonVersion`.

```
yml
```

```
- task: UsePythonVersion@0
  inputs:
    versionSpec: '$(pythonVersion)'
    displayName: 'Use Python $(pythonVersion)'
```

2. En este paso se usa un script para crear un entorno de Python virtual e instalar las dependencias de la aplicación contenidas en el `requirements.txt` parámetro. The `workingDirectory` especifica la ubicación del código de la aplicación.

```
yml
```

```
- script: |
  python -m venv antenv
  source antenv/bin/activate
  python -m pip install --upgrade pip
  pip install setuptools
  pip install -r ./requirements.txt
  workingDirectory: $(projectRoot)
  displayName: "Install requirements"
```

3. La tarea `ArchiveFiles` crea el archivo `.zip` archivo que contiene la aplicación web. El archivo `.zip` se carga en la canalización como el artefacto denominado `drop`. El archivo `.zip` se usa en la fase de implementación para implementar la aplicación en la aplicación web.

```
yml
```

```
- task: ArchiveFiles@2
  displayName: 'Archive files'
  inputs:
    rootFolderOrFile: '$(projectRoot)'
    includeRootFolder: false
    archiveType: zip
    archiveFile: $(Build.ArtifactStagingDirectory)/$(Build.BuildId).zip
    replaceExistingArchive: true

- upload: $(Build.ArtifactStagingDirectory)/$(Build.BuildId).zip
  displayName: 'Upload package'
  artifact: drop
```

 Expandir tabla

Parámetro	Descripción
<code>rootFolderOrFile</code>	Ubicación del código de la aplicación.

Parámetro	Descripción
<code>includeRootFolder</code>	Indica si se debe incluir la carpeta raíz en el archivo .zip. Establezca este parámetro en <code>false</code> ; de lo contrario, el contenido del archivo .zip se coloca en una carpeta denominada <code>s</code> y App Service en el contenedor de Linux no encuentra el código de la aplicación.
<code>archiveType</code>	El tipo de archivo que se va a crear. Establécelo en <code>zip</code> .
<code>archiveFile</code>	Ubicación del archivo .zip que se va a crear.
<code>replaceExistingArchive</code>	Indica si se va a reemplazar un archivo existente si el archivo ya existe. Establécelo en <code>true</code> .
<code>upload</code>	Ubicación del archivo .zip que se va a cargar.
<code>artifact</code>	Nombre del artefacto que se va a crear.

Fase de implementación

La fase de implementación se ejecuta si la fase de compilación se completa correctamente. Las siguientes palabras clave definen este comportamiento:

```
yml
```

```
dependsOn: Build
condition: succeeded()
```

La fase de implementación contiene un único trabajo de implementación configurado con las siguientes palabras clave:

```
yml
```

```
- deployment: DeploymentJob
pool:
  vmImage: $(vmImageName)
  environment: $(environmentName)
```

 Expandir tabla

Palabra clave	Descripción
<code>deployment</code>	Indica que el trabajo es un trabajo de implementación que tiene como destino un entorno .

Palabra clave	Descripción
pool	Especifica el grupo de agentes de implementación. Grupo de agentes predeterminado si no se especifica el nombre. La palabra clave <code>vmImage</code> identifica el sistema operativo de la imagen de máquina virtual del agente.
environment	Especifica el entorno en el que se va a implementar. El entorno se crea automáticamente en el proyecto cuando se ejecuta el trabajo.

La palabra clave `strategy` se usa para definir la estrategia de implementación. La palabra clave `runOnce` especifica que el trabajo de implementación se ejecuta una vez. La palabra clave `deploy` especifica los pasos que se van a ejecutar en el trabajo de implementación.

yml

```
strategy:
  runOnce:
    deploy:
      steps:
```

Los `steps` de la canalización son los siguientes:

1. Use la tarea [UsePythonVersion](#) para especificar la versión de Python que se va a usar en el agente. La versión se define en la variable `pythonVersion`.

yml

```
- task: UsePythonVersion@0
  inputs:
    versionSpec: '$(pythonVersion)'
    displayName: 'Use Python version'
```

2. Implemente la aplicación web mediante [AzureWebApp@1](#). Esta tarea implementa el artefacto de canalización `drop` en la aplicación web.

yml

```
- task: AzureWebApp@1
  displayName: 'Deploy Azure Web App : <your-web-app-name>'
  inputs:
    azureSubscription: $(azureServiceConnectionId)
    appName: $(webAppName)
    package: $(Pipeline.Workspace)/drop/$(Build.BuildId).zip
```

Parámetro	Descripción
<code>azureSubscription</code>	El identificador o nombre de la conexión de servicio de Azure Resource Manager que se va a utilizar.
<code>appName</code>	el nombre de la aplicación web.
<code>package</code>	Ubicación del archivo <code>.zip</code> que se va a implementar.

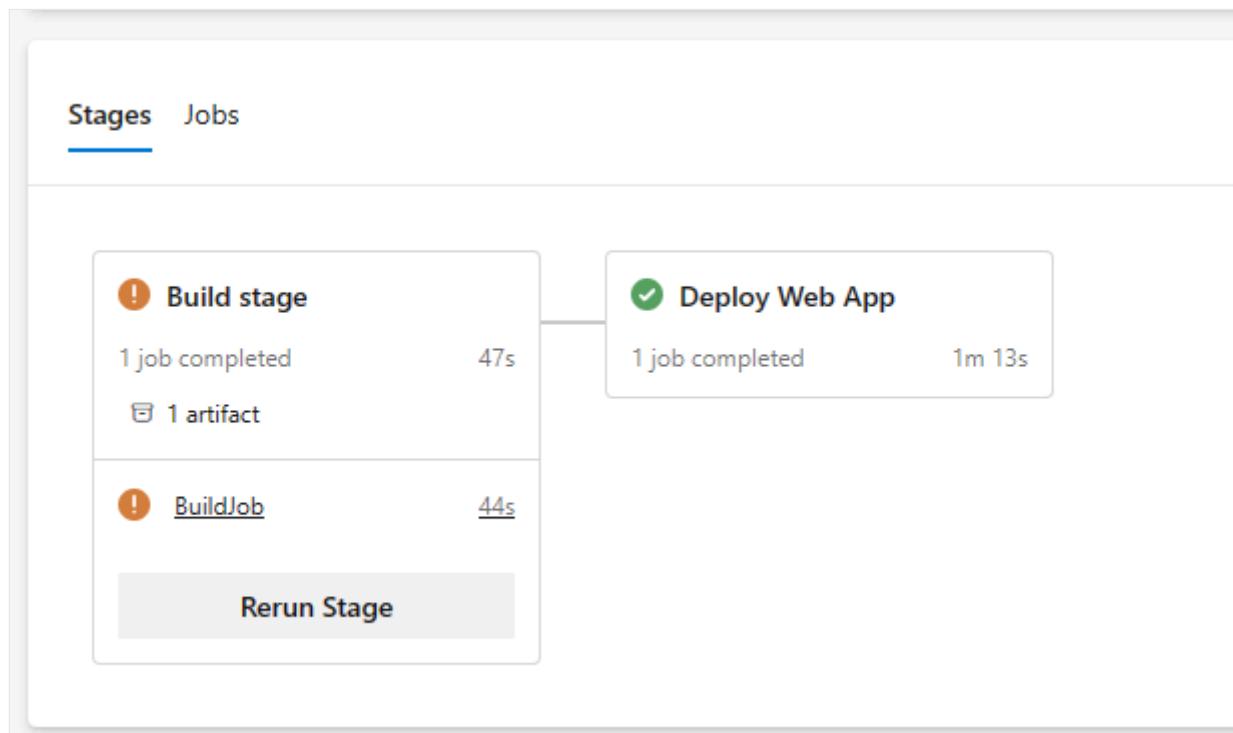
Además, debido a que el repositorio `python-vscode-flask-tutorial` contiene el mismo comando de inicio en un archivo llamado `startup.txt`, puede especificar ese archivo agregando el parámetro: `startUpCommand: 'startup.txt'`.

Ejecución de la canalización

Ya está todo listo para probarlo.

1. En el editor, seleccione **Guardar y ejecutar**.
2. En el cuadro de diálogo **Guardar y ejecutar**, agregue un mensaje de confirmación y, a continuación, seleccione **Guardar y ejecutar**.

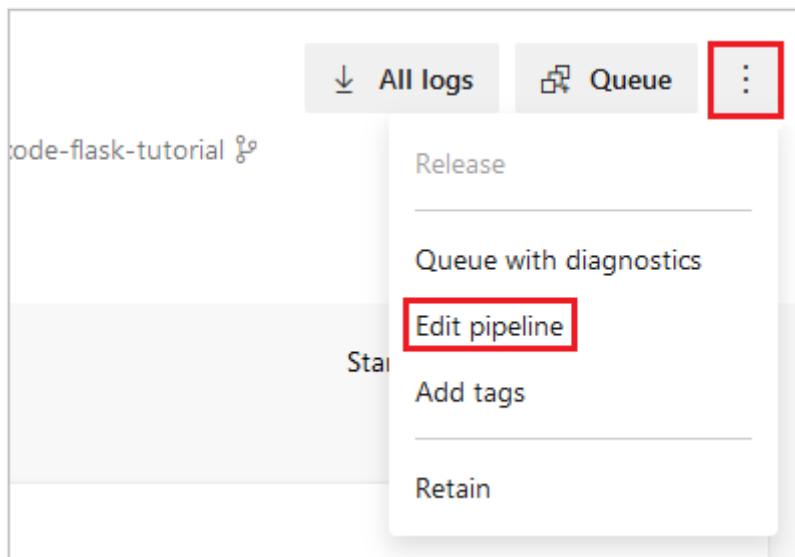
Puede ver la canalización a medida que se ejecuta seleccionando las fases o los trabajos en el resumen de ejecución de la canalización.



Hay marcas de verificación verdes junto a cada fase y trabajo a medida que se completa correctamente. Si se producen errores, se muestran en el resumen o en los pasos del trabajo.

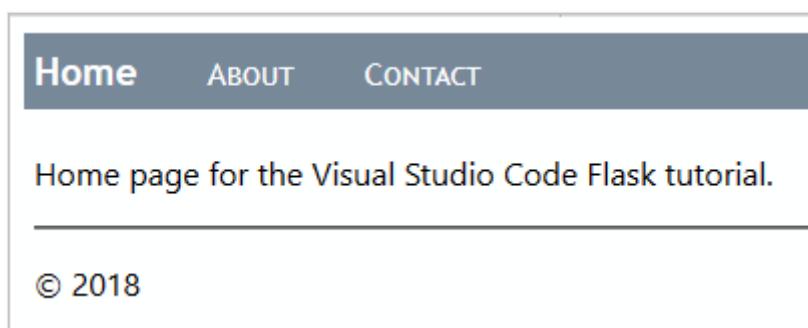
← Jobs in run #20240312.1		
username.python-sample-vscode-flask-tutorial		
Build stage		
▼	BuildJob	54s
	Initialize job	5s
	Checkout username/py...	2s
	Use Python 3.12	<1s
	Install requirements	10s
	Archive files	2s
	Upload package	6s
	Post-job: Checkout us...	<1s
	Finalize Job	<1s
Deploy Web App		
▼	DeploymentJob	59s
	Initialize job	4s
	Download Artifact	4s
	Use Python version	<1s
	Deploy Azure Web Ap...	36s
	Finalize Job	<1s

Para volver rápidamente al editor de YAML, seleccione los puntos verticales en la parte superior derecha de la página **Resumen** y seleccione **Editar canalización**:



3. En el trabajo de implementación, seleccione la tarea **Implementar aplicación web de Azure** para mostrar su salida. Para visitar el sitio implementado, mantenga presionada la tecla `ctrl` y seleccione la dirección URL tras `App Service Application URL`.

Si usa la aplicación de ejemplo, la aplicación debería aparecer de la siguiente manera:



ⓘ Importante

Si se produce un error en la aplicación debido a una dependencia que falta, significa que el archivo `requirements.txt` no se procesó durante la implementación. Este comportamiento se produce si creó la aplicación web directamente en el portal en lugar de usar el comando `az webapp up` como se muestra en este artículo.

El comando `az webapp up` establece específicamente la acción de compilación `SCM_DO_BUILD_DURING_DEPLOYMENT` en `true`. Si aprovisionó el servicio de aplicaciones a través del portal, esta acción no se establece automáticamente.

Los pasos siguientes establecen la acción:

1. Abra [Azure Portal](#), seleccione la instancia de App Service y, a continuación, elija **Configuración**.

2. En la pestaña **Configuración de aplicaciones**, seleccione **Nueva configuración de aplicación**.
3. En el menú emergente que aparece, establezca **Nombre** en `SCM_DO_BUILD_DURING_DEPLOYMENT`, establezca **Valor** en `true` y seleccione **Aceptar**.
4. Seleccione **Guardar** en la parte superior de la página **Configuración**.
5. Repetición de la ejecución de la canalización Las dependencias deben instalarse durante la implementación.

Desencadenamiento de una ejecución de la canalización

Para desencadenar una ejecución de canalización, confirme un cambio en el repositorio. Por ejemplo, puede agregar una nueva característica a la aplicación o actualizar las dependencias de la aplicación.

1. Vaya al repositorio de GitHub.
2. Realice un cambio en el código, como cambiar el título de la aplicación.
3. Confirme el cambio en el repositorio.
4. Vaya a su pipeline y verifique que se ha creado una nueva ejecución.
5. Cuando finalice la ejecución, compruebe que la nueva creación se ha implementado en la aplicación web.
 - a. En Azure Portal, vaya a la aplicación web.
 - b. Seleccione **Centro de implementación** y seleccione la pestaña **Registros**.
 - c. Compruebe que aparece la nueva implementación.

Consideraciones para Django

Puede usar Azure Pipelines para implementar aplicaciones de Django en Azure App Service en Linux si usa una base de datos independiente. No se puede usar una base de datos SQLite, ya que App Service bloquea el archivo `db.sqlite3`, lo que impide tanto las lecturas como las escrituras. Este comportamiento no afecta a una base de datos externa.

Como se describe en [Configuración de la aplicación de Python en App Service: proceso de inicio del contenedor](#), App Service busca automáticamente un archivo `wsgi.py` en el código de la aplicación, que normalmente contiene el objeto de aplicación. Si desea personalizar el comando de inicio de cualquier manera, use el parámetro `startUpCommand` en el paso `AzureWebApp@1` del archivo de canalización de YAML, como se describe en la sección anterior.

Al usar Django, normalmente quiere migrar los modelos de datos mediante `manage.py migrate` después de implementar el código de la aplicación. Puede agregar `startUpCommand` con un script posterior a la implementación para este propósito. Por ejemplo, esta es la propiedad `startUpCommand` de la tarea AzureWebApp@1.

```
yml
```

```
- task: AzureWebApp@1
  displayName: 'Deploy Azure Web App : $(webAppName)'
  inputs:
    azureSubscription: $(azureServiceConnectionId)
    appName: $(webAppName)
    package: $(Pipeline.Workspace)/drop/$(Build.BuildId).zip
    startUpCommand: 'python manage.py migrate'
```

Ejecución de pruebas en el agente de compilación

Como parte del proceso de compilación, puede que desee ejecutar pruebas en el código de la aplicación. Las pruebas se ejecutan en el agente de compilación, por lo que debe instalar las dependencias en un entorno virtual en el agente de compilación. Una vez ejecutadas las pruebas, elimine el entorno virtual antes de crear el archivo .zip para la implementación. Los siguientes elementos de script muestran este proceso. Colóquelos antes de la tarea `ArchiveFiles@2` en el archivo *azure-pipelines.yml*. Para más información, consulte [Ejecución de scripts multiplataforma](#).

```
yml
```

```
# The | symbol is a continuation character, indicating a multi-line script.
# A single-line script can immediately follow "- script:".
- script: |
    python -m venv .env
    source .env/bin/activate
    pip install setuptools
    pip install -r requirements.txt

# The displayName shows in the pipeline UI when a build runs
displayName: 'Install dependencies on build agent'

- script: |
    # Put commands to run tests here
    displayName: 'Run tests'

- script: |
    echo Deleting .env
    deactivate
```

```
rm -rf .env  
displayName: 'Remove .env before zip'
```

También puede usar una tarea como [PublishTestResults@2](#) para publicar los resultados de las pruebas en la canalización. Para más información, consulte [Compilación de aplicaciones de Python: ejecución de pruebas](#).

Limpieza de recursos

Para evitar incurrir en cargos en los recursos de Azure creados en este tutorial:

- Elimine el proyecto que creó. Al eliminar el proyecto, se elimina la canalización y la conexión de servicio.
- Elimine el grupo de recursos de Azure que contiene App Service y el plan de App Service. En Azure Portal, vaya al grupo de recursos, seleccione **Eliminar grupo de recursos** y siga las indicaciones.
- Elimine la cuenta de almacenamiento que mantiene el sistema de archivos para Cloud Shell. Cierre Cloud Shell y, a continuación, vaya al grupo de recursos que comienza con **cloud-shell-storage-**, seleccione **Eliminar grupo de recursos** y siga las indicaciones.

Pasos siguientes

- [Personalización de aplicaciones de Python en Azure Pipelines](#)
- [Configuración de Python en App Service](#)

Quickstart: Sign in users in a sample web app

08/04/2025

Applies to:  Workforce tenants  External tenants ([learn more](#))

In this quickstart, you use a sample web app to show you how to sign in users and call Microsoft Graph API in your workforce tenant. The sample app uses the [Microsoft Authentication Library](#) to handle authentication.

Before you begin, use the **Choose a tenant type** selector at the top of this page to select tenant type. Microsoft Entra ID provides two tenant configurations, [workforce](#) and [external](#). A workforce tenant configuration is for your employees, internal apps, and other organizational resources. An external tenant is for your customer-facing apps.

Prerequisites

- An Azure account with an active subscription. If you don't already have one, [Create an account for free](#).
- This Azure account must have permissions to manage applications. Any of the following Microsoft Entra roles include the required permissions:
 - Application Administrator
 - Application Developer
- A workforce tenant. You can use your Default Directory or [set up a new tenant](#).
- [Visual Studio Code](#) or another code editor.

Node

- Register a new app in the [Microsoft Entra admin center](#), configured for *Accounts in this organizational directory only*. Refer to [Register an application](#) for more details. Record the following values from the application **Overview** page for later use:
 - Application (client) ID
 - Directory (tenant) ID
- Add the following redirect URIs using the **Web** platform configuration. Refer to [How to add a redirect URI in your application](#) for more details.
 - **Redirect URI:** `http://localhost:3000/auth/redirect`
 - **Front-channel logout URL:** `https://localhost:5001/signout-callback-oidc`
- Add a client secret to your app registration. **Do not** use client secrets in production apps. Use certificates or federated credentials instead. For more information, see [add credentials to your application](#).

- [Node.js](#)

Clone or download sample web application

To obtain the sample application, you can either clone it from GitHub or download it as a `.zip` file.

Node

- [Download the `.zip` file](#), then extract it to a file path where the length of the name is fewer than 260 characters or clone the repository:
- To clone the sample, open a command prompt and navigate to where you wish to create the project, and enter the following command:

Consola

```
git clone https://github.com/Azure-Samples/ms-identity-node.git
```

Configure the sample web app

For you to sign in users with the sample app, you need to update it with your app and tenant details:

Node

In the *ms-identity-node* folder, open the *App/.env* file, then replace the following placeholders:

[Expandir tabla](#)

Variable	Description	Example(s)
<code>Enter_the_Cloud_Instance_Id_Here</code>	The Azure cloud instance in which your application is registered	https://login.microsoftonline.com/ (include the trailing forward-slash)

Variable	Description	Example(s)
Enter_the_Tenant_Info_here	Tenant ID or Primary domain	contoso.microsoft.com or aaaabbbb-0000-cccc-1111-dddd2222eeee
Enter_the_Application_Id_Here	Client ID of the application you registered	00001111-aaaa-2222-bbbb-3333cccc4444
Enter_the_Client_Secret_Here	Client secret of the application you registered	A1b-C2d_E3f.H4i,J5k?L6m!N7o-P8q_R9s.T0u
Enter_the_Graph_Endpoint_Here	The Microsoft Graph API cloud instance that your app calls	https://graph.microsoft.com/ (include the trailing forward-slash)
Enter_the_Express_Session_Secret_Here	A random string of characters used to sign the Express session cookie	A1b-C2d_E3f.H4...

After you make changes, your file should look similar to the following snippet:

```
env

CLOUD_INSTANCE=https://login.microsoftonline.com/
TENANT_ID=aaaabbbb-0000-cccc-1111-dddd2222eeee
CLIENT_ID=00001111-aaaa-2222-bbbb-3333cccc4444
CLIENT_SECRET=A1b-C2d_E3f.H4...

REDIRECT_URI=http://localhost:3000/auth/redirect
POST_LOGOUT_REDIRECT_URI=http://localhost:3000

GRAPH_API_ENDPOINT=https://graph.microsoft.com/

EXPRESS_SESSION_SECRET=6DP6v09eLiW7f1E65B8k
```

Run and test sample web app

You've configured your sample app. You can proceed to run and test it.

Node

1. To start the server, run the following commands from within the project directory:

```
Consola
```

```
cd App  
npm install  
npm start
```

2. Go to <http://localhost:3000/>.

3. Select **Sign in** to start the sign-in process.

The first time you sign in, you're prompted to provide your consent to allow the application to sign you in and access your profile. After you're signed in successfully, you'll be redirected back to the application home page.

How the app works

The sample hosts a web server on localhost, port 3000. When a web browser accesses this address, the app renders the home page. Once the user selects **Sign in**, the app redirects the browser to Microsoft Entra sign-in screen, via the URL generated by the MSAL Node library. After user consents, the browser redirects the user back to the application home page, along with an ID and access token.

Related content

Node

- Learn how to build a Node.js web app that signs in users and calls Microsoft Graph API in [Tutorial: Sign in users and acquire a token for Microsoft Graph in a Node.js & Express web app](#).

Inicio rápido: Biblioteca cliente de secretos de Azure Key Vault para Python

Artículo • 14/04/2025

Introducción a la biblioteca cliente de secretos de Azure Key Vault para Python. Siga estos pasos para instalar el paquete y probar el código de ejemplo para realizar tareas básicas. Si usa Key Vault para almacenar secretos, no tendrá que almacenarlos en el código, lo que aumenta la seguridad de las aplicaciones.

[Documentación de referencia de API](#) | [Código fuente de la biblioteca](#) ↗ | [Paquete \(índice de paquetes de Python\)](#) ↗

Prerrequisitos

- Una suscripción a Azure: [cree una cuenta gratuita](#) ↗ .
- [Python 3.7+](#).
- La [CLI de Azure](#) o [Azure PowerShell](#).

En este inicio rápido se supone que está ejecutando la [CLI de Azure](#) o [Azure PowerShell](#) en una ventana de terminal de Linux.

Configuración de un entorno local

En este inicio rápido se usa la biblioteca de identidades de Azure con la CLI de Azure o Azure PowerShell para autenticar el usuario en los servicios de Azure. Los desarrolladores también pueden usar Visual Studio o Visual Studio Code para autenticar sus llamadas. Para más información, consulte [Autenticación del cliente mediante la biblioteca cliente Azure Identity](#).

Inicio de sesión en Azure

CLI de Azure

1. Ejecute el comando `az login`.

Azure CLI

`az login`

Si la CLI puede abrir el explorador predeterminado, lo hará y cargará una página de inicio de sesión de Azure.

En caso contrario, abra una página del explorador en <https://aka.ms/devicelogin> y escriba el código de autorización que se muestra en el terminal.

2. Inicie sesión con las credenciales de su cuenta en el explorador.

Instalación de los paquetes

1. En un símbolo del sistema o en un terminal, cree una carpeta de proyecto adecuada y, después, cree y active un entorno virtual de Python, como se describe en el apartado [Uso de entornos virtuales de Python](#).

2. Instale la biblioteca de identidades de Microsoft Entra:

```
terminal  
pip install azure-identity
```

3. Instale la biblioteca de secretos de Key Vault:

```
terminal  
pip install azure-keyvault-secrets
```

Creación de un grupo de recursos y de un almacén de claves

CLI de Azure

1. Uso del comando `az group create` para crear un grupo de recursos:

```
Azure CLI  
az group create --name myResourceGroup --location eastus
```

Si lo prefiere, puede cambiar "eastus" a una ubicación más próxima.

2. Use `az keyvault create` para crear el almacén de claves:

```
Azure CLI
```

```
az keyvault create --name <your-unique-keyvault-name> --resource-group myResourceGroup
```

Reemplace `<your-unique-keyvault-name>` por un nombre que sea único en todo Azure. Normalmente, se usa el nombre personal o de la empresa, junto con otros números e identificadores.

Establecimiento de la variable de entorno KEY_VAULT_NAME

Nuestro script usará el valor asignado a la variable de entorno `KEY_VAULT_NAME` como el nombre del almacén de claves. Por lo tanto, debe establecer este valor mediante el comando siguiente:

Consola

```
export KEY_VAULT_NAME=<your-unique-keyvault-name>
```

Concesión de acceso al almacén de claves

Para obtener permisos para el almacén de claves mediante [Control de acceso basado en roles \(RBAC\)](#), asigne un rol a su "Nombre principal de usuario" (UPN) mediante el comando de la CLI de Azure `az role assignment create`.

Azure CLI

```
az role assignment create --role "Key Vault Secrets Officer" --assignee "<upn>" --scope "/subscriptions/<subscription-id>/resourceGroups/<resource-group-name>/providers/Microsoft.KeyVault/vaults/<your-unique-keyvault-name>"
```

Reemplace `<upn>`, `<subscription-id>`, `<resource-group-name>` y `<your-unique-keyvault-name>` por los valores reales. El UPN normalmente tendrá el formato de una dirección de correo electrónico (por ejemplo, `username@domain.com`).

Creación del código de ejemplo

La biblioteca cliente de secretos de Azure Key Vault para Python permite administrar los secretos. El siguiente código de ejemplo muestra cómo crear un cliente y cómo establecer, recuperar y eliminar un secreto.

Cree un archivo llamado `kv_secrets.py` que contenga este código.

Python

```
import os
from azure.keyvault.secrets import SecretClient
from azure.identity import DefaultAzureCredential

keyVaultName = os.environ["KEY_VAULT_NAME"]
KVUri = f"https://{keyVaultName}.vault.azure.net"

credential = DefaultAzureCredential()
client = SecretClient(vault_url=KVUri, credential=credential)

secretName = input("Input a name for your secret > ")
secretValue = input("Input a value for your secret > ")

print(f"Creating a secret in {keyVaultName} called '{secretName}' with the value
'{secretValue}' ...")

client.set_secret(secretName, secretValue)

print(" done.")

print(f"Retrieving your secret from {keyVaultName}.")
retrieved_secret = client.get_secret(secretName)

print(f"Your secret is '{retrieved_secret.value}'.")
print(f"Deleting your secret from {keyVaultName} ...")

poller = client.begin_delete_secret(secretName)
deleted_secret = poller.result()

print(" done.")
```

Ejecución del código

Asegúrese de que el código de la sección anterior se encuentra en un archivo llamado `kv_secrets.py`. Luego, ejecute el código con el siguiente comando:

terminal

```
python kv_secrets.py
```

- Si se producen errores de permisos, asegúrese de que ha ejecutado el comando `az keyvault set-policy` o `Set-AzKeyVaultAccessPolicy`.

- Al volver a ejecutar el código con el mismo nombre de secreto, puede aparecer el error "Conflict) Secret <name> is currently in a deleted but recoverable state". Use otro nombre de secreto.

Detalles del código

Autenticación y creación de un cliente

Deben autorizarse las solicitudes de aplicación a la mayor parte de servicios de Azure. El uso de la clase `DefaultAzureCredential` que proporciona la [biblioteca cliente de Azure Identity](#) es el enfoque recomendado para implementar conexiones sin contraseña a los servicios de Azure en el código. `DefaultAzureCredential` admite varios métodos de autenticación y determina el que se debe usar en tiempo de ejecución. Este enfoque permite que la aplicación use diferentes métodos de autenticación en distintos entornos (local frente a producción) sin implementar código específico del entorno.

En este inicio rápido, `DefaultAzureCredential` se autentica en el almacén de claves mediante las credenciales del usuario de desarrollo local que inició sesión en la CLI de Azure. Cuando la aplicación se implementa en Azure, el mismo código `DefaultAzureCredential` puede detectar y usar automáticamente una identidad administrada asignada a una instancia de App Service, máquina virtual u otros servicios. Para más información, consulte [Introducción a la identidad administrada](#).

En el ejemplo siguiente, el nombre del almacén de claves se expande usando el valor de la variable `KVUri`, con el formato "https://<nombre-del-almacén-de-claves>.vault.azure.net".

Python

```
credential = DefaultAzureCredential()  
client = SecretClient(vault_url=KVUri, credential=credential)
```

Almacenamiento de un secreto

Una vez que haya obtenido el objeto de cliente para el almacén de claves, puede almacenar un secreto mediante el método `set_secret`:

Python

```
client.set_secret(secretName, secretValue)
```

La llamada a `set_secret` genera una llamada a la API REST de Azure para el almacén de claves.

Cuando Azure administra la solicitud, autentica la identidad del autor de la llamada (la entidad de servicio) mediante el objeto de credencial que proporcionó al cliente.

Recuperación de un secreto

Para leer un secreto de Key Vault, use el método [get_secret](#):

Python

```
retrieved_secret = client.get_secret(secretName)
```

El valor de secreto se incluye en `retrieved_secret.value`.

Los secretos también se pueden recuperar con el comando [az keyvault secret show](#) de la CLI de Azure o con el cmdlet [Get-AzKeyVaultSecret](#) de Azure PowerShell.

Eliminación de un secreto

Para eliminar un secreto, use el método [begin_delete_secret](#):

Python

```
poller = client.begin_delete_secret(secretName)
deleted_secret = poller.result()
```

El método `begin_delete_secret` es asíncrono y devuelve un objeto de sondeador. La llamada al método `result` del sondeador espera hasta su finalización.

Puede comprobar que se ha eliminado el secreto con el comando [az keyvault secret show](#) de la CLI de Azure o con el cmdlet [Get-AzKeyVaultSecret](#) de Azure PowerShell.

Aun después de haberse eliminado, los secretos permanecen en estado eliminado, pero recuperable, durante un tiempo. Si vuelve a ejecutar el código, use otro nombre de secreto.

Limpieza de recursos

Si también desea experimentar con [certificados](#) y [claves](#), puede volver a usar la instancia de Key Vault que se ha creado en este artículo.

De lo contrario, cuando haya terminado con los recursos creados en este artículo, utilice el siguiente comando para eliminar el grupo de recursos y todos los recursos que contiene:

CLI de Azure

Azure CLI

```
az group delete --resource-group myResourceGroup
```

Pasos siguientes

- [Información general de Azure Key Vault](#)
- [Guía del desarrollador de Azure Key Vault](#)
- [Introducción a la seguridad de Key Vault](#)
- [Autenticación con Key Vault](#)

Inicio rápido: Creación e implementación de código de función en Azure mediante Visual Studio Code

15/08/2025

Use Visual Studio Code para crear una función que responda a las solicitudes HTTP de una plantilla. Use GitHub Copilot para mejorar el código de función generado, comprobar las actualizaciones de código localmente y, a continuación, implementarlo en el plan de hospedaje de consumo flexible sin servidor en Azure Functions.

Completar este inicio rápido acarrea un coste pequeño, de unos pocos centavos de dólar o menos, en su cuenta de Azure.

Asegúrese de seleccionar el lenguaje de desarrollo preferido en la parte superior del artículo.

Prerrequisitos

- Una cuenta de Azure con una suscripción activa. [Cree una cuenta gratuita](#).
- [Visual Studio Code](#) en una de las [plataformas admitidas](#).
- La [extensión de Azure Functions](#) para Visual Studio Code.
- Use versiones de Python [compatibles con Azure Functions](#). Para obtener más información, vea [Instalación de Python](#).
- La [extensión de Python](#) para Visual Studio Code

Instalar o actualizar Core Tools

La extensión de Azure Functions para Visual Studio Code se integra con Azure Functions Core Tools para que pueda ejecutar y depurar las funciones localmente en Visual Studio Code mediante el entorno de ejecución de Azure Functions. Antes de empezar, es una buena idea instalar Core Tools localmente o actualizar una instalación existente para usar la versión más reciente.

En Visual Studio Code, seleccione F1 para abrir la paleta de comandos y, a continuación, busque y ejecute el comando **Azure Functions: Instalar o actualizar Core Tools**.

Este comando intenta iniciar una instalación basada en paquetes de la versión más reciente de Core Tools o bien actualizar una instalación basada en paquetes existente. Si no tiene npm o

Homebrew instalado en el equipo local, en su lugar debe [instalar o actualizar manualmente Core Tools](#).

Creación del proyecto local

En esta sección, usará Visual Studio Code para crear un proyecto local de Azure Functions en su lenguaje preferido. Más adelante en el artículo, actualizará, ejecutará y, a continuación, publicará el código de función en Azure.

1. En Visual Studio Code, presione `F1` para abrir la paleta de comandos y busque y ejecute el comando `Azure Functions: Create New Project...`.
2. Elija una ubicación de directorio para el área de trabajo del proyecto y, después, seleccione el botón **Seleccionar**. Debe crear una nueva carpeta o elegir una carpeta vacía en la que ubicar el área de trabajo del proyecto. No elija una carpeta de proyecto que ya forme parte de un área de trabajo.
3. Escriba la siguiente información cuando se le indique:

 Expandir tabla

Pronto	Selección
Seleccione un lenguaje	Elija <code>Python</code> .
Seleccionar el intérprete de Python para crear un entorno virtual	Elija el intérprete de Python que prefiera usar. Si no se muestra una opción, escriba la ruta de acceso completa al archivo binario de Python.
Seleccione una plantilla para la primera función del proyecto	Elija <code>HTTP trigger</code> .
Nombre de la función que desea crear	Escribe <code>HttpExample</code> .
Nivel de autorización	Elija el valor <code>ANONYMOUS</code> , que permitirá que cualquier cliente llame al punto de conexión de la función. Para más información, consulte Nivel de autorización .
Seleccione cómo desea abrir el proyecto	Elija <code>Open in current window</code> .

Con esta información, Visual Studio Code genera un proyecto de código para Azure Functions con un punto de conexión de función de desencadenador HTTP. Los archivos del proyecto locales se pueden ver en Explorer. Para obtener más información sobre los archivos que se crean, consulte [Archivos del proyecto generados](#).

4. En el archivo local.settings.json, actualice la configuración AzureWebJobsStorage como en el ejemplo siguiente:

JSON

"AzureWebJobsStorage": "UseDevelopmentStorage=true",

Esto indica al host de Functions local que use el emulador de almacenamiento para la conexión de almacenamiento que requiere el modelo Python v2. Al publicar el proyecto en Azure, esta configuración usa la cuenta de almacenamiento predeterminada en su lugar. Si usa una cuenta de Azure Storage durante el desarrollo local, establezca la cadena de conexión de la cuenta de almacenamiento aquí.

Inicio del emulador

1. En Visual Studio Code, presione **F1** para abrir la paleta de comandos. En la paleta de comandos, busque y seleccione **Azurite: Start**.
 2. Compruebe la barra inferior y compruebe que los servicios de emulación de Azurite se están ejecutando. Si es así, ahora puede ejecutar la función localmente.

Ejecución local de la función

Visual Studio Code se integra con [Azure Functions Core Tools](#) para que pueda ejecutar este proyecto en un equipo de desarrollo local antes de publicarlo en Azure.

1. Para iniciar la función localmente, presione **F5** o el icono **Ejecutar y depurar** en la barra de actividad del lado izquierdo. En el panel **Terminal** se mostrará la salida de Core Tools. La aplicación se inicia en el panel **Terminal**. Puede ver el punto de conexión de la dirección URL de la función desencadenada por HTTP que se ejecuta localmente.

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    AZURE: ACTIVITY LOG    ⚙ host start - Task ✓    +    □    🗑    ^    ×

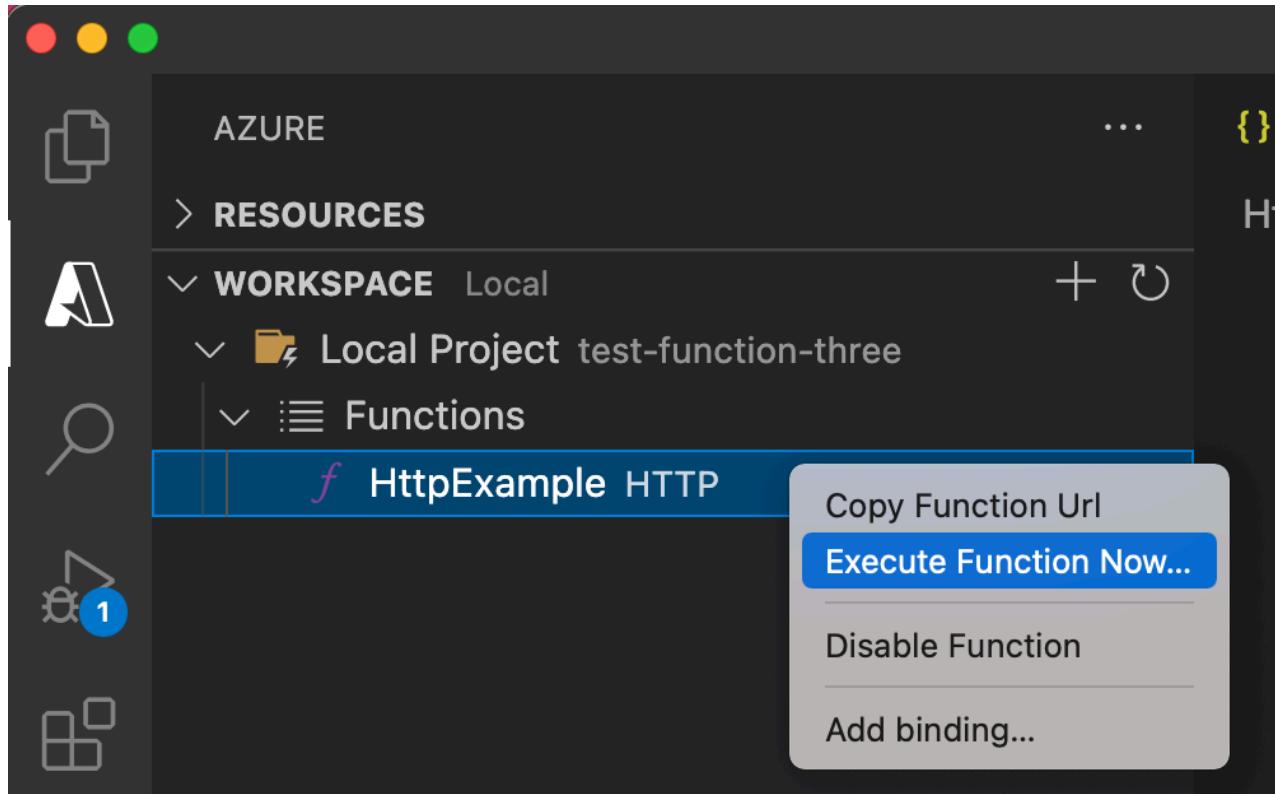
Functions:

HttpExample: [GET,POST] http://localhost:7071/api/HttpExample

For detailed output, run func with --verbose flag.
[2022-05-25T18:45:53.470Z] Worker process started and initialized.
[2022-05-25T18:45:54.960Z] Host lock lease acquired by instance ID '000000000000000000000000E24CCCCAC'.
```

Si tiene problemas para ejecutarlo en Windows, asegúrese de que el terminal predeterminado de Visual Studio Code no esté establecido en **WSL Bash**.

2. Mientras Core Tools todavía se ejecuta en el **Terminal**, elija el ícono de Azure en la barra de actividades. En el **Área de trabajo**, expanda el desplegable **Proyecto local>Funciones**. Haga clic con el botón derecho (Windows) o `ctrl - clic` (macOS) en la nueva función y seleccione **Ejecutar función ahora....**



3. En **Enter request body** (Especificar el cuerpo de la solicitud) verá el valor del cuerpo del mensaje de solicitud de `{ "name": "Azure" }`. Presione Entrar para enviar este mensaje de solicitud a la función.
4. Cuando la función se ejecuta localmente y devuelve una respuesta, se genera una notificación en Visual Studio Code. La información sobre la ejecución de la función se muestra en el panel **Terminal**.
5. Con el panel **Terminal** enfocado, presione `ctrl + c` para detener Core Tools y desconectar el depurador.

Después de comprobar que la función se ejecuta correctamente en el equipo local, puede usar opcionalmente herramientas de IA, como GitHub Copilot en Visual Studio Code, para actualizar el código de función generado por plantillas.

Uso de IA para normalizar y validar los datos

Se trata de un mensaje de ejemplo para El chat de Copilot que actualiza el código de función existente para recuperar parámetros de la cadena de consulta o el cuerpo JSON, aplicar conversiones de formato o tipo y devolverlos como JSON en la respuesta:

Solicitud de Copilot

Modify the function to accept name, email, and age from the JSON body of the request. If any of these parameters are missing from the query string, read them from the JSON body. Return all three parameters in the JSON response, applying these rules:

Title-case the name

Lowercase the email

Convert age to an integer if possible, otherwise return "not provided"

Use sensible defaults if any parameter is missing

Puede personalizar la solicitud para agregar detalles según sea necesario y, a continuación, volver a ejecutar la aplicación localmente y comprobar que funciona según lo previsto después de que el código cambie. Esta vez, use un cuerpo del mensaje como:

JSON

```
{ "name": "devon torres", "email": "torres.devon@contoso.com", "age": "34" }
```

Sugerencia

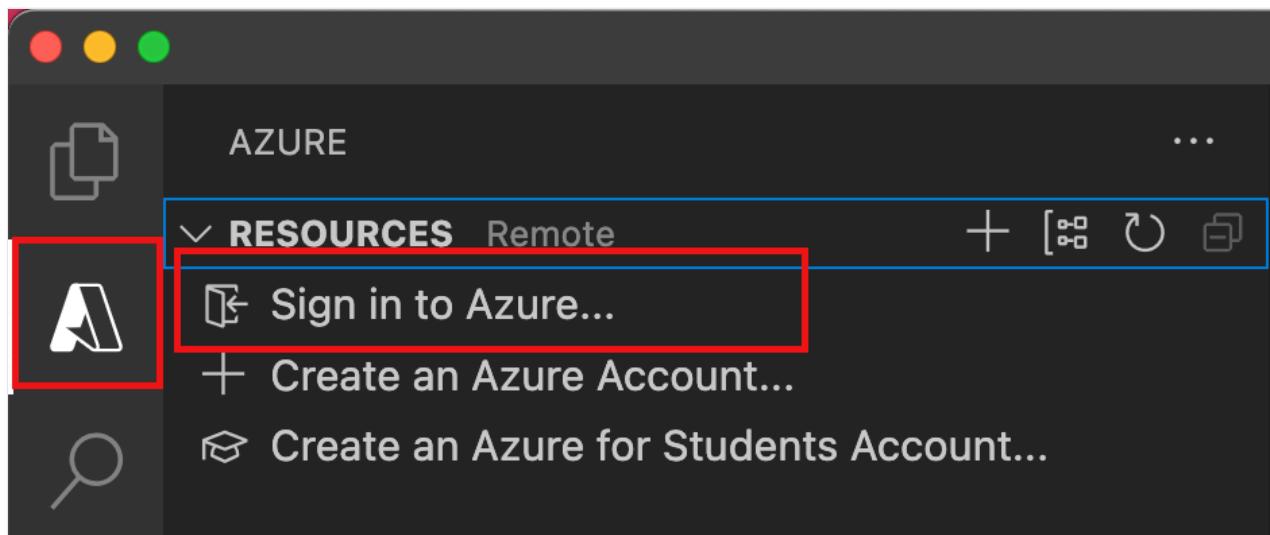
GitHub Copilot se basa en la inteligencia artificial, por lo que son posibles sorpresas y errores. Si se producen errores durante la ejecución, pegue el mensaje de error en la ventana de chat, seleccione **Modo de agente** y pida a Copilot que le ayude a resolver el error. Para obtener más información, consulte [Preguntas más frecuentes sobre Copilot](#).

Cuando esté satisfiado con la aplicación, puede usar Visual Studio Code para publicar el proyecto directamente en Azure.

Inicio de sesión en Azure

Para poder crear recursos de Azure o publicar la aplicación, debe iniciar sesión en Azure.

1. Si aún no ha iniciado sesión, en la **barra de actividad**, seleccione el ícono de Azure. A continuación, en **Recursos**, seleccione **Iniciar sesión en Azure**.



Si ya ha iniciado sesión y puede ver las suscripciones existentes, vaya a la siguiente sección. Si aún no tiene una cuenta de Azure, seleccione **Crear una cuenta de Azure**. Los alumnos pueden seleccionar **Crear una cuenta de Azure for Students**.

2. Cuando se le solicite en el explorador, seleccione su cuenta de Azure e inicie sesión con las credenciales de la cuenta de Azure. Si opta por crear una cuenta, podrá iniciar sesión una vez que haya completado el proceso de creación.
3. Después de iniciar sesión correctamente, puede cerrar la nueva ventana del explorador. Las suscripciones que pertenecen a su cuenta de Azure se muestran en la barra lateral.

Cree la aplicación de funciones en Azure

En esta sección, creará una aplicación de funciones en el plan de Consumo flexible junto con los recursos relacionados de la suscripción de Azure. Muchas de las decisiones de creación de recursos se toman en función de los comportamientos predeterminados. Para obtener más control sobre los recursos creados, en su lugar debe [crear la aplicación de funciones con opciones avanzadas](#).

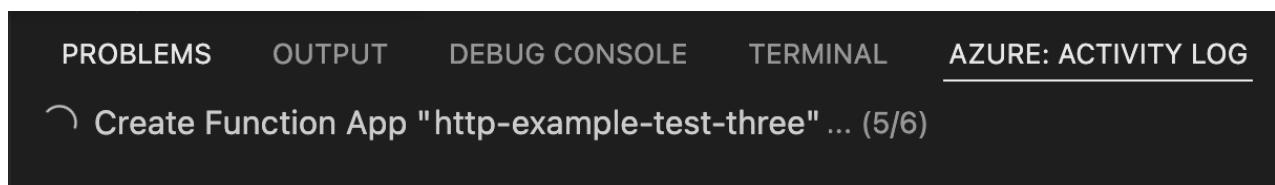
1. En Visual Studio Code, seleccione F1 para abrir la paleta de comandos. En el símbolo del sistema (>), escriba y seleccione **Azure Functions: Crear aplicación de funciones en Azure**.
2. Escriba la siguiente información cuando se le indique:

Expandir tabla

Pronto	Acción
Selección de la suscripción	Seleccione la suscripción de Azure que se va a usar. El símbolo del sistema no aparece cuando solo tiene una suscripción visible en

Pronto	Acción
	Recursos.
Escriba un nuevo nombre de aplicación de funciones.	Escriba un nombre único global que sea válido en una ruta del URL. El nombre que escriba se valida para asegurarse de que es único en Azure Functions.
Seleccionar una ubicación para los nuevos recursos	Seleccionar una región de Azure. Para mejorar el rendimiento, seleccione una región cerca de usted. Solo se muestran las regiones admitidas por los planes de consumo flexible.
Seleccione una pila en tiempo de ejecución	Seleccione la versión de idioma que se ejecuta actualmente localmente.
Selección del tipo de autenticación de recursos	Seleccione Identidad administrada , que es la opción más segura para conectarse a la cuenta de almacenamiento de host predeterminada .

En el panel **Azure: Registro de actividad**, la extensión de Azure muestra el estado de los recursos individuales a medida que se crean en Azure.



3. Cuando se crea la aplicación de funciones, se crean los siguientes recursos relacionados en la suscripción de Azure. Los recursos se denominan en función del nombre especificado para la aplicación de funciones.

- Un [grupo de recursos](#), que es un contenedor lógico de recursos relacionados.
- Una aplicación de funciones, que proporciona el entorno para ejecutar el código de función. Una aplicación de funciones permite agrupar funciones como una unidad lógica para facilitar la administración, la implementación y el uso compartido de recursos en el mismo plan de hospedaje.
- Un plan de Azure App Service, que define el host subyacente para la aplicación de funciones.
- Una [cuenta de Azure Storage](#) estándar que usa el host de Functions para mantener el estado y otra información sobre tu aplicación de funciones.
- Una instancia de Application Insights conectada a la aplicación de funciones y que realiza un seguimiento del uso de las funciones en la aplicación.
- Una identidad administrada asignada por el usuario que se agrega al rol [Colaborador de datos de Storage Blob](#) en la nueva cuenta de almacenamiento de host predeterminada.

Una vez que se haya creado la aplicación de función se mostrará una notificación y se aplicará el paquete de implementación.

Sugerencia

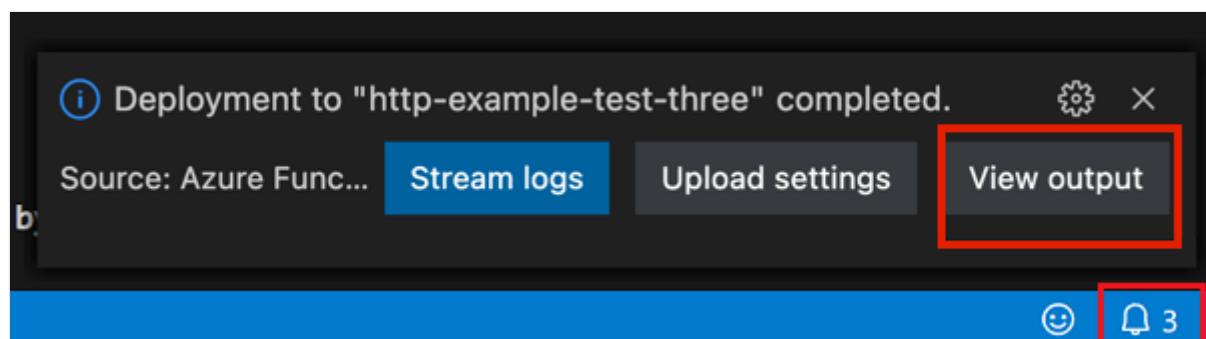
De forma predeterminada, los recursos de Azure necesarios para la aplicación de funciones se crean según el nombre que escriba para la aplicación de funciones. De manera predeterminada, los recursos se crean con la aplicación de funciones en el mismo grupo de recursos nuevo. Si desea personalizar los nombres de los recursos asociados o reutilizar los recursos existentes, [publique el proyecto con opciones de creación avanzadas](#).

Implementar el proyecto en Azure

Importante

Los procesos de implementación en aplicaciones de funciones existentes siempre sobrescriben el contenido de esas aplicaciones en Azure.

1. En la paleta de comandos, escriba y seleccione **Azure Functions: implementar la aplicación de funciones**.
2. Seleccione la aplicación de funciones que acaba de crear. Cuando se le solicite sobrescribir las implementaciones anteriores, seleccione **Implementar** para implementar el código de función en el nuevo recurso de aplicación de funciones.
3. Cuando se complete la implementación, seleccione **Ver la salida** para ver los resultados de creación e implementación, incluidos los recursos de Azure que creó. Si se pierde la notificación, seleccione el ícono de campana en la esquina inferior derecha para verlo de nuevo.



Ejecución de la función en Azure

1. Presione **F1** para mostrar la paleta de comandos y luego busque y ejecute el comando **Azure Functions:Execute Function Now...**. Si se le solicita, seleccione la suscripción.
2. Seleccione el nuevo recurso de la aplicación de funciones y **HttpExample** como función.
3. En **Escriba el cuerpo de la solicitud** escriba `{ "name": "Contoso", "email": "me@contoso.com", "age": "34" }`, y presione Entrar para enviar este mensaje de solicitud a la función.
4. Cuando la función se ejecuta en Azure, la respuesta se muestra en el área de notificación. Expanda la notificación para revisar la respuesta completa.

Solución de problemas

Use la tabla siguiente para resolver los problemas más comunes detectados al usar este artículo.

 Expandir tabla

Problema	Solución
¿No puede crear un proyecto de función local?	Asegúrese de tener instalada la extensión de Azure Functions .
¿No puede ejecutar la función localmente?	Asegúrese de tener instalada la última versión de Azure Functions Core Tools . En la ejecución en Windows, asegúrese de que el shell del terminal predeterminado de Visual Studio Code no se haya establecido en WSL Bash.
¿No puede implementar la función en Azure?	Revise la salida para obtener información sobre el error. El ícono de campana que aparece en la esquina inferior derecha también permite ver la salida. ¿Ha realizado la publicación en una aplicación de funciones existente? Esa acción sobrescribe el contenido de la aplicación en Azure.
¿No se pudo ejecutar la aplicación de funciones basada en la nube?	No olvide usar la cadena de consulta para enviar parámetros.

Limpieza de recursos

Si va al [paso siguiente](#) y agrega un enlace de cola de Azure Storage a una función, tendrá que conservar todos los recursos intactos para basarse en lo que ya ha hecho.

De lo contrario, puede usar los pasos siguientes para eliminar la aplicación de funciones y sus recursos relacionados para evitar incurrir en costos adicionales.

1. En Visual Studio Code, seleccione el icono de Azure para abrir el explorador de Azure.
2. En la sección Grupos de recursos, busque el grupo de recursos.
3. Haga clic con el botón derecho en el grupo de recursos y seleccione **Eliminar**.

Para más información sobre los costos de Functions, consulte [Estimación de los costos según el plan de consumo](#).

Pasos siguientes

Ha usado [Visual Studio Code](#) para crear una aplicación de función con una función simple desencadenada por HTTP. En los artículos siguientes, ampliará esa función mediante la conexión a Azure Cosmos DB o Azure Storage. Para más información sobre cómo conectarse a otros servicios de Azure, consulte [Incorporación de enlaces a una función existente de Azure Functions](#). Si desea más información sobre la seguridad, consulte [Protección de Azure Functions](#).

[Conexión a Azure Cosmos DB](#)

[Conexión a Azure Queue Storage](#)

Inicio rápido: Creación de una función en Azure desde la línea de comandos

15/08/2025

En este artículo, usará herramientas de línea de comandos locales para crear una función que responda a las solicitudes HTTP. Después de comprobar el código localmente, se implementa en un plan de hospedaje de consumo flexible sin servidor en Azure Functions.

Completar este inicio rápido acarrea un coste pequeño, de unos pocos centavos de dólar o menos, en su cuenta de Azure.

Asegúrese de seleccionar el lenguaje de desarrollo preferido en la parte superior del artículo.

Prerequisites

- Una cuenta de Azure con una suscripción activa. [Cree una cuenta gratuita ↗](#).
- [Python 3.11 ↗](#).
- [Azure CLI](#)
- El [jq procesador JSON de la línea de comandos ↗](#), que se usa para analizar la salida JSON, y también está disponible en Azure Cloud Shell.

Instalación de Azure Functions Core Tools

La manera recomendada de instalar Core Tools depende del sistema operativo del equipo de desarrollo local.

Windows

En los pasos siguientes se utiliza Windows Installer (MSI) para instalar Core Tools v4.x. Para más información sobre otros instaladores basados en paquetes, consulte el [archivo Léame de Core Tools ↗](#).

Descargue y ejecute el instalador de Core Tools según su versión de Windows:

- v4.x: [Windows de 64 bits ↗](#) (recomendado. [La depuración de Visual Studio Code requiere 64 bits.](#))
- v4.x: [Windows de 32 bits ↗](#)

Si ha usado previamente Windows Installer (MSI) para instalar Core Tools en Windows, debe desinstalar la versión anterior desde Agregar/Quitar programas antes de instalar la versión más reciente.

Creación y activación de un entorno virtual

En una carpeta adecuada, ejecute los comandos siguientes para crear y activar un entorno virtual denominado `.venv`. Asegúrese de usar una de las [versiones de Python](#) compatibles con Azure Functions.

```
bash
```

```
Bash
```

```
python -m venv .venv
```

```
Bash
```

```
source .venv/bin/activate
```

Si Python no instaló el paquete `venv` en la distribución de Linux, ejecute el siguiente comando:

```
Bash
```

```
sudo apt-get install python3-venv
```

Ejecute todos los comandos siguientes en este entorno virtual activado.

Creación de un proyecto de código local y una función

En Azure Functions, el proyecto de código es una aplicación que contiene una o varias funciones individuales que cada una responde a un desencadenador específico. Todas las funciones de un proyecto comparten las mismas configuraciones y se implementan como una unidad en Azure. En esta sección, creará un proyecto de código que contiene una sola función.

1. En un terminal o consola, ejecute este `func init` comando para crear un proyecto de aplicación de funciones en la carpeta actual.

Consola

```
func init --worker-runtime python
```

2. Use este [func new](#) comando para agregar una función al proyecto:

Consola

```
func new --name HttpExample --template "HTTP trigger" --authlevel "anonymous"
```

Se agrega un nuevo archivo de código al proyecto. En este caso, el `--name` argumento es el nombre único de la función (`HttpExample`) y el `--template` argumento especifica un desencadenador HTTP.

La carpeta raíz del proyecto contiene varios archivos para el proyecto, incluidos los archivos de configuración denominados [local.settings.json](#) y [host.json](#). Dado que `local.settings.json` puede contener secretos descargados de Azure, el archivo se excluye del control de código fuente de forma predeterminada en el archivo `.gitignore`.

Ejecución local de la función

Compruebe la nueva función ejecutando el proyecto localmente y llamando al punto de conexión de la función.

1. Use este comando para iniciar el host en tiempo de ejecución de Azure Functions local en la raíz de la carpeta del proyecto:

Consola

```
func start
```

Hacia el final de la salida, deberían aparecer las líneas siguientes:

```
...
```

```
Now listening on: http://0.0.0.0:7071  
Application started. Press Ctrl+C to shut down.
```

```
Http Functions:
```

```
HttpExample: [GET,POST] http://localhost:7071/api/HttpExample
```

```
...
```

ⓘ Nota

Si el punto de conexión `HttpExample` no aparece como se esperaba, es probable que haya iniciado el host desde fuera de la carpeta raíz del proyecto. En ese caso, use `Ctrl+C` para detener el host, vaya a la carpeta raíz del proyecto y vuelva a ejecutar el comando anterior.

2. Copie la dirección URL de la función `HttpExample` de esta salida en un explorador y vaya a la dirección URL de la función y recibirá una respuesta correcta con un mensaje de tipo "hola mundo".
3. Cuando haya terminado, use `Ctrl+C` y elija `y` detener el host de funciones.

Creación de recursos auxiliares de Azure para la función

Para poder implementar el código de función en Azure, debe crear estos recursos:

- Un [grupo de recursos](#), que es un contenedor lógico de recursos relacionados.
- Una [cuenta de almacenamiento](#) predeterminada, que usa el host de Functions para mantener el estado y otra información sobre las funciones.
- Una [identidad administrada asignada por el usuario](#), que el host de Functions usa para conectarse a la cuenta de almacenamiento predeterminada.
- Una aplicación de funciones, que proporciona el entorno para ejecutar el código de función. Una aplicación de funciones se asigna al proyecto de funciones y le permite agrupar funciones como una unidad lógica para facilitar la administración, la implementación y el uso compartido de recursos.

Use los comandos de la CLI de Azure en estos pasos para crear los recursos necesarios.

1. Si todavía no lo ha hecho, inicie sesión en Azure:

```
Azure CLI
```

```
az login
```

El comando `az login` inicia sesión en la cuenta de Azure. Omita este paso cuando se ejecute en Azure Cloud Shell.

2. Si aún no lo ha hecho, use este `az extension add` comando para instalar la extensión de Application Insights:

```
Azure CLI
```

```
az extension add --name application-insights
```

3. Use este comando `az group create` para crear un grupo de recursos denominado `AzureFunctionsQuickstart-rg` en la región elegida:

```
Azure CLI
```

```
az group create --name "AzureFunctionsQuickstart-rg" --location "<REGION>"
```

En este ejemplo, reemplace `<REGION>` por una región cercana que admite el plan de consumo flexible. Use el comando `az functionapp list-flecxconsumption-locations` para ver la lista de regiones admitidas actualmente.

4. Use este comando `az storage account create` para crear una cuenta de almacenamiento de uso general en el grupo de recursos y la región:

```
Azure CLI
```

```
az storage account create --name <STORAGE_NAME> --location "<REGION>" --resource-group "AzureFunctionsQuickstart-rg" \ --sku "Standard_LRS" --allow-blob-public-access false --allow-shared-key-access false
```

En este ejemplo, reemplace por `<STORAGE_NAME>` un nombre adecuado para usted y único en Azure Storage. Los nombres deben contener entre 3 y 24 caracteres y solo letras minúsculas. `Standard_LRS` especifica una cuenta de uso general, que es compatible con Functions. El acceso a esta nueva cuenta solo es posible mediante identidades autenticadas por Microsoft. Entra a las que se les han otorgado permisos para recursos específicos.

5. Use este script para crear una identidad administrada asignada por el usuario, analizar las propiedades JSON devueltas del objeto mediante `jq` y conceder `Storage Blob Data Owner` permisos en la cuenta de almacenamiento predeterminada:

```
Azure CLI
```

```
output=$(az identity create --name "func-host-storage-user" --resource-group "AzureFunctionsQuickstart-rg" --location <REGION> \ --query "{userId:id, principalId: principalId, clientId: clientId}" -o json)
```

```
userId=$(echo $output | jq -r '.userId')
principalId=$(echo $output | jq -r '.principalId')
clientId=$(echo $output | jq -r '.clientId')

storageId=$(az storage account show --resource-group
"AzureFunctionsQuickstart-rg" --name <STORAGE_NAME> --query 'id' -o tsv)
az role assignment create --assignee-object-id $principalId --assignee-
principal-type ServicePrincipal \
--role "Storage Blob Data Owner" --scope $storageId
```

Si no tiene la `jq` utilidad en la shell de Bash local, está disponible en Azure Cloud Shell. En este ejemplo, reemplace `<STORAGE_NAME>` y `<REGION>` por el nombre y la región de la cuenta de almacenamiento predeterminados, respectivamente.

El comando `az identity create` crea una identidad denominada `func-host-storage-user`. El `principalId` devuelto se utiliza para asignar permisos a esta nueva identidad en la cuenta de almacenamiento predeterminada mediante el comando `az role assignment create`. El `az storage account show` comando se usa para obtener el identificador de la cuenta de almacenamiento.

6. Use este comando `az functionapp create` para crear la aplicación de funciones en Azure:

Azure CLI

```
az functionapp create --resource-group "AzureFunctionsQuickstart-rg" --name
<APP_NAME> --flexconsumption-location <REGION> \
--runtime python --runtime-version <LANGUAGE_VERSION> --storage-account
<STORAGE_NAME> \
--deployment-storage-auth-type UserAssignedIdentity --deployment-storage-
auth-value "func-host-storage-user"
```

En este ejemplo, reemplace estos marcadores de posición por los valores adecuados:

- `<APP_NAME>`: un nombre único global adecuado para usted. `<APP_NAME>` también es el dominio DNS predeterminado de la aplicación de función.
- `<STORAGE_NAME>`: el nombre de la cuenta que usó en el paso anterior.
- `<REGION>`: la región actual.
- `<LANGUAGE_VERSION>`: use la misma [versión de pila de idiomas compatible](#) que ha comprobado localmente.

Este comando crea una función de aplicación que se ejecuta en el tiempo de ejecución del lenguaje especificado en Linux, bajo el [Plan de Consumo Flexible](#), el cual es gratuito para el nivel de uso que se incurre aquí. El comando también crea una instancia de Azure Application Insights asociada en el mismo grupo de recursos, con el que puede usar para

supervisar las ejecuciones de la aplicación de funciones y ver los registros. Para más información, consulte [Supervisión de Azure Functions](#). La instancia no incurrirá en ningún costo hasta que se active.

7. Use este script para agregar la identidad administrada asignada por el usuario al rol [Publicador de métricas de supervisión](#) en la instancia de Application Insights:

Azure CLI

```
appInsights=$(az monitor app-insights component show --resource-group "AzureFunctionsQuickstart-rg" \
    --app <APP_NAME> --query "id" --output tsv)
principalId=$(az identity show --name "func-host-storage-user" --resource-group "AzureFunctionsQuickstart-rg" \
    --query principalId -o tsv)
az role assignment create --role "Monitoring Metrics Publisher" --assignee $principalId --scope $appInsights
```

En este ejemplo, reemplace `<APP_NAME>` por el nombre de la aplicación de funciones. El comando `az role assignment create` agrega el usuario al rol. El identificador de recurso de la instancia de Application Insights y el ID principal de su usuario se obtienen mediante los comandos `az monitor app-insights component show` y `az identity show`, respectivamente.

Actualización de la configuración de la aplicación

Para permitir que el host de Functions se conecte a la cuenta de almacenamiento predeterminada mediante secretos compartidos, debe reemplazar la cadena de conexión `AzureWebJobsStorage` por varias opciones de configuración que tengan el prefijo `AzureWebJobsStorage__`. Esta configuración define una configuración compleja que la aplicación usa para conectarse al almacenamiento y Application Insights con una identidad administrada asignada por el usuario.

1. Use este script para obtener el identificador de cliente de la identidad administrada asignada por el usuario y usarlo para definir conexiones de identidad administrada tanto al almacenamiento como a Application Insights:

Azure CLI

```
clientId=$(az identity show --name func-host-storage-user \
    --resource-group AzureFunctionsQuickstart-rg --query 'clientId' -o tsv)
az functionapp config appsettings set --name <APP_NAME> --resource-group "AzureFunctionsQuickstart-rg" \
    --settings AzureWebJobsStorage__accountName=<STORAGE_NAME> \
    AzureWebJobsStorage__credential=managedidentity
```

```
AzureWebJobsStorage__clientId=$clientId \
APPLICATIONINSIGHTS_AUTHENTICATION_STRING="ClientId=$clientId;Authorization=AD"
```

En este script, reemplace `<APP_NAME>` y `<STORAGE_NAME>` por los nombres de la aplicación de funciones y la cuenta de almacenamiento, respectivamente.

- Ejecute el comando `az functionapp config appsettings delete` para quitar la configuración de cadena de conexión existente `AzureWebJobsStorage`, que contiene una clave secreta compartida:

Azure CLI

```
az functionapp config appsettings delete --name <APP_NAME> --resource-group "AzureFunctionsQuickstart-rg" --setting-names AzureWebJobsStorage
```

En este ejemplo, reemplace por `<APP_NAME>` los nombres de la aplicación de funciones.

En este momento, el host de Functions puede conectarse a la cuenta de almacenamiento de forma segura mediante identidades administradas en lugar de secretos compartidos. Ahora puede implementar el código del proyecto en los recursos de Azure.

Implementación del proyecto de función en Azure

Después de haber creado correctamente su aplicación de funciones en Azure, estará listo para implementar el proyecto de funciones local mediante el comando `func azure functionapp publish`.

En la carpeta del proyecto raíz, ejecute este comando `func azure functionapp publish`:

Consola

```
func azure functionapp publish <APP_NAME>
```

En este ejemplo, reemplace `<APP_NAME>` por el nombre de la aplicación. Una implementación correcta muestra resultados similares a la salida siguiente (truncada por motivos de simplicidad):

```
...
```

```
Getting site publishing info...
Creating archive for current directory...
Performing remote build for functions project.
```

```
...  
Deployment successful.  
Remote build succeeded!  
Syncing triggers...  
Functions in msdocs-azurefunctions-qs:  
    HttpExample - [httpTrigger]  
        Invoke url: https://msdocs-azurefunctions-  
qs.azurewebsites.net/api/httpexample
```

Invocación de la función en Azure

Como la función usa un desencadenador HTTP y admite solicitudes GET, realice una solicitud HTTP a su dirección URL para invocarla. Es más fácil ejecutar una solicitud GET en un explorador.

Copie la dirección **URL de invocación** completa que se muestra en la salida del comando publish en una barra de direcciones del explorador.

La dirección URL del punto de conexión debe tener un aspecto similar al de este ejemplo:

```
https://contoso-app.azurewebsites.net/api/httpexample
```

Cuando vaya a esta dirección URL, el explorador debe mostrar una salida similar a cuando ejecutó la función localmente.

Limpieza de recursos

Si continúa con el [paso siguiente](#) y agrega un enlace de salida de cola de Azure Storage, mantenga todos los recursos en su lugar, ya que se basará en lo que ya ha hecho.

De lo contrario, use el siguiente comando para eliminar el grupo de recursos y todos los recursos que contiene para evitar incurrir en costos adicionales.

Azure CLI

```
Azure CLI
```

```
az group delete --name AzureFunctionsQuickstart-rg
```

Pasos siguientes

Conexión a Azure Queue Storage

Conexión de Azure Functions a Azure Storage mediante Visual Studio Code

Artículo • 25/04/2024

Azure Functions le permite conectar servicios de Azure y otros recursos a funciones sin tener que escribir su propio código de integración. Estos *enlaces*, que representan la entrada y la salida, se declaran dentro de la definición de función. Los datos de los enlaces se proporcionan a la función como parámetros. Un *desencadenador* es un tipo especial de enlace de entrada. Si bien una función tiene un único desencadenador, puede tener varios enlaces de entrada y salida. Para más información, consulte [Conceptos básicos sobre los enlaces y desencadenadores de Azure Functions](#).

En este artículo se muestra cómo usar Visual Studio Code para conectar el servicio Azure Storage a la función que creó al completar el anterior artículo de inicio rápido. El enlace de salida que se agrega a esta función escribe datos de la solicitud HTTP en un mensaje de la cola de Azure Queue Storage.

La mayoría de los enlaces requieren una cadena de conexión almacenada que se usa en Functions para acceder al servicio enlazado. Para que el proceso sea más fácil, usará la cuenta de almacenamiento que creó junto con la aplicación de funciones. La conexión a esta cuenta ya está almacenada en una configuración de aplicación llamada `AzureWebJobsStorage`.

Configuración del entorno local

Antes de comenzar, deberá cumplir los siguientes requisitos:

- Instale la [extensión de Azure Storage para Visual Studio Code](#).
- Instale el [Explorador de Azure Storage](#). El Explorador de Storage es una herramienta que utilizará para examinar los mensajes en cola que genere el enlace de salida. El Explorador de Storage se admite en sistemas operativos basados en Linux, Windows y macOS.
- Complete los pasos de la [parte 1 del inicio rápido de Visual Studio Code](#).

En este artículo se da por supuesto que ya inició sesión en la suscripción de Azure desde Visual Studio Code. Puede iniciar sesión mediante la ejecución de `Azure: Sign In` desde la paleta de comandos.

Descarga de la configuración de la aplicación de función

En el [artículo de inicio rápido anterior](#) creó una aplicación de funciones en Azure junto con la cuenta de almacenamiento que se requería para ello. La cadena de conexión de esta cuenta se almacena de forma segura en la configuración de la aplicación en Azure. En este artículo, escribirá mensajes en una cola de almacenamiento de la misma cuenta. Si la función se ejecuta localmente, deberá descargar la configuración de la aplicación al archivo `local.settings.json` para conectarse a la cuenta de almacenamiento.

1. Presione `F1` para abrir la paleta de comandos y busque y ejecute el comando

```
Azure Functions: Download Remote Settings....
```

2. Elija la aplicación de funciones que creó en el artículo anterior. Seleccione **Sí a todo** para sobrescribir la configuración local existente.

i Importante

El archivo `local.settings.json` contiene secretos, así que este nunca se publica y, además, se excluye de las características de control de código fuente.

3. Copie el valor `AzureWebJobsStorage`, que constituye la clave del valor de la cadena de conexión de la cuenta de almacenamiento. Esta conexión se usa para comprobar que el enlace de salida funciona según lo previsto.

Registro de extensiones de enlace

Dado que está utilizando un enlace de salida de Queue Storage, para poder ejecutar el proyecto debe tener instalada la extensión de enlaces de Storage.

El proyecto se ha configurado para usar [conjuntos de extensiones](#), que instalan automáticamente un conjunto predefinido de paquetes de extensiones.

Los conjuntos de extensiones ya están habilitados en el archivo `host.json` en la raíz del proyecto, que tiene un aspecto similar al siguiente ejemplo:

JSON

```
{  
  "version": "2.0",  
  "extensionBundle": {  
    "id": "Microsoft.Azure.Functions.ExtensionBundle",  
    "version": "[3.*, 4.0.0)"]  
}
```

```
}
```

Ahora podrá agregar el enlace de salida de almacenamiento al proyecto.

Adición de un enlace de salida

Los atributos de enlace se definen mediante la decoración de código de función específico en el archivo *function_app.py*. Use el decorador `queue_output` para agregar un [enlace de salida de Azure Queue Storage](#).

Al usar el decorador `queue_output`, la dirección de enlace es implícitamente "out" y el tipo es Cola de Azure Storage. Agregue el siguiente decorador al código de función en *HttpExample\function_app.py*:

Python

```
@app.queue_output(arg_name="msg", queue_name="outqueue",
connection="AzureWebJobsStorage")
```

En este código, `arg_name` identifica el parámetro de enlace al que se hace referencia en el código, `queue_name` es el nombre de la cola en la que escribe el enlace y `connection` es el nombre de una configuración de la aplicación que contiene la cadena de conexión para la cuenta de Storage. En los inicios rápidos se usa la misma cuenta de almacenamiento que la aplicación de funciones, que se encuentra en la configuración `AzureWebJobsStorage`. Cuando no existe `queue_name`, el enlace lo crea durante el primer uso.

Adición de código que utilice el enlace de salida

Una vez definido el enlace, podrá usar el valor de `name` de este para acceder a él como atributo en la firma de función. Con un enlace de salida, no tiene que usar el código del SDK de Azure Storage para autenticarse, obtener una referencia de cola o escribir datos. El sistema en tiempo de ejecución de Functions y el enlace de salida de cola realizan esas tareas automáticamente.

Actualice *HttpExample\function_app.py* para que se ajuste al siguiente código y agregue el parámetro `msg` a la definición de la función y `msg.set(name)` en la instrucción `if name:`

Python

```
import azure.functions as func
import logging

app = func.FunctionApp(http_auth_level=func.AuthLevel.ANONYMOUS)

@app.route(route="HttpExample")
@app.queue_output(arg_name="msg", queue_name="outqueue",
connection="AzureWebJobsStorage")
def HttpExample(req: func.HttpRequest, msg: func.Out[func.QueueMessage]) ->
func.HttpResponse:
    logging.info('Python HTTP trigger function processed a request.')

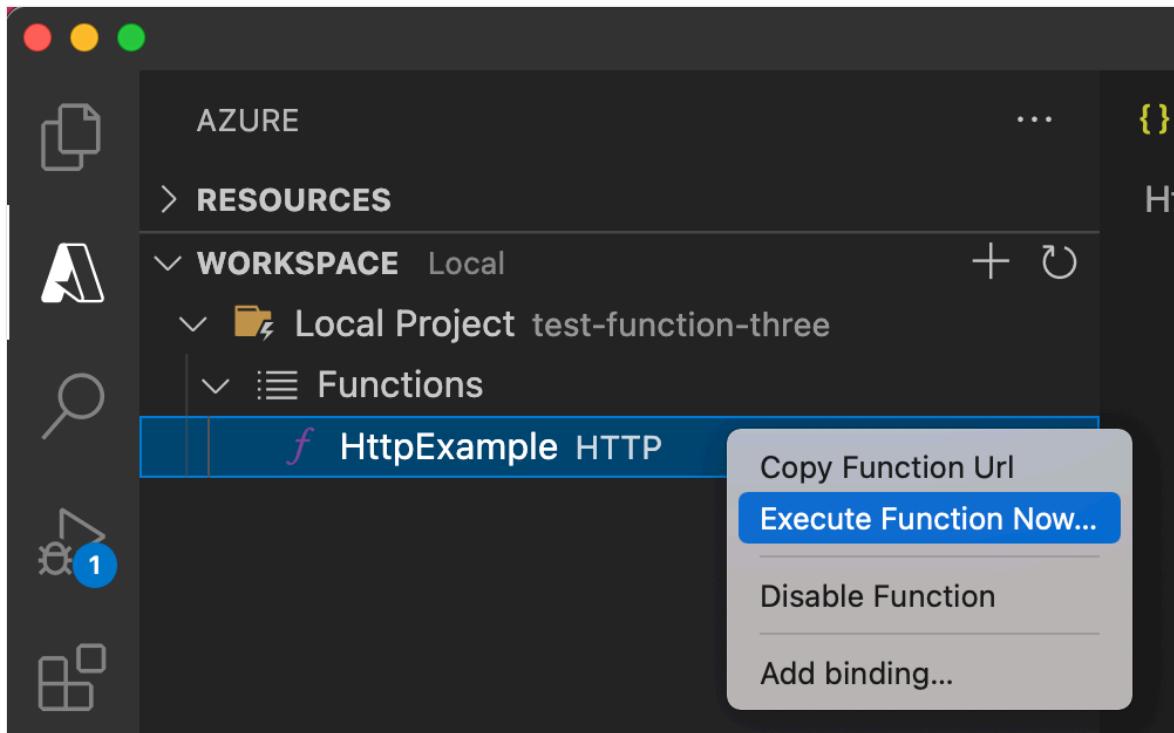
    name = req.params.get('name')
    if not name:
        try:
            req_body = req.get_json()
        except ValueError:
            pass
        else:
            name = req_body.get('name')

    if name:
        msg.set(name)
        return func.HttpResponse(f"Hello, {name}. This HTTP triggered
function executed successfully.")
    else:
        return func.HttpResponse(
            "This HTTP triggered function executed successfully. Pass a
name in the query string or in the request body for a personalized
response.",
            status_code=200
        )
```

El parámetro `msg` es una instancia de `azure.functions.Out class`. El método `set` escribe un mensaje de cadena en la cola. En este caso, es el `name` que pasa a la función en la cadena de consulta de URL.

Ejecución local de la función

1. Como en el artículo anterior, presione `F5` para iniciar el proyecto de aplicación de función y Core Tools.
2. Mientras se ejecuta Core Tools, vaya al área **Azure: Funciones**. En **Functions**, expanda **Proyecto local>Functions**. Haga clic con el botón derecho (Ctrl + clic en Mac) en la función `HttpExample` y elija **Ejecutar la función ahora...**



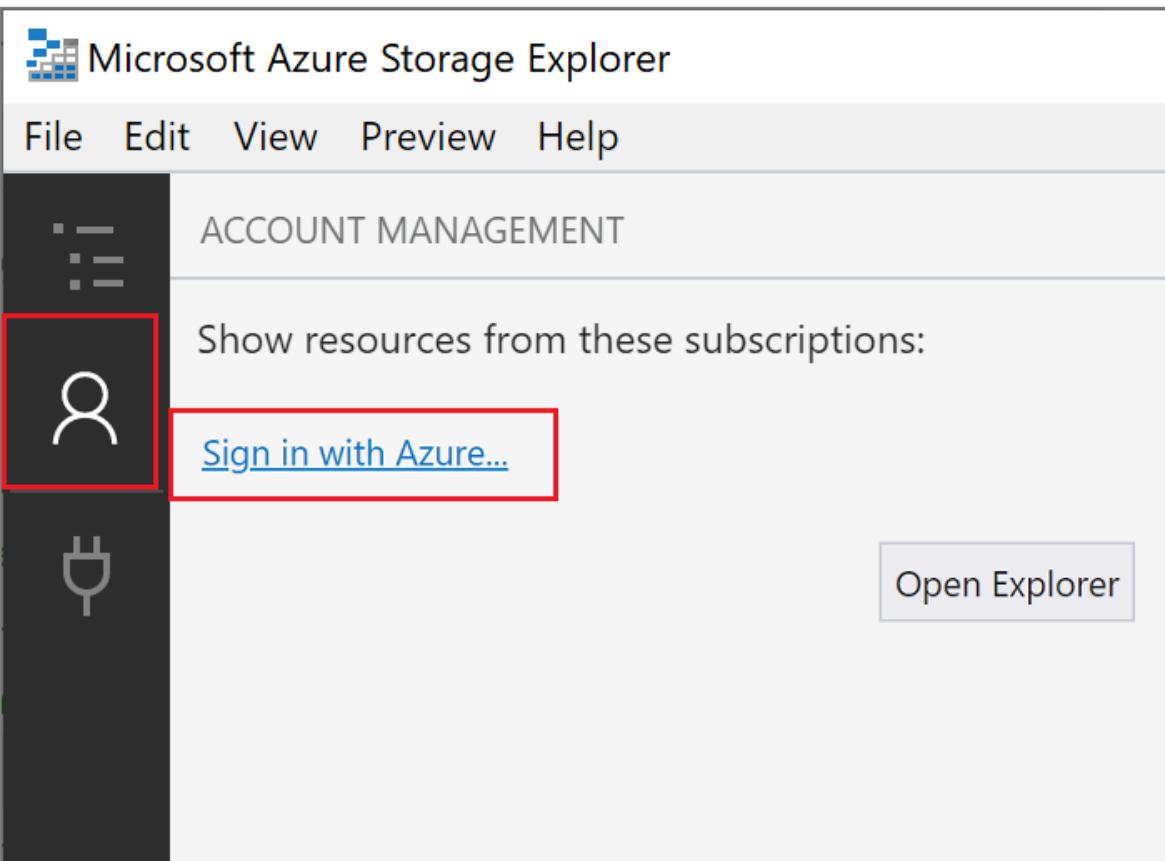
3. En **Especificar el cuerpo de la solicitud**, verá el valor `{ "name": "Azure" }` en el cuerpo del mensaje de solicitud. Presione `Entrar` para enviar este mensaje de solicitud a la función.
4. Una vez que se haya devuelto una respuesta, presione `ctrl + c` para detener Core Tools.

Dado que está usando la cadena de conexión de almacenamiento, la función se conectará a la cuenta de almacenamiento de Azure durante la ejecución local. Cuando el enlace de salida se usa por primera vez, el entorno de ejecución de Functions crea una nueva cola denominada **outqueue** en la cuenta de almacenamiento. Usará el Explorador de Storage para comprobar que se han creado la cola y un mensaje.

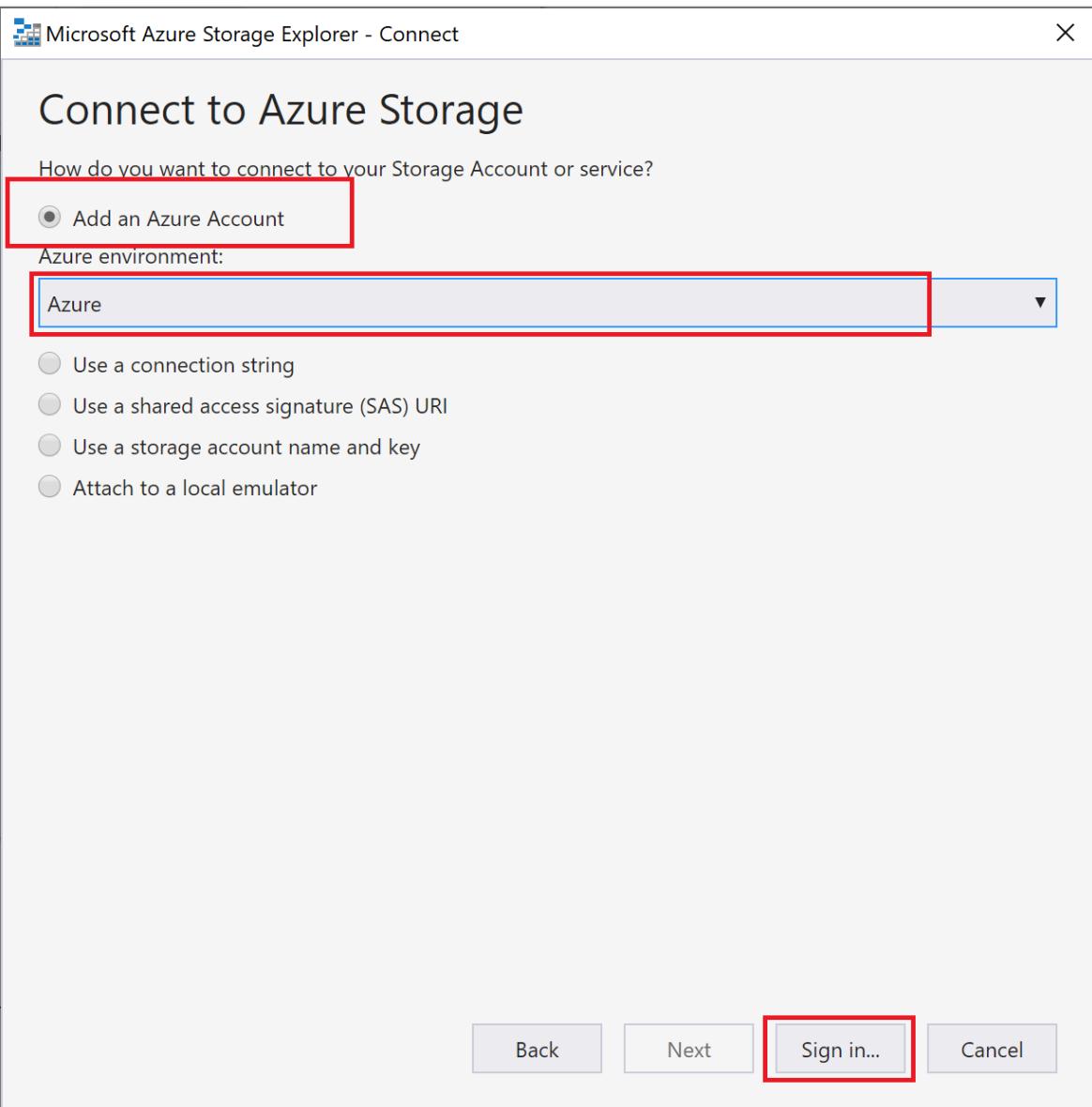
Conexión del Explorador de Storage con la cuenta

Omita esta sección si ya instaló el Explorador de Azure Storage y lo conectó a su cuenta de Azure.

1. Ejecute la herramienta [Explorador de Azure Storage](#), y seleccione el ícono de conexión de la izquierda y **Agregar una cuenta**.



2. En el cuadro de diálogo **Conectar**, seleccione **Agregar una cuenta de Azure** y, a continuación, elija el **entorno de Azure** que quiera usar y seleccione **Iniciar sesión...**

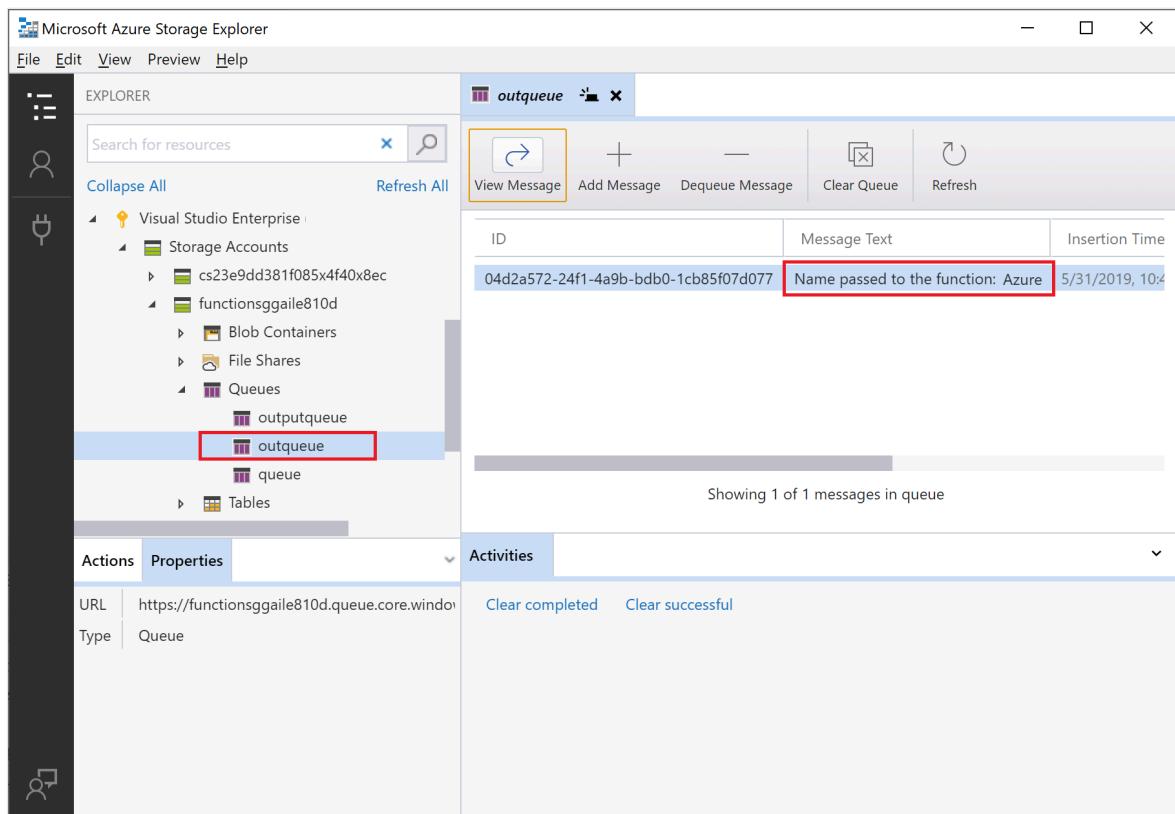


Después de iniciar sesión correctamente en su cuenta, verá todas las suscripciones de Azure asociadas con ella. Elija la suscripción y selección **Abrir Explorer**.

Examen de la cola de salida

1. En Visual Studio Code, presione la tecla `F1` para abrir la paleta de comandos y, después, busque y ejecute el comando `Azure Storage: Open in Storage Explorer` y elija el nombre de la cuenta de almacenamiento que quiere usar. En el menú de la cuenta de almacenamiento, seleccione el Explorador de Azure Storage.
2. Expanda el nodo **Colas** y, después, seleccione la cola con el nombre **outqueue**.

La cola contiene el mensaje que creó el enlace de salida de la cola al ejecutar la función desencadenada por HTTP. Si se invoca la función con el valor predeterminado `name` de Azure, el mensaje de cola es *Name passed to the function: Azure* (Nombre pasado a la función: Azure).



3. Vuelva a ejecutar la función y envíe otra solicitud. A continuación, verá que hay un nuevo mensaje en la cola.

Ahora, es el momento de volver a publicar la aplicación de función actualizada en Azure.

Reimplementación y comprobación de la aplicación actualizada

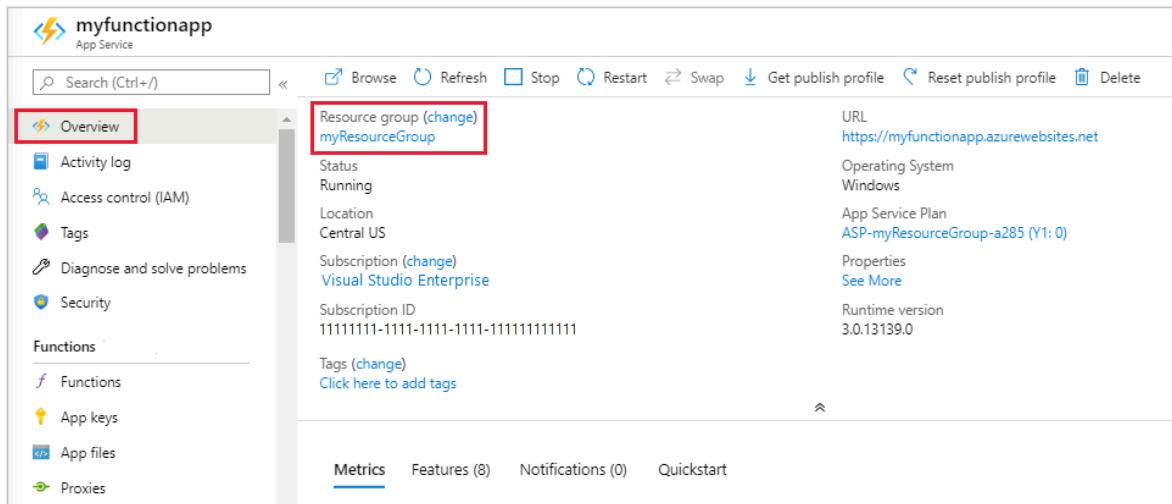
1. En Visual Studio Code, presione **F1** para abrir la paleta de comandos. En la paleta de comandos, busque y seleccione **Azure Functions: Deploy to function app....**
2. Elija la aplicación de funciones que creó en el primer artículo. Dado que va a volver a implementar el proyecto en la misma aplicación, seleccione **Implementar** para descartar la advertencia sobre la sobrescritura de archivos.
3. Una vez que haya finalizado la implementación, podrá volver a usar la característica **Ejecutar la función ahora...** para desencadenar la función en Azure.
4. Vuelva a [consultar el mensaje de la cola de almacenamiento](#) para comprobar que el enlace de salida genera un nuevo mensaje en la cola.

Limpieza de recursos

En Azure, los *recursos* son aplicaciones de funciones, funciones o cuentas de almacenamiento, entre otros. Se agrupan en *grupos de recursos* y se puede eliminar todo el contenido de un grupo si este se elimina.

Para completar estas guías de inicio rápido, ha creado varios recursos. Se le pueden facturar por estos recursos, dependiendo del [estado de la cuenta de los servicios](#) y [precios de los servicios](#). Si ya no necesita los recursos, aquí se indica cómo eliminarlos:

1. En Visual Studio Code, presione **F1** para abrir la paleta de comandos. En la paleta de comandos, busque y seleccione **Azure: Open in portal**.
2. Elija una aplicación de funciones y presione la tecla **ENTRAR**. La página de la aplicación de funciones se abre en Azure Portal.
3. En la pestaña **Información general**, seleccione el vínculo con nombre junto a **Grupo de recursos**.



The screenshot shows the Azure Portal interface for a function app named "myfunctionapp". The left sidebar has links like "Overview", "Activity log", "Access control (IAM)", "Tags", "Diagnose and solve problems", "Security", "Functions", "Functions", "App keys", "App files", and "Proxies". The main content area is titled "Overview" and shows details for the resource group "myResourceGroup". The "myResourceGroup" link is highlighted with a red box. The details include:

- Status: Running
- Location: Central US
- Subscription (change): Visual Studio Enterprise
- Subscription ID: 11111111-1111-1111-1111-111111111111
- Tags (change): Click here to add tags
- Metrics, Features (8), Notifications (0), Quickstart buttons at the bottom

On the right side, there are additional details:

- URL: https://myfunctionapp.azurewebsites.net
- Operating System: Windows
- App Service Plan: ASP-myResourceGroup-a285 (Y1: 0)
- Properties: See More
- Runtime version: 3.0.13139.0

4. En la página **Grupo de recursos**, revise la lista de recursos incluidos y compruebe que estos sean los que desea eliminar.
5. Seleccione **Eliminar grupo de recursos** y siga las instrucciones.

El proceso de eliminación tardará un par de minutos. Cuando termine, aparece una notificación durante unos segundos. También puede seleccionar el ícono de campana en la parte superior de la página para ver la notificación.

Pasos siguientes

Ha actualizado la función desencadenada por HTTP para escribir datos en una cola de almacenamiento. Ahora, puede obtener más información sobre el desarrollo de Functions mediante Visual Studio Code:

- Desarrollo de Azure Functions mediante Visual Studio Code
- Enlaces y desencadenadores de Azure Functions.
- Ejemplos de proyectos de Function completos en Python.
- Guía de Azure Functions para desarrolladores de Python

Conexión de Azure Functions a Azure Storage mediante herramientas de línea de comandos

Artículo • 29/12/2024

En este artículo, integrará una cola de Azure Storage con la función y la cuenta de almacenamiento que creó en el artículo de inicio rápido anterior. Para lograr esta integración se usa un *enlace de salida* que escribe los datos de una solicitud HTTP en un mensaje de la cola. Completar este artículo no supone ningún costo extraordinario, más allá del pequeño importe del inicio rápido anterior. Para más información acerca de los enlaces, consulte [Conceptos básicos sobre los enlaces y desencadenadores de Azure Functions](#).

Configuración del entorno local

Antes de empezar, debe completar el artículo [Inicio rápido: Creación de un proyecto de Azure Functions desde la línea de comandos](#). Si ya ha limpiado los recursos al final de ese artículo, vuelva a recorrer los pasos para crear de nuevo la aplicación de función y los recursos relacionados en Azure.

Recuperación de la cadena de conexión de Azure Storage

Importante

En este artículo se muestra cómo conectarse a una cuenta de Azure Storage mediante la cadena de conexión, que contiene una clave secreta compartida. El uso de una cadena de conexión facilita la comprobación de las actualizaciones de datos en la cuenta de almacenamiento. Para obtener la mejor seguridad posible, debe usar identidades administradas al conectarse a su cuenta de almacenamiento. Para más información, consulte [Conexiones](#) en la Guía para desarrolladores.

Anteriormente, creó una cuenta de Azure Storage para el uso de la aplicación de funciones. La cadena de conexión de esta cuenta se almacena de forma segura en la configuración de la aplicación en Azure. Mediante la descarga de la configuración en el archivo `local.settings.json`, puede usar la conexión para escribir en una cola de Storage de la misma cuenta cuando ejecute la función de forma local.

1. En la raíz del proyecto, ejecute el siguiente comando, pero reemplace <APP_NAME> por el nombre de la aplicación de funciones del paso anterior. Este comando sobrescribe los valores existentes en el archivo.

```
func azure functionapp fetch-app-settings <APP_NAME>
```

2. Abra el archivo *local.settings.json* y busque el valor denominado `AzureWebJobsStorage`, que es la cadena de conexión de la cuenta de almacenamiento. Usará el nombre `AzureWebJobsStorage` y la cadena de conexión en otras secciones de este artículo.

 **Importante**

Como el archivo *local.settings.json* contiene secretos descargados de Azure, excluya siempre este archivo del control de código fuente. El archivo *.gitignore* que se creó con un proyecto de Functions local excluye el archivo de forma predeterminada.

Incorporación de una definición de enlace de salida a la función

Aunque ninguna de las funciones puede tener más de un desencadenador, puede tener varios enlaces de entrada y salida que le permiten conectarse a otros servicios y recursos de Azure sin escribir código de integración personalizado.

Si se usa el [modelo de programación de Python v2](#), los atributos de enlace se definen directamente en el archivo *function_app.py* como decoradores. En el inicio rápido anterior, el archivo *function_app.py* ya contiene un enlace basado en decorador:

Python

```
import azure.functions as func
import logging

app = func.FunctionApp()

@app.function_name(name="HttpTrigger1")
@app.route(route="hello", auth_level=func.AuthLevel.ANONYMOUS)
```

El decorador `route` agrega el enlace `HttpTrigger` y `HttpOutput` a la función, lo que permite que la función se desencadene cuando las solicitudes http alcancen la ruta

especificada.

Para escribir en una cola de Azure Storage desde esta función, agregue el decorador `queue_output` al código de función:

Python

```
@app.queue_output(arg_name="msg", queue_name="outqueue",
connection="AzureWebJobsStorage")
```

En el decorador, `arg_name` identifica el parámetro de enlace al que se hace referencia en el código, `queue_name` es el nombre de la cola en la que escribe el enlace y `connection` es el nombre de una configuración de la aplicación que contiene la cadena de conexión para la cuenta de Storage. En los inicios rápidos se usa la misma cuenta de almacenamiento que la aplicación de funciones, que se encuentra en la configuración `AzureWebJobsStorage` (desde el archivo `local.settings.json`). Cuando no existe `queue_name`, el enlace lo crea durante el primer uso.

Para más información sobre los enlaces, consulte [Conceptos básicos sobre los enlaces y desencadenadores de Azure Functions](#) y [configuración de la cola de salida](#).

Adición de código para usar el enlace de salida

Con el enlace de cola definido, ahora puede actualizar la función para que reciba el parámetro de salida `msg` y escribir mensajes en la cola.

Actualice `HttpExample\function_app.py` para que se ajuste al siguiente código y agregue el parámetro `msg` a la definición de la función y `msg.set(name)` en la instrucción `if name::`

Python

```
import azure.functions as func
import logging

app = func.FunctionApp(http_auth_level=func.AuthLevel.ANONYMOUS)

@app.route(route="HttpExample")
@app.queue_output(arg_name="msg", queue_name="outqueue",
connection="AzureWebJobsStorage")
def HttpExample(req: func.HttpRequest, msg: func.Out[func.QueueMessage]) ->
func.HttpResponse:
    logging.info('Python HTTP trigger function processed a request.')

    name = req.params.get('name')
```

```

if not name:
    try:
        req_body = req.get_json()
    except ValueError:
        pass
    else:
        name = req_body.get('name')

if name:
    msg.set(name)
    return func.HttpResponse(f"Hello, {name}. This HTTP triggered
function executed successfully.")
else:
    return func.HttpResponse(
        "This HTTP triggered function executed successfully. Pass a
name in the query string or in the request body for a personalized
response.",
        status_code=200
)

```

El parámetro `msg` es una instancia de `azure.functions.Out class`. El método `set` escribe un mensaje de cadena en la cola. En este caso, es el `name` que pasa a la función en la cadena de consulta de URL.

Observe que *no* necesita escribir código para la autenticación, para obtener una referencia de la cola ni para escribir datos. Todas estas tareas de integración se administran de manera adecuada en el entorno de ejecución de Azure Functions y en el enlace de salida de la cola.

Ejecución local de la función

1. Para ejecutar la función, inicie el host en tiempo de ejecución local de Azure Functions desde la carpeta `LocalFunctionProj`.

Consola

`func start`

En la parte final de la salida, deberán aparecer las líneas siguientes:

```
Azure Functions Core Tools
Core Tools Version:      4.0.5049 Commit hash: N/A  (64-bit)
Function Runtime Version: 4.15.2.20177

[2023-03-17T03:27:12.372Z] Worker process started and initialized.

Functions:

    HttpExample: [GET,POST] http://localhost:7071/api/HttpExample
```

ⓘ Nota

Si `HttpExample` no aparece como se ha mostrado arriba, es probable que haya iniciado el host desde fuera de la carpeta raíz del proyecto. En ese caso, use `Ctrl+C` para detener el host, vaya a la carpeta raíz del proyecto y vuelva a ejecutar el comando anterior.

2. Copie la URL de su función HTTP de esta salida a un navegador y agregue la cadena de consulta `?name=<YOUR_NAME>`, lo que hace que la URL completa sea como `http://localhost:7071/api/HttpExample?name=Functions`. El explorador debe mostrar un mensaje de respuesta que devuelve el valor de la cadena de consulta. El terminal en el que inició el proyecto también muestra la salida del registro cuando realiza solicitudes.
3. Cuando termine, presione `Ctrl + C` y escriba `y` para detener el host de Azure Functions.

💡 Sugerencia

Durante el inicio, el host descarga e instala la [extensión de enlace de Storage](#) y otras extensiones de enlace de Microsoft. Esta instalación se produce porque las extensiones de enlace se habilitan de forma predeterminada en el archivo `host.json` con las siguientes propiedades:

JSON

```
{
  "version": "2.0",
  "extensionBundle": {
    "id": "Microsoft.Azure.Functions.ExtensionBundle",
    "version": "[1.*, 2.0.0)"
  }
}
```

Si encuentra algún error relacionado con las extensiones de enlace, compruebe que las propiedades anteriores están presentes en `host.json`.

Visualización del mensaje en la cola de Azure Storage

La cola se puede ver en [Azure Portal](#) o en el [Explorador de Microsoft Azure Storage](#). También puede ver la cola en la CLI de Azure, como se describe en los pasos siguientes:

1. Abra el archivo `local.setting.json` del proyecto de Functions y copie el valor de la cadena de conexión. En una ventana de terminal o de comandos, ejecute el siguiente comando para crear una variable de entorno denominada `AZURE_STORAGE_CONNECTION_STRING` y pegue la cadena de conexión concreta en lugar de `<MY_CONNECTION_STRING>`. (Esta variable de entorno significa que no es necesario proporcionar la cadena de conexión a cada comando posterior mediante el argumento `--connection-string`).

```
bash
Bash
export AZURE_STORAGE_CONNECTION_STRING=<MY_CONNECTION_STRING>
```

2. (Opcional) Puede usar el comando [az storage queue list](#) para ver las colas de Storage de la cuenta. La salida de este comando debe incluir una cola denominada `outqueue`, la cual se creó cuando la función escribió su primer mensaje en esa cola.

```
Azure CLI
az storage queue list --output tsv
```

3. Use el comando [az storage message get](#) para leer el mensaje de esta cola, que debería ser el valor que indicó al probar la función anteriormente. El comando lee y quita el primer mensaje de la cola.

```
bash
Azure CLI
```

```
echo `echo $(az storage message get --queue-name outqueue -o tsv --query '[].{Message:content}') | base64 --decode`
```

Dado que el cuerpo del mensaje se almacena [codificado en Base64](#), el mensaje se debe decodificar antes de visualizarse. Después de ejecutar `az storage message get`, el mensaje se quita de la cola. Si solo había un mensaje en `outqueue`, no se recuperará al ejecutar este comando una segunda vez y, en su lugar, recibirá un error.

Nueva implementación del proyecto en Azure

Tras comprobar localmente que la función escribió un mensaje en la cola de Azure Storage, puede volver a implementar el proyecto para actualizar el punto de conexión que se ejecuta en Azure.

En la carpeta `LocalFunctionsProj`, use el comando [func azure functionapp publish](#) para volver a implementar el proyecto y reemplace `<APP_NAME>` el nombre de la aplicación.

```
func azure functionapp publish <APP_NAME>
```

Comprobación en Azure

1. Como en el inicio rápido anterior, use un explorador o CURL para probar la función que ha vuelto a implementar.

Browser

Copie la **dirección URL de invocación** completa que se muestra en la salida del comando de publicación en una barra de direcciones del explorador, y anexe el parámetro de consulta `&name=Functions`. El explorador debe mostrar la misma salida a cuando ejecutó la función localmente.

2. Vuelva a examinar la cola de Storage, como se describe en la sección anterior, para comprobar que contiene el nuevo mensaje escrito en la cola.

Limpieza de recursos

Cuando termine, use el siguiente comando para eliminar el grupo de recursos y todos los recursos que contiene para evitar incurrir en costos adicionales.

Azure CLI

```
az group delete --name AzureFunctionsQuickstart-rg
```

Pasos siguientes

Ha actualizado la función desencadenada por HTTP para escribir datos en una cola de almacenamiento. Ahora puede aprender más sobre el desarrollo de Functions desde la línea de comandos mediante Core Tools y la CLI de Azure:

- [Uso de Azure Functions Core Tools](#)
- [Enlaces y desencadenadores de Azure Functions](#)
- [Ejemplos de proyectos de Function completos en Python.](#)
- [Guía de Azure Functions para desarrolladores de Python](#)

Comentarios

¿Le ha resultado útil esta página?

 Sí

 No

[Proporcionar comentarios sobre el producto](#) | [Obtener ayuda en Microsoft Q&A](#)

Soluciones de datos para aplicaciones de Python en Azure

05/06/2025

Azure ofrece una amplia gama de soluciones de almacenamiento y base de datos totalmente administradas, incluidas las bases de datos relacionales, NoSQL y en memoria, con compatibilidad con tecnologías propietarias y de código abierto. También puede elegir entre los servicios de objetos, bloques y almacenamiento de archivos. Los artículos siguientes pueden ayudarle a empezar a usar estas opciones con Python en Azure.

Bases de datos

- **PostgreSQL:** compile aplicaciones empresariales escalables, seguras y totalmente administradas mediante PostgreSQL de código abierto. Puede escalar PostgreSQL de un solo nodo para un alto rendimiento o migrar cargas de trabajo de PostgreSQL y Oracle existentes a la nube.
 - [Inicio rápido: Uso de Python para conectarse y consultar datos en Azure Database for PostgreSQL: servidor flexible](#)
 - [Inicio rápido: Uso de Python para conectarse y consultar datos en Azure Database for PostgreSQL: servidor único](#)
 - [Desplegar una aplicación web de Python \(Django o Flask\) con PostgreSQL en Azure App Service](#)
- **MySQL:** cree aplicaciones escalables mediante una base de datos MySQL totalmente administrada e inteligente en la nube.
 - [Inicio rápido: Use Python para conectarse a datos y consultarlos en Azure Database for MySQL con la opción Servidor flexible](#)
 - [Inicio rápido: Uso de Python para conectarse y consultar datos en Azure Database for MySQL](#)
- **Azure SQL:** cree aplicaciones escalables con una plataforma de base de datos SQL totalmente administrada e inteligente en la nube.
 - [Inicio rápido: Uso de Python para consultar una base de datos de Azure SQL o Azure SQL Managed Instance](#)

NoSQL, blobs, tablas, archivos, grafos y memorias caché

- **Cosmos DB:** cree aplicaciones de baja latencia, alta disponibilidad a escala global o migre Cassandra, MongoDB y otras cargas de trabajo noSQL a la nube.
 - [Inicio rápido: Biblioteca cliente de Azure Cosmos DB for NoSQL para Python](#)
 - [Inicio rápido: Azure Cosmos DB para MongoDB para Python con el controlador de MongoDB](#)
 - [Inicio rápido: Compilación de una aplicación de Cassandra con el SDK de Python y Azure Cosmos DB](#)
 - [Inicio rápido: Creación de una aplicación de API para Table con el SDK de Python y Azure Cosmos DB](#)
 - [Inicio rápido: Biblioteca de Azure Cosmos DB for Apache Gremlin para Python](#)
- **Blob Storage:** almacenamiento seguro y escalable de objetos para aplicaciones nativas de nube, lagos de datos, archivos, informática de alto rendimiento (HPC) y aprendizaje automático.
 - [Inicio rápido: Biblioteca cliente de Azure Blob Storage para Python](#)
 - [Ejemplos de Azure Storage con bibliotecas cliente de Python v12](#)
- **Azure Data Lake Storage Gen2:** lago de datos escalable y seguro optimizado para análisis de alto rendimiento.
 - [Uso de Python para administrar directorios y archivos en Azure Data Lake Storage Gen2](#)
 - [Uso de Python para administrar ACL en Azure Data Lake Storage Gen2](#)
- **Almacenamiento de archivos:** compartición de archivos en la nube de nivel empresarial, sencilla, segura y sin servidor.
 - [Desarrollo para Azure Files con Python](#)
- **Redis Cache:** acelere el rendimiento de las aplicaciones con un almacén de datos escalable en memoria compatible con código abierto.
 - [Inicio rápido: Uso de Azure Cache for Redis con Python](#)

Macrodatos y análisis

- **Análisis de Azure Data Lake:** servicio de análisis de pago por trabajo totalmente administrado que ofrece un procesamiento de datos paralelo eficaz con seguridad, auditoría y soporte técnico integrados de nivel empresarial.
 - [Administración de Azure Data Lake Analytics mediante Python](#)
 - [Desarrollo de U-SQL con Python para Azure Data Lake Analytics en Visual Studio Code](#)
- **Azure Data Factory:** un servicio de integración de datos totalmente administrado que le permite crear, orquestar y automatizar visualmente el movimiento y la transformación de datos en varios orígenes de datos.

- Inicio rápido: Creación de una factoría de datos y una canalización con Python
 - Transformación de datos mediante la ejecución de una actividad de Python en Azure Databricks
- **Azure Event Hubs:** un servicio de ingesta de telemetría totalmente administrado y a gran escala diseñado para recopilar, transformar y almacenar millones de eventos por segundo desde aplicaciones y dispositivos conectados.
 - Envío o recepción de eventos desde Event Hubs mediante Python
 - Capturar datos de Event Hubs en Azure Storage y leerlos mediante Python (azure-eventhub)
- **HDInsight:** un servicio en la nube totalmente administrado que ejecuta marcos populares de código abierto, como Hadoop y Spark, respaldados por un Acuerdo de Nivel de Servicio de 99.9% para el análisis de macrodatos de nivel empresarial.
 - Usar las herramientas Spark y Hive para Visual Studio Code
- **Azure Databricks:** una plataforma de análisis basada en Apache® Spark™ totalmente administrada, rápida y colaborativa optimizada para cargas de trabajo de macrodatos e inteligencia artificial en Azure.
 - Conexión a Azure Databricks desde Excel, Python o R
 - Introducción a Azure Databricks
 - Tutorial: Azure Data Lake Storage Gen2, Azure Databricks & Spark
- **Azure Synapse Analytics:** un servicio de análisis totalmente administrado que unifica la integración de datos, el almacenamiento de datos empresariales y el análisis de macrodatos en una sola plataforma.
 - Inicio rápido: Uso de Python para consultar una base de datos en Azure SQL Database o Azure SQL Managed Instance (incluye Azure Synapse Analytics)

Aprendizaje automático para aplicaciones de Python en Azure

12/06/2025

Los artículos siguientes le ayudarán a empezar a trabajar con Azure Machine Learning. Las API REST de Azure Machine Learning v2, la extensión de la CLI de Azure y el SDK de Python están diseñadas para simplificar todo el ciclo de vida de aprendizaje automático y acelerar los flujos de trabajo de producción. Los vínculos de este artículo tienen como destino la versión 2, que se recomienda si va a iniciar un nuevo proyecto de aprendizaje automático.

Cómo empezar

En Azure Machine Learning, el área de trabajo es el recurso principal que organiza y administra todo lo que se crea, como conjuntos de datos, modelos y experimentos.

- [Inicio rápido: Introducción a Azure Machine Learning](#)
- [Administración de áreas de trabajo de Azure Machine Learning en el portal o con el SDK de Python \(v2\)](#)
- [Ejecuta cuadernos de Jupyter en tu área de trabajo](#)
- [Tutorial: Desarrollo de modelos en una estación de trabajo en la nube](#)

Implementación de modelos

Implemente modelos para predicciones de aprendizaje automático en tiempo real y de baja latencia.

- [Tutorial: Diseñador: Implementación de un modelo de Machine Learning](#)
- [Implementación y puntuación de un modelo de aprendizaje automático mediante un punto de conexión en línea](#)

Aprendizaje Automático Automatizado

MI automatizado (AutoML) hace referencia al proceso de optimización del desarrollo de modelos de aprendizaje automático mediante la automatización de sus tareas repetitivas y lentas.

- [Entrenamiento de un modelo de regresión con AutoML y Python \(SDK v1\)](#)
- [Configuración del entrenamiento de AutoML para datos tabulares con la CLI de Azure Machine Learning y el SDK de Python \(v2\)](#)

Acceso a datos

Con Azure Machine Learning, puede importar datos desde el equipo local o conectarse a los servicios de almacenamiento en la nube existentes.

- [Creación y administración de recursos de datos](#)
- [Tutorial: Carga, acceso y exploración de los datos en Azure Machine Learning](#)
- [Acceso a datos en un trabajo](#)

Flujos de aprendizaje automático

Use canalizaciones de aprendizaje automático para crear flujos de trabajo que conecten distintas fases del proceso de aprendizaje automático.

- [Uso de Azure Pipelines con Azure Machine Learning](#)
- [Creación y ejecución de canalizaciones de aprendizaje automático mediante componentes con el SDK de Azure Machine Learning v2](#)
- [Tutorial: Creación de canalizaciones de ML de producción con el SDK v2 de Python en un cuaderno de Jupyter Notebook](#)

Introducción a Python Container Apps en Azure

Artículo • 03/05/2025

En este artículo se explica cómo tomar un proyecto de Python (como una aplicación web) e implementarlo como un contenedor de Docker en Azure. Abarca el flujo de trabajo de contenedorización general, las opciones de implementación de Azure para los contenedores y las configuraciones de contenedor específicas de Python en Azure. La compilación e implementación de contenedores de Docker en Azure sigue un proceso estándar entre lenguajes, con configuraciones específicas de Python en dockerfile, requirements.txt y la configuración de marcos web como [Django](#), [Flask](#) y [FastAPI](#).

Escenarios de flujo de trabajo de contenedor

En el caso del desarrollo de contenedores de Python, se describen algunos flujos de trabajo típicos para pasar de código a contenedor en la tabla siguiente.

[+] Expandir tabla

Escenario	Descripción	Flujo de trabajo
Dev	Compile imágenes de Docker de Python localmente en el entorno de desarrollo.	<p>Código: clone el código de la aplicación localmente mediante Git (con Docker instalado).</p> <p>Compilación: use la CLI de Docker, VS Code (con extensiones), PyCharm (con el complemento docker). Se describe en la sección Trabajar con imágenes y contenedores de Docker de Python.</p> <p>Prueba: ejecute y pruebe el contenedor localmente.</p> <p>Inserción: inserte la imagen en un registro de contenedor como Azure Container Registry, Docker Hub o registro privado.</p> <p>Implementación: implemente el contenedor desde el registro en un servicio de Azure.</p>
Híbrido	Compile imágenes de Docker en Azure, pero inicie el proceso desde el entorno local.	<p>Código: clone el código localmente (no es necesario para instalar Docker).</p> <p>Compilación: para desencadenar compilaciones en Azure, use VS Code (con extensiones remotas) o la CLI de Azure.</p>

Escenario	Descripción	Flujo de trabajo
		<p>Publicar: Publique la imagen compilada en Azure Container Registry.</p> <p>Implementación: implemente el contenedor desde el registro en un servicio de Azure.</p>
Azure	Use Azure Cloud Shell para compilar e implementar contenedores completamente en la nube.	<p>Código: clone el repositorio de GitHub en Azure Cloud Shell.</p> <p>Compilación: use la CLI de Azure o la CLI de Docker en Cloud Shell.</p> <p>Inserción: inserte la imagen en un registro como Azure Container Registry, Docker Hub o registro privado.</p> <p>Implementación: implemente el contenedor desde el registro en un servicio de Azure.</p>

El objetivo final de estos flujos de trabajo es tener un contenedor que se ejecute en uno de los recursos de Azure que admiten contenedores de Docker, como se muestra en la sección siguiente.

Un entorno de desarrollo puede ser:

- Estación de trabajo local con Visual Studio Code o PyCharm
- [Codespaces](#) (un entorno de desarrollo hospedado en la nube)
- [Contenedores de desarrollo de Visual Studio](#) (un contenedor como entorno de desarrollo)

Opciones de contenedor de implementación en Azure

Las aplicaciones de contenedor de Python se admiten en los siguientes servicios.

[] Expandir tabla

Servicio	Descripción
Aplicación web para contenedores	Azure App Service es una plataforma de hospedaje totalmente administrada para aplicaciones web en contenedores, incluidos sitios web y API web. Admite implementaciones escalables e se integra perfectamente con flujos de trabajo de CI/CD mediante Docker Hub, Azure Container Registry y GitHub. Este servicio es ideal para los desarrolladores que desean una ruta de acceso sencilla y eficaz para implementar aplicaciones en contenedores, a la vez que se benefician de las

Servicio	Descripción
	<p>funcionalidades completas de la plataforma de Azure App Service. Al empaquetar la aplicación y todas sus dependencias en un único contenedor implementable, se obtiene la portabilidad y la facilidad de administración, sin necesidad de administrar la infraestructura.</p> <p>Ejemplo: Implementación de una aplicación web de Flask o FastAPI en Azure App Service.</p>
Azure Container Apps (ACA)	<p>Azure Container Apps (ACA) es un servicio de contenedores totalmente administrado y sin servidor impulsado por Kubernetes y tecnologías de código abierto como Dapr, KEDA y envoy. Su diseño incorpora procedimientos recomendados del sector y está optimizado para ejecutar contenedores de uso general. ACA abstrae la complejidad de administrar una infraestructura de Kubernetes: no se requiere ni admite el acceso directo a la API de Kubernetes. En su lugar, ofrece construcciones de aplicación de nivel superior, como revisiones, escalado, certificados y entornos para simplificar los flujos de trabajo de desarrollo e implementación. Este servicio es ideal para los equipos de desarrollo que buscan compilar e implementar microservicios en contenedores con una sobrecarga operativa mínima, lo que les permite centrarse en la lógica de la aplicación en lugar de en la administración de la infraestructura.</p> <p>Ejemplo: Implementación de una aplicación web de Flask o FastAPI en Azure Container Apps.</p>
Azure contenedores de instancias (ACI)	<p>Azure Container Instances (ACI) es una oferta sin servidor que proporciona un único pod de Hyper-V contenedores aislados a petición. La facturación se basa en el consumo real de recursos en lugar de la infraestructura asignada previamente, lo que hace que sea adecuado para cargas de trabajo de corta duración o ampliables. A diferencia de otros servicios de contenedor, ACI no incluye compatibilidad integrada con conceptos como el escalado, el equilibrio de carga o los certificados TLS. En su lugar, normalmente funciona como un bloque de creación de contenedores fundamentales, a menudo integrado con servicios de Azure como Azure Kubernetes Service (AKS) para la orquestación. ACI se destaca como opción ligera cuando las abstracciones y características de nivel superior de Azure Container Apps no son necesarias</p> <p>Ejemplo: Creación de una imagen de contenedor para la implementación en Azure Container Instances. (El tutorial no es específico de Python, pero los conceptos que se muestran se aplican a todos los lenguajes).</p>
Azure Kubernetes Service (AKS)	<p>Azure Kubernetes Service (AKS) es una opción de Kubernetes totalmente administrada en Azure que proporciona un control completo sobre el entorno de Kubernetes. Admite el acceso directo a la API de Kubernetes y puede ejecutar cualquier carga de trabajo estándar de Kubernetes. El clúster completo reside en su suscripción, con las configuraciones y las operaciones del clúster bajo su control y responsabilidad. ACI es ideal para los equipos que buscan una solución de contenedor totalmente administrada, mientras que AKS le proporciona control total sobre el clúster de Kubernetes, lo que requiere administrar configuraciones, redes, escalado y operaciones. Azure controla el plano de control y el aprovisionamiento de</p>

Servicio	Descripción
	<p>infraestructura, pero la operación diaria y la seguridad del clúster están dentro del control del equipo. Este servicio es ideal para los equipos que desean la flexibilidad y la eficacia de Kubernetes con la ventaja adicional de la infraestructura administrada de Azure, a la vez que mantienen la plena propiedad sobre el entorno del clúster.</p> <p>Ejemplo: Implementación de un clúster de Azure Kubernetes Service mediante la CLI de Azure.</p>
Funciones de Azure	<p>Azure Functions ofrece una plataforma de funciones como servicio (FaaS) controlada por eventos que permite ejecutar pequeños fragmentos de código (funciones) en respuesta a eventos, sin necesidad de administrar la infraestructura. Azure Functions comparte muchas características con Azure Container Apps en torno a la escala y la integración con eventos, pero está optimizado para funciones de corta duración implementadas como código o contenedores. Al para los equipos que buscan desencadenar la ejecución de funciones en eventos; por ejemplo, para enlazar a otros orígenes de datos. Al igual que Azure Container Apps, Azure Functions admite el escalado automático y la integración con orígenes de eventos (por ejemplo, solicitudes HTTP, colas de mensajes o actualizaciones de Blob Storage). Este servicio es ideal para los equipos que crean flujos de trabajo ligeros desencadenados por eventos, como el procesamiento de cargas de archivos o la respuesta a los cambios de base de datos, en Python u otros lenguajes.</p> <p>Ejemplo: Creación de una función en Linux mediante un contenedor personalizado.</p>

Para obtener una comparación más detallada de estos servicios, consulte [Comparación de aplicaciones de contenedor con otras opciones de contenedor de Azure](#).

Entornos virtuales y contenedores

Los entornos virtuales de Python aislan las dependencias del proyecto de las instalaciones de Python de nivel de sistema, lo que garantiza la coherencia entre entornos de desarrollo. Un entorno virtual incluye su propio intérprete aislado de Python, junto con las bibliotecas y scripts necesarios para ejecutar el código de proyecto específico dentro de ese entorno. Las dependencias de los proyectos de Python se administran a través del archivo *requirements.txt*. Al especificar dependencias en un archivo *derequirements.txt*, los desarrolladores pueden reproducir el entorno exacto necesario para su proyecto. Este enfoque facilita transiciones más fluidas a implementaciones en contenedores como Azure App Service, donde la coherencia del entorno es esencial para un rendimiento confiable de las aplicaciones.

Sugerencia

En los proyectos de Python en contenedores, los entornos virtuales suelen ser innecesarios porque los contenedores de Docker proporcionan entornos aislados con su

propio intérprete y dependencias de Python. Sin embargo, puede usar entornos virtuales para el desarrollo local o las pruebas. Para mantener las imágenes de Docker ajustadas, excluya los entornos virtuales mediante un archivo `.dockerignore`, lo que impide copiar archivos innecesarios en la imagen.

Puede pensar en contenedores de Docker como ofrecer funcionalidades similares a los entornos virtuales de Python, pero con ventajas más amplias en la reproducibilidad, el aislamiento y la portabilidad. A diferencia de los entornos virtuales, los contenedores de Docker se pueden ejecutar de forma coherente en distintos sistemas operativos y entornos, siempre y cuando esté disponible un entorno de ejecución de contenedor.

Un contenedor de Docker incluye el código del proyecto de Python junto con todo lo que necesita para ejecutarse, como dependencias, configuración del entorno y bibliotecas del sistema. Para crear un contenedor, primero se compila una imagen de Docker a partir del código y la configuración del proyecto y, a continuación, se inicia un contenedor, que es una instancia ejecutable de esa imagen.

Para incluir en contenedores proyectos de Python, los archivos de clave se describen en la tabla siguiente:

 Expandir tabla

Archivo de proyecto	Descripción
<code>requirements.txt</code>	Este archivo contiene la lista definitiva de dependencias de Python necesarias para la aplicación. Docker usa esta lista durante el proceso de compilación de imágenes para instalar todos los paquetes necesarios. Esto garantiza la coherencia entre entornos de desarrollo e implementación.
<code>Dockerfile</code>	Este archivo contiene instrucciones para compilar la imagen de Docker de Python, incluida la selección de imágenes base, la instalación de dependencias, la copia de código y los comandos de inicio del contenedor. Define el entorno de ejecución completo para la aplicación. Para obtener más información, consulte la sección Instrucciones de Dockerfile para Python .
<code>.dockerignore</code>	Este archivo especifica los archivos y directorios que se deben excluir al copiar contenido en la imagen de Docker con el <code>COPY</code> comando en el Dockerfile. Este archivo usa patrones similares a <code>.gitignore</code> para definir exclusiones. El archivo <code>.dockerignore</code> admite patrones de exclusión similares a los archivos <code>.gitignore</code> . Para obtener más información, vea archivo .dockerignore .

Configuración de contenedor para marcos web

Normalmente, los marcos web se enlazan a puertos predeterminados (como 5000 para Flask, 8000 para FastAPI). Al implementar contenedores en servicios de Azure, como Azure Container Instances, Azure Kubernetes Service (AKS) o App Service for Containers, es fundamental exponer y configurar explícitamente el puerto de escucha del contenedor para garantizar el enrutamiento adecuado del tráfico entrante. La configuración del puerto correcto garantiza que la infraestructura de Azure pueda dirigir las solicitudes al punto de conexión correcto dentro del contenedor.

 Expandir tabla

Marco web	Puerto
Django 	8.000
Flask 	5000 o 5002
FastAPI  (uvicorn  <td>8000 o 80</td>	8000 o 80

En la tabla siguiente se muestra cómo establecer el puerto para diferentes soluciones de contenedor de Azure.

 Expandir tabla

Solución de contenedor de Azure	Cómo establecer el puerto de la aplicación web
Aplicación web para contenedores	De forma predeterminada, App Service asume que el contenedor personalizado está escuchando en el puerto 80 o en el puerto 8080. Si el contenedor escucha a otro puerto, establezca la opción <code>WEBSITES_PORT</code> de la aplicación en App Service. Para más información, consulte Configuración de un contenedor personalizado para Azure App Service .
Aplicaciones de contenedores de Azure	Azure Container Apps le permite exponer la aplicación contenedora a la web pública, a la red virtual o a otras aplicaciones de contenedor dentro del mismo entorno habilitando la entrada. Establezca el <code>targetPort</code> de entrada en el puerto en el que escucha el contenedor para las solicitudes entrantes. El punto de conexión de entrada de la aplicación siempre se expone en el puerto 443. Para obtener más información, consulte Configurar el ingreso HTTPS o TCP en Azure Container Apps .
Azure Container Instances, Azure Kubernetes	Defines el puerto en el que tu aplicación está escuchando durante la creación de contenedores o pods. La imagen de contenedor debe incluir un marco web, un servidor de aplicaciones (por ejemplo, gunicorn, uvicorn) y, opcionalmente, un servidor web (por ejemplo, nginx). En escenarios más complejos, puede dividir las responsabilidades entre dos contenedores, una para el servidor de aplicaciones y otra

Solución de contenedor de Azure

Cómo establecer el puerto de la aplicación web

para el servidor web. En ese caso, el contenedor de servidores web normalmente expone los puertos 80 o 443 para el tráfico externo.

Python Dockerfile

Un Dockerfile es un archivo de texto que contiene instrucciones para compilar una imagen de Docker para una aplicación de Python. La primera instrucción normalmente especifica la imagen base desde la que empezar. A continuación, las instrucciones posteriores detallan acciones como instalar el software necesario, copiar archivos de aplicación y configurar el entorno para crear una imagen ejecutable. En la tabla siguiente se proporcionan ejemplos específicos de Python para obtener instrucciones de Dockerfile usadas habitualmente.

 Expandir tabla

Instrucción	Propósito	Ejemplo
FROM	Establece la imagen base para obtener instrucciones posteriores.	<code>FROM python:3.8-slim</code>
EXPOSE	Indica a Docker que el contenedor escucha un puerto especificado en tiempo de ejecución.	<code>EXPOSE 5000</code>
COPIAR	Copia archivos o directorios del origen especificado y los agrega al sistema de archivos del contenedor en la ruta de acceso de destino especificada.	<code>COPY . /app</code>
RUN	Ejecuta un comando dentro de la imagen de Docker. Por ejemplo, incorpore dependencias. El comando se ejecuta una vez en tiempo de compilación.	<code>RUN python -m pip install -r requirements.txt</code>
CMD	El comando proporciona el valor predeterminado para ejecutar un contenedor. Solo puede haber una instrucción CMD.	<code>CMD ["gunicorn", "--bind", "0.0.0.0:5000", "wsgi:app"]</code>

El comando docker build compila imágenes de Docker a partir de un Dockerfile y un contexto. El contexto de una compilación es el conjunto de archivos ubicados en la ruta de acceso o dirección URL especificadas. Normalmente, se compila una imagen a partir de la raíz del proyecto de Python y la ruta de acceso del comando de compilación es "." como se muestra en el ejemplo siguiente.

Bash

```
docker build --rm --pull --file "Dockerfile" --tag "mywebapp:latest" .
```

El proceso de compilación puede hacer referencia a cualquiera de los archivos del contexto. Por ejemplo, la compilación puede usar una instrucción COPY para hacer referencia a un archivo en el contexto. Este es un ejemplo de un Dockerfile para un proyecto de Python mediante el marco [de Flask](#) :

Dockerfile

```
FROM python:3.8-slim

EXPOSE 5000

# Keeps Python from generating .pyc files in the container.
ENV PYTHONDONTWRITEBYTECODE=1

# Turns off buffering for easier container logging
ENV PYTHONUNBUFFERED=1

# Install pip requirements.
COPY requirements.txt .
RUN python -m pip install -r requirements.txt

WORKDIR /app
COPY . /app

# Creates a non-root user with an explicit UID and adds permission to access the
# /app folder.
RUN adduser -u 5678 --disabled-password --gecos "" appuser && chown -R appuser
/app
USER appuser

# Provides defaults for an executing container; can be overridden with Docker CLI.
CMD ["gunicorn", "--bind", "0.0.0.0:5000", "wsgi:app"]
```

Puede crear un Dockerfile manualmente o crearlo automáticamente con VS Code y la extensión de Docker. Para obtener más información, consulte [Generación de archivos de Docker](#).

El comando docker build forma parte de la CLI de Docker. Cuando se usan IDE como VS Code o PyCharm, los comandos de interfaz de usuario para trabajar con imágenes de Docker llaman al comando de compilación automáticamente y automatizan la especificación de opciones.

Trabajar con imágenes de Python Docker y contenedores

VS Code y PyCharm

Los entornos de desarrollo integrados (IDE), como Visual Studio Code (VS Code) y PyCharm simplifican el desarrollo de contenedores de Python mediante la integración de tareas de Docker en el flujo de trabajo. Con extensiones o complementos, estos IDE simplifican la creación de imágenes de Docker, la ejecución de contenedores y la implementación en servicios de Azure, como App Service o Container Instances. Estas son algunas de las cosas que puede hacer con VS Code y PyCharm.

- Descargue y compile imágenes de Docker.
 - Cree imágenes en el entorno de desarrollo.
 - Compile imágenes de Docker en Azure sin Docker instaladas en el entorno de desarrollo. (Para PyCharm, use la CLI de Azure para compilar imágenes en Azure).
- Cree y ejecute contenedores de Docker desde una imagen existente, una imagen extraída o directamente desde un Dockerfile.
- Ejecute aplicaciones multicontenedor con Docker Compose.
- Conéctese y trabaje con registros de contenedor como Docker Hub, GitLab, JetBrains Space, Docker V2 y otros registros de Docker autohospedados.
- (Solo VS Code) Agregue un Dockerfile y archivos de Docker Compose que se adapten a tu proyecto de Python.

Para configurar VS Code y PyCharm para ejecutar contenedores de Docker en el entorno de desarrollo, siga estos pasos.

Código de VS

Si aún no lo ha hecho, instale [Azure Tools para VS Code](#).

[+] Expandir tabla

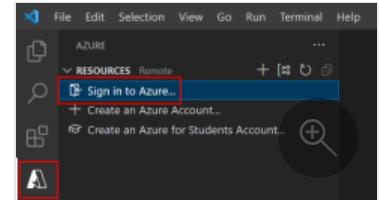
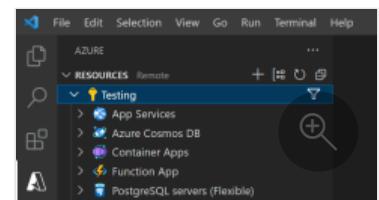
Instrucciones

Paso 1: Use MAYÚS + ALT + A para abrir la extensión de Azure y confirmar que está conectado a Azure.

También puede seleccionar el ícono de Azure en la barra de extensiones de VS Code.

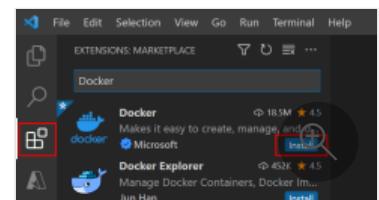
Si no ha iniciado sesión, seleccione **Iniciar sesión en Azure** y siga las indicaciones.

Si tiene problemas para acceder a la suscripción de Azure, puede deberse a que está detrás de un proxy. Para resolver problemas de conexión, consulte [Conexiones de red en Visual Studio Code](#).

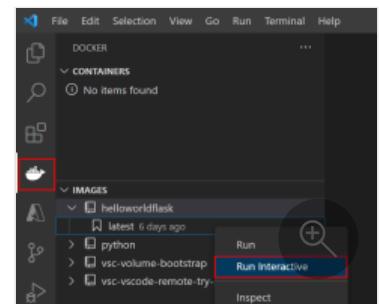


Paso 2: Use CTRL + MAYÚS + X para abrir **Extensiones**, busque la extensión de Docker e instale la extensión.

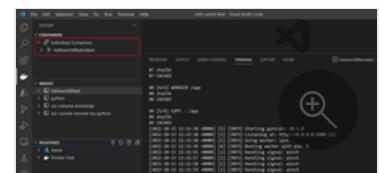
También puede seleccionar el ícono **Extensiones** en la barra de extensiones de VS Code.



Paso 3: Seleccione el ícono de Docker en la barra de extensiones, expanda imágenes y haga clic con el botón derecho en una imagen de Docker ejecutándola como contenedor.



Paso 4: Supervisar la salida de ejecución de Docker en la ventana Terminal .



CLI de Azure y CLI de Docker

También puede trabajar con imágenes y contenedores de Docker de Python mediante la [CLI de Azure](#) y la [CLI de Docker](#). VS Code y PyCharm tienen terminales donde puede ejecutar estas CLIs.

Use una CLI cuando desee un control más preciso sobre los argumentos de compilación y ejecución, y para la automatización. Por ejemplo, el siguiente comando muestra cómo usar la CLI de Azure `az acr build` para especificar el nombre de la imagen de Docker.

Bash

```
az acr build --registry <registry-name> \  
  --resource-group <resource-group> \  
  --target pythoncontainerwebapp:latest .
```

Como otro ejemplo, considere el siguiente comando que muestra cómo usar el comando [run](#) de la CLI de Docker. En el ejemplo se muestra cómo ejecutar un contenedor de Docker que se comunica con una instancia de MongoDB en el entorno de desarrollo, fuera del contenedor. Los distintos valores para completar el comando son más fáciles de automatizar cuando se especifican en una línea de comandos.

Bash

```
docker run --rm -it \  
  --publish <port>:<port> --publish 27017:27017 \  
  --add-host mongoservice:<your-server-IP-address> \  
  --env CONNECTION_STRING=mongodb://mongoservice:27017 \  
  --env DB_NAME=<database-name> \  
  --env COLLECTION_NAME=<collection-name> \  
  containermongo:latest
```

Para obtener más información sobre este escenario, consulte [Compilación y prueba de una aplicación web de Python en contenedores localmente](#).

Variables de entorno en contenedores

Los proyectos de Python suelen usar variables de entorno para pasar datos de configuración al código de la aplicación. Este enfoque permite una mayor flexibilidad en distintos entornos. Por ejemplo, los detalles de conexión de base de datos se pueden almacenar en variables de entorno, lo que facilita el cambio entre bases de datos de desarrollo, pruebas y producción sin modificar el código. Esta separación de la configuración del código promueve implementaciones más limpias y mejora la seguridad y el mantenimiento.

Los paquetes como [python-dotenv](#) se usan a menudo para leer pares clave-valor de un archivo `.env` y establecerlos como variables de entorno. Un archivo `.env` es útil cuando se ejecuta en un entorno virtual, pero no se recomienda al trabajar con contenedores. **No copie el archivo `.env` en la imagen de Docker, especialmente si contiene información confidencial y el contenedor se hará público.** Use el archivo `.dockerignore` para excluir que los archivos se copien en la imagen de Docker. Para obtener más información, consulte la sección [Entornos virtuales y contenedores](#) de este artículo.

Puede pasar variables de entorno a contenedores de varias maneras:

1. Se define en el Dockerfile como instrucciones [enV](#) .

2. Se pasa como `--build-arg` argumentos con el comando de compilación [Docker](#).
3. Se pasa como argumentos de `--secret` con el comando `build` de Docker y [BuildKit](#) [back-end](#).
4. Se pasa como argumentos `--env` o `--env-file` con el comando [Docker run](#).

Las dos primeras opciones tienen el mismo inconveniente que se indica con los archivos `.env`, es decir, que está codificando información potencialmente confidencial en una imagen de Docker. Puede inspeccionar una imagen de Docker y ver las variables de entorno, por ejemplo, con el comando [docker image inspect](#).

La tercera opción con BuildKit permite pasar información secreta que se usará en el Dockerfile para compilar imágenes de Docker de una manera segura que no termine almacenada en la imagen final.

La cuarta opción de pasar variables de entorno con el comando `Docker run` significa que la imagen de Docker no contiene las variables. Sin embargo, las variables siguen siendo visibles inspeccionando la instancia de contenedor (por ejemplo, con [la inspección del contenedor de Docker](#)). Esta opción puede ser aceptable cuando el acceso a la instancia de contenedor se controla o en escenarios de pruebas o desarrollo.

Este es un ejemplo de cómo pasar variables de entorno mediante el comando `run` de la CLI de Docker y mediante el `--env` argumento .

Bash

```
# PORT=8000 for Django and 5000 for Flask
export PORT=<port-number>

docker run --rm -it \
    --publish $PORT:$PORT \
    --env CONNECTION_STRING=<connection-info> \
    --env DB_NAME=<database-name> \
    <dockerimagename:tag>
```

En VS Code (extensión de Docker) o PyCharm (complemento de Docker), las herramientas de interfaz de usuario simplifican la administración de imágenes y contenedores de Docker mediante la ejecución de comandos estándar de la CLI de Docker (como `docker build`, `docker run`) en segundo plano.

Por último, especificar variables de entorno al implementar un contenedor en Azure es diferente de usar variables de entorno en el entorno de desarrollo. Por ejemplo:

- En el caso de Web App for Containers, se configuran las opciones de la aplicación durante la configuración de App Service. Esta configuración está disponible para el código de la aplicación como variables de entorno y se puede acceder a ella mediante el patrón

estándar [os.environ](#). Puede cambiar los valores después de la implementación inicial cuando sea necesario. Para obtener más información, consulte [Access app settings as environment variables \(Configuración de la aplicación de Access como variables de entorno\)](#).

- Para Azure Container Apps, puede configurar variables de entorno durante la configuración inicial de la aplicación contenedora. La modificación posterior de las variables de entorno crea una [revisión](#) del contenedor. Además, Azure Container Apps permite definir secretos en el nivel de aplicación y, a continuación, hacer referencia a ellos en variables de entorno. Para obtener más información, consulte [Secretos administrados en Azure Container Apps](#).

Como otra opción, puede usar [Service Connector](#) para ayudarle a conectar los servicios de proceso de Azure a otros servicios de respaldo. Este servicio configura la configuración de red y la información de conexión (por ejemplo, la generación de variables de entorno) entre los servicios de proceso y los servicios de respaldo de destino en el plano de administración.

Visualización de registros de contenedor

Vea los registros de instancia de contenedor para ver la salida de mensajes de diagnóstico del código y solucionar problemas en el código del contenedor. Estas son varias maneras de ver los registros al ejecutar un contenedor en el [entorno de desarrollo](#):

- Al ejecutar un contenedor con VS Code o PyCharm, como se muestra en la sección [VS Code y PyCharm](#), puede ver los registros en las ventanas de terminal abiertas cuando se ejecuta Docker.
- Si usa [el comando run](#) de la CLI de Docker con la marca interactiva `-it`, verá la salida siguiendo el comando.
- En [Docker Desktop](#), también puede ver los registros de un contenedor en ejecución.

Al implementar un contenedor en [Azure](#), también tiene acceso a los registros de contenedor. Estos son varios servicios de Azure y cómo acceder a los registros de contenedor en Azure Portal.

[\[+\] Expandir tabla](#)

Servicio de Azure	Acceso a los registros en Azure Portal
Aplicación web para contenedores	Vaya al recurso Diagnosticar y resolver problemas para ver los registros. El diagnóstico es una experiencia inteligente e interactiva para ayudarle a solucionar problemas de la aplicación sin que se requiera ninguna configuración. Para obtener una vista en tiempo real de los registros, vaya al flujo de registro de

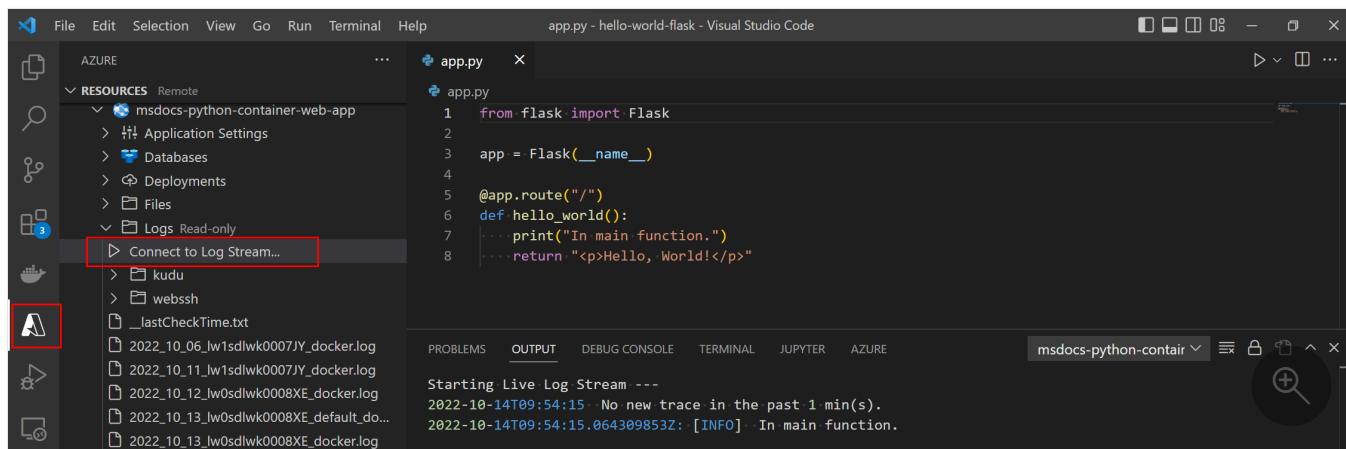
Servicio de Azure	Acceso a los registros en Azure Portal
	supervisión - . Para obtener consultas y configuraciones de registros más detalladas, consulte los demás recursos en Supervisión .
Azure Container Apps (Aplicaciones de Contenedores de Azure)	Vaya al recurso de entorno Diagnóstico y solución de problemas para solucionar problemas de entorno. Con mayor frecuencia, se quiere ver los registros de contenedor. En el recurso de contenedor, en Administración - <i>derevisiones</i> de aplicaciones, seleccione la revisión y, desde allí, puede ver los registros del sistema y de la consola. Para obtener consultas de registro y configuración más detalladas, consulte los recursos en Supervisión .
Azure Container Instances (Instancias de Contenedor de Azure)	Vaya al recurso Contenedores y seleccione Registros .

Para estos servicios, estos son los comandos de la CLI de Azure para acceder a los registros.

[\[+\] Expandir tabla](#)

Servicio de Azure	Comando de la CLI de Azure para acceder a los registros
Aplicación web para contenedores	az webapp log
Azure Container Apps (Aplicaciones de Contenedores de Azure)	az containerapps logs
Azure Container Instances	az container logs

También se admite la visualización de registros en VS Code. Debe tener [instaladas las herramientas de Azure para VS Code](#). A continuación se muestra un ejemplo de visualización de registros de Web Apps for Containers (App Service) en VS Code.



Pasos siguientes

- Aplicación web de Python en contenedor en Azure con MongoDB
- Implementación de una aplicación web de Python en Azure Container Apps con PostgreSQL

Implementar una aplicación web de Flask o FastAPI contenedorizada en Azure App Service

10/06/2025

En este tutorial se muestra cómo implementar una aplicación web [Flask](#) o [FastAPI](#) de Python en [Azure App Service](#) mediante la característica [Web App for Containers](#). Este enfoque proporciona una ruta de acceso simplificada para los desarrolladores que desean las ventajas de una plataforma totalmente administrada al implementar su aplicación como un único artefacto contenedorizado con todas las dependencias incluidas. Para obtener más información sobre el uso de contenedores en Azure, consulte [Comparación de las opciones de contenedor de Azure](#).

En este tutorial, usa la [CLI de Docker](#) y [Docker](#) para, opcionalmente, compilar y probar una imagen de Docker localmente. Después, use la [CLI de Azure](#) para insertar la imagen de Docker en [Azure Container Registry](#) (ACR) e implementarla en Azure App Service. La aplicación web se configura con su [identidad administrada](#) asignada por el sistema para obtener acceso seguro y sin contraseña para extraer la imagen de ACR mediante el control de acceso basado en rol (RBAC) de Azure. También puede implementar con [Visual Studio Code](#) con la [extensión de Azure Tools](#) instalada.

Para obtener un ejemplo de cómo compilar y crear una imagen de Docker para ejecutarse en Azure Container Apps, consulte [Implementación de una aplicación web Flask o FastAPI en Azure Container Apps](#).

Nota

En este tutorial se muestra cómo crear una imagen de Docker que se pueda implementar en Azure App Service. Sin embargo, no es necesario usar una imagen de Docker para implementarla en App Service. También puede implementar el código de la aplicación directamente desde el área de trabajo local en App Service sin crear una imagen de Docker. Para ver ejemplo, consulte [Inicio rápido: implementación de una aplicación web de Python \(Django o Flask\) en Azure App Service](#).

Requisitos previos

Para completar este tutorial, necesita:

- Una cuenta de Azure donde pueda implementar una aplicación web en [Azure App Service](#) y [Azure Container Registry](#). Si no tiene una suscripción a Azure, cree una [cuenta gratuita](#) antes de empezar.
- [CLI de Azure](#) para crear una imagen de Docker e implementarla en App Service. Y, opcionalmente, [Docker](#) y la [CLI de Docker](#) para crear un Docker y probarlo en el entorno local.

Obtención del código de ejemplo

En el entorno local, obtenga el código.

Flask

Bash

```
git clone https://github.com/Azure-Samples/msdocs-python-flask-webapp-quickstart.git
cd msdocs-python-flask-webapp-quickstart
```

Adición de archivos Dockerfile y .dockerignore

Agregue un *Dockerfile* para indicar a Docker cómo compilar la imagen. El *Dockerfile* especifica el uso de [Gunicorn](#), un servidor web de nivel de producción que reenvía las solicitudes web a los marcos Flask y FastAPI. Los comandos ENTRYPOINT y CMD indican a Gunicorn que controle las solicitudes del objeto de aplicación.

Flask

Dockerfile

```
# syntax=docker/dockerfile:1
FROM python:3.11
WORKDIR /code
COPY requirements.txt .
RUN pip3 install -r requirements.txt
COPY . .
```

```
EXPOSE 50505
```

```
ENTRYPOINT ["gunicorn", "app:app"]
```

50505 se usa para el puerto de contenedor (interno) en este ejemplo, pero puede usar cualquier puerto libre.

Compruebe el archivo *requirements.txt* para asegurarse de que contiene gunicorn.

```
Python
```

```
Flask==3.1.0
```

```
gunicorn
```

Agregue un archivo *.dockerignore* para excluir archivos innecesarios de la imagen.

```
dockerignore
```

```
.git*
**/*.pyc
.venv/
```

Configuración de gunicorn

Gunicorn se puede configurar con un archivo *gunicorn.conf.py*. Cuando el archivo *gunicorn.conf.py* se encuentra en el mismo directorio donde se ejecuta gunicorn, no es necesario especificar su ubicación en el *Dockerfile*. Para obtener más información sobre cómo especificar el archivo de configuración, consulte [Configuración de Gunicorn](#).

En este tutorial, el archivo de configuración sugerido configura gunicorn para aumentar su número de trabajos en función del número de núcleos de CPU disponibles. Para obtener más información sobre la configuración de archivos *gunicorn.conf.py*, consulte [Configuración de Gunicorn](#).

```
Flask
```

```
text
```

```
# Gunicorn configuration file
import multiprocessing

max_requests = 1000
max_requests_jitter = 50
```

```
log_file = "-"

bind = "0.0.0.0:50505"

workers = (multiprocessing.cpu_count() * 2) + 1
threads = workers

timeout = 120
```

Compilación y ejecución local de la imagen

Compile la imagen localmente.

Flask

Bash

```
docker build --tag flask-demo .
```

⚠ Nota

Si el comando `docker build` devuelve un error, asegúrese de que se está ejecutando el demonio del docker. En Windows, asegúrese de que Docker Desktop se esté ejecutando.

Ejecute la imagen localmente en un contenedor de Docker.

Flask

Bash

```
docker run --detach --publish 5000:50505 flask-demo
```

Abra la URL `http://localhost:5000` en el explorador para ver la aplicación web que se ejecuta localmente.

La opción `--detach` ejecuta el contenedor en segundo plano. La opción `--publish` asigna el puerto de contenedor a un puerto en el host. El puerto host (externo) es primero en el par y el puerto de contenedor (interno) es el segundo. Para más información, consulte [Referencia de ejecución de Docker](#).

Creación de un grupo de recursos y una instancia de Azure Container Registry

- Ejecute el comando `az login` para iniciar sesión en Azure.

```
Azure CLI
```

```
az login
```

- Ejecute el comando `az upgrade` para asegurarse de que la versión de la CLI de Azure está actualizada.

```
Azure CLI
```

```
az upgrade
```

- Cree un grupo con el comando `az group create`.

```
Azure CLI
```

```
RESOURCE_GROUP_NAME=<resource-group-name>
LOCATION=<location>
az group create --name $RESOURCE_GROUP_NAME --location $LOCATION
```

Un grupo de recursos de Azure es un contenedor lógico en el que se implementan y se administran los recursos de Azure. Al crear un grupo de recursos, especifique una ubicación como *eastus*. Reemplace por `<location>` la ubicación que elija. Algunas SKU no están disponibles en determinadas ubicaciones, por lo que es posible que reciba un error que indique esto. Use otra ubicación e inténtelo de nuevo.

- Cree una instancia de Azure Container Registry con el comando `az acr create`. Reemplace `<container-registry-name>` por un nombre único para la instancia.

```
Azure CLI
```

```
COUNTAINER_REGISTRY_NAME=<container-registry-name>
az acr create --resource-group $RESOURCE_GROOP_NAME \
--name $COUTAINER_REGISTRY_NAME --sku Basic
```

ⓘ Nota

El nombre del registro debe ser único en Azure. Si recibe un error, pruebe con un nombre diferente. Los nombres de registro pueden tener de 5 a 50 caracteres alfanuméricos. No se permiten guiones ni caracteres de subrayado. Para obtener más información, consulte [Reglas de nombres de instancias de Azure Container Registry](#). Si usa un nombre diferente, asegúrese de usar su nombre en lugar de `webappacr123` en los comandos que hacen referencia al registro y a los artefactos del registro en las secciones siguientes.

Azure Container Registry es un registro privado de Docker que almacena imágenes para su uso en Azure Container Instances, Azure App Service, Azure Kubernetes Service y otros servicios. Al crear un registro, especifique un nombre, una SKU y un grupo de recursos.

Compilación de la imagen en Azure Container Registry

Compile la imagen de Docker en Azure con el comando `az acr build`. El comando usa el Dockerfile en el directorio actual e inserta la imagen en el registro.

Azure CLI

```
az acr build \
--resource-group $RESOURCE_GROUP_NAME \
--registry $CONTAINER_REGISTRY_NAME \
--image webappsimple:latest .
```

La opción `--registry` especifica el nombre del registro y la opción `--image` especifica el nombre de la imagen. El nombre de la imagen está en el formato `registry.azurecr.io/repository:tag`.

Implementación de la aplicación web en Azure

1. Cree un plan de App Service con el comando [az appservice plan](#).

Azure CLI

```
az appservice plan create \
--name webplan \
--resource-group $RESOURCE_GROUP_NAME \
--sku B1 \
--is-linux
```

2. Establezca una variable de entorno para el ID suscripción. Se usa en el parámetro `--scope` en el comando siguiente.

```
Azure CLI
```

```
SUBSCRIPTION_ID=$(az account show --query id --output tsv)
```

El comando para crear la variable de entorno se muestra para el shell de Bash. Cambie la sintaxis según corresponda para otros entornos.

3. Cree la aplicación web con el comando `az webapp create`.

```
Azure CLI
```

```
export MSYS_NO_PATHCONV=1 # This line is for Windows users to prevent path
conversion issues in Git Bash.
az webapp create \
--resource-group $RESOURCE_GROUP_NAME \
--plan webplan --name <container-registry-name> \
--assign-identity [system] \
--role AcrPull \
--scope /subscriptions/$SUBSCRIPTION_ID/resourceGroups/$RESOURCE_GROUP_NAME \
--acr-use-identity --acr-identity [system] \
--container-image-name
$CONTAINER_REGISTRY_NAME.azurecr.io/webappsimple:latest
```

Notas:

- El nombre de la aplicación web debe ser único en Azure. Si recibe un error, pruebe con un nombre diferente. El nombre puede constar de caracteres alfanuméricos y guiones, pero no puede empezar ni terminar con un guion. Para obtener más información, consulte [Reglas de nombres de Microsoft.Web](#).
- Si usa un nombre diferente de `webappacr123` para su instancia de Azure Container Registry, asegúrese de actualizar el parámetro `--container-image-name` correctamente.
- Los parámetros `--assign-identity`, `--role` y `--scope` habilitan la identidad administrada asignada por el sistema en la aplicación web y le asignan el rol de [AcrPull](#) en el grupo de recursos. Esto proporciona permiso de identidad administrada para extraer imágenes de cualquier instancia de Azure Container Registry en el grupo de recursos.
- Los parámetros `--acr-use-identity` y `--acr-identity` configuran la aplicación web para que use su identidad administrada asignada por el sistema para extraer

imágenes de Azure Container Registry.

- La creación de la aplicación web puede tardar unos minutos. Puede comprobar los registros de implementación con el comando `az webapp log tail`. Por ejemplo, `az webapp log tail --resource-group web-app-simple-rg --name webappsimple123`. Si ve entradas con "warmup", significa que el contenedor se está implementando.
- La URL de la aplicación web es `<web-app-name>.azurewebsites.net`, por ejemplo, `https://webappsimple123.azurewebsites.net`.

Actualizaciones y nuevas implementaciones

Después de realizar cambios en el código, puede volver a implementar en App Service con los comandos `az acr build` y `az webapp update`.

Limpiar

Todos los recursos de Azure creados en este tutorial están en el mismo grupo de recursos. Al quitar el grupo de recursos se quitan todos los recursos que haya dentro; es la manera más rápida de quitar todos los recursos de Azure usados para la aplicación.

Para quitar recursos, use el comando `az group delete`.

Azure CLI

```
az group delete --name $RESOURCE_GROUP_NAME --yes --no-wait
```

También puede quitar el grupo en [Azure Portal](#) o en [Visual Studio Code](#) y la [extensión de Azure Tools](#).

Pasos siguientes

Para obtener más información, consulte los siguientes recursos:

- [Implementación de una aplicación web de Python en Azure Container Apps](#)
- [Inicio rápido: implementación de una aplicación web de Python \(Django o Flask\) en Azure App Service](#)

Información general: Aplicación web de Python en contenedor en Azure con MongoDB

Artículo • 16/04/2025

En esta serie de tutoriales se muestra cómo incluir en contenedores una aplicación web de Python y, a continuación, ejecutarla localmente o implementarla en [Azure App Service](#). App Service [Web App for Containers](#) le permite centrarse en la creación de contenedores sin preocuparse de administrar y mantener un orquestador de contenedores subyacente. Al compilar aplicaciones web, Azure App Service es una buena opción para realizar los primeros pasos con los contenedores. Esta aplicación web de contenedor puede usar una instancia local de [MongoDB](#) o [MongoDB para Azure Cosmos DB](#) para almacenar datos. Para obtener más información sobre el uso de contenedores en Azure, consulte [Comparación de las opciones de contenedor de Azure](#).

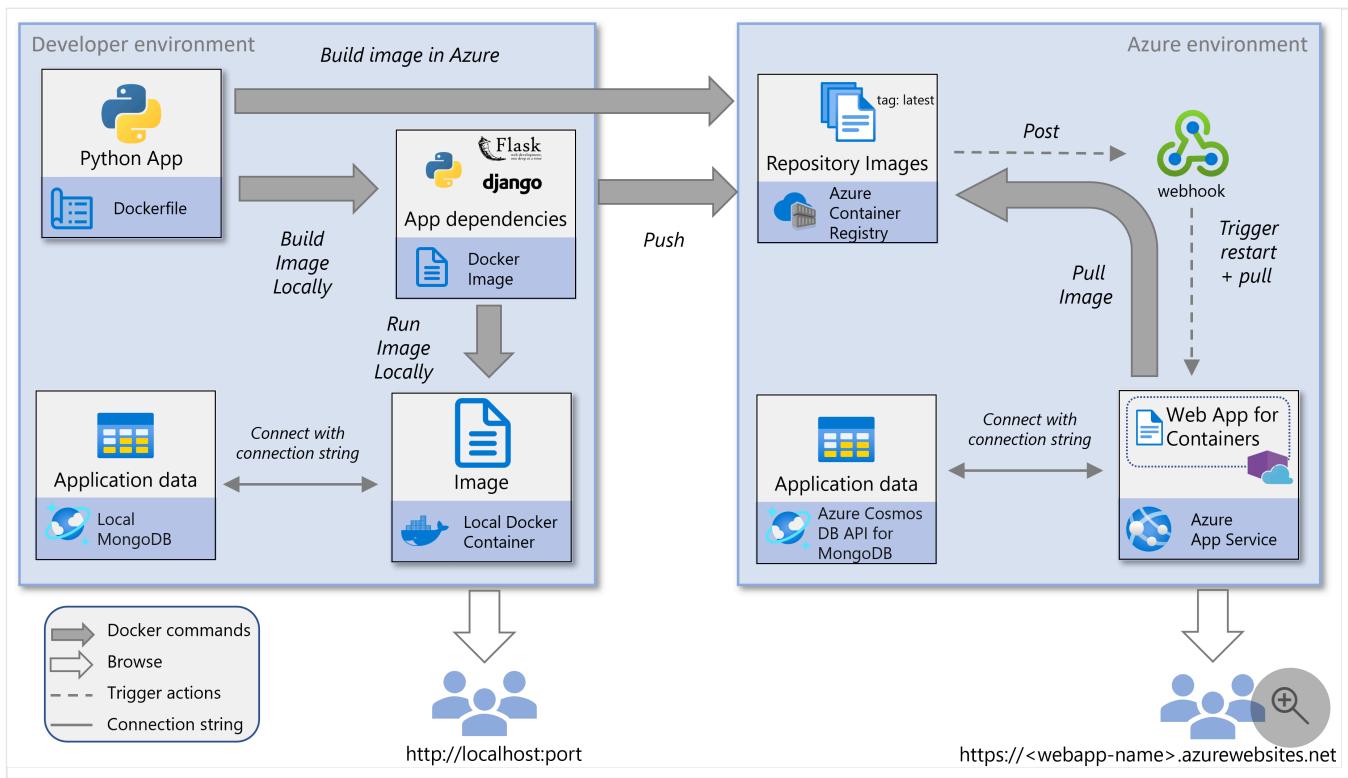
En este tutorial:

- Compile y ejecute un contenedor [Docker](#) localmente. Consulte [Compilación y ejecución de una aplicación web de Python en contenedores localmente](#).
- Compile una imagen de contenedor [Docker](#) directamente en Azure. Consulte [Compilación de una aplicación web de Python en contenedor en Azure](#).
- Configure una instancia de App Service para crear una aplicación web basada en la imagen de contenedor de Docker. Consulte [Implementación de una aplicación de Python en contenedor en App Service](#).

Después de completar los artículos de esta serie de tutoriales, tendrá la base para la integración continua (CI) y la implementación continua (CD) de una aplicación web de Python en Azure.

Introducción al servicio

El diagrama de servicio que admite este tutorial muestra dos entornos: entorno de desarrollador y entorno de Azure. Resalta los servicios clave de Azure que se usan en el proceso de desarrollo.



Entorno para desarrolladores

Entre los componentes que admiten el entorno para desarrolladores de este tutorial se incluyen:

- **sistema de desarrollo local:** un equipo personal que se usa para codificar, compilar y probar el contenedor de Docker.
- **contenedorización de Docker:** Docker se emplea para empaquetar la aplicación y sus dependencias en un contenedor portátil.
- **Herramientas de desarrollo:** incluye un editor de código y otras herramientas necesarias para el desarrollo de software.
- **instancia local de MongoDB:** una base de datos local de MongoDB utilizada para el almacenamiento de datos durante el desarrollo.
- **Conexión de MongoDB:** acceso a la base de datos local de MongoDB proporcionada a través de una cadena de conexión.

Entorno de Azure

Los componentes que admiten el entorno de Azure en este tutorial incluyen:

- [Azure App Service ↗](#)

- En Azure App Service, Web App for Containers usa la tecnología de contenedor [de Docker](#) para proporcionar hospedaje de contenedores de imágenes integradas e imágenes personalizadas mediante Docker.
 - Web App for Containers usa un webhook en Azure Container Registry (ACR) para recibir notificaciones de nuevas imágenes. Cuando se inserta una nueva imagen en el registro, la notificación de webhook desencadena App Service para extraer la actualización y reiniciar la aplicación.
- [Registro de contenedores de Azure](#)
 - Azure Container Registry permite almacenar y administrar imágenes de Docker y sus componentes en Azure. Proporciona un registro ubicado cerca de las implementaciones de Azure que ofrece la capacidad de controlar el acceso mediante los permisos y grupos de Microsoft Entra.
 - En este tutorial, Azure Container Registry es el origen del registro, pero también puede usar Docker Hub o un registro privado con modificaciones menores.
- [Azure Cosmos DB para MongoDB](#)
 - Azure Cosmos DB para MongoDB es una base de datos NoSQL que se usa en este tutorial para el almacenamiento de datos.
 - La aplicación en contenedor se conecta y accede al recurso de Azure Cosmos DB mediante una cadena de conexión, que se almacena como una variable de entorno y se proporciona a la aplicación.

Autenticación

En este tutorial, creará una imagen de Docker localmente o en Azure y, a continuación, la implementará en Azure App Service. App Service extrae la imagen de contenedor de un repositorio de Azure Container Registry.

Para extraer imágenes del repositorio de forma segura, App Service usa una identidad administrada asignada por el sistema. Esta identidad administrada concede a la aplicación web permisos para interactuar con otros recursos de Azure, lo que elimina la necesidad de credenciales explícitas. En este tutorial, la identidad administrada se configura durante la instalación de App Service para usar una imagen de contenedor del registro.

La aplicación web de ejemplo del tutorial usa MongoDB para almacenar datos. El código de ejemplo se conecta a Azure Cosmos DB a través de una cadena de conexión.

Prerrequisitos

Para completar este tutorial, necesita lo siguiente:

- Una cuenta de Azure donde puede crear:
 - [Registro de contenedores de Azure ↗](#)
 - [Azure App Service ↗](#)
 - [Azure Cosmos DB para MongoDB](#) (o acceso a un equivalente). Para crear una base de datos de Azure Cosmos DB para MongoDB, siga los pasos [descritos en la parte 2 de este tutorial](#).
- [Visual Studio Code ↗](#) o [CLI de Azure](#), según la herramienta que prefiera. Si usa Visual Studio Code, necesita la extensión de Docker [↗](#) y la extensión de Azure App Service [↗](#).
- Estos paquetes de Python:
 - [Shell de MongoDB \(mongosh\) ↗](#) para conectarse a MongoDB.
 - [Flask ↗](#) o [Django ↗](#) como marco web.
- [Docker ↗](#) instalado localmente.

Aplicación de ejemplo

El resultado final de este tutorial es una aplicación de revisión de restaurantes, implementada y en ejecución en Azure, que tiene un aspecto similar a la siguiente captura de pantalla.

The screenshot shows a web application interface for a restaurant review. At the top, there's a header bar with the Azure logo and the text "Azure Restaurant Review" on the left, and "Azure Docs ▾" on the right. Below the header, the title "Contoso Café" is displayed. Underneath the title, there are three sections: "Street address:" with "1 Main Street", "Description:" with "Friendly coffee shop.", and "Rating:" with a yellow star icon and "4.5 (2 reviews)". Below these details, a section titled "Reviews" contains a button "Add new review". A table lists two reviews: one from "Davide Sagese" on 26/07/2022 at 14:46:54 with a rating of 5 and the review text "Great cappuccino.", and another from "Francesca Lombo" on 26/07/2022 at 14:47:58 with a rating of 4 and the review text "Healthy breakfast choices.". To the right of the reviews table is a circular search icon with a magnifying glass and a plus sign.

Date	User	Rating	Review
26/07/2022 14:46:54	Davide Sagese	5	Great cappuccino.
26/07/2022 14:47:58	Francesca Lombo	4	Healthy breakfast choices.

En este tutorial, creará una aplicación de revisión de restaurantes de Python que usa MongoDB para el almacenamiento de datos. Para ver una aplicación de ejemplo con PostgreSQL, consulte

Creación e implementación de una aplicación web de Flask en Azure con una identidad administrada.

Paso siguiente

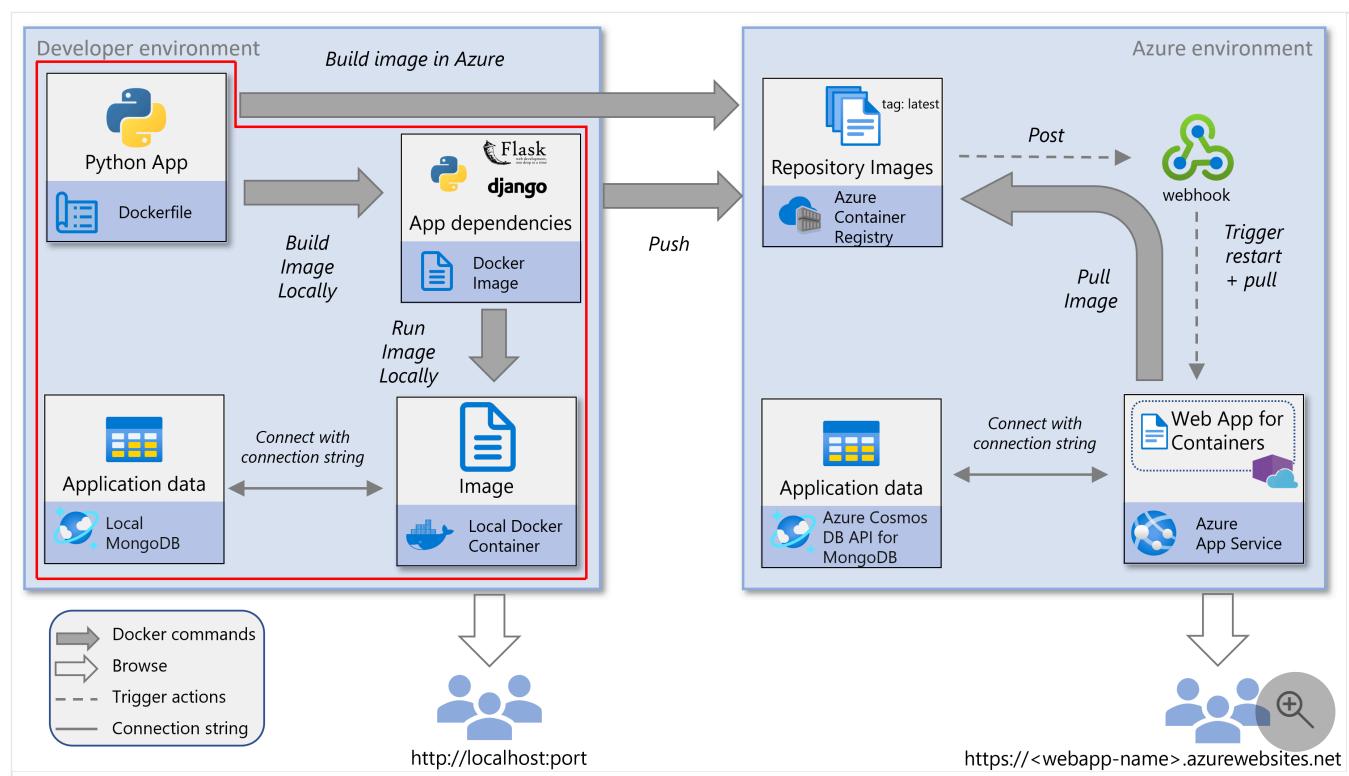
Compilación y prueba local

Compilación y ejecución de una aplicación web de Python en contenedores localmente

Artículo • 23/04/2025

En esta parte de la serie de tutoriales, aprenderás a crear y ejecutar una aplicación web de Python contenedorizada usando [Django](#) o [Flask](#) en tu equipo local. Para almacenar datos para esta aplicación, puede usar una instancia local de MongoDB o [Azure Cosmos DB para MongoDB](#). Este artículo es la parte 2 de una serie de tutoriales de 5 partes. Se recomienda completar [la parte 1](#) antes de comenzar este artículo.

En el diagrama de servicio siguiente se resaltan los componentes locales descritos en este artículo. En este artículo, también aprenderá a usar Azure Cosmos DB para MongoDB con una imagen de Docker local, en lugar de una instancia local de MongoDB.



Clonar o descargar la aplicación de Python de ejemplo

En esta sección, clonará o descargará la aplicación de Python de ejemplo que se usa para compilar una imagen de Docker. Puede elegir entre una aplicación web de Django o Flask para Python. Si tiene su propia aplicación web de Python, puede optar por usarla en su lugar. Si usa su propia aplicación web de Python, asegúrese de que la aplicación tiene un *Dockerfile* en la carpeta raíz y puede conectarse a una base de datos de MongoDB.

Clonación de Git

1. Clone el repositorio de Django o Flask en una carpeta local mediante uno de los siguientes comandos:

Django

Consola

```
# Django
git clone https://github.com/Azure-Samples/msdocs-python-django-
container-web-app.git
```

2. Vaya a la carpeta raíz del repositorio clonado.

Django

Consola

```
# Django
cd msdocs-python-django-container-web-app
```

Creación de una imagen de Docker

En esta sección, creará una imagen de Docker para la aplicación web de Python mediante Visual Studio Code o la CLI de Azure. La imagen de Docker contiene la aplicación web de Python, sus dependencias y el entorno de ejecución de Python. La imagen de Docker se crea a partir de un *Dockerfile* que define el contenido y el comportamiento de la imagen. *El Dockerfile* se encuentra en la carpeta raíz de la aplicación de ejemplo que ha clonado o descargado (o proporcionado usted mismo).

Sugerencia

Si no está familiarizado con la CLI de Azure, consulte [Introducción a la CLI de Azure](#) para obtener información sobre cómo descargar e instalar la CLI de Azure localmente o cómo ejecutar comandos de la CLI de Azure en Azure Cloud Shell.

Docker [🔗](#) es necesario para compilar la imagen de Docker mediante la CLI de Docker. Una vez instalado Docker, abra una ventana de terminal y vaya a la carpeta de ejemplo.

⚠ Nota

Los pasos de esta sección requieren que se ejecute el demonio Docker. En algunas instalaciones, por ejemplo, en Windows, debe abrir [Docker Desktop](#) [🔗](#), que inicia el servicio, antes de continuar.

1. Confirme que Docker es accesible mediante la ejecución del siguiente comando en la carpeta raíz de la aplicación de ejemplo.

Consola

```
docker
```

Si, después de ejecutar este comando, ve la ayuda para la [CLI de Docker](#) [🔗](#), significa que se puede acceder a Docker. De lo contrario, asegúrese de que Docker esté instalado y de que el shell tenga acceso a la CLI de Docker.

2. Compile la imagen de Docker para la aplicación web de Python mediante el comando [de compilación de Docker](#) [🔗](#).

La forma general del comando es `docker build --rm --pull --file "<path-to-project-root>/Dockerfile" --label "com.microsoft.created-by=docker-cli" --tag "<container-name>:latest" "<path-to-project-root>"`.

Si está en la carpeta raíz del proyecto, use el siguiente comando para compilar la imagen de Docker. El punto (".") al final del comando hace referencia al directorio actual en el que se ejecuta el comando. Para forzar una recompilación, agregue `--no-cache`.

Bash

Consola

```
#!/bin/bash
docker build --rm --pull \
--file "Dockerfile" \
--label "com.microsoft.create-by=docker-cli" \
```

```
--tag "msdocspythoncontainerwebapp:latest" \
```

```
.
```

3. Confirme que la imagen se ha compilado correctamente mediante el comando [Imágenes de Docker](#).

Consola

```
docker images
```

El comando devuelve una lista de imágenes por nombre de repositorio, etiqueta y fecha de creación, entre otras características de imagen.

En este momento, tiene una imagen de Docker local denominada "msdocspythoncontainerwebapp" con la etiqueta "latest". Las etiquetas ayudan a definir los detalles de la versión, el uso previsto, la estabilidad y otra información relevante. Para más información, consulte [Recomendaciones para el etiquetado y control de versiones de las imágenes de contenedor](#).

 **Nota**

Las imágenes compiladas a partir de VS Code o mediante la CLI de Docker directamente también se pueden ver con la aplicación [Docker Desktop](#).

Configuración de MongoDB

La aplicación web de Python requiere una base de datos de MongoDB denominada *restaurants_reviews* y se requiere una colección denominada *restaurants_reviews* para almacenar datos. En este tutorial, usará una instalación local de MongoDB y una instancia de [Azure Cosmos DB para MongoDB](#) para crear y acceder a la base de datos y la colección.

 **Importante**

No use una base de datos de MongoDB que use en producción. En este tutorial, almacenará la cadena de conexión de MongoDB en una de estas instancias de MongoDB en una variable de entorno (que es observable por cualquier persona capaz de inspeccionar el contenedor, como mediante `docker inspect`).

Local MongoDB

Comencemos creando una instancia local de MongoDB mediante la CLI de Azure.

1. Instale [MongoDB](#) (si aún no está instalado).

Puede comprobar la instalación de MongoDB mediante el [Shell de MongoDB \(mongosh\)](#). Si los siguientes comandos no funcionan, es posible que tenga que [instalar explícitamente mongosh](#) o [conectar mongosh al servidor de MongoDB](#).

- Use el siguiente comando para abrir el shell de MongoDB y obtener la versión del shell de MongoDB y el servidor de MongoDB:

```
Consola
```

```
mongosh
```

Sugerencia

Para devolver solo la versión del servidor de MongoDB instalado en el sistema, cierre y vuelva a abrir el shell de MongoDB y use el siguiente comando: `mongosh --quiet --exec 'db.version()'`

En algunas configuraciones, también puede invocar directamente el demonio de Mongo en el shell de Bash.

```
Consola
```

```
mongod --version
```

2. Edite el archivo `mongod.cfg` en la `\MongoDB\Server\8.0\bin` carpeta y agregue la dirección IP local del equipo a la `bindIP` clave.

La `bindip` clave del [archivo de configuración de MongoDB](#) define los nombres de host y las direcciones IP que MongoDB escucha para las conexiones de cliente. Agregue la dirección IP actual del equipo de desarrollo local. La aplicación web de Python de ejemplo que se ejecuta localmente en un contenedor de Docker se comunica con el equipo host con esta dirección.

Por ejemplo, parte del archivo de configuración debe tener este aspecto:

```
yml
```

```
net:  
  port: 27017  
  bindIp: 127.0.0.1,<local-ip-address>
```

3. Guarde los cambios realizados en este archivo de configuración.

ⓘ Importante

Necesita privilegios administrativos para guardar los cambios realizados en este archivo de configuración.

4. Reinicie MongoDB para recoger los cambios en el archivo de configuración.

5. Abra un shell de MongoDB y ejecute el siguiente comando para establecer el nombre de la base de datos en "restaurants_reviews" y el nombre de la colección en "restaurants_reviews". También puede crear una base de datos y una colección con la extensión MongoDB ↗ de VS Code o cualquier otra herramienta compatible con MongoDB.

```
mongosh  
> help  
> use restaurants_reviews  
> db.restaurants_reviews.insertOne({})  
> show dbs  
> exit
```

Después de completar el paso anterior, la cadena de conexión local de MongoDB es "mongodb://127.0.0.1:27017/", el nombre de la base de datos es "restaurants_reviews" y el nombre de la colección es "restaurants_reviews".

Azure Cosmos DB para MongoDB

Ahora, también vamos a crear una instancia de Azure Cosmos DB para MongoDB mediante la CLI de Azure.

! Nota

En la parte 4 de esta serie de tutoriales, usará la instancia de Azure Cosmos DB para MongoDB para ejecutar la aplicación web en Azure App Service.

Antes de ejecutar el siguiente script, reemplace la ubicación, el grupo de recursos y el nombre de la cuenta de Azure Cosmos DB para MongoDB por los valores adecuados (opcional). Se recomienda usar el mismo grupo de recursos para todos los recursos de Azure creados en este tutorial para que sean más fáciles de eliminar cuando haya terminado.

El script tarda unos minutos en ejecutarse.

Bash

```
#!/bin/bash
# LOCATION: The Azure region. Use the "az account list-locations -o table" command to find a region near you.
LOCATION='westus'
# RESOURCE_GROUP_NAME: The resource group name, which can contain underscores, hyphens, periods, parenthesis, letters, and numbers.
RESOURCE_GROUP_NAME='msdocs-web-app-rg'
# ACCOUNT_NAME: The Azure Cosmos DB for MongoDB account name, which can contain lowercase letters, hyphens, and numbers.
ACCOUNT_NAME='msdocs-cosmos-db-account-name'

# Create a resource group
echo "Creating resource group $RESOURCE_GROUP_NAME in $LOCATION..."
az group create --name $RESOURCE_GROUP_NAME --location $LOCATION

# Create a Cosmos account for MongoDB API
echo "Creating $ACCOUNT_NAME. This command may take a while to complete."
az cosmosdb create --name $ACCOUNT_NAME --resource-group $RESOURCE_GROUP_NAME --kind MongoDB

# Create a MongoDB API database
echo "Creating database restaurants_reviews"
az cosmosdb mongodb database create --account-name $ACCOUNT_NAME --resource-group $RESOURCE_GROUP_NAME --name restaurants_reviews

# Create a MongoDB API collection
echo "Creating collection restaurants_reviews"
az cosmosdb mongodb collection create --account-name $ACCOUNT_NAME --resource-group $RESOURCE_GROUP_NAME --database-name restaurants_reviews --name restaurants_reviews

# Get the connection string for the MongoDB database
echo "Get the connection string for the MongoDB account"
az cosmosdb keys list --name $ACCOUNT_NAME --resource-group $RESOURCE_GROUP_NAME --type connection-strings

echo "Copy the Primary MongoDB Connection String from the list above"
```

Cuando se complete el script, copie la *cadena de conexión principal de MongoDB* de la salida del último comando al Portapapeles o a otra ubicación.

```
Resultados

{
  "connectionStrings": [
    {
      "connectionString": """mongodb://msdocs-cosmos-
db:pnaMGVtGIRAZHUjsg4GJBCZMBJ0trV4eg2IcZf1TqV...5oONz0WX14Ph0Ha5IeYACDbuVrBPA==@ms
docs-cosmos-db.mongo.cosmos.azure.com:10255/?
ssl=true&replicaSet=globaldb&retrywrites=false&maxIdleTimeMS=120000&appName=@msdoc
s-cosmos-db""",
      "description": "Primary MongoDB Connection String",
      "keyKind": "Primary",
      "type": "MongoDB"
    },
    ...
  ]
}
```

Después de completar el paso anterior, tiene una cadena de conexión de Azure Cosmos DB para MongoDB con el formato `mongodb://<server-name>:<password>@<server-
name>.mongo.cosmos.azure.com:10255/?ssl=true&<other-parameters>`, una base de datos denominada `restaurants_reviews` y una colección denominada `restaurants_reviews`.

Para más información sobre cómo usar la CLI de Azure para crear una cuenta de Cosmos DB para MongoDB y crear bases de datos y colecciones, consulte [Creación de una base de datos y una colección para MongoDB para Azure Cosmos DB mediante la CLI de Azure](#). También puede usar [PowerShell](#), la [extensión Azure Databases](#) de VS Code y Azure Portal.

Sugerencia

En la extensión Azure Databases de VS Code, puede hacer clic con el botón derecho en el servidor de MongoDB y obtener la cadena de conexión.

Ejecute la imagen localmente en un contenedor

Ya está listo para ejecutar el contenedor de Docker localmente mediante la instancia local de MongoDB o la instancia de Cosmos DB para MongoDB. En esta sección del tutorial, aprenderá a usar VS Code o la CLI de Azure para ejecutar la imagen localmente. La aplicación de ejemplo espera que la información de conexión de MongoDB se le proporcione mediante variables de entorno. Hay varias maneras de obtener las variables de entorno que se pasan al contenedor

localmente. Cada uno tiene ventajas y desventajas en términos de seguridad. Debe evitar la comprobación de cualquier información confidencial o dejar información confidencial en el código del contenedor.

! Nota

Cuando la aplicación web se implementa en Azure, la aplicación web obtiene información de conexión de los valores de entorno establecidos como opciones de configuración de App Service y no se aplica ninguna de las modificaciones para el escenario de entorno de desarrollo local.

CLI de Azure

MongoDB local

Use los siguientes comandos con la instancia local de MongoDB para ejecutar la imagen de Docker localmente.

1. Ejecute la versión más reciente de la imagen.

Bash

```
Bash

#!/bin/bash

# Define variables
# Set the port number based on the framework being used:
# 8000 for Django, 5000 for Flask
export PORT=<port-number> # Replace with actual port (e.g., 8000 or
5000)

# Set your computer's IP address (replace with actual IP)
export YOUR_IP_ADDRESS=<your-computer-ip-address> # Replace with
actual IP address

# Run the Docker container with the required environment variables
docker run --rm -it \
--publish "$PORT:$PORT" \
--publish 27017:27017 \
--add-host "mongoservice:$YOUR_IP_ADDRESS" \
--env CONNECTION_STRING=mongodb://mongoservice:27017 \
--env DB_NAME=restaurants_reviews \
--env COLLECTION_NAME=restaurants_reviews \
```

```
--env SECRET_KEY="supersecretkeythatispassedtopythonapp" \
msdocspythoncontainerwebapp:latest
```

2. Confirme que el contenedor está en ejecución. En otra ventana de consola, ejecute el comando [docker container ls](#).

Consola

```
docker container ls
```

Consulte el contenedor "msdocspythoncontainerwebapp:latest:latest" en la lista. Observe la `NAMES` columna de la salida y la `PORTS` columna. Use el nombre del contenedor para detener el contenedor.

3. Pruebe la aplicación web.

Vaya a "http://127.0.0.1:8000" para Django y "http://127.0.0.1:5000/" para Flask.

4. Apague el contenedor.

Consola

```
docker container stop <container-name>
```

Azure Cosmos DB para MongoDB

Use los siguientes comandos con la instancia de Azure Cosmos DB para MongoDB para ejecutar la imagen de Docker en Azure.

1. Ejecute la versión más reciente de la imagen.

Bash

```
#!/bin/bash
# PORT=8000 for Django and 5000 for Flask
export PORT=<port-number>
export CONNECTION_STRING="<connection-string>

docker run --rm -it \
--publish $PORT:$PORT/tcp \
--env CONNECTION_STRING=$CONNECTION_STRING \
--env DB_NAME=restaurants_reviews \
--env COLLECTION_NAME=restaurants_reviews \
```

```
--env SECRET_KEY=supersecretkeythatyougenerate \
msdocspythoncontainerwebapp:latest
```

Pasar información confidencial solo se muestra con fines de demostración. La información de la cadena de conexión se puede ver inspeccionando el contenedor con el comando [docker container inspect](#). Otra manera de controlar secretos es usar la funcionalidad [BuildKit](#) de Docker.

2. Abra una nueva ventana de consola y ejecute el siguiente comando [docker container ls](#) para confirmar que el contenedor se está ejecutando.

Consola

```
docker container ls
```

Consulte el contenedor "msdocspythoncontainerwebapp:latest:latest" en la lista. Observe la `NAMES` columna de la salida y la `PORTS` columna. Use el nombre del contenedor para detener el contenedor.

3. Pruebe la aplicación web.

Vaya a "http://127.0.0.1:8000" para Django y "http://127.0.0.1:5000/" para Flask.

4. Apague el contenedor.

Consola

```
docker container stop <container-name>
```

También puede iniciar un contenedor desde una imagen y detenerlo con la aplicación [Docker Desktop](#).

Paso siguiente

[Creación de una imagen de contenedor en Azure](#)

Compilación de una aplicación web de Python en contenedor en Azure

Artículo • 17/04/2025

En esta parte de la serie de tutoriales, aprenderá a compilar una aplicación web de Python en contenedor directamente en [Azure Container Registry](#) sin instalar Docker localmente. La creación de la imagen de Docker en Azure suele ser más rápida y sencilla que crear la imagen localmente y, a continuación, insertarla en Azure Container Registry. Además, la creación de imágenes basada en la nube elimina la necesidad de que Docker se ejecute en el entorno de desarrollo.

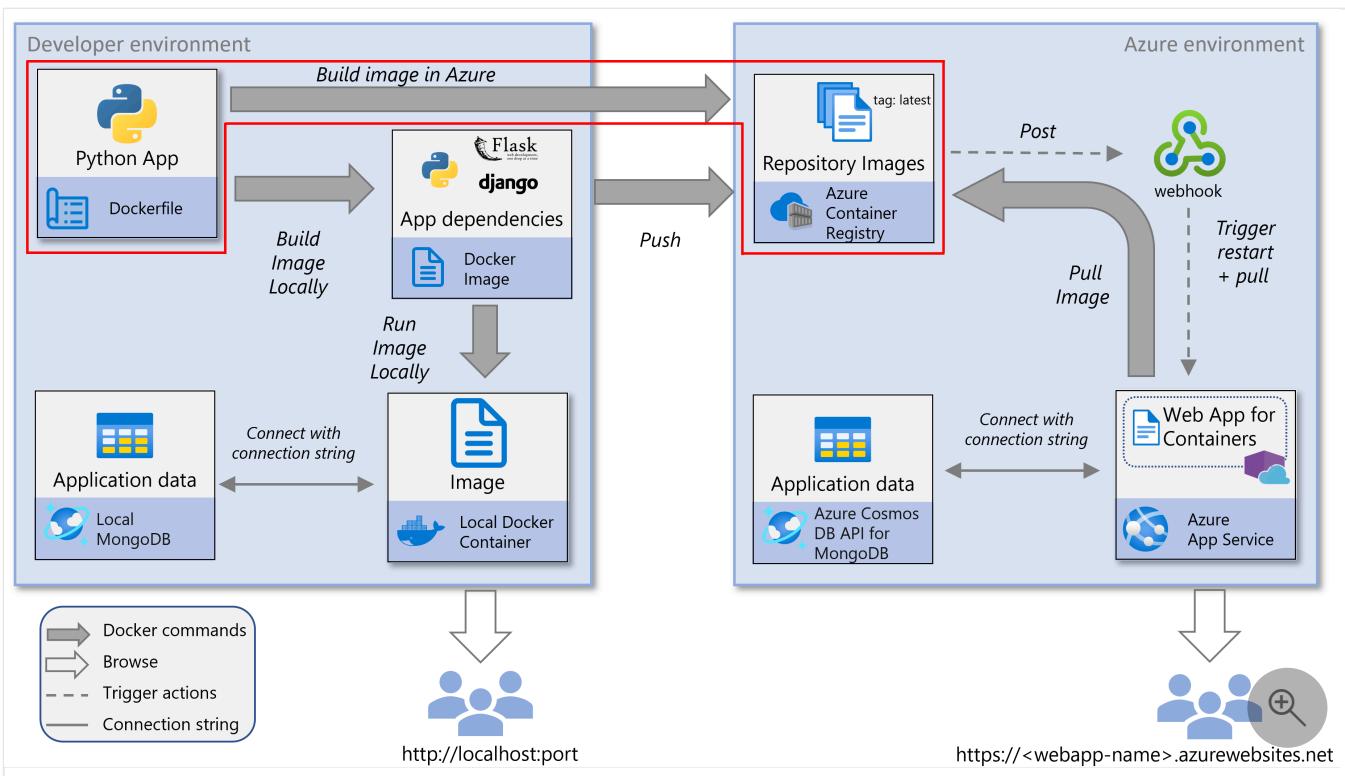
App Service le permite ejecutar aplicaciones web en contenedores e implementarlas a través de las funcionalidades de integración continua e implementación continua (CI/CD) de Docker Hub, Azure Container Registry y Visual Studio Team Services. Este artículo es la parte 3 de una serie de tutoriales de 5 partes sobre cómo incluir e implementar una aplicación web de Python en Azure App Service. En esta parte del tutorial, aprenderá a compilar la aplicación web de Python en contenedores en Azure.

Azure App Service le permite implementar y ejecutar aplicaciones web en contenedores mediante canalizaciones de CI/CD desde plataformas como Docker Hub, Azure Container Registry y Azure DevOps. Este artículo es la parte 3 de una serie de tutoriales de 5 partes.

En [la parte 2 de esta serie de tutoriales](#), ha compilado y ejecutado la imagen de contenedor localmente. En cambio, en esta parte del tutorial, compilará (contenedorizará) la misma aplicación web de Python directamente en una imagen de Docker en [Azure Container Registry](#). La creación de la imagen en Azure suele ser más rápida y sencilla que compilar localmente y, a continuación, insertar la imagen en un registro. Además, la compilación en la nube no requiere que Docker se ejecute en el entorno de desarrollo.

Una vez que la imagen de Docker está en Azure Container Registry, se puede implementar en Azure App Service.

En este diagrama de servicio se resaltan los componentes descritos en este artículo.



CLI de Azure

Creación de una instancia de Azure Container Registry

Si tiene una instancia de Azure Container Registry existente que desea usar, omita este paso siguiente y continúe con el paso siguiente. De lo contrario, cree un nuevo registro de contenedor de Azure mediante la CLI de Azure.

Los comandos de la CLI de Azure se pueden ejecutar en [Azure Cloud Shell](#) o en el entorno de desarrollo local con la [CLI de Azure instalada](#).

! Nota

Use los mismos nombres que en la parte 2 de esta serie de tutoriales.

1. Cree un registro de contenedor de Azure con el comando [az acr create](#) .

Bash

Azure CLI

```
#!/bin/bash
# Use the resource group that you created in part 2 of this tutorial
# series.
RESOURCE_GROUP_NAME='msdocs-web-app-rg'
# REGISTRY_NAME must be unique within Azure and contain 5-50
# alphanumeric characters.
REGISTRY_NAME='msdocscontainerregistryname'

echo "Creating Azure Container Registry $REGISTRY_NAME..."
az acr create -g $RESOURCE_GROUP_NAME -n $REGISTRY_NAME --sku
Standard
```

En la salida JSON del comando, busque el valor `loginServer`. Este valor representa el nombre completo del Registro (todo en minúsculas) y contiene el nombre del Registro.

- Si usa la CLI de Azure en el equipo local, ejecute el comando `az acr login` para iniciar sesión en el registro de contenedor.

Azure CLI

```
az acr login -n $REGISTRY_NAME
```

El comando agrega "azurecr.io" al nombre para crear el nombre de registro completamente calificado. Si se ejecuta correctamente, verá el mensaje "Login Succeeded".

⚠️ Nota

En Azure Cloud Shell, el az `acr login command` no es necesario, ya que la autenticación se controla automáticamente a través de la sesión de Cloud Shell. Sin embargo, si encuentra problemas de autenticación, puede seguir utilizándolo.

Creación de una imagen en Azure Container Registry

Puede generar la imagen de contenedor directamente en Azure a través de varios enfoques:

- Azure Cloud Shell permite construir la imagen completamente en la nube, independientemente del entorno local.
- Como alternativa, puede usar VS Code o la CLI de Azure para crearlo en Azure a partir de la configuración local, sin necesidad de que Docker se ejecute localmente.

Los comandos de la CLI de Azure se pueden ejecutar en el entorno de desarrollo local con la [CLI de Azure instalada](#) o en [Azure Cloud Shell](#).

1. En la consola, vaya a la carpeta raíz del repositorio clonado de la parte 2 de esta serie de tutoriales.
2. Compile la imagen de contenedor mediante el comando [az acr build](#).

```
Azure CLI

az acr build -r $REGISTRY_NAME -g $RESOURCE_GROUP_NAME -t
msdocpythoncontainerwebapp:latest .
# When using Azure Cloud Shell, run one of the following commands
instead:
# az acr build -r $REGISTRY_NAME -g $RESOURCE_GROUP_NAME -t
msdocpythoncontainerwebapp:latest https://github.com/Azure-
Samples/msdocs-python-django-container-web-app.git
# az acr build -r $REGISTRY_NAME -g $RESOURCE_GROUP_NAME -t
msdocpythoncontainerwebapp:latest https://github.com/Azure-
Samples/msdocs-python-flask-container-web-app.git
```

El último argumento del comando es la ruta completa al repositorio. Cuando se ejecuta en Azure Cloud Shell, use <https://github.com/Azure-Samples/msdocs-python-django-container-web-app.git> para la aplicación de ejemplo de Django y <https://github.com/Azure-Samples/msdocs-python-flask-container-web-app.git> para la aplicación de ejemplo de Flask.

3. Confirme que se creó la imagen de contenedor con el comando [az acr repository list](#).

```
Azure CLI

az acr repository list -n $REGISTRY_NAME
```

Paso siguiente

[Implementar una aplicación web](#)

Implementación de una aplicación de Python en contenedor en App Service

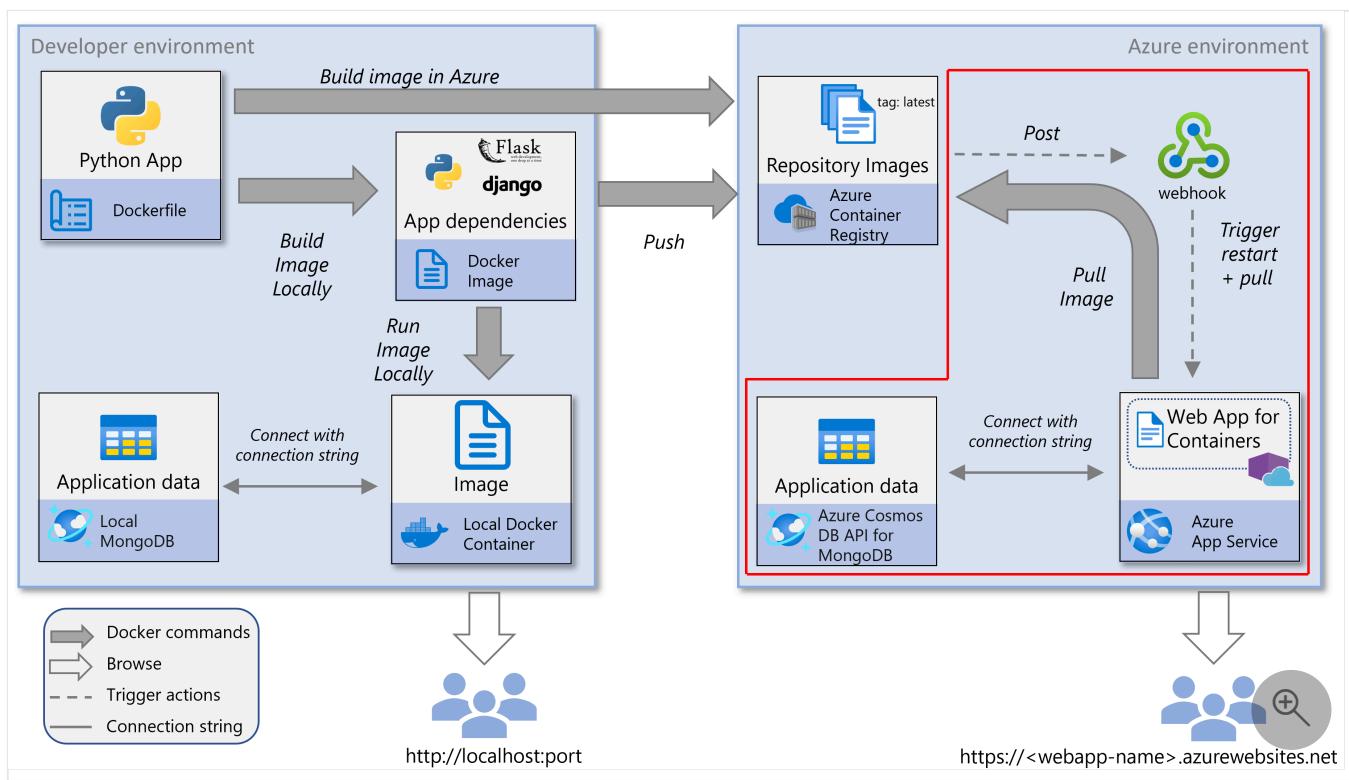
21/05/2025

En esta parte de la serie de tutoriales, aprenderá a implementar una aplicación web de Python en contenedor en [Azure App Service Web App for Containers](#). Este servicio totalmente administrado le permite ejecutar aplicaciones en contenedores sin tener que mantener su propio orquestador de contenedores.

App Service simplifica la implementación mediante canalizaciones de integración continua e implementación continua (CI/CD) que funcionan con Docker Hub, Azure Container Registry, Azure Key Vault y otras herramientas de DevOps. Este tutorial es la parte 4 de una serie de tutoriales de 5 partes.

Al final de este artículo, tiene una aplicación web de App Service segura y lista para producción que se ejecuta desde una imagen de contenedor de Docker. La aplicación usa una **identidad administrada asignada por el sistema** para extraer la imagen de Azure Container Registry y recuperar secretos de Azure Key Vault.

En este diagrama de servicio se resaltan los componentes descritos en este artículo.



CLI de Azure

Los comandos de la CLI de Azure se pueden ejecutar en [Azure Cloud Shell](#) o en una máquina local con la [CLI de Azure instalada](#).

ⓘ Importante

Se recomienda usar **Azure Cloud Shell** para todos los pasos basados en la CLI de este tutorial porque:

- Viene autenticado previamente con su cuenta de Azure, evitando problemas de inicio de sesión.
- Incluye todas las extensiones de la CLI de Azure necesarias de forma predeterminada.
- Garantiza un comportamiento coherente independientemente del sistema operativo local o del entorno.
- No requiere ninguna instalación local, ideal para los usuarios sin derechos de administrador
- Proporciona acceso directo a los servicios de Azure desde el portal: no se requiere ninguna configuración local de Docker o de red.
- Evita problemas de configuración de red o firewall local

Creación de Key Vault con autorización de RBAC

Azure Key Vault es un servicio seguro para almacenar secretos, claves de API, cadenas de conexión y certificados. En este script, este almacena la **cadena de conexión de MongoDB** y la **SECRET_KEY** aplicación web.

Key Vault está configurado para usar el **control de acceso basado en rol (RBAC)** para administrar el acceso a través de roles de Azure en lugar de directivas de acceso tradicionales. La aplicación web usa su **identidad administrada asignada por el sistema** para recuperar secretos de forma segura en tiempo de ejecución.

ⓘ Nota

La creación temprana de Key Vault garantiza que los roles se puedan asignar antes de cualquier intento de acceder a los secretos. También ayuda a evitar retrasos de propagación en las asignaciones de funciones. Dado que Key Vault no depende de App Service, aprovisionarlo temprano mejora la confiabilidad y la secuenciación.

1. En este paso, usará el comando `az keyvault create` para crear una instancia de Azure Key Vault con RBAC habilitado.

Juerga

Azure CLI

```
#!/bin/bash
RESOURCE_GROUP_NAME="msdocs-web-app-rg"
LOCATION="westus"
KEYVAULT_NAME="${RESOURCE_GROUP_NAME}-kv"

az keyvault create \
--name "$KEYVAULT_NAME" \
--resource-group "$RESOURCE_GROUP_NAME" \
--location "$LOCATION" \
--enable-rbac-authorization true
```

Creación del plan de App Service y la aplicación web

El plan de App Service define los recursos de proceso, el plan de tarifa y la región de la aplicación web. La aplicación web ejecuta la aplicación en contenedor y se aprovisiona con una identidad administrada asignada por el sistema que se usa para autenticarse de forma segura en Azure Container Registry (ACR) y Azure Key Vault.

En este paso, realizará las siguientes tareas:

- Crear un plan de App Service
- Creación de la aplicación web con su identidad administrada
- Configuración de la aplicación web para implementar mediante una imagen de contenedor específica
- Preparación para la implementación continua a través de ACR

⚠ Nota

La aplicación web debe crearse antes de asignar acceso a ACR o Key Vault porque la **identidad administrada solo se crea en el momento de la implementación**. Además, asignar la imagen de contenedor durante la creación garantiza que la aplicación se inicie correctamente con la configuración prevista.

1. En este paso, usará el comando `az appservice plan create` para aprovisionar el entorno de proceso para la aplicación.

Juerga

Azure CLI

```
#!/bin/bash
APP_SERVICE_PLAN_NAME="msdocs-web-app-plan"

az appservice plan create \
    --name "$APP_SERVICE_PLAN_NAME" \
    --resource-group "$RESOURCE_GROUP_NAME" \
    --sku B1 \
    --is-linux
```

2. En este paso, usará el comando `az webapp create` para crear la aplicación web. Este comando también habilita una **identidad administrada asignada por el sistema** y establece la imagen de contenedor que ejecuta la aplicación.

Juerga

Azure CLI

```
#!/bin/bash
APP_SERVICE_NAME="msdocs-website-name" #APP_SERVICE_NAME must be
globally unique as it becomes the website name in the URL
`https://<website-name>.azurewebsites.net`.
# Use the same registry name as in part 2 of this tutorial series.
REGISTRY_NAME="msdocscontainerregistryname" #REGISTRY_NAME is the
registry name you used in part 2 of this tutorial.
CONTAINER_NAME="$REGISTRY_NAME.azurecr.io/msdocspythoncontainerwebapp
:latest" #CONTAINER_NAME is of the form
"yourregistryname.azurecr.io/repo_name:tag".

az webapp create \
    --resource-group "$RESOURCE_GROUP_NAME" \
    --plan "$APP_SERVICE_PLAN_NAME" \
    --name "$APP_SERVICE_NAME" \
    --assign-identity '[system]' \
    --deployment-container-image-name "$CONTAINER_NAME"
```

① Nota

Al ejecutar este comando, es posible que vea el siguiente error:

Resultados

No credential was provided to access Azure Container Registry. Trying to look up...

Retrieving credentials failed with an exception:'Failed to retrieve container registry credentials. Please either provide the credentials or run 'az acr update -n msdocscontainerregistryname --admin-enabled true' to enable admin first.'

Este error se produce porque la aplicación web intenta usar credenciales de administrador para acceder a ACR, que las credenciales están deshabilitadas de forma predeterminada. Es seguro omitir este mensaje: el siguiente paso configura la aplicación web para usar su identidad administrada para autenticarse con ACR.

Concesión del rol del oficial de secretos al usuario que ha iniciado sesión

Para almacenar secretos en Azure Key Vault, el usuario que ejecuta el script debe tener el rol **De oficial de secretos de Key Vault**. Este rol permite crear y administrar datos confidenciales dentro de la bóveda.

En este paso, el script asigna ese rol al usuario que ha iniciado sesión actualmente. A continuación, este usuario puede almacenar de forma segura secretos de aplicación, como la cadena de conexión de MongoDB y la aplicación `SECRET_KEY`.

Esta asignación de roles es la primera de dos asignaciones de roles relacionadas con Key Vault. Más adelante, a la identidad administrada asignada por el sistema de la aplicación web se le concede acceso para recuperar secretos del almacén.

El uso de **Azure RBAC** garantiza un acceso seguro y auditible basado en la identidad, lo que elimina la necesidad de credenciales codificadas de forma rígida.

! Nota

El usuario debe tener asignado el rol de **Key Vault Secrets Officerantes** de intentar almacenar secretos en el Key Vault. Esta asignación se realiza mediante el comando [az role assignment create](#) en el ámbito del Key Vault.

1. En este paso, usará el comando [az role assignment create](#) para asignar el rol en el ámbito de Key Vault.

Azure CLI

```
#!/bin/bash
CALLER_ID=$(az ad signed-in-user show --query id -o tsv)
echo $CALLER_ID # Verify this value retrieved successfully. In
production, poll to verify this value is retrieved successfully.

az role assignment create \
--role "Key Vault Secrets Officer" \
--assignee "$CALLER_ID" \
--scope "/subscriptions/$(az account show --query id -o
tsv)/resourceGroups/$RESOURCE_GROUP_NAME/providers/Microsoft.KeyVault
/vaults/$KEYVAULT_NAME"
```

Concesión de acceso web a ACR mediante identidad administrada

Para extraer imágenes de Azure Container Registry (ACR) de forma segura, la aplicación web debe configurarse para usar su **identidad administrada asignada por el sistema**. El uso de identidad administrada evita la necesidad de credenciales de administrador y admite la implementación segura sin credenciales.

Este proceso implica dos acciones clave:

- Habilitación de la aplicación web para usar su identidad administrada al acceder a ACR
 - Asignación del rol **AcrPull** a esa identidad en el ACR de destino
1. En este paso, recuperará el **ID principal** (identificador de objeto único) de la identidad administrada de la aplicación web mediante el comando **az webapp identity show**. A continuación, habilitará el uso de la identidad administrada para la autenticación de ACR estableciendo la propiedad en **true** mediante **az webapp config set**. A continuación, asigne el rol **AcrPull** a la identidad administrada de la aplicación web mediante el comando **az role assignment create**. Este rol concede permiso a la aplicación web para extraer imágenes del registro.

```

#!/bin/bash
PRINCIPAL_ID=$(az webapp identity show \
    --name "$APP_SERVICE_NAME" \
    --resource-group "$RESOURCE_GROUP_NAME" \
    --query principalId \
    -o tsv)
echo $PRINCIPAL_ID # Verify this value retrieved successfully. In
production, poll for successful 'AcrPull' role assignment using `az
role assignment list`.

az webapp config set \
    --resource-group "$RESOURCE_GROUP_NAME" \
    --name "$APP_SERVICE_NAME" \
    --generic-configurations '{"acrUseManagedIdentityCreds": true}'

az role assignment create \
    --role "AcrPull" \
    --assignee "$PRINCIPAL_ID" \
    --scope "/subscriptions/$(az account show --query id -o
tsv)/resourceGroups/$RESOURCE_GROUP_NAME/providers/Microsoft.Containe
rRegistry/registries/$REGISTRY_NAME"

```

Conceder acceso a la bóveda de claves a la identidad administrada de la aplicación web

La aplicación web necesita permiso para acceder a secretos como la cadena de conexión de MongoDB y `SECRET_KEY`. Para conceder estos permisos, debe asignar el rol **Usuario de secretos de Key Vault** a la identidad administrada asignada por el sistema de la aplicación web.

1. En este paso, usará el ID principal de la identidad administrada asignada automáticamente por el sistema en la aplicación web para concederle acceso al Key Vault con el rol **Usuario de secretos de Key Vault**, mediante el comando `az role assignment create`.

Juerga

Azure CLI

```

#!/bin/bash

az role assignment create \
    --role "Key Vault Secrets User" \
    --assignee "$PRINCIPAL_ID" \

```

```
--scope "/subscriptions/$(az account show --query id -o tsv)/resourceGroups/$RESOURCE_GROUP_NAME/providers/Microsoft.KeyVault/vaults/$KEYVAULT_NAME"
```

Almacenamiento de secretos en Key Vault

Para evitar la codificación de secretos en la aplicación, este paso almacena la **cadena de conexión de MongoDB** y la **clave secreta** de la aplicación web en Azure Key Vault. A continuación, la aplicación web puede acceder a estos secretos de forma segura a través de su identidad administrada, sin necesidad de almacenar credenciales en el código o la configuración.

! Nota

Aunque en este tutorial solo se almacena la cadena de conexión y la clave secreta en el almacén de claves, también puede almacenar otras opciones de configuración de la aplicación, como el nombre de la base de datos de MongoDB o el nombre de la colección en Key Vault.

1. En este paso, usará el comando [az cosmosdb keys list](#) para recuperar la cadena de conexión de MongoDB. A continuación, use el comando [az keyvault secret set](#) para almacenar la cadena de conexión y una clave secreta generada aleatoriamente en Key Vault.

Juerga

Azure CLI

```
#!/bin/bash
ACCOUNT_NAME="msdocs-cosmos-db-account-name"

MONGO_CONNECTION_STRING=$(az cosmosdb keys list \
    --name "$ACCOUNT_NAME" \
    --resource-group "$RESOURCE_GROUP_NAME" \
    --type connection-strings \
    --query "connectionStrings[?description=='Primary MongoDB Connection String'].connectionString" -o tsv)

SECRET_KEY=$(openssl rand -base64 32 | tr -dc 'a-zA-Z0-9')
# This key is cryptographically secure, using OpenSSL's strong random number generator.

az keyvault secret set \
    --vault-name "$KEYVAULT_NAME" \
```

```
--name "MongoConnectionString" \
--value "$MONGO_CONNECTION_STRING"

az keyvault secret set \
--vault-name "$KEYVAULT_NAME" \
--name "MongoSecretKey" \
--value "$SECRET_KEY"
```

Configuración de la aplicación web para usar secretos de Kay Vault

Para acceder a secretos de forma segura en tiempo de ejecución, la aplicación web debe configurarse para hacer referencia a los secretos almacenados en Azure Key Vault. Este paso se realiza mediante **referencias de Key Vault**, que insertan los valores secretos en el entorno de la aplicación a través de su **identidad administrada asignada por el sistema**.

Este enfoque evita la codificación rígida de secretos y permite a la aplicación recuperar de forma segura valores confidenciales, como la cadena de conexión de MongoDB y la clave secreta durante la ejecución.

1. En este paso, usará el comando `az webapp config appsettings set` para agregar configuraciones de la aplicación que hacen referencia a los secretos del Key Vault. En concreto, establece las configuraciones de la `MongoConnectionString` y `MongoSecretKey` aplicación para hacer referencia a los secretos correspondientes almacenados en Key Vault.

Juerga

Azure CLI

```
#!/bin/bash
MONGODB_NAME="restaurants_reviews"
MONGODB_COLLECTION_NAME="restaurants_reviews"

az webapp config appsettings set \
--resource-group "$RESOURCE_GROUP_NAME" \
--name "$APP_SERVICE_NAME" \
--settings \

CONNECTION_STRING="@Microsoft.KeyVault(SecretUri=https://$KEYVAULT_NAME.vault.azure.net/secrets/MongoConnectionString)" \
SECRET_KEY="@Microsoft.KeyVault(SecretUri=https://$KEYVAULT_NAME.vault.azure.net/secrets/MongoSecretKey)" \
```

```
DB_NAME="$MONGODB_NAME" \
COLLECTION_NAME="$MONGODB_COLLECTION_NAME"
```

Habilitación de la implementación continua desde ACR

La habilitación de la implementación continua permite que la aplicación web extraiga y ejecute automáticamente la imagen de contenedor más reciente siempre que se inserte en Azure Container Registry (ACR). Esto reduce los pasos de implementación manuales y ayuda a garantizar que la aplicación permanezca actualizada.

! Nota

En el siguiente paso, registrarás un webhook en Azure Container Registry (ACR) para notificar a la aplicación web cuando se suba una nueva imagen.

1. En este paso, usará el comando [az webapp deployment container config](#) para habilitar la implementación continua desde ACR a la aplicación web.

Juerga

Azure CLI

```
#!/bin/bash
az webapp deployment container config \
--name "$APP_SERVICE_NAME" \
--resource-group "$RESOURCE_GROUP_NAME" \
--enable-cd true
```

Registra un webhook de ACR para la implementación continua

Para automatizar las implementaciones, registre un webhook en Azure Container Registry (ACR) que notifique la aplicación web cada vez que se inserte una nueva imagen de contenedor. El webhook permite que la aplicación extraiga y ejecute automáticamente la versión más reciente.

El webhook configurado en Azure Container Registry (ACR) envía una solicitud POST al punto de conexión SCM (SERVICE_URI) de la aplicación web cada vez que se inserta una nueva imagen en el repositorio msdocpythoncontainerwebapp. Esta acción desencadena la aplicación web para extraer e implementar la imagen actualizada, completando la canalización de implementación continua entre ACR y Azure App Service.

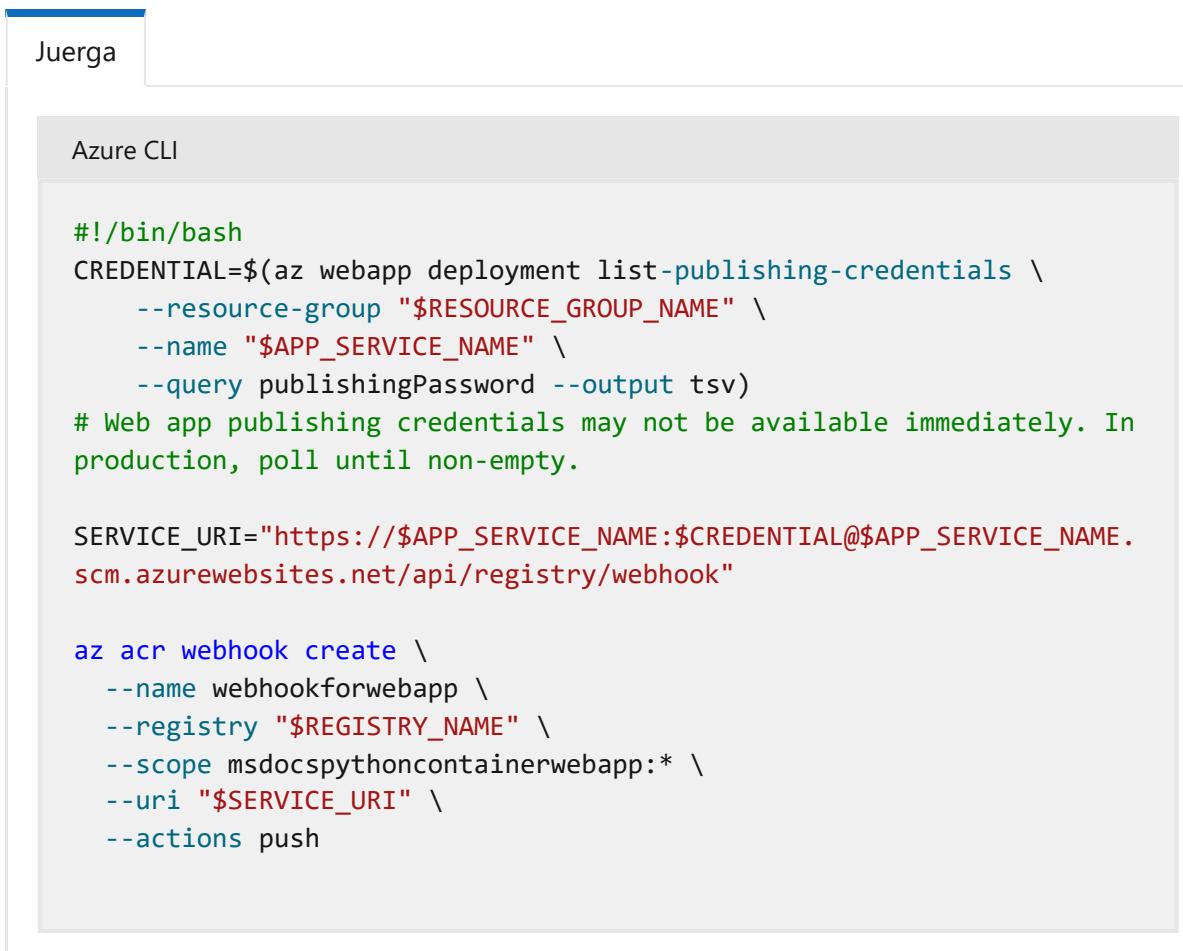
! Nota

El URI del webhook debe seguir este formato:

```
https://<app-name>.scm.azurewebsites.net/api/registry/webhook
```

Debe terminar con /api/registry/webhook. Si recibe un error de URI, confirme que la ruta de acceso es correcta.

1. En este paso, use el comando `az acr webhook create` para registrar el webhook y configurarlo para que se desencadene en `push` eventos.



```
#!/bin/bash
CREDENTIAL=$(az webapp deployment list-publishing-credentials \
    --resource-group "$RESOURCE_GROUP_NAME" \
    --name "$APP_SERVICE_NAME" \
    --query publishingPassword --output tsv)
# Web app publishing credentials may not be available immediately. In
# production, poll until non-empty.

SERVICE_URI="https://$APP_SERVICE_NAME:$CREDENTIAL@$APP_SERVICE_NAME.
scm.azurewebsites.net/api/registry/webhook"

az acr webhook create \
    --name webhookforwebapp \
    --registry "$REGISTRY_NAME" \
    --scope msdocpythoncontainerwebapp:* \
    --uri "$SERVICE_URI" \
    --actions push
```

Navegar por el sitio

Para comprobar que la aplicación web se está ejecutando, abra `https://<website-name>.azurewebsites.net` y reemplace `<website-name>` con el nombre de su Servicio de Aplicación. Deberías ver la aplicación de ejemplo de reseñas de restaurantes. Puede tardar unos instantes en cargarse la primera vez.

Una vez que aparezca el sitio, intente agregar un restaurante y envíe una revisión para confirmar que la aplicación funciona correctamente.

! Nota

El comando `az webapp browse` no se admite en Cloud Shell. Si usa Cloud Shell, abra manualmente un explorador y vaya a la dirección URL del sitio.

Si usa la CLI de Azure localmente, puede usar el comando `az webapp browse` para abrir el sitio en el explorador predeterminado:

Azure CLI

```
az webapp browse --name $APP_SERVICE_NAME --resource-group  
$RESOURCE_GROUP_NAME
```

! Nota

El comando `az webapp browse` no se admite en Cloud Shell. Abra una ventana del explorador y vaya a la dirección URL del sitio web en su lugar.

Solución de problemas de implementación

Si no ve la aplicación de ejemplo, pruebe los pasos siguientes.

- Con la implementación de contenedores y App Service, compruebe siempre la página **Centro de implementación / Registros** en Azure Portal. Confirme que el contenedor se ha extraído y se está ejecutando. La extracción inicial y la ejecución del contenedor pueden tardar unos instantes.
- Intente reiniciar App Service y compruebe si se resuelve el problema.
- Si hay errores de programación, esos errores se muestran en los registros de la aplicación. En la página del portal de Azure para App Service, seleccione **Diagnosticar y resolver problemas/Registros de aplicaciones**.

- La aplicación de ejemplo se basa en una conexión a Azure Cosmos DB para MongoDB. Confirme que App Service tiene la configuración de la aplicación con la información de conexión correcta.
- Confirme que la identidad administrada está habilitada para App Service y se usa en el Centro de implementación. En la página del portal Azure para el App Service, vaya al recurso App Service **Centro de implementación** y confirme que la **Autenticación** está establecida en **Identidad administrada**.
- Compruebe que el webhook está definido en Azure Container Registry. El webhook permite que App Service extraiga la imagen del contenedor. En concreto, compruebe que el URI del servicio termina con "/api/registry/webhook". Si no es así, agréguelo.
- [Los diferentes SKU de Azure Container Registry](#) tienen diferentes características, incluyendo el número de webhooks. Si reutiliza un registro existente, podría ver el mensaje "Cuota superada para webhooks de tipo de recurso para la SKU Básica del registro. Obtenga más información sobre las diferentes cuotas de SKU y el proceso de actualización: <https://aka.ms/acr/tiers>". Si ve este mensaje, utilice un nuevo registro o reduzca el número de [webhooks de registro](#) en uso.

Paso siguiente

[Limpieza de recursos](#)

Limpieza del tutorial de contenedorización y próximos pasos

Artículo • 21/04/2025

En esta parte de la serie de tutoriales, aprenderá a limpiar los recursos usados en Azure para que no incurra en otros cargos y ayude a mantener la suscripción de Azure desordenada.

Limpieza de recursos

Al final de un tutorial o proyecto, es importante limpiar los recursos de Azure que ya no necesite. Esto le ayuda a:

- Evitar cargos innecesarios: los recursos que quedan en ejecución pueden seguir acumulando costos.
- Mantener organizada la suscripción de Azure: la eliminación de recursos no utilizados facilita la administración y navegación por la suscripción.

En este tutorial, todos los recursos de Azure se crearon en el mismo grupo de recursos. Al quitar el grupo de recursos, se quitan todos los recursos del grupo de recursos y se trata de la manera más rápida de quitar todos los recursos de Azure usados para la aplicación.

Sugerencia

Si tiene previsto continuar con el desarrollo o las pruebas, puede dejar los recursos en funcionamiento. Tenga en cuenta los costos potenciales.

CLI de Azure

Los comandos de Azure CLI se pueden ejecutar en el [Azure Cloud Shell](#) o en una estación de trabajo con la CLI de Azure [instalada](#).

Elimine el grupo de recursos con el comando [az group delete](#).

Opcionalmente, puede agregar el argumento `--no-wait` para permitir que el comando vuelva antes de que se complete la operación.

Código de VS

Para trabajar con recursos de Azure en VS Code, debe tener instalado [el paquete de extensión de Azure Tools](#) y iniciar sesión en Azure desde VS Code.

1. En la vista de Azure en VS Code (desde la extensión Herramientas de Azure), expanda **RECURSOS** y busque la suscripción.
2. Asegúrese de que la vista está establecida en **Agrupar por grupo de recursos**.
3. Busque el grupo de recursos que desea eliminar.
4. Haga clic con el botón derecho en el grupo de recursos y seleccione **Eliminar grupo de recursos**.
5. En el cuadro de diálogo de confirmación, escriba el nombre exacto del grupo de recursos.
6. Presione **ENTRAR** para confirmar y eliminar el grupo de recursos.

Pasos siguientes

Después de completar este tutorial, estos son algunos pasos siguientes que puede seguir para desarrollar sobre lo que ha aprendido y mover el código del tutorial y la implementación más cerca de estar listo para producción:

- [Implementación de una aplicación web desde una instancia de Azure Container Registry con replicación geográfica](#)
- [Revisión de la seguridad en Azure Cosmos DB](#)
- Asigne un nombre DNS personalizado a su aplicación, consulte el [Tutorial: Asigne un nombre DNS personalizado a su aplicación](#).
- Supervise App Service para la disponibilidad, el rendimiento y el funcionamiento. Consulte [Supervisión de App Service](#) y [Configurar Azure Monitor para su aplicación Python](#).
- Habilite la implementación continua en Azure App Service, consulte [Implementación continua en Azure App Service](#), [Uso de CI/CD para implementar una aplicación web de Python en Azure App Service](#) en Linux y [Diseño de una canalización de CI/CD mediante Azure DevOps](#).
- Cree una infraestructura reutilizable como código con [CLI para Desarrolladores de Azure \(azd\)](#).

Módulos de Learn relacionados

A continuación se muestran algunos módulos de Learn que exploran las tecnologías y los temas tratados en este tutorial:

- [Introducción a python](#)
- [Introducción a Django](#)
- [Crear vistas y plantillas en Django](#)
- [Creación de sitios web controlados por datos mediante el marco de Python Django](#)
- [Implementación de una aplicación de Django en Azure mediante PostgreSQL](#)
- [Introducción a la API de MongoDB en Azure Cosmos DB](#)
- [migrar bases de datos locales de MongoDB a Azure Cosmos DB](#)
- [Compilación de una aplicación web en contenedor con Docker](#)

Implementación de una aplicación web de Flask o FastAPI en Azure Container Apps

18/06/2025

En este tutorial se muestra cómo incluir en contenedores una aplicación web de Python [Flask](#) o [FastAPI](#) e implementarla en [Azure Container Apps](#). Azure Container Apps usa la tecnología de contenedor [de Docker](#) para hospedar imágenes integradas e imágenes personalizadas. Para obtener más información sobre el uso de contenedores en Azure, consulte [Comparación de las opciones de contenedor de Azure](#).

En este tutorial, usará la [CLI de Docker](#) y la [CLI de Azure](#) para crear una imagen de Docker e implementarla en Azure Container Apps. También puede implementar con [Visual Studio Code](#) y la [extensión de Herramientas de Azure](#).

Prerrequisitos

Para completar este tutorial, necesita lo siguiente:

- Una cuenta de Azure donde puede implementar una aplicación web en [Azure Container Apps](#). (Se crea un área de trabajo de [Azure Container Registry](#) y [Log Analytics](#) automáticamente en el proceso).
- [CLI de Azure](#), [Docker](#) y la [CLI de Docker](#) instaladas en el entorno local.

Obtención del código de ejemplo

En el entorno local, obtenga el código.

```
Flask
Bash
git clone https://github.com/Azure-Samples/msdocs-python-flask-webapp-quickstart.git
```

Adición de archivos Dockerfile y .dockerignore

Agregue un *Dockerfile* para indicar a Docker cómo compilar la imagen. El *Dockerfile* especifica el uso de [Gunicorn](#), un servidor web de nivel de producción que reenvía las solicitudes web a

los marcos Flask y FastAPI. Los comandos ENTRYPPOINT y CMD indican a Gunicorn que controle las solicitudes del objeto de aplicación.

Flask

```
Dockerfile

# syntax=docker/dockerfile:1

FROM python:3.11

WORKDIR /code

COPY requirements.txt .

RUN pip3 install -r requirements.txt

COPY . .

EXPOSE 50505

ENTRYPOINT ["gunicorn", "app:app"]
```

50505 se usa para el puerto de contenedor (interno) en este ejemplo, pero puede usar cualquier puerto libre.

Compruebe el archivo *requirements.txt* para asegurarse de que contiene gunicorn.

Python

```
Flask==3.1.0
gunicorn
```

Configuración de gunicorn

Gunicorn se puede configurar con un archivo *gunicorn.conf.py*. Cuando el archivo *gunicorn.conf.py* se encuentra en el mismo directorio donde gunicorn se ejecuta, no es necesario especificar su ubicación en la ENTRYPPOINT instrucción ni CMD del Dockerfile. Para obtener más información sobre cómo especificar el archivo de configuración, consulte [Configuración de Gunicorn](#).

En este tutorial, el archivo de configuración sugerido configura Gunicorn para aumentar su número de trabajos en función del número de núcleos de CPU disponibles. Para obtener más información sobre la configuración de archivos *gunicorn.conf.py*, consulte [Configuración de Gunicorn](#).

text

```
# Gunicorn configuration file
import multiprocessing

max_requests = 1000
max_requests_jitter = 50

log_file = "-"

bind = "0.0.0.0:50505"

workers = (multiprocessing.cpu_count() * 2) + 1
threads = workers

timeout = 120
```

Agregue un archivo `.dockerignore` para excluir archivos innecesarios de la imagen.

`dockerignore`

```
.git*
**/*.pyc
.venv/
```

Compilación y ejecución local de la imagen

Compile la imagen localmente.

Flask

Bash

```
docker build --tag flask-demo .
```

Ejecute la imagen localmente en un contenedor de Docker.

Flask

Bash

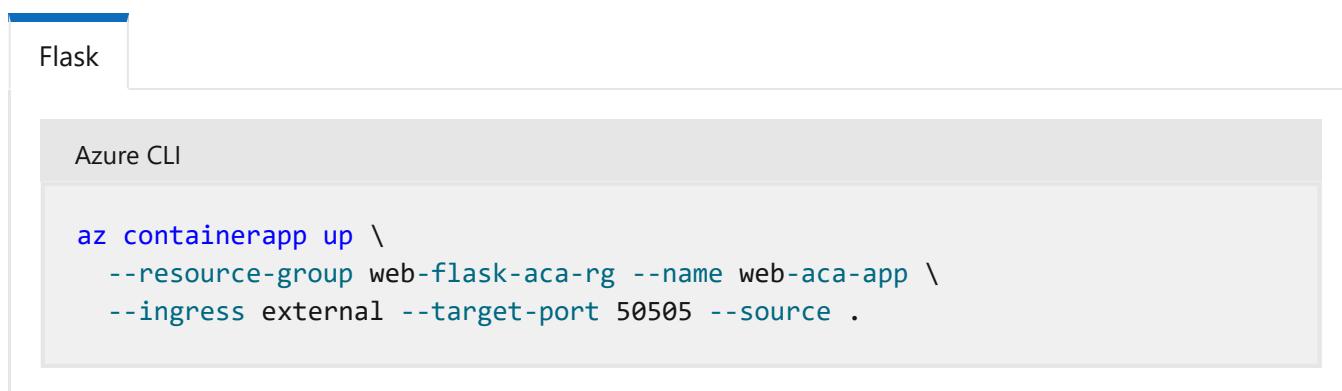
```
docker run --detach --publish 5000:50505 flask-demo
```

Abra la URL `http://localhost:5000` en el explorador para ver la aplicación web que se ejecuta localmente.

La opción `--detach` ejecuta el contenedor en segundo plano. La opción `--publish` asigna el puerto de contenedor a un puerto en el host. El puerto host (externo) es primero en el par y el puerto de contenedor (interno) es el segundo. Para más información, consulte [Referencia de ejecución de Docker](#).

Implementación de la aplicación web en Azure

Para implementar la imagen de Docker en Azure Container Apps, use el comando `az containerapp up`. (Se muestran los siguientes comandos para el shell de Bash. Cambie el carácter de continuación (`\`) según sea necesario para otros shells.



The screenshot shows a terminal window with a blue header bar. In the header bar, the word "Flask" is visible on the left. The main area of the terminal contains the following text:

```
Azure CLI

az containerapp up \
--resource-group web-flask-aca-rg --name web-aca-app \
--ingress external --target-port 50505 --source .
```

Cuando se complete la implementación, tendrá un grupo de recursos con los siguientes recursos dentro de ella:

- Instancia de Azure Container Registry
- Un entorno de Container Apps
- Una Container App que ejecuta la imagen de la aplicación web
- Un área de trabajo de Log Analytics

La dirección URL de la aplicación implementada se encuentra en la salida del `az containerapp up` comando. Abra la dirección URL en el explorador para ver la aplicación web que se ejecuta en Azure. El formato de la dirección URL será similar al siguiente `https://<generated-text>.<location-info>.azurecontainerapps.io`, donde `<generated-text>` y `<location-info>` son únicos para su implementación.

Actualizaciones y nuevas implementaciones

Después de realizar actualizaciones de código, puede volver a ejecutar el comando anterior `az containerapp up`, que vuelve a generar la imagen y la vuelve a implementar en Azure

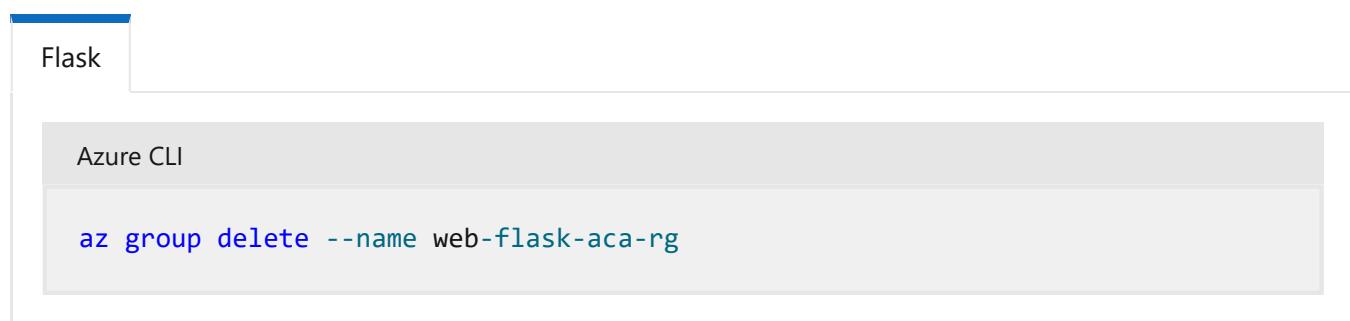
Container Apps. La ejecución del comando vuelve a tener en cuenta que el grupo de recursos y la aplicación ya existen y actualiza solo la aplicación contenedora.

En escenarios de actualización más complejos, puede volver a implementar con los comandos [az acr build](#) y [az containerapp update](#) juntos para actualizar la aplicación contenedora.

Limpieza

Todos los recursos de Azure creados en este tutorial están en el mismo grupo de recursos. Al quitar el grupo de recursos, se quitan todos los recursos del grupo de recursos y se trata de la manera más rápida de quitar todos los recursos de Azure usados para la aplicación.

Para quitar recursos, use el comando [az group delete](#).



A screenshot of a terminal window. The title bar says "Flask". The main area shows the command "az group delete --name web-flask-aca-rg" in blue text, indicating it is a clickable link.

También puede quitar el grupo en [Azure Portal](#) o en [Visual Studio Code](#) y la extensión de [Azure Tools](#).

Pasos siguientes

Para obtener más información, consulte los siguientes recursos:

- [Implementación de Azure Container Apps con el comando az containerapp up](#)
- [Inicio rápido: Implementación en Azure Container Apps mediante Visual Studio Code](#)
- [Extracción de imágenes de Azure Container Apps con identidad administrada](#)

Tutorial: Información general sobre los conceptos de implementación de una aplicación web de Python en Azure Container Apps

Artículo • 16/01/2025

En esta serie de tutoriales se muestra cómo incluir en contenedores una aplicación web de Python e implementarla en [Azure Container Apps](#). Una aplicación web de ejemplo está en contenedor y la imagen de Docker se almacena en [Azure Container Registry](#). Azure Container Apps está configurado para extraer la imagen de Docker de Container Registry y crear un contenedor. La aplicación de ejemplo se conecta a [azure Database for PostgreSQL](#) para demostrar la comunicación entre Container Apps y otros recursos de Azure.

Hay varias opciones para compilar e implementar aplicaciones web de Python nativas y en contenedores en la nube en Azure. En esta serie de tutoriales se describe Azure Container Apps. Container Apps es adecuado para ejecutar contenedores de uso general, especialmente para aplicaciones que abarcan muchos microservicios implementados en contenedores.

En esta serie de tutoriales, creará un contenedor. Para implementar una aplicación web de Python como contenedor en Azure App Service, consulte [aplicación web de Python en contenedor en Azure con MongoDB](#).

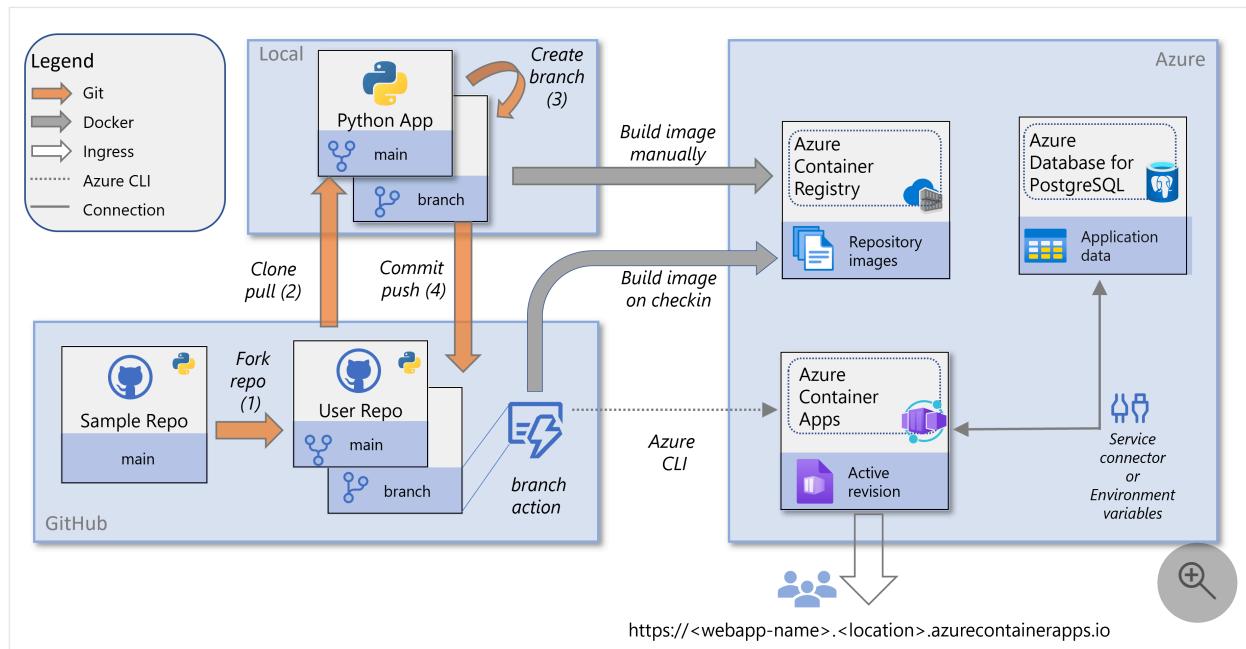
Los procedimientos de esta serie de tutoriales le guían para completar estas tareas:

- ✓ Compile una imagen de Docker desde una aplicación web de Python y almacene la imagen en Azure Container Registry.
- ✓ Configurar [Azure Container Apps](#) para hospedar la imagen de Docker.
- ✓ Configure las [acciones de GitHub](#) para actualizar el contenedor con una nueva imagen de Docker al detectar cambios en su repositorio de GitHub. *Este paso es opcional.*
- ✓ Configure la integración continua y la entrega continua (CI/CD) de una aplicación web de Python en Azure.

En esta primera parte de la serie, aprenderá conceptos básicos para implementar una aplicación web de Python en Azure Container Apps.

Introducción al servicio

En el diagrama siguiente se muestra cómo usará el entorno local, los repositorios de GitHub y los servicios de Azure de esta serie de tutoriales.



El diagrama incluye estos componentes:

- **Aplicaciones de Contenedores de Azure:**

Azure Container Apps permite ejecutar microservicios y aplicaciones en contenedores en una plataforma sin servidor. Una plataforma sin servidor significa que disfruta de las ventajas de ejecutar contenedores con una configuración mínima. Con Azure Container Apps, las aplicaciones se pueden escalar dinámicamente en función de características como el tráfico HTTP, el procesamiento controlado por eventos o la carga de cpu o memoria.

Container Apps extrae imágenes de Docker de Azure Container Registry. Los cambios en las imágenes de contenedor desencadenan una actualización en el contenedor implementado. También puede configurar Acciones de GitHub para desencadenar actualizaciones.

- **Azure Container Registry:**

Azure Container Registry permite trabajar con imágenes de Docker en Azure. Dado que Container Registry está cerca de las implementaciones en Azure, tiene control sobre el acceso. Puede usar los grupos y permisos de Microsoft Entra para controlar el acceso a las imágenes de Docker.

En esta serie de tutoriales, el origen del registro es Azure Container Registry. Pero también puede usar Docker Hub o un registro privado con modificaciones menores.

- [Base de datos de Azure para PostgreSQL](#):

El código de ejemplo almacena los datos de la aplicación en una base de datos PostgreSQL. La aplicación contenedora se conecta a PostgreSQL mediante una identidad administrada asignada por el usuario . La información de conexión se almacena en variables de entorno configuradas explícitamente o mediante un conector de servicio de Azure .

- [GitHub](#) :

El código de ejemplo de esta serie de tutoriales se encuentra en un repositorio de GitHub que puedes bifurcar y clonar localmente. Para configurar un flujo de trabajo de CI/CD con [GitHub Actions](#) , necesitáis una cuenta de GitHub.

Todavía puede seguir esta serie de tutoriales sin una cuenta de GitHub, si trabaja localmente o en [Azure Cloud Shell](#) para compilar la imagen de contenedor desde el repositorio de código de ejemplo.

Revisiones y CI/CD

Para realizar cambios en el código e insertarlos en un contenedor, cree una nueva imagen de Docker con los cambios. A continuación, inserte la imagen en Container Registry y cree una nueva revisión de la aplicación contenedora.

Para automatizar este proceso, un paso opcional de la serie de tutoriales muestra cómo construir un pipeline de CI/CD utilizando GitHub Actions. La canalización compila e implementa automáticamente el código en Container Apps cada vez que se inserta una nueva confirmación en el repositorio de GitHub.

Autenticación y seguridad

En esta serie de tutoriales, creará una imagen de contenedor de Docker directamente en Azure e la implementará en Azure Container Apps. Container Apps se ejecuta en el contexto de un entorno de , que es compatible con una red virtual de Azure . Las redes virtuales son un bloque de creación fundamental para la red privada en Azure. Container Apps permite exponer la aplicación contenedora a la web pública habilitando la entrada.

Para configurar CI/CD, autoriza Azure Container Apps como una aplicación OAuth [🔗](#) para tu cuenta de GitHub. Como aplicación de OAuth, Container Apps escribe un archivo de flujo de trabajo de Acciones de GitHub en el repositorio con información sobre los recursos y trabajos de Azure para actualizarlos. El flujo de trabajo actualiza los recursos de Azure mediante las credenciales de una entidad de servicio de Microsoft

Entra (o una existente) con acceso basado en roles para Container Apps y un nombre de usuario y contraseña para Azure Container Registry. Las credenciales se almacenan de forma segura en el repositorio de GitHub.

Por último, la aplicación web de ejemplo de esta serie de tutoriales almacena datos en una base de datos PostgreSQL. El código de ejemplo se conecta a PostgreSQL a través de una cadena de conexión. Cuando la aplicación se ejecuta en Azure, se conecta a la base de datos PostgreSQL mediante una identidad administrada asignada por el usuario. El código usa [DefaultAzureCredential](#) para actualizar dinámicamente la contraseña en la cadena de conexión con un token de acceso de Microsoft Entra durante el tiempo de ejecución. Este mecanismo evita la necesidad de codificar la contraseña en la cadena de conexión o en una variable de entorno, y proporciona una capa adicional de seguridad.

La serie de tutoriales le guía a través de la creación de la identidad administrada y la concesión de un rol y permisos de PostgreSQL adecuados para que pueda acceder a la base de datos y actualizarla. Durante la configuración de Container Apps, la serie de tutoriales le guía a través de la configuración de la identidad administrada en la aplicación y la configuración de variables de entorno que contienen información de conexión para la base de datos. También puede usar un conector de servicio de Azure para lograr lo mismo.

Prerrequisitos

Para completar esta serie de tutoriales, necesita:

- Una cuenta de Azure donde puede crear:
 - Una instancia de Azure Container Registry.
 - Un entorno de Azure Container Apps.
 - Una instancia de la base de datos de Azure para PostgreSQL.
- Visual Studio Code o la CLI de Azure , dependiendo de la herramienta que utilices:
 - Para Visual Studio Code, se necesita la [extensión de Container Apps](#) ↗.
 - Puede usar la CLI de Azure a través de [azure Cloud Shell](#).
- Paquetes de Python:
 - [psycopg2-binary](#) ↗ para conectarse a PostgreSQL.
 - [Flask](#) ↗ o [Django](#) ↗ como marco de trabajo web.

Aplicación de ejemplo

La aplicación de ejemplo de Python es una aplicación de revisión de restaurantes que guarda el restaurante y revisa los datos en PostgreSQL. Al final de la serie de tutoriales, tendrá una aplicación de revisión de restaurante implementada y en ejecución en Azure Container Apps que tiene un aspecto similar a la captura de pantalla siguiente.

The screenshot shows a web application titled "Azure Restaurant Review". At the top, there is a logo and a link to "Azure Docs". The main section displays a restaurant profile for "Contoso Café". It includes the street address (1 Main Street), a description ("Friendly coffee shop."), and a rating of 4.5 based on 2 reviews, accompanied by a star icon. Below this, a section titled "Reviews" contains a button to "Add new review". A table lists two reviews: one from Davide Sogese on 26/07/2022 at 14:46:54 rating 5, saying "Great cappuccino.", and another from Francesca Lombo on 26/07/2022 at 14:47:58 rating 4, saying "Healthy breakfast choices.". There is also a magnifying glass icon with a plus sign, likely for adding a new review.

Paso siguiente

Compilación e implementación de una aplicación web de Python con Azure Container Apps y PostgreSQL

Comentarios

¿Le ha resultado útil esta página?

Sí

No

[Proporcionar comentarios sobre el producto](#) | [Obtener ayuda en Microsoft Q&A](#)

Tutorial: Compilación e implementación de una aplicación web de Python con Azure Container Apps y PostgreSQL

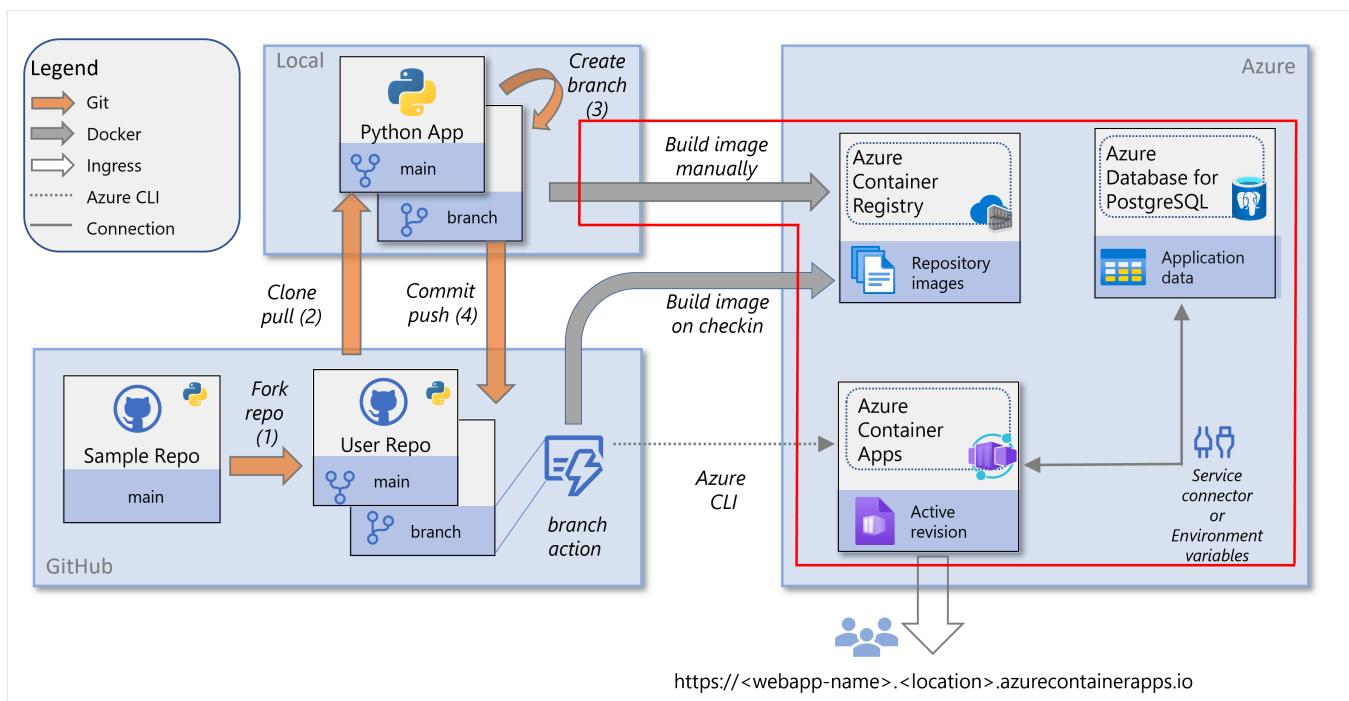
18/06/2025

Este artículo forma parte de una serie de tutoriales sobre cómo incluir e implementar una aplicación web de Python en [Azure Container Apps](#). Container Apps permite implementar aplicaciones contenerizadoras sin necesidad de administrar una infraestructura compleja.

En este tutorial, tú:

- ✓ Contenerizar un ejemplo de aplicación web de Python (Django o Flask) mediante la creación de una imagen de contenedor en la nube.
- ✓ Implemente la imagen de contenedor en Azure Container Apps.
- ✓ Defina variables de entorno que permitan que la aplicación contenedora se conecte a una instancia de [Azure Database for PostgreSQL](#): servidor flexible, donde la aplicación de ejemplo almacena los datos.

En el diagrama siguiente se resaltan las tareas de este tutorial: compilar e implementar una imagen de contenedor.



Prerrequisitos

Si no tiene una suscripción de Azure, cree una cuenta gratuita [»](#) antes de comenzar.

Puede ejecutar comandos de la CLI de Azure en [Azure Cloud Shell](#) o en una estación de trabajo con [la CLI de Azure](#) instalada.

Si se ejecuta localmente, siga estos pasos para iniciar sesión e instalar los módulos necesarios para este tutorial:

1. Inicie sesión en Azure y autentíquese, si es necesario:

```
Azure CLI
```

```
az login
```

2. Asegúrese de que ejecuta la versión más reciente de la CLI de Azure:

```
Azure CLI
```

```
az upgrade
```

3. Instale o actualice las extensiones de la CLI de Azure *containerapp* y *rdbms-connect* mediante el comando `az extension add`:

```
Azure CLI
```

```
az extension add --name containerapp --upgrade  
az extension add --name rdbms-connect --upgrade
```

Nota:

Para enumerar las extensiones instaladas en el sistema, puede usar el comando [az extension list](#). Por ejemplo:

```
Azure CLI
```

```
az extension list --query [].name --output tsv
```

Obtención de la aplicación de ejemplo

Hacer un fork y clonar el código de ejemplo en el entorno de desarrollo:

1. Vaya al repositorio GitHub de la aplicación de ejemplo ([Django](#) o [Flask](#)) y seleccione **Bifurcar**.

Sigue los pasos para hacer un fork del repositorio en tu cuenta de GitHub. También puede descargar el repositorio de código directamente en su máquina local sin necesidad de bifurcarlo ni usar una cuenta de GitHub. Pero si usa el método de descarga, no podrá configurar la integración continua y la entrega continua (CI/CD) en el siguiente tutorial de esta serie.

2. En el símbolo del sistema de la consola, use el comando [git clone](#) para clonar el repositorio bifurcado en la carpeta *python-container*:

```
Consola

# Django
git clone https://github.com/<github-username>/msdocs-python-django-azure-
container-apps.git python-container

# Flask
# git clone https://github.com/<github-username>/msdocs-python-flask-azure-
container-apps.git python-container
```

3. Cambie el directorio:

```
Consola

cd python-container
```

Construir una imagen de contenedor a partir del código de la aplicación web

Después de seguir estos pasos, tendrá una instancia de Azure Container Registry que contiene una imagen de contenedor de Docker compilada a partir del código de ejemplo.

[CLI de Azure](#) [Código de VS](#) [Azure Portal](#)

1. Si ejecuta comandos en un shell de Git Bash en un equipo Windows, escriba el siguiente comando antes de continuar:

```
Azure CLI

#!/bin/bash
```

```
export MSYS_NO_PATHCONV=1
```

2. Cree un grupo de recursos con el comando `az group create`:

Azure CLI

```
#!/bin/bash
RESOURCE_GROUP_NAME=<resource-group-name>
LOCATION=<location>
az group create \
    --name $RESOURCE_GROUP_NAME \
    --location $LOCATION
```

3. Cree un registro de contenedor mediante el comando `az acr create`:

Azure CLI

```
#!/bin/bash
REGISTRY_NAME=<registry-name> #The name that you use for *\\<registry-
name>* must be unique within Azure, and it must contain 5 to 50
alphanumeric characters.
az acr create \
    --resource-group $RESOURCE_GROUP_NAME \
    --name $REGISTRY_NAME \
    --sku Basic \
    --admin-enabled true
```

4. Inicie sesión en el Registro con el comando `az acr login` :

Azure CLI

```
az acr login --name $REGISTRY_NAME
```

El comando añade "azurecr.io" al nombre para crear el nombre de registro completo. Si el inicio de sesión se realiza correctamente, aparece el mensaje "Inicio de sesión correcto". Si está accediendo al registro desde una suscripción diferente de aquella en la que se creó el registro, use el comutador `--suffix`.

Si se produce un error en el inicio de sesión, asegúrese de que el demonio de Docker se está ejecutando en su sistema.

5. Compile la imagen mediante el comando `az acr build`:

Azure CLI

```
#!/bin/bash
az acr build \
```

```
--registry $REGISTRY_NAME \
--resource-group $RESOURCE_GROUP_NAME \
--image pythoncontainer:latest .
```

Estas consideraciones se aplican:

- El punto (.) al final del comando indica la ubicación del código fuente que se va a compilar. Si no ejecuta este comando en el directorio raíz de la aplicación de ejemplo, especifique la ruta de acceso al código.
- Si ejecuta el comando en Azure Cloud Shell, use `git clone` para extraer primero el repositorio en el entorno de Cloud Shell. A continuación, cambie el directorio a la raíz del proyecto para que el punto (.) se interprete correctamente.
- Si omite la opción `-t` (igual que `--image`), el comando pone en cola una compilación de contexto local sin enviarla al registro. Compilar sin enviar puede ser útil para comprobar que la imagen se compila.

6. Confirme que la imagen de contenedor se creó mediante el comando `az acr repository list`:

```
Azure CLI

az acr repository list --name $REGISTRY_NAME
```

Nota:

Los pasos de esta sección crean un registro de contenedor en el nivel de servicio Básico. Este nivel está optimizado para costos, con un conjunto de características y un rendimiento dirigidos a escenarios de desarrollador, y es adecuado para los requisitos de este tutorial. En escenarios de producción, lo más probable es que use el nivel de servicio Estándar o Premium. Estos niveles proporcionan niveles mejorados de almacenamiento y rendimiento.

Para obtener más información, consulte [Niveles de servicio de Azure Container Registry](#). Para obtener información sobre los precios, consulte los [Precios de Azure Container Registry](#).

Crear una instancia de servidor flexible de PostgreSQL.

La aplicación de ejemplo ([Django](#) o [Flask](#)) almacena los datos de revisión del restaurante en una base de datos PostgreSQL. En estos pasos, se crea el servidor que contendrá la base de

datos.

CLI de Azure

Código de VS

Azure Portal

1. Use el comando `az postgres flexible-server create` para crear el servidor PostgreSQL en Azure. No es raro que este comando se ejecute durante unos minutos antes de que finalice.

Azure CLI

```
#!/bin/bash
ADMIN_USERNAME=demoadmin
ADMIN_PASSWORD=<admin-password> # Use a strong password that meets the
                                requirements for PostgreSQL.
POSTGRES_SERVER_NAME=<postgres-server-name>
az postgres flexible-server create \
    --resource-group $RESOURCE_GROUP_NAME \
    --name $POSTGRES_SERVER_NAME \
    --location $LOCATION \
    --admin-user $ADMIN_USERNAME \
    --admin-password $ADMIN_PASSWORD \
    --version 16 \
    --tier Burstable \
    --sku-name Standard_B1ms \
    --public-access 0.0.0.0 \
    --microsoft-entra-auth Enabled \
    --storage-size 32 \
    --backup-retention 7 \
    --high-availability Disabled \
    --yes
```

Use estos valores:

- <*postgres-server-name*>: el nombre del servidor de base de datos PostgreSQL. Este nombre debe ser único en todas las instancias de Azure. El punto de conexión del servidor es `https://<postgres-server-name>.postgres.database.azure.com`. Los caracteres permitidos son de A a Z, 0 a 9 y el guion (-).
- <*ubicación*>: Use la misma ubicación que utilizaste para la aplicación web. <*location*> es uno de los valores de `Name` de ubicación de Azure de la salida del comando `az account list-locations -o table`.
- <*admin-username*>: el nombre de usuario de la cuenta de administrador. No puede ser `azure_superuser`, `admin`, `administrator`, `root`, `guest` o `public`. Usa

`demoadmin` para este tutorial.

- <contraseña del usuario administrador>: la contraseña del usuario administrador. Debe contener entre 8 y 128 caracteres de tres de las siguientes categorías: Letras del alfabeto inglés mayúsculas y minúsculas, números y caracteres no alfanuméricos.

Importante

Al crear nombres de usuario o contraseñas, *no* usar el carácter de signo de dólar (\$). Más adelante, al crear variables de entorno con estos valores, ese carácter tiene un significado especial dentro del contenedor de Linux que se usa para ejecutar aplicaciones de Python.

- `--version`: utiliza `16`. Especifica la versión de PostgreSQL que se va a usar para el servidor.
- `--tier`: utiliza `Burstable`. Especifica el plan de tarifa para el servidor. El nivel Elástico es una opción de menor costo para las cargas de trabajo que no necesitan utilizar la CPU al completo de forma continua, y es adecuada para los requisitos de este tutorial.
- `--sku-name`: El nombre del plan de tarifa y la configuración del proceso, por ejemplo, `Standard_B1ms`. Para más información, consulte los [precios de Azure Database for PostgreSQL](#). Para enumerar los niveles disponibles, use `az postgres flexible-server list-skus --location <location>`.
- `--public-access`: utiliza `0.0.0.0`. Permite el acceso público al servidor desde cualquier servicio de Azure, como Container Apps.
- `--microsoft-entra-auth`: utiliza `Enabled`. Habilita la autenticación de Microsoft Entra en el servidor.
- `--storage-size`: utiliza `32`. Especifica el tamaño de almacenamiento en gigabytes (GB) para el servidor. El mínimo es de 32 GB.
- `--backup-retention`: utiliza `7`. Especifica el número de días que se conservan las copias de seguridad del servidor. El mínimo es de 7 días.
- `--high-availability`: utiliza `Disabled`. Deshabilita la alta disponibilidad para el servidor. No se requiere alta disponibilidad para este tutorial.
- `--yes`: acepta los términos de uso del servidor PostgreSQL.

Nota:

Si tiene previsto trabajar con el servidor PostgreSQL desde la estación de trabajo local mediante herramientas, debe agregar una regla de firewall para la dirección IP de la estación de trabajo mediante el comando [az postgres flexible-server firewall-rule create](#).

2. Utilice el comando [az ad signed-in-user show](#) para obtener el ID de objeto de su cuenta de usuario. Use este identificador en el siguiente comando.

Azure CLI

```
#!/bin/bash
CALLER_OBJECT_ID=$(az ad signed-in-user show --query id -o tsv)
CALLER_DISPLAY_NAME=$(az ad signed-in-user show --query userPrincipalName
-o tsv)
```

3. Use el comando [az postgres flexible-server ad-admin create](#) para agregar su cuenta de usuario como administrador de Microsoft Entra en el servidor PostgreSQL.

Azure CLI

```
#!/bin/bash
az postgres flexible-server microsoft-entra-admin create \
--server-name "$POSTGRES_SERVER_NAME" \
--resource-group "$RESOURCE_GROUP_NAME" \
--display-name "$CALLER_DISPLAY_NAME" \
--object-id "$CALLER_OBJECT_ID" \
--type User
```

4. Use el comando [az postgres flexible-server firewall-rule create](#) para agregar una regla que permita a una aplicación web acceder al servidor flexible de PostgreSQL. En el comando siguiente, configurará el firewall del servidor para aceptar conexiones de la estación de trabajo de desarrollo mediante la dirección IP pública:

Azure CLI

```
MY_IP=$(curl -s ifconfig.me)
az postgres flexible-server firewall-rule create \
--name "$POSTGRES_SERVER_NAME" \
--resource-group "$RESOURCE_GROUP_NAME" \
--rule-name allow-my-ip \
--start-ip-address "$MY_IP" \
--end-ip-address "$MY_IP"
```
```

Nota:

Los pasos en esta sección le permiten crear un servidor PostgreSQL con un único vCore y memoria limitada en el nivel de precios ampliable. El nivel Elástico es una opción de menor costo para las cargas de trabajo que no necesitan utilizar la CPU al completo de forma continua, y es adecuada para los requisitos de este tutorial. En el caso de las cargas de trabajo de producción, puede actualizar al nivel de precios de uso general u optimizado para memoria. Estos niveles proporcionan un mayor rendimiento, pero aumentan los costos.

Para obtener más información, consulte [Opciones de proceso en Azure Database for PostgreSQL - Servidor flexible](#). Para obtener más información sobre los precios, consulte [Precios de Azure Database for PostgreSQL](#).

## Crear una base de datos en el servidor

En este punto, tiene un servidor PostgreSQL. En esta sección, se crea una base de datos en el servidor.

CLI de Azure

Código de VS

Azure Portal

Use el comando [az postgres flexible-server db create](#) para crear una base de datos denominada *restaurants\_reviews*:

```
Azure CLI

#!/bin/bash
DATABASE_NAME=restaurants_reviews
az postgres flexible-server db create \
 --resource-group $RESOURCE_GROUP_NAME \
 --server-name $POSTGRES_SERVER_NAME \
 --database-name $DATABASE_NAME
```

También podría usar el comando [az postgres flexible-server connect](#) para conectarse a la base de datos y luego trabajar con comandos [psql](#). Al trabajar con psql, a menudo es más fácil usar [Azure Cloud Shell](#), ya que el shell incluye todas las dependencias automáticamente.

También puede conectarse al servidor flexible de Azure Database for PostgreSQL y crear una base de datos mediante [psql](#) o un IDE que admita PostgreSQL, como [Azure Data Studio](#). Para conocer los pasos con psql, consulte [Configuración de la identidad administrada en la base de datos PostgreSQL](#) más adelante en este artículo.

# Creación de una identidad administrada asignada por el usuario

Cree una identidad administrada asignada por el usuario para usarla como identidad para la aplicación contenedora cuando se ejecute en Azure.

Nota:

Para crear una identidad administrada asignada por el usuario, la cuenta requiere la asignación del rol [Colaborador de identidades administradas](#).

CLI de Azure

Código de VS

Azure Portal

Utiliza el comando [az identity create](#) para crear una identidad gestionada asignada al usuario.

Azure CLI

```
UA_MANAGED_IDENTITY_NAME=<managed-identity-name> # Use a unique name for the
managed identity, such as - "my-ua-managed-id".
az identity create \
 --name $UA_MANAGED_IDENTITY_NAME
 --resource-group $RESOURCE_GROUP_NAME
```

## Configuración de la identidad administrada en la base de datos PostgreSQL

Configure la identidad administrada como rol en el servidor PostgreSQL y conceda los permisos necesarios para la base de datos de *restaurants\_reviews*. Ya sea que use la CLI de Azure o psql, debe conectarse al servidor de Azure PostgreSQL con un usuario que se haya configurado como administrador de Microsoft Entra en la instancia del servidor. Solo las cuentas de Microsoft Entra configuradas como administrador de PostgreSQL pueden configurar identidades administradas y otros roles de administrador de Microsoft en el servidor.

CLI de Azure

psql

1. Obtenga un token de acceso para la cuenta de Azure mediante el comando `az account get-access-token`. Use el token de acceso en los pasos siguientes.

#### Azure CLI

```
#!/bin/bash
MY_ACCESS_TOKEN=$(az account get-access-token --resource-type oss-rdbms -
--output tsv --query accessToken)
echo $MY_ACCESS_TOKEN
```

2. Agregue la identidad administrada asignada por el usuario como rol de base de datos en el servidor PostgreSQL mediante el comando [az postgres flexible-server execute](#).

#### Azure CLI

```
#!/bin/bash
az postgres flexible-server execute \
--name "$POSTGRES_SERVER_NAME" \
--admin-user "$CALLER_DISPLAY_NAME" \
--admin-password "$ACCESS_TOKEN" \
--database-name postgres \
--querytext "SELECT * FROM
pg_aadauth_create_principal('$UA_MANAGED_IDENTITY_NAME', false, false);"
```

Nota:

Si ejecuta el comando `az postgres flexible-server execute` en la estación de trabajo local, asegúrese de agregar una regla de firewall para la dirección IP de la estación de trabajo. Puede agregar una regla mediante el comando [az postgres flexible-server firewall-rule create](#). El mismo requisito también existe para el comando en el paso siguiente.

3. Conceda a la identidad administrada asignada por el usuario los permisos necesarios en la base de datos de *restaurants\_reviews* usando el siguiente comando [az postgres flexible-server execute](#):

#### Azure CLI

```
#!/bin/bash
SQL_GRANTS=$(cat <<EOF
GRANT CONNECT ON DATABASE $DATABASE_NAME TO "$UA_MANAGED_IDENTITY_NAME";
GRANT USAGE, CREATE ON SCHEMA public TO "$UA_MANAGED_IDENTITY_NAME";
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO
"$UA_MANAGED_IDENTITY_NAME";
ALTER DEFAULT PRIVILEGES IN SCHEMA public GRANT SELECT, INSERT, UPDATE,
DELETE ON TABLES TO "$UA_MANAGED_IDENTITY_NAME";
EOF
)

az postgres flexible-server execute \
```

```
--name "$POSTGRES_SERVER_NAME" \
--admin-user "$CALLER_DISPLAY_NAME" \
--admin-password "$MY_ACCESS_TOKEN" \
--database-name "$DATABASE_NAME" \
--querytext "$SQL_GRANTS"
```

Este comando de la CLI de Azure se conecta a la base de datos de *restaurants\_reviews* en el servidor y emite los siguientes comandos SQL:

SQL

```
GRANT CONNECT ON DATABASE restaurants_reviews TO "my-ua-managed-id";
GRANT USAGE ON SCHEMA public TO "my-ua-managed-id";
GRANT CREATE ON SCHEMA public TO "my-ua-managed-id";
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO "my-ua-managed-
id";
ALTER DEFAULT PRIVILEGES IN SCHEMA public
GRANT SELECT, INSERT, UPDATE, DELETE ON TABLES TO "my-ua-managed-id";
```

## Implementación de la aplicación web en Container Apps

Las aplicaciones de contenedor se implementan en entornos *de Azure Container Apps*, que actúan como límites seguros. En los siguientes pasos, crearás el entorno y un contenedor dentro de este. A continuación, configure el contenedor para que el sitio web sea visible externamente.

CLI de Azure

Código de VS

Azure Portal

Estos pasos requieren la extensión Azure Container Apps, *containerapp*.

1. Cree un entorno de Container Apps con el comando [az containerapp env create](#):

Azure CLI

```
#!/bin/bash
APP_ENV_NAME=<app-env-name> # Use a unique name for the environment, such
as "python-container-env".
az containerapp env create \
--name python-container-env \
--resource-group $RESOURCE_GROUP_NAME \
--location $LOCATION
```

2. Obtenga las credenciales de inicio de sesión de la instancia de Azure Container Registry mediante el comando [az acr credential show](#):

```
Azure CLI

#!/bin/bash
REGISTRY_CREDS=$(az acr credential show -n "$REGISTRY_NAME" --query "[username,passwords[0].value]" -o tsv)
REGISTRY_USERNAME=$(echo "$REGISTRY_CREDS" | head -n1)
REGISTRY_PASSWORD=$(echo "$REGISTRY_CREDS" | tail -n1)
```

Use el nombre de usuario y una de las contraseñas devueltas desde la salida del comando al crear la aplicación contenedora en el paso 5.

3. Use el comando [az identity show](#) para obtener el identificador de cliente y el identificador de recurso de la identidad administrada asignada por el usuario:

```
Azure CLI

UA_CLIENT_ID=$(az identity show \
 --name "$UA_MANAGED_IDENTITY_NAME" \
 --resource-group "$RESOURCE_GROUP" \
 --query clientId -o tsv)
UA_RESOURCE_ID=$(az identity show \
 --name "$UA_MANAGED_IDENTITY_NAME" \
 --resource-group "$RESOURCE_GROUP" \
 --query id -o tsv)
```

Use el valor del identificador de cliente (GUID) y el identificador de recurso de la salida del comando al crear la aplicación contenedora en el paso 5. El identificador de recurso tiene el siguiente formato: /subscriptions/<subscription-id>/resourcegroups/pythoncontainer-rg/providers/Microsoft.ManagedIdentity/userAssignedIdentities/my-ua-managed-id.

4. Ejecute el siguiente comando para generar un valor de clave secreta:

```
Azure CLI

AZURE_SECRET_KEY=$(python -c 'import secrets;
print(secrets.token_hex())')
```

Use el valor de clave secreta para establecer una variable de entorno al crear la aplicación contenedora en el paso 5.

Nota:

El comando que muestra este paso es para un shell de Bash. En función del entorno, es posible que tenga que invocar Python mediante `python3`. En Windows, debe incluir el comando en el parámetro `-c` entre comillas dobles en lugar de comillas simples. También es posible que tenga que invocar Python mediante `py` o `py -3`, en función de su entorno.

5. Crea una aplicación contenedora en el entorno con el comando [az containerapp create](#):

```
Azure CLI

az containerapp create \
--name "$CONTAINER_APP_NAME" \
--resource-group "$RESOURCE_GROUP" \
--environment "$APP_ENV" \
--image "$REGISTRY_NAME.azurecr.io/$IMAGE_NAME" \
--target-port "$TARGET_PORT" \
--ingress external \
--registry-server "$REGISTRY_NAME.azurecr.io" \
--registry-username "$REGISTRY_USERNAME" \
--registry-password "$REGISTRY_PASSWORD" \
--user-assigned "$UA_RESOURCE_ID" \
--env-vars \
 DBHOST="$POSTGRES_SERVER_NAME" \
 DBNAME="$DATABASE_NAME" \
 DBUSER="$UA_MANAGED_IDENTITY_NAME" \
 RUNNING_IN_PRODUCTION=1 \
 AZURE_CLIENT_ID="$UA_CLIENT_ID" \
 AZURE_SECRET_KEY="$AZURE_SECRET_KEY"
```
```

6. Para Django solamente, migre y cree un esquema de base de datos. (En la app de ejemplo Flask, se hace automáticamente, y puede saltarse este paso).

Conéctese utilizando el comando [az containerapp exec](#).

```
Azure CLI

az containerapp exec \
  --name $CONTAINER_APP_NAME \
  --resource-group $RESOURCE_GROUP_NAME
```

A continuación, en la línea de comandos del shell, introduzca `python manage.py migrate`.

No es necesario migrar para las revisiones del contenedor.

7. Pruebe el sitio web.

El comando `az containerapp create` que escribió anteriormente genera una dirección URL de aplicación que puede usar para ir a la aplicación. La dirección URL termina en `azurecontainerapps.io`. Vaya a la dirección URL en un explorador. También puede usar el comando `az containerapp browse`.

Este es un ejemplo del sitio web de ejemplo después de la adición de un restaurante y dos opiniones.

The screenshot shows a web application titled "Azure Restaurant Review". At the top left is the logo and title. On the right is a menu icon (three horizontal lines). Below the title, the word "Restaurants" is displayed. A table lists one restaurant: "Contoso Café" with a rating of "4.0 (2 reviews)". To the right of the rating is a blue "Details" button. At the bottom of the table is a green "Add new restaurant" button. The entire interface is contained within a white box with rounded corners.

Solución de problemas de implementación

Olvidó la dirección URL de la aplicación para acceder al sitio web.

En Azure Portal:

- Vaya a la página **Información general** de la aplicación de contenedor y busque **URL de aplicación**.

En VS Code:

1. Vaya a **Vista de Azure** (Ctrl+Shift+A) y expanda la suscripción en la que está trabajando.
2. Expanda el nodo **Container Apps**, luego expanda el entorno administrado y haga clic con el botón derecho en **python-container-app** y seleccione **Examinar**. VS Code abre el explorador con la dirección URL de la aplicación.

En la CLI de Azure:

- Use el comando `az containerapp show -g pythoncontainer-rg -n python-container-app -query properties.configuration.ingress.fqdn`.

En VS Code, la tarea Compilar imagen en Azure devuelve un error.

Si ve el mensaje "Error: no se pudo descargar el contexto. Compruebe si la dirección URL es incorrecta en la ventana de salida de **VS Code** y actualice el registro en la extensión de Docker. Para actualizar, seleccione la extensión de Docker, vaya a la sección **Registros**, busque el registro y selecciónelo.

Si vuelve a ejecutar la tarea **Compilar imagen en Azure**, compruebe si existe el registro de una ejecución anterior. Si es así, úselo.

En Azure Portal, aparece un error de acceso durante la creación de una aplicación de contenedor.

Se produce un error de acceso con el mensaje "No se puede acceder al ACR '<>.azurecr.io'" si se deshabilitan las credenciales de administrador en una instancia de Azure Container Registry.

Para comprobar el estado del administrador en el portal, vaya a la instancia de Azure Container Registry, seleccione el recurso de **Claves de acceso** y asegúrese de que **usuario administrador** esté habilitado.

La imagen de contenedor no aparece en la instancia de Azure Container Registry

- Compruebe la salida del comando CLI de Azure o salida de VS Code y busque mensajes que se ha realizado correctamente.
- Compruebe que el nombre del registro se ha especificado correctamente en el comando de compilación con la CLI de Azure o en los mensajes de tarea de VS Code.
- Asegúrese de que sus credenciales no hayan caducado. Por ejemplo, en VS Code, busque el registro de destino en la extensión de Docker y actualícelo. En la CLI de Azure, ejecute `az login`.

El sitio web devuelve "Solicitud incorrecta (400)"

Si recibe un error de "Solicitud incorrecta (400)", compruebe las variables de entorno de PostgreSQL pasadas al contenedor. El error 400 a menudo indica que el código Python no puede conectarse a la instancia PostgreSQL.

El código de ejemplo usado en este tutorial comprueba la existencia de la variable de entorno de contenedor `RUNNING_IN_PRODUCTION`, que se puede establecer en cualquier valor (como `1`).

El sitio web devuelve "No encontrado (404)"

- Compruebe el valor de **URL de aplicación** en la página de **Información general** del contenedor. Si la dirección URL de la aplicación contiene la palabra "internal", el acceso no se establece correctamente.
- Compruebe la entrada del contenedor. Por ejemplo, en Azure Portal, vaya al recurso **Entrada** del contenedor. Asegúrese de que **Entrada HTTP** está habilitado y **Aceptar tráfico desde cualquier lugar** esté seleccionado.

El sitio web no se inicia, recibes "tiempo de espera de transmisión" o no se devuelve nada.

- Compruebe los registros:
 - En Azure Portal, vaya al recurso de administración de revisiones de la aplicación contenedora y compruebe **Estado de aprovisionamiento** para el contenedor:
 - Si el estado es **Aprovisionamiento**, espere hasta que finalice el aprovisionamiento.
 - Si el estado es **Error**, seleccione la revisión y vea los registros de la consola. Elija el orden de las columnas para mostrar **Hora de generación**, **Stream_s** y **Log_s**. Ordene los registros por más reciente y busque los mensajes `stderr` de Python y `stdout` en la columna **Stream_s**. La salida de `print` de Python son mensajes `stdout`.
 - En la CLI de Azure, use el comando `az containerapp logs show`.
- Si usa el marco django, compruebe si las tablas de *restaurants_reviews* existen en la base de datos. Si no es así, use una consola para acceder al contenedor y ejecute `python manage.py migrate`.

Paso siguiente

[Configuración de la implementación continua para una aplicación web de Python en Azure Container Apps](#)

Tutorial: Configuración de la implementación continua para una aplicación web de Python en Azure Container Apps

18/06/2025

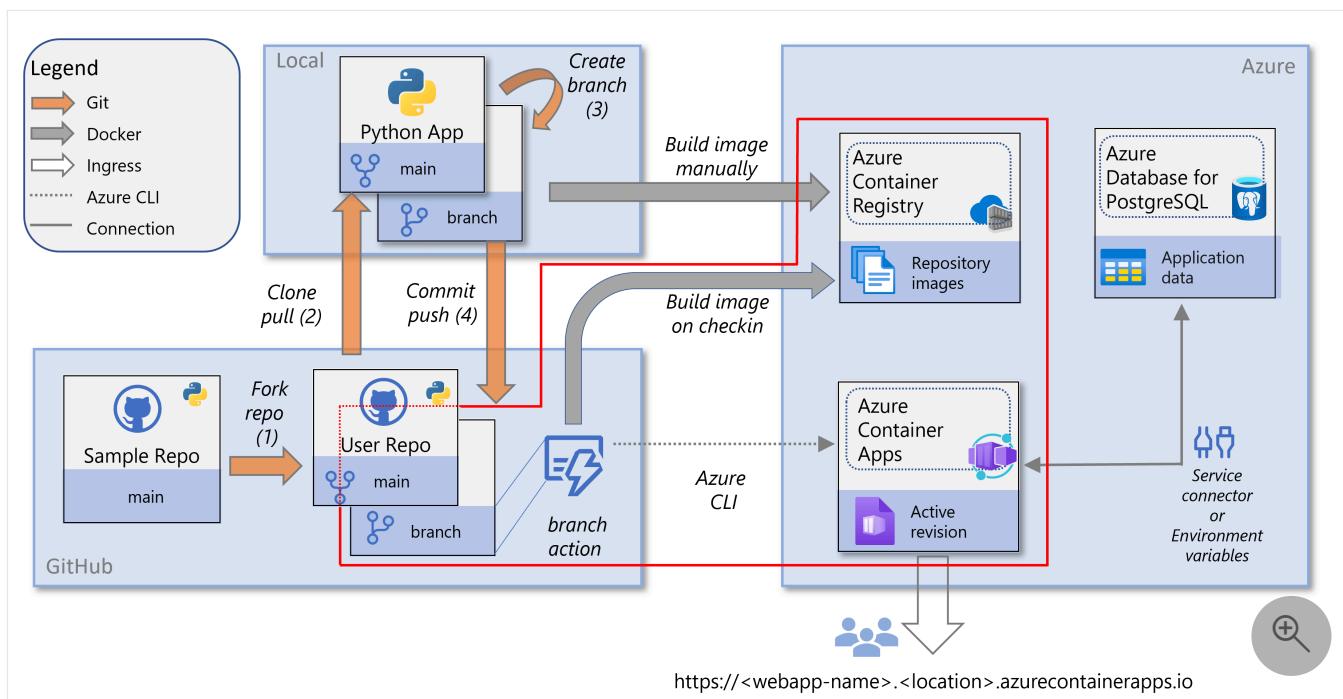
Este artículo forma parte de una serie de tutoriales sobre cómo incluir e implementar una aplicación web de Python en [Azure Container Apps](#). Container Apps permite implementar aplicaciones en contenedor sin administrar infraestructura compleja.

En este tutorial ha:

- ✓ Configura la implementación continua de una aplicación contenedora mediante un flujo de trabajo de GitHub Actions .
- ✓ Realice un cambio en una copia del repositorio de ejemplo para desencadenar el flujo de trabajo de las acciones de GitHub.
- ✓ Resuelva los problemas que pueda encontrarse con la configuración de la implementación continua.
- ✓ Quite los recursos que no necesite al finalizar la serie de tutoriales.

La implementación continua está relacionada con la práctica de DevOps de integración continua y entrega continua (CI/CD), que es la automatización del flujo de trabajo de desarrollo de aplicaciones.

En el diagrama siguiente se resaltan las tareas de este tutorial.



Prerrequisitos

Para configurar la implementación continua, necesita:

- Los recursos (y su configuración) que creó en el [tutorial anterior](#), que incluye una instancia de [Azure Container Registry](#) y una aplicación de contenedor de [Azure Container Apps](#).
- Una cuenta de GitHub en la que bifurcaste el código de ejemplo ([Django](#) o [Flask](#)) y a la que se pueda conectar desde Azure Container Apps. (Si descargó el código de ejemplo en lugar de bifurcar, asegúrese de subir el repositorio local a su cuenta de GitHub).
- Opcionalmente, tener [Git](#) instalado en el entorno de desarrollo para realizar cambios en el código y subirlos al repositorio en GitHub. Como alternativa, puede realizar los cambios directamente en GitHub.

Configuración de la implementación continua para un contenedor

En el tutorial anterior, creó y configuró una aplicación de contenedor en Azure Container Apps. Parte de la configuración estaba extrayendo una imagen de Docker de una instancia de Azure Container Registry. La imagen de contenedor se extrae del registro cuando se crea un contenedor [revisión](#), como cuando se configura por primera vez la aplicación contenedora.

En esta sección, configurará la implementación continua mediante un flujo de trabajo de Acciones de GitHub. Con la implementación continua, se crea una nueva imagen de Docker y una revisión de contenedor en función de un desencadenador. El desencadenador de este tutorial es cualquier cambio en la rama *principal* del repositorio, como con una solicitud de incorporación de cambios. Cuando se desencadena el flujo de trabajo, crea una nueva imagen de Docker, la inserta en la instancia de Azure Container Registry y actualiza la aplicación contenedora a una nueva revisión mediante la nueva imagen.

CLI de Azure

Puede ejecutar los comandos de la CLI de Azure en [Azure Cloud Shell](#) o en una estación de trabajo con la [CLI de Azure](#) instalada.

Si ejecuta comandos en un shell de Git Bash en un equipo Windows, escriba el siguiente comando antes de continuar:

Bash

```
export MSYS_NO_PATHCONV=1
```

1. Cree una entidad de servicio mediante el comando `az ad sp create-for-rbac`:

Azure CLI

```
az ad sp create-for-rbac \
--name <app-name> \
--role Contributor \
--scopes "/subscriptions/<subscription-ID>/resourceGroups/<resource-group-name>"
```

En el comando :

- `<app-name>` es un nombre para mostrar opcional para la entidad de servicio. Si se omite la opción `--name`, se genera un GUID como nombre para mostrar.
- `<ID de suscripción>` es el GUID que identifica de forma única tu suscripción en Azure. Si no conoce su ID de suscripción, puede ejecutar el comando `az account show` y copiarlo desde la propiedad `id` en la salida.
- `<resource-group-name>` es el nombre de un grupo de recursos que contiene el contenedor de Azure Container Apps. El control de acceso basado en rol (RBAC) está en el nivel de grupo de recursos. Si ha seguido los pasos del tutorial anterior, el nombre del grupo de recursos es `pythoncontainer-rg`.

Guarde la salida de este comando para el paso siguiente. En concreto, guarde el identificador de cliente (propiedad `appId`), el secreto de cliente (propiedad `password`) y el ID de tenant (propiedad `tenant`).

2. Configure acciones de GitHub mediante el comando `az containerapp github-action add`:

Azure CLI

```
az containerapp github-action add \
--resource-group <resource-group-name> \
--name python-container-app \
--repo-url <https://github.com/userid/repo> \
--branch main \
--registry-url <registry-name>.azurecr.io \
--service-principal-client-id <client-id> \
--service-principal-tenant-id <tenant-id> \
--service-principal-client-secret <client-secret> \
--login-with-github
```

En el comando :

- <resource-group-name> es el nombre del grupo de recursos. En este tutorial, es `pythoncontainer-rg`.
- <<https://github.com/userid/repo>> es la dirección URL del repositorio de GitHub. En este tutorial, escoja entre `https://github.com/userid/msdocs-python-django-azure-container-apps` o `https://github.com/userid/msdocs-python-flask-azure-container-apps`. En esas direcciones URL, `userid` es el identificador de usuario de GitHub.
- <nombre del registro> es la instancia de Azure Container Registry existente que creó en el tutorial anterior o una que puede usar.
- <> client-id es el valor de la propiedad `appId` del comando `az ad sp create-for-rbac` anterior. El Id. es un GUID con el formato `00000000-0000-0000-0000-00000000`.
- <tenant-id> es el valor de la propiedad `tenant` del comando `az ad sp create-for-rbac` anterior. El ID también es un GUID, similar al identificador del cliente.
- <client-secret> es el valor de la propiedad `password` del comando `az ad sp create-for-rbac` anterior.

En la configuración de la implementación continua, una entidad de servicio [llamada](#) permite que GitHub Actions acceda a los recursos de Azure y los modifique. Los roles asignados al principal de servicio restringen el acceso a los recursos. A la entidad de servicio se le asignó el rol integrado de [Colaborador](#) en el grupo de recursos que contiene la aplicación de contenedor.

Si ha seguido los pasos del portal, la entidad de servicio se habrá creado automáticamente. Si has seguido los pasos de la CLI de Azure, has creado explícitamente el principal de servicio antes de configurar la implementación continua.

Reimplementación de la aplicación web con Acciones de GitHub

En esta sección, realizas un cambio en la copia clonada del repositorio de ejemplo. Después, puedes confirmar que el cambio se implementa automáticamente en el sitio web.

Si aún no lo ha hecho, realice una [bifurcación](#) del repositorio de ejemplo ([Django](#) o [Flask](#)). Puede realizar cambios en su código directamente en [GitHub](#) o en su entorno de desarrollo desde la línea de comandos con [Git](#).

1. Vaya a la bifurcación del repositorio de ejemplo e inicie en la rama *principal*.

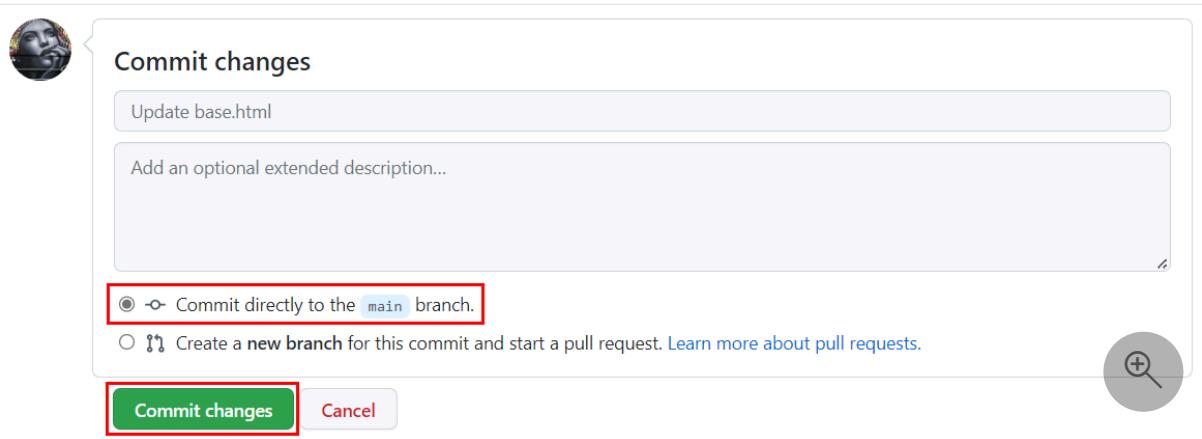
The screenshot shows a GitHub repository page for 'username / msdocs-python-flask-azure-container-app'. The 'Code' tab is selected, and the 'main' branch dropdown is highlighted with a red box. The repository description states: 'A Python/Flask sample web app targeting deployment in Azure Container Apps.' It includes links to 'Readme' and 'MIT license'.

2. Realice un cambio:

- Vaya al archivo */templates/base.html*. (Para Django, la ruta es *restaurant_review/templates/restaurant_review/base.html*).
- Seleccione **Editar** y cambie la frase `Azure Restaurant Review` a `Azure Restaurant Review - Redeployed`.

The screenshot shows the GitHub file editor for 'base.html'. The 'Edit' button is highlighted with a red box. The code has been modified at line 17 to read: `Azure Restaurant Review - Redeployed`.

3. En la parte inferior de la página que está editando, asegúrese de que **Confirmar directamente en la rama principal** está seleccionado. A continuación, seleccione el botón **Confirmar cambios**.



La confirmación inicia el flujo de trabajo de Acciones de GitHub.

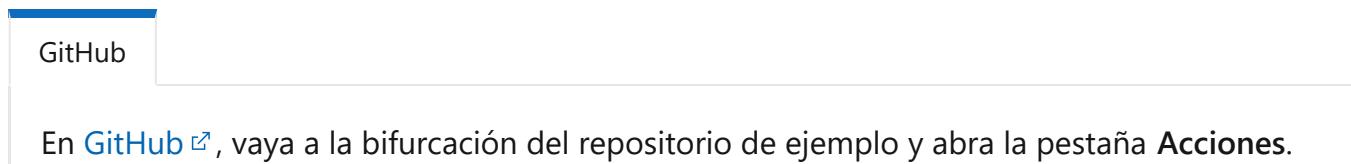
ⓘ Nota

En este tutorial se muestra cómo realizar un cambio directamente en la rama *principal*. En los flujos de trabajo de software típicos, se realiza un cambio en una rama distinta de *principal* y, a continuación, se crea un pull request para combinar el cambio en *principal*. Las solicitudes de incorporación de cambios también inician el flujo de trabajo.

Descripción de acciones de GitHub

Visualización del historial de flujos de trabajo

Si necesita ver el historial de flujos de trabajo, use uno de los procedimientos siguientes.



En [GitHub](#), vaya a la bifurcación del repositorio de ejemplo y abra la pestaña **Acciones**.

The screenshot shows the GitHub Actions page for a repository named 'username/msdocs-python-flask-azure-container-app'. The 'Actions' tab is selected, highlighted with a red box. Below it, the 'All workflows' section shows a single workflow named 'Update base.html' with a red box around it. The workflow run details show it was triggered by 'Trigger auto deployment for ...' and pushed by 'username'. The status is 'In progress' and it was updated 2 minutes ago.

Secretos de flujo de trabajo

El archivo de flujo de trabajo `.github/workflows/<workflow-name>.yml` que se agregó al repositorio incluye marcadores de posición para las credenciales necesarias para los trabajos de creación y actualización de la aplicación de contenedor del flujo de trabajo. La información de credenciales se almacena cifrada en el área **Configuración**, en **Seguridad>Secretos y variables>Acciones**.

The screenshot shows the 'Repository secrets' page for the same repository. The left sidebar has a red box around the 'Actions' section. The main table lists three secrets:

Name	Last updated	Actions
PYTHONCONTAINERAPP_AZURE_CREDENTIALS	2 hours ago	
PYTHONCONTAINERAPP_REGISTRY_PASSWORD	2 hours ago	
PYTHONCONTAINERAPP_REGISTRY_USERNAME	2 hours ago	

Si cambia la información de credenciales, puede actualizarla aquí. Por ejemplo, si se vuelven a generar las contraseñas de Azure Container Registry, debe actualizar el valor de `REGISTRY_PASSWORD`. Para obtener más información, consulte [secretos cifrados](#) en la documentación de GitHub.

Aplicaciones autorizadas de OAuth

Al configurar la implementación continua, designa Azure Container Apps como una aplicación de OAuth autorizada para su cuenta de GitHub. Container Apps usa el acceso autorizado para crear un archivo YAML de acciones de GitHub en `.github/workflows/<workflow-name>.yml`. Puede ver las aplicaciones autorizadas y revocar permisos en su cuenta, en **Integrations>Applications**.

The screenshot shows the GitHub 'Applications' page. On the left sidebar, there are sections for Public profile, Account, Appearance, Accessibility, Notifications, Access, Billing and plans, Emails, Password and authentication, and SSH and GPG keys. The main area is titled 'Applications' and has tabs for 'Installed GitHub Apps', 'Authorized GitHub Apps', and 'Authorized OAuth Apps'. The 'Authorized OAuth Apps' tab is highlighted with a red box. Below it, a message says 'You have granted 9 applications access to your account.' There are two entries listed: 'Azure App Service' (last used within the last 2 months, owned by AzureAppService) and 'Azure App Service Container Apps' (last used within the last week, owned by AzureAppService). Both entries have a blue icon. To the right of the entries are 'Sort' and 'Revoke all' buttons, and a search icon.

Solución de problemas

Experimenta errores al configurar un principal de servicio mediante la CLI de Azure.

Esta sección puede ayudarle a solucionar los errores que obtiene al configurar una entidad de servicio mediante el comando `az ad sp create-for-rba` de la CLI de Azure.

Si recibe un error que contiene "InvalidSchema: No se encontraron adaptadores de conexión":

- Compruebe el shell en el que se está ejecutando. Si usa un shell de Bash, establezca las variables de `MSYS_NO_PATHCONV` como `export MSYS_NO_PATHCONV=1`.

Para obtener más información, consulte el problema de GitHub [No se puede crear una entidad de servicio con la CLI de Azure desde el shell de Git Bash ↗](#).

Si recibe un error que contiene "Más de una aplicación tiene el mismo nombre para mostrar":

- El nombre ya se usa en la entidad de servicio. Elija otro nombre o deje el argumento `--name`. Se generará automáticamente un GUID como nombre para mostrar.

Error en el flujo de trabajo de Acciones de GitHub

Para comprobar el estado de un flujo de trabajo, vaya a la pestaña Acciones del repositorio:

- Si hay un flujo de trabajo con errores, explore en profundidad el archivo de flujo de trabajo. Debería haber dos trabajos: compilar e implementar. Para un trabajo fallido, compruebe la salida de las tareas del trabajo para buscar problemas.
- Si hay un mensaje de error que indica "Tiempo de espera del protocolo de enlace TLS", ejecute el flujo de trabajo manualmente. En el repositorio, en la pestaña **Acciones**, seleccione **Desencadenar implementación automática** para verificar si el tiempo de espera es un problema temporal.

- Si configura la implementación continua para la aplicación contenedora como se muestra en este tutorial, el archivo de flujo de trabajo (`.github/workflows/<nombre de flujo de trabajo>.yml`) se crea automáticamente. No es necesario modificar este archivo para este tutorial. Si lo hizo, revierta los cambios e intente el flujo de trabajo.

El sitio web no muestra los cambios que ha fusionado en la rama principal

En GitHub:

- Compruebe que se ejecutó el flujo de trabajo de Acciones de GitHub y que ha comprobado el cambio en la rama que desencadena el flujo de trabajo.

En Azure Portal:

- Compruebe la instancia de Azure Container Registry para ver si se creó una imagen de Docker con una marca de tiempo después del cambio en la rama.
- Compruebe los registros de la aplicación contenedora para ver si hay un error de programación:
 1. Vaya a la aplicación de contenedor y, a continuación, vaya a **Administración de revisión**>*<contenedor activo>*>**Detalles de revisión**>**Registros de consola**.
 2. Elija el orden de las columnas para mostrar **Hora de generación**, **Stream_s** y **Log_s**.
 3. Ordene los registros por más reciente y busque los mensajes `stderr` de Python y `stdout` en la columna **Stream_s**. La salida de `print` de Python son mensajes `stdout`.

En la CLI de Azure:

- Utiliza el comando `az containerapp logs show`.

Quiere detener la implementación continua.

Detener la implementación continua significa desconectar la aplicación contenedora del repositorio.

En Azure Portal:

- Vaya a la aplicación de contenedor. En el menú de servicio, seleccione **Implementación continua**, a continuación, seleccione **Desconectar**.

En la CLI de Azure:

- Utilice el comando `az container app github-action remove`.

Después de desconectar:

- El archivo `.github/workflows/<workflow-name>.yml` se quita del repositorio.
- Las claves secretas no se quitan del repositorio.
- Azure Container Apps permanece como una aplicación de OAuth autorizada para su cuenta de GitHub.
- En Azure, el contenedor se queda con el último contenedor que fue implementado. Puede volver a conectar la aplicación contenedora con la instancia de Azure Container Registry para que las nuevas revisiones de contenedor recojan la imagen más reciente.
- En Azure, las entidades de servicio que ha creado y usado para la implementación continua no se eliminan.

Quitar recursos

Si ha terminado con la serie de tutoriales y no quiere incurrir en costos adicionales, quite los recursos que usó.

La eliminación de un grupo de recursos quita todos los recursos del grupo y es la manera más rápida de quitar recursos. Para ver un ejemplo de cómo quitar grupos de recursos, consulte [Tutorial sobre contenedorización y limpieza](#).

Contenido relacionado

Si tiene previsto desarrollar a partir de este tutorial, estos son algunos próximos pasos que puede seguir:

- [Establecimiento de reglas de escalado en Azure Container Apps](#)
- [Enlazar certificados y nombres de dominio personalizados en Azure Container Apps](#)
- [Supervisión de una aplicación en Azure Container Apps](#)

Inicio rápido: Implementación de un clúster de Azure Kubernetes Service (AKS) con la CLI de Azure

19/08/2025



Deploy to Azure



Azure Kubernetes Service (AKS) es un servicio de Kubernetes administrado que le permite implementar y administrar clústeres rápidamente. En esta guía de inicio rápido, ha aprendido a hacer lo siguiente:

- Implementación de un clúster de AKS con la CLI de Azure.
- Ejecute una aplicación de varios contenedores de ejemplo con un grupo de microservicios y front-end web que simulan un escenario comercial.

ⓘ Nota

En este artículo se incluyen los pasos para implementar un clúster con la configuración predeterminada solo con fines de evaluación. Antes de implementar un clúster listo para producción, se recomienda familiarizarse con nuestra [arquitectura de referencia de línea base](#) para tener en cuenta cómo se alinea con sus requisitos empresariales.

Antes de empezar

En esta guía rápida se presupone un conocimiento básico de los conceptos de Kubernetes.

Para más información, consulte [Conceptos básicos de Kubernetes de Azure Kubernetes Service \(AKS\)](#).

- Si no tiene una cuenta de Azure, cree una [cuenta gratuita](#) antes de comenzar.
- Use el entorno de Bash en [Azure Cloud Shell](#). Para más información, consulte [Introducción a Azure Cloud Shell](#).



Launch Cloud Shell



- Si prefiere ejecutar comandos de referencia de la CLI localmente, [instale](#) la CLI de Azure. Si utiliza Windows o macOS, considere la posibilidad de ejecutar la CLI de Azure en un contenedor Docker. Para más información, vea [Ejecución de la CLI de Azure en un contenedor de Docker](#).

- Si usa una instalación local, inicie sesión en la CLI de Azure mediante el comando [az login](#). Siga los pasos que se muestran en el terminal para completar el proceso de autenticación. Para ver otras opciones de inicio de sesión, consulte [Autenticación en Azure mediante la CLI de Azure](#).
 - En caso de que se le solicite, instale las extensiones de la CLI de Azure la primera vez que la use. Para obtener más información sobre las extensiones, consulte [Uso y administración de extensiones con la CLI de Azure](#).
 - Ejecute [az version](#) para buscar cuál es la versión y las bibliotecas dependientes que están instaladas. Para realizar la actualización a la versión más reciente, ejecute [az upgrade](#).
- Asegúrese de que la identidad que usará para crear el clúster tenga los permisos mínimos adecuados. Para más información sobre el acceso y la identidad en AKS, consulte [Opciones de acceso e identidad en Azure Kubernetes Service \(AKS\)](#).
 - Si tiene varias suscripciones de Azure, seleccione el identificador de suscripción adecuado en el que se deben facturar los recursos con el comando [az account set](#). Para más información, consulte [Administración de suscripciones de Azure: CLI de Azure](#).
 - En función de la suscripción de Azure, es posible que tenga que solicitar un aumento de cuota de vCPU. Para más información, consulte [Aumento de las cuotas de vCPU de la familia de máquinas virtuales](#).

Registro de proveedores de recursos

Es posible que tenga que registrar proveedores de recursos en la suscripción de Azure. Por ejemplo, `Microsoft.ContainerService` es necesario.

Ejecute el siguiente comando para comprobar el estado del registro.

Azure CLI

```
az provider show --namespace Microsoft.ContainerService --query registrationState
```

Si es necesario, registre el proveedor de recursos.

Azure CLI

```
az provider register --namespace Microsoft.ContainerService
```

Definición de las variables de entorno

Defina las siguientes variables de entorno para usarlas en este inicio rápido.

Azure CLI

```
export RANDOM_ID=$(openssl rand -hex 3)
export MY_RESOURCE_GROUP_NAME="myAKSResourceGroup$RANDOM_ID"
export REGION="westus"
export MY_AKS_CLUSTER_NAME="myAKSCluster$RANDOM_ID"
export MY_DNS_LABEL="mydnslabel$RANDOM_ID"
```

El `RANDOM_ID` valor de la variable es un valor alfanumérico de seis caracteres anexado al grupo de recursos y al nombre del clúster para que los nombres sean únicos. Use el `echo` comando para ver valores de variable como `echo $RANDOM_ID`.

Crear un grupo de recursos

Un [grupo de recursos de Azure](#) es un grupo lógico en el que se implementan y administran recursos de Azure. Cuando crea un grupo de recursos, se le pide que especifique una ubicación. Esta ubicación es la ubicación de almacenamiento de los metadatos del grupo de recursos y donde se ejecutan los recursos en Azure si no se especifica otra región durante la creación de recursos.

Cree un grupo de recursos con el comando [az group create](#).

Azure CLI

```
az group create --name $MY_RESOURCE_GROUP_NAME --location $REGION
```

El resultado es similar al ejemplo siguiente.

Resultados

```
{
  "id": "/subscriptions/aaaa0a0a-bb1b-cc2c-dd3d-
eeeeee4e4e/resourceGroups/myAKSResourceGroup<randomIDValue>",
  "location": "westus",
  "managedBy": null,
  "name": "myAKSResourceGroup<randomIDValue>",
  "properties": {
    "provisioningState": "Succeeded"
  },
  "tags": null,
  "type": "Microsoft.Resources/resourceGroups"
}
```

Creación de un clúster de AKS

Cree un clúster de AKS con el comando [az aks create](#). En el ejemplo siguiente se crea un clúster con un nodo y se habilita una identidad administrada asignada por el sistema.

Azure CLI

```
az aks create \
--resource-group $MY_RESOURCE_GROUP_NAME \
--name $MY_AKS_CLUSTER_NAME \
--node-count 1 \
--generate-ssh-keys
```

! Nota

Al crear un nuevo clúster, AKS crea automáticamente un segundo grupo de recursos para almacenar los recursos de AKS. Para más información, consulte [¿Por qué se crean dos grupos de recursos con AKS?](#)

Conectarse al clúster

Para administrar un clúster de Kubernetes, use [kubectl](#), el cliente de línea de comandos de Kubernetes. Si usa Azure Cloud Shell, `kubectl` ya está instalado. Para instalar `kubectl` localmente, use el comando [az aks install-cli](#).

1. Para configurar `kubectl` para conectarse a su clúster de Kubernetes, use el comando [az aks get-credentials](#). Con este comando se descargan las credenciales y se configura la CLI de Kubernetes para usarlas.

Azure CLI

```
az aks get-credentials --resource-group $MY_RESOURCE_GROUP_NAME --name
$MY_AKS_CLUSTER_NAME
```

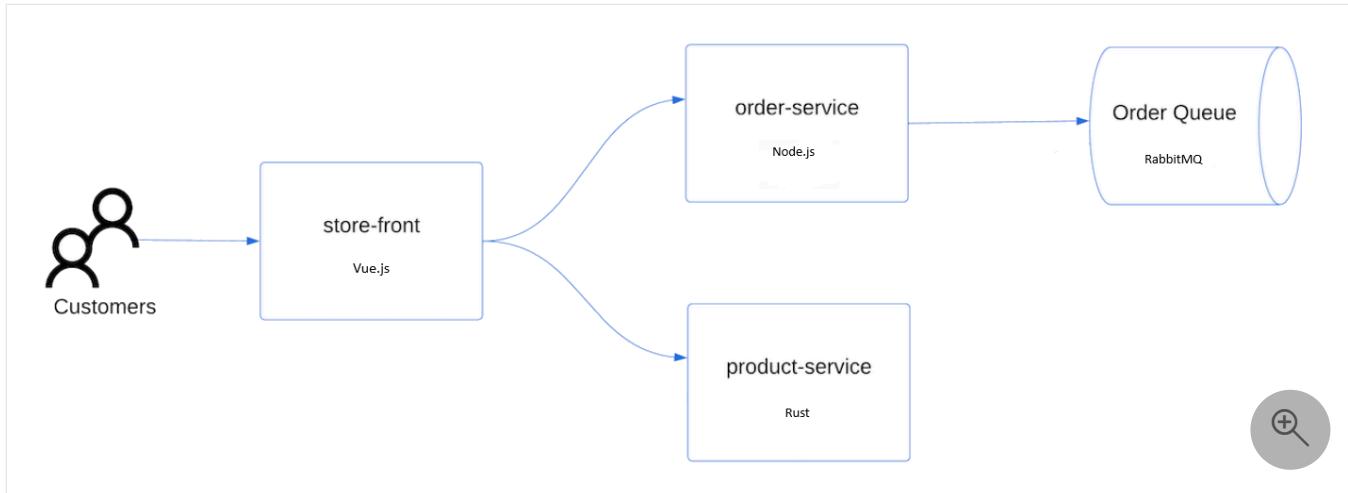
2. Compruebe la conexión al clúster con el comando [kubectl get](#). Este comando devuelve una lista de los nodos del clúster.

Azure CLI

```
kubectl get nodes
```

Implementación de la aplicación

Para implementar la aplicación, se usa un archivo de manifiesto para crear todos los objetos necesarios para ejecutar la [aplicación AKS Store](#). Un archivo de manifiesto de Kubernetes define el estado deseado de un clúster, como las imágenes de contenedor que se van a ejecutar. El manifiesto incluye las siguientes implementaciones y servicios de Kubernetes:



- Frente a la tienda: aplicación web para que los clientes vean productos y realicen pedidos.
- Servicio de producto: muestra la información del producto.
- Servicio de pedidos: realiza pedidos.
- RabbitMQ: cola de mensajes para una cola de pedidos.

Nota

No se recomienda ejecutar contenedores con estado, como RabbitMQ, sin almacenamiento persistente para producción. Aquí se usa para simplificar, pero se recomienda usar servicios administrados, como Azure CosmosDB o Azure Service Bus.

1. Cree un archivo denominado *aks-store-quickstart.yaml* y copie en el siguiente manifiesto.

YAML

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: rabbitmq
spec:
  serviceName: rabbitmq
  replicas: 1
  selector:
    matchLabels:
      app: rabbitmq
  template:
```

```

metadata:
  labels:
    app: rabbitmq
spec:
  nodeSelector:
    "kubernetes.io/os": linux
  containers:
    - name: rabbitmq
      image: mcr.microsoft.com/mirror/docker/library/rabbitmq:3.10-
management-alpine
      ports:
        - containerPort: 5672
          name: rabbitmq-amqp
        - containerPort: 15672
          name: rabbitmq-http
      env:
        - name: RABBITMQ_DEFAULT_USER
          value: "username"
        - name: RABBITMQ_DEFAULT_PASS
          value: "password"
      resources:
        requests:
          cpu: 10m
          memory: 128Mi
        limits:
          cpu: 250m
          memory: 256Mi
      volumeMounts:
        - name: rabbitmq-enabled-plugins
          mountPath: /etc/rabbitmq/enabled_plugins
          subPath: enabled_plugins
      volumes:
        - name: rabbitmq-enabled-plugins
          configMap:
            name: rabbitmq-enabled-plugins
            items:
              - key: rabbitmq_enabled_plugins
                path: enabled_plugins
---
apiVersion: v1
data:
  rabbitmq_enabled_plugins: |
    [rabbitmq_management,rabbitmq_prometheus,rabbitmq_amqp1_0].
kind: ConfigMap
metadata:
  name: rabbitmq-enabled-plugins
---
apiVersion: v1
kind: Service
metadata:
  name: rabbitmq
spec:
  selector:
    app: rabbitmq
  ports:

```

```
- name: rabbitmq-amqp
  port: 5672
  targetPort: 5672
- name: rabbitmq-http
  port: 15672
  targetPort: 15672
  type: ClusterIP
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: order-service
spec:
  replicas: 1
  selector:
    matchLabels:
      app: order-service
  template:
    metadata:
      labels:
        app: order-service
  spec:
    nodeSelector:
      "kubernetes.io/os": linux
    containers:
      - name: order-service
        image: ghcr.io/azure-samples/aks-store-demo/order-service:latest
        ports:
          - containerPort: 3000
        env:
          - name: ORDER_QUEUE_HOSTNAME
            value: "rabbitmq"
          - name: ORDER_QUEUE_PORT
            value: "5672"
          - name: ORDER_QUEUE_USERNAME
            value: "username"
          - name: ORDER_QUEUE_PASSWORD
            value: "password"
          - name: ORDER_QUEUE_NAME
            value: "orders"
          - name: FASTIFY_ADDRESS
            value: "0.0.0.0"
        resources:
          requests:
            cpu: 1m
            memory: 50Mi
          limits:
            cpu: 75m
            memory: 128Mi
        startupProbe:
          httpGet:
            path: /health
            port: 3000
          failureThreshold: 5
          initialDelaySeconds: 20
```

```
    periodSeconds: 10
  readinessProbe:
    httpGet:
      path: /health
      port: 3000
      failureThreshold: 3
      initialDelaySeconds: 3
      periodSeconds: 5
  livenessProbe:
    httpGet:
      path: /health
      port: 3000
      failureThreshold: 5
      initialDelaySeconds: 3
      periodSeconds: 3
  initContainers:
  - name: wait-for-rabbitmq
    image: busybox
    command: ['sh', '-c', 'until nc -zv rabbitmq 5672; do echo waiting for rabbitmq; sleep 2; done;']
    resources:
      requests:
        cpu: 1m
        memory: 50Mi
      limits:
        cpu: 75m
        memory: 128Mi
  ---
apiVersion: v1
kind: Service
metadata:
  name: order-service
spec:
  type: ClusterIP
  ports:
  - name: http
    port: 3000
    targetPort: 3000
  selector:
    app: order-service
  ---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: product-service
spec:
  replicas: 1
  selector:
    matchLabels:
      app: product-service
  template:
    metadata:
      labels:
        app: product-service
  spec:
```

```
nodeSelector:
  "kubernetes.io/os": linux
containers:
- name: product-service
  image: ghcr.io/azure-samples/aks-store-demo/product-service:latest
  ports:
  - containerPort: 3002
  env:
  - name: AI_SERVICE_URL
    value: "http://ai-service:5001/"
  resources:
    requests:
      cpu: 1m
      memory: 1Mi
    limits:
      cpu: 2m
      memory: 20Mi
  readinessProbe:
    httpGet:
      path: /health
      port: 3002
    failureThreshold: 3
    initialDelaySeconds: 3
    periodSeconds: 5
  livenessProbe:
    httpGet:
      path: /health
      port: 3002
    failureThreshold: 5
    initialDelaySeconds: 3
    periodSeconds: 3
---
apiVersion: v1
kind: Service
metadata:
  name: product-service
spec:
  type: ClusterIP
  ports:
  - name: http
    port: 3002
    targetPort: 3002
  selector:
    app: product-service
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: store-front
spec:
  replicas: 1
  selector:
    matchLabels:
      app: store-front
  template:
```

```
metadata:
  labels:
    app: store-front
spec:
  nodeSelector:
    "kubernetes.io/os": linux
  containers:
    - name: store-front
      image: ghcr.io/azure-samples/aks-store-demo/store-front:latest
      ports:
        - containerPort: 8080
          name: store-front
      env:
        - name: VUE_APP_ORDER_SERVICE_URL
          value: "http://order-service:3000/"
        - name: VUE_APP_PRODUCT_SERVICE_URL
          value: "http://product-service:3002/"
      resources:
        requests:
          cpu: 1m
          memory: 200Mi
        limits:
          cpu: 1000m
          memory: 512Mi
      startupProbe:
        httpGet:
          path: /health
          port: 8080
          failureThreshold: 3
          initialDelaySeconds: 5
          periodSeconds: 5
      readinessProbe:
        httpGet:
          path: /health
          port: 8080
          failureThreshold: 3
          initialDelaySeconds: 3
          periodSeconds: 3
      livenessProbe:
        httpGet:
          path: /health
          port: 8080
          failureThreshold: 5
          initialDelaySeconds: 3
          periodSeconds: 3
  ---
apiVersion: v1
kind: Service
metadata:
  name: store-front
spec:
  ports:
    - port: 80
      targetPort: 8080
  selector:
```

```
app: store-front
type: LoadBalancer
```

Para obtener un desglose de los archivos de manifiesto de YAML, consulte [Implementaciones y manifiestos de YAML](#).

Si crea y guarda el archivo YAML localmente, puede cargar el archivo de manifiesto en el directorio predeterminado de Cloud Shell seleccionando el botón **Cargar y descargar archivos** y seleccionando el archivo del sistema de archivos local.

2. Implemente la aplicación mediante el comando [kubectl apply](#) y especifique el nombre del manifiesto de YAML:

Azure CLI

```
kubectl apply -f aks-store-quickstart.yaml
```

Prueba de la aplicación

Puede validar que la aplicación se está ejecutando visitando la dirección IP pública o la dirección URL de la aplicación.

Obtenga la dirección URL de la aplicación mediante los siguientes comandos:

Azure CLI

```
runtime="5 minutes"
endtime=$(date -ud "$runtime" +%s)
while [[ $(date -u +%s) -le $endtime ]]
do
    STATUS=$(kubectl get pods -l app=store-front -o 'jsonpath=
{..status.conditions[?(@.type=="Ready")].status}')
    echo $STATUS
    if [ "$STATUS" == 'True' ]
    then
        export IP_ADDRESS=$(kubectl get service store-front --output 'jsonpath=
{..status.loadBalancer.ingress[0].ip}')
        echo "Service IP Address: $IP_ADDRESS"
        break
    else
        sleep 10
    fi
done
```

Azure CLI

```
curl $IP_ADDRESS
```

Results:

HTML

```
<!doctype html>
<html lang="">
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width,initial-scale=1">
    <link rel="icon" href="/favicon.ico">
    <title>store-front</title>
    <script defer="defer" src="/js/chunk-vendors.df69ae47.js"></script>
    <script defer="defer" src="/js/app.7e8cfbb2.js"></script>
    <link href="/css/app.a5dc49f6.css" rel="stylesheet">
  </head>
  <body>
    <div id="app"></div>
  </body>
</html>
```

Resultados

```
echo "You can now visit your web server at $IP_ADDRESS"
```

Para ver el sitio web de la aplicación, abra un explorador y escriba la dirección IP. La página tiene un aspecto similar al del ejemplo siguiente.

The screenshot shows a web application for a pet store. At the top, there is a header with the Contoso logo, a search bar, and links for 'Products' and 'Cart (0)'. Below the header, there are five product cards arranged horizontally, each featuring a product image, the product name, a brief description, the price, and a quantity selector with an 'Add to Cart' button.

Product	Description	Price	Quantity	Action
Contoso Catnip's Friend	Watch your feline friend embark on a fishing adventure with Contoso Catnip's Friend toy. Packed with irresistible catnip and dangling fish lure.	9.99	1	Add to Cart
Salty Sailor's Squeaky Squid	Let your dog set sail with the Salty Sailor's Squeaky Squid. This interactive toy provides hours of fun, featuring multiple squeakers and crinkle tentacles.	6.99	1	Add to Cart
Mermaid's Mice Trio	Entertain your kitty with the Mermaid's Mice Trio. These adorable plush mice are dressed as mermaids and filled with catnip to captivate their curiosity.	12.99	1	Add to Cart
Ocean Explorer's Puzzle Ball	Challenge your pet's problem-solving skills with the Ocean Explorer's Puzzle Ball. This interactive toy features hidden compartments and treats, providing mental stimulation and entertainment.	11.99	1	Add to Cart
Pirate Parrot Teaser Wand	Engage your cat in a playful pursuit with the Pirate Parrot Teaser Wand. The colorful feathers and jingling bells mimic the mischievous charm of a pirate's parrot.	8.99	1	Add to Cart

Eliminación del clúster

Si no planea pasar por el [tutorial de AKS](#), limpie los recursos innecesarios para evitar cargos de facturación de Azure. Puede quitar el grupo de recursos, el servicio de contenedor y todos los recursos relacionados mediante el comando [az group delete](#).

Azure CLI

```
az group delete --name $MY_RESOURCE_GROUP_NAME
```

El clúster de AKS se creó con una identidad administrada asignada por el sistema, que es la opción de identidad predeterminada utilizada en este inicio rápido. La plataforma administra esta identidad para que no tenga que quitarla manualmente.

Pasos siguientes

En este inicio rápido, ha implementado un clúster de Kubernetes y luego ha implementado en él una aplicación simple de varios contenedores. Esta aplicación de ejemplo es solo para fines de demostración y no representa todos los procedimientos recomendados para las aplicaciones de Kubernetes. Para obtener instrucciones sobre cómo crear soluciones completas con AKS para producción, consulte la [guía de soluciones de AKS](#).

Para más información sobre AKS y realizar un ejemplo completo de código a implementación, continúe con el tutorial del clúster de Kubernetes.

[Tutorial de AKS](#)

Uso de las bibliotecas de Azure (SDK) para Python

Artículo • 06/02/2025

Las bibliotecas de Azure de código abierto para Python simplifican el aprovisionamiento, la administración y el uso de recursos de Azure desde el código de aplicación de Python.

Los detalles que realmente quiere saber

- Las bibliotecas de Azure son la manera de comunicarse con los servicios de Azure *desde* código de Python que se ejecuta localmente o en la nube. (Si puede ejecutar código de Python dentro del ámbito de un servicio determinado depende de si ese propio servicio admite actualmente Python).
- Las bibliotecas admiten [Python ↗](#) 3.8 o posterior. Para obtener más información sobre las versiones de Python compatibles, consulte [la política de soporte de las versiones de Python en los SDK de Azure ↗](#). Si usa [PyPy ↗](#), asegúrese de que la versión que usa al menos admite la versión de Python mencionada anteriormente.
- El SDK de Azure para Python se compone únicamente de más de 180 bibliotecas individuales de Python relacionadas con servicios específicos de Azure. No hay otras herramientas en el SDK.
- Al ejecutar código localmente, la autenticación con Azure se basa en variables de entorno, como se describe en [Autenticación de aplicaciones de Python en servicios de Azure mediante el SDK de Azure para Python](#).
- Para instalar paquetes de biblioteca con pip, use `pip install <library_name>` mediante nombres de biblioteca del índice de paquete de . Para instalar paquetes de biblioteca en entornos de Conda, use `conda install <package_name>` mediante nombres del canal de Microsoft de [en anaconda.org ↗](#). Para más información, consulte [Instalación de paquetes de biblioteca de Azure](#).
- Hay bibliotecas de administración **distintas** y bibliotecas de cliente (a veces denominadas bibliotecas de "plano de administración" y "plano de datos"). Cada conjunto sirve para distintos propósitos y se usa en diferentes tipos de código. Para obtener más información, consulte las secciones siguientes más adelante en este artículo:

- Creación y administración de recursos de Azure con bibliotecas de administración
 - Conectarse a recursos de Azure y usarlos con bibliotecas cliente
- La documentación de las bibliotecas se encuentra en la [Referencia de Azure para Python](#), que está organizada por el servicio de Azure o el explorador de api de Python , que se organiza por nombre de paquete.
- Para probar las bibliotecas por su cuenta, primero se recomienda [configuración del entorno de desarrollo local](#). A continuación, puede probar cualquiera de los siguientes ejemplos independientes (en cualquier orden): [Ejemplo: Crear un grupo de recursos](#), [Ejemplo: Crear y usar Azure Storage](#), [Ejemplo: Crear e implementar una aplicación web](#), [Ejemplo: Crear y consultar una base de datos MySQL](#) [Ejemplo: Crear una máquina virtual](#).
- Para ver vídeos de demostración, consulte [Introducción al SDK de Azure para Python](#) ↗ (PyCon 2021) y [Uso de SDK de Azure para interactuar con los recursos de Azure](#) ↗ (PyCon 2020).

Detalles no esenciales, pero aún interesantes

- Dado que el de la CLI de Azure se escribe en Python mediante las bibliotecas de administración, cualquier cosa que pueda hacer con los comandos de la CLI de Azure, también puede hacerlo desde un script de Python. Dicho esto, los comandos de la CLI proporcionan muchas características útiles, como realizar varias tareas juntas, controlar automáticamente las operaciones asincrónicas, dar formato a la salida como cadenas de conexión, etc. Por lo tanto, el uso de la CLI (o su equivalente, [Azure PowerShell](#)) para scripts de creación y administración automatizados puede ser más cómodo que escribir el código de Python equivalente, a menos que desee tener un grado de control mucho más exacto sobre el proceso.
- Las bibliotecas de Azure para Python se basan en la API REST de Azure subyacente , lo que le permite usar esas API a través de paradigmas conocidos de Python. Sin embargo, siempre puede usar la API REST directamente desde código de Python, si lo desea.
- Puede encontrar el código fuente de las bibliotecas de Azure en <https://github.com/Azure/azure-sdk-for-python> ↗ . Como proyecto de código abierto, las contribuciones son bienvenidas.
- Aunque puedes usar las bibliotecas con intérpretes, como IronPython y Jython, los cuales no probamos, es posible que encuentres problemas aislados e

incompatibilidades.

- El repositorio de origen de la documentación de referencia de la API de biblioteca reside en <https://github.com/MicrosoftDocs/azure-docs-sdk-python/>.
- A partir de 2019, actualizamos las bibliotecas de Python de Azure para compartir patrones comunes en la nube, como protocolos de autenticación, registro, seguimiento, protocolos de transporte, respuestas almacenadas en búfer y reintentos. Las bibliotecas actualizadas cumplen [las directrices actuales del SDK de Azure](#).
- El 31 de marzo de 2023, hemos retirado la compatibilidad con las bibliotecas de Azure SDK que no se ajustan a las actuales directrices de Azure SDK. Aunque las bibliotecas anteriores todavía se pueden usar más allá del 31 de marzo de 2023, ya no recibirán soporte técnico oficial ni actualizaciones de Microsoft. Para más información, consulte el aviso [Actualización de las bibliotecas de Azure SDK](#).
- Para evitar que falten actualizaciones de seguridad y rendimiento en los SDK de Azure, actualice a [las bibliotecas más recientes del SDK de Azure](#) antes del 31 de marzo de 2023.
- Para comprobar qué bibliotecas de Python se ven afectadas, consulte [Azure SDK Deprecated Releases for Python](#).
- Para obtener más información sobre las directrices que aplicamos a las bibliotecas, consulte las directrices de Python : [Introducción](#).

Creación y administración de recursos de Azure con bibliotecas de administración

Las bibliotecas de administración de *del SDK* (o "plano de administración"), los nombres de los cuales comienzan por `azure-mgmt-`, le ayudan a crear, configurar y administrar recursos de Azure desde scripts de Python. Todos los servicios de Azure tienen bibliotecas de administración correspondientes. Para más información, consulte [plano de control y plano de datos de Azure](#).

Con las bibliotecas de administración, puede escribir scripts de configuración e implementación para realizar las mismas tareas que puede realizar a través de la [Azure Portal](#) o la CLI de Azure . (Como se indicó anteriormente, la CLI de Azure se escribe en Python y usa las bibliotecas de administración para implementar sus distintos comandos).

En los ejemplos siguientes se muestra cómo usar algunas de las bibliotecas de administración principales:

- [Crear un grupo de recursos](#)
- [Enumerar grupos de recursos en una suscripción](#)
- [Creación de una cuenta de Azure Storage y un contenedor de Blob Storage](#)
- [Creación e implementación de una aplicación web en App Service](#)
- [Creación y consulta de una base de datos de Azure MySQL](#)
- [Crear una máquina virtual](#)

Para obtener más información sobre cómo trabajar con cada biblioteca de administración, consulte el archivo *README.md* o *README.rst* ubicado en la carpeta del proyecto de la biblioteca en el repositorio de GitHub del SDK de [SDK](#). También puede encontrar más fragmentos de código en la documentación de referencia de [y Azure Ejemplos](#).

Migración desde bibliotecas de administración anteriores

Si va a migrar código de versiones anteriores de las bibliotecas de administración, consulte los detalles siguientes:

- Si usa la clase `ServicePrincipalCredentials`, consulte [Autenticar con credenciales de token](#).
- Los nombres de las API asincrónicas han cambiado como se describe en los patrones de uso de [Library: operaciones asincrónicas](#). Los nombres de las API asincrónicas de las bibliotecas más recientes comienzan por `begin_`. En la mayoría de los casos, la firma de api sigue siendo la misma.

Conexión y uso de recursos de Azure con bibliotecas cliente

Las bibliotecas de cliente *del SDK* (o "plano de datos") le ayudan a desarrollar código de aplicaciones en Python para interactuar con servicios ya existentes. Las bibliotecas cliente solo existen para aquellos servicios que admiten una API de cliente.

El artículo titulado [Ejemplo: Uso de Azure Storage](#) proporciona una ilustración básica del uso de la biblioteca cliente.

Los distintos servicios de Azure también proporcionan ejemplos con estas bibliotecas. Consulte las páginas de índice siguientes para ver otros vínculos:

- [Hospedaje de aplicaciones](#)

- Servicios Cognitivos
- soluciones de datos
- Identidad y seguridad
- Aprendizaje Automático
- Mensajería e IoT
- Otros servicios

Para obtener más información sobre cómo trabajar con cada biblioteca cliente, consulte el archivo *README.md* o *README.rst* que se encuentra en la carpeta del proyecto de la biblioteca en el repositorio de GitHub del SDK de [. También puede encontrar más fragmentos de código en la documentación de referencia de y los \[Ejemplos de Azure\]\(#\).](#)

Obtención de ayuda y conexión con el equipo del SDK

- Visite las bibliotecas de Azure de [para la documentación de Python](#)
- Publique preguntas a la comunidad en [Stack Overflow](#)
- Problemas abiertos contra el SDK en [de GitHub](#)
- Mencionar [@AzureSDK](#) en Twitter
- [Completar una breve encuesta sobre el SDK de Azure para Python](#)

Paso siguiente

Se recomienda encarecidamente realizar una configuración única del entorno de desarrollo local para que pueda usar fácilmente cualquiera de las bibliotecas de Azure para Python.

[Configuración del entorno de desarrollo local >>>](#)

Comentarios

¿Le ha resultado útil esta página?

 Sí

 No

[Proporcionar comentarios sobre el producto](#) | [Obtener ayuda en Microsoft Q&A](#)

Bibliotecas de Azure para patrones de uso de Python

Artículo • 30/04/2025

El SDK de Azure para Python se compone de muchas bibliotecas independientes, que se enumeran en el [índice de paquetes del SDK de Python](#).

Todas las bibliotecas comparten ciertas características comunes y patrones de uso, como la instalación y el uso de JSON insertado para argumentos de objeto.

Configuración del entorno de desarrollo local

Si aún no lo ha hecho, puede configurar un entorno en el que pueda ejecutar este código. Estas son algunas opciones:

- Configure un entorno virtual de Python mediante `venv` o la herramienta que prefiera. Puede crear el entorno virtual localmente o en [azure Cloud Shell](#) y ejecutar el código allí. Asegúrese de activar el entorno virtual para empezar a usarlo. Para instalar Python, consulte [Instalación de Python](#).

Bash

```
python -m venv .venv
source .venv/bin/activate # Linux or macOS
.venv\Scripts\activate # Windows
```

- Use un [entorno de Conda](#). Para instalar Conda, consulte [Instalación de Miniconda](#).
- Usa un contenedor de desarrollo en [Visual Studio Code](#) o en [GitHub Codespaces](#).

Instalación de la biblioteca

Elija el método de instalación correspondiente a la herramienta de administración del entorno de Python, ya sea pip o conda.

pip

Para instalar un paquete de biblioteca específico, use `pip install`:

Símbolo del sistema de Windows

```
REM Install the management library for Azure Storage  
pip install azure-mgmt-storage
```

Símbolo del sistema de Windows

```
REM Install the client library for Azure Blob Storage  
pip install azure-storage-blob
```

Símbolo del sistema de Windows

```
REM Install the azure identity library for Azure authentication  
pip install azure-identity
```

`pip install` recupera la versión más reciente de una biblioteca en el entorno de Python actual.

También puede usar `pip` para desinstalar bibliotecas e instalar versiones específicas, incluidas las versiones preliminares. Para más información, consulte [Instalación de paquetes de biblioteca de Azure para Python](#).

Operaciones asincrónicas

Bibliotecas asincrónicas

Muchas bibliotecas de administración y cliente proporcionan versiones asincrónicas (`.aio`). La `asyncio` biblioteca está disponible desde Python 3.4 y las palabras clave `async/await` se introdujeron en Python 3.5. Las versiones asincrónicas de las bibliotecas están diseñadas para usarse con Python 3.5 y versiones posteriores.

Algunos ejemplos de bibliotecas del SDK de Azure Python con versiones asincrónicas son : `azure.storage.blob.aio`, `azure.servicebus.aio`, `azure.mgmt.keyvault.aio` y `azure.mgmt.compute.aio`.

Estas bibliotecas necesitan un transporte asincrónico como `aiohttp` para funcionar. La `azure-core` biblioteca proporciona un transporte asincrónico, `AioHttpTransport` que usa las bibliotecas asincrónicas, por lo que es posible que no sea necesario instalar `aiohttp` por separado.

En el código siguiente se muestra cómo crear un archivo de Python que muestra cómo crear un cliente para la versión asincrónica de la biblioteca de Azure Blob Storage:

Python

```
credential = DefaultAzureCredential()

async def run():

    async with BlobClient(
        storage_url,
        container_name="blob-container-01",
        blob_name=f"sample-blob-{str(uuid.uuid4())[0:5]}.txt",
        credential=credential,
    ) as blob_client:

        # Open a local file and upload its contents to Blob Storage
        with open("./sample-source.txt", "rb") as data:
            await blob_client.upload_blob(data)
            print(f"Uploaded sample-source.txt to {blob_client.url}")

        # Close credential
        await credential.close()

asyncio.run(run())
```

El ejemplo completo está en GitHub en [use_blob_auth_async.py](#). Para obtener la versión sincrónica de este código, consulte [Ejemplo: Carga de un blob](#).

Operaciones de larga duración

Algunas operaciones de administración que se invocan (como `ComputeManagementClient.virtual_machines.begin_create_or_update` y `WebAppsClient.web_apps.begin_create_or_update`) devuelven un sondeo para operaciones de larga duración, `LROPoller[<type>]`, donde `<type>` es específico de la operación en cuestión.

! Nota

Puede observar diferencias en los nombres de método de una biblioteca en función de su versión y de si se basa en `azure.core`. Las bibliotecas anteriores que no se basan en `azure.core` suelen usar nombres como `create_or_update`. Las bibliotecas basadas en `azure.core` agregan el `begin_` prefijo a los nombres de método para indicar mejor que son operaciones de sondeo largas. Migrar código antiguo a una biblioteca basada en `azure.core` más reciente normalmente significa agregar el `begin_` prefijo a los nombres de método, ya que la mayoría de las firmas de método siguen siendo las mismas.

El `LROPoller` tipo de valor devuelto significa que la operación es asíncrona. En consecuencia, debe llamar al método `result` del sondeador para esperar a que finalice la operación y

obtener el resultado.

El código siguiente, tomado de [Ejemplo: Creación e implementación de una aplicación web](#), muestra un ejemplo del uso del sondeo para esperar un resultado:

Python

```
# Step 3: With the plan in place, provision the web app itself, which is the
process that can host
# whatever code we want to deploy to it.

poller = app_service_client.web_apps.begin_create_or_update(RESOURCE_GROUP_NAME,
    WEB_APP_NAME,
    {
        "location": LOCATION,
        "server_farm_id": plan_result.id,
        "site_config": {
            "linux_fx_version": "python|3.8"
        }
    }
)

web_app_result = poller.result()
```

En este caso, el valor devuelto de `begin_create_or_update` es de tipo

`AzureOperationPoller[Site]`, lo que significa que el valor devuelto de `poller.result()` es un objeto Site.

Excepciones

En general, las bibliotecas de Azure generan excepciones cuando las operaciones no funcionan según lo previsto, incluidas las solicitudes HTTP erróneas a la API REST de Azure. En el caso del código de la aplicación, puede usar bloques `try...except` en torno a las operaciones de biblioteca.

Para obtener más información sobre el tipo de excepciones que se pueden generar, consulte la documentación de la operación en cuestión.

Registro

Las bibliotecas de Azure más recientes usan la biblioteca estándar `logging` de Python para generar la salida del registro. Puede establecer el nivel de registro para bibliotecas individuales, grupos de bibliotecas o todas las bibliotecas. Una vez registrado un controlador de flujo de

registro, puede habilitar el registro para un objeto de cliente específico o una operación específica. Para más información, consulte [Registro en las bibliotecas de Azure](#).

Configuración de proxy

Para especificar un proxy, puede usar variables de entorno o argumentos opcionales. Para obtener más información, consulte [Configuración de servidores proxy](#).

Argumentosopcionales para los objetos y métodos de cliente

En la documentación de referencia de la biblioteca, a menudo se ve un argumento `**kwargs` o `**operation_config` en la firma de un constructor de objeto cliente o de un método de operación específico. Estos marcadores de posición indican que el objeto o el método en cuestión pueden admitir otros argumentos con nombre. Normalmente, la documentación de referencia indica los argumentos específicos que puede usar. También hay algunos argumentos generales que a menudo se admiten como se describe en las secciones siguientes.

Argumentos para bibliotecas basadas en `azure.core`

Estos argumentos se aplican a esas bibliotecas enumeradas en Python: [nuevas bibliotecas](#). Por ejemplo, este es un subconjunto de los argumentos de palabra clave para `azure-core`. Para obtener una lista completa, consulte el archivo LÉAME de GitHub para [azure-core](#).

[] Expandir tabla

Nombre	Tipo	Predeterminado	Descripción
<code>logging_enable</code>	<code>bool</code>	Falso	Habilita el registro. Para más información, consulte Registro en las bibliotecas de Azure .
<code>proxies</code>	<code>dict</code>	{}	Direcciones URL del servidor proxy. Para obtener más información, consulte Configuración de servidores proxy .
<code>use_env_settings</code>	<code>bool</code>	Cierto	Si es True, permite el uso de las variables de entorno <code>HTTP_PROXY</code> y <code>HTTPS_PROXY</code> para servidores proxy. Si es False, se omiten las variables de entorno. Para obtener más información, consulte Configuración de servidores proxy .
<code>connection_timeout</code>	<code>int</code>	300	Tiempo de espera en segundos para realizar una conexión a los puntos de conexión de la API REST de

Nombre	Tipo	Predeterminado	Descripción
			Azure.
read_timeout	int	300	Tiempo de espera en segundos para completar una operación de API REST de Azure (es decir, esperar una respuesta).
retry_total	int	10	Número de reintentos permitidos para las llamadas a la API REST. Utilice <code>retry_total=0</code> para deshabilitar los reintentos.
modo de reinicio	enum	exponential	Aplica el tiempo de reinicio de forma lineal o exponencial. Si es "single", los reintentos se realizarán a intervalos regulares. Si es "exponencial", cada reinicio espera el doble de tiempo que el reinicio anterior.

Las bibliotecas individuales no están obligadas a admitir ninguno de estos argumentos, por lo que siempre consulte la documentación de referencia de cada biblioteca para obtener detalles exactos. Además, cada biblioteca puede admitir otros argumentos. Por ejemplo, para argumentos específicos de clave de blob storage, ver el README de GitHub para [azure-storage-blob ↗](#).

Patrón JSON en línea para argumentos de objeto

Muchas operaciones dentro de las bibliotecas de Azure permiten expresar argumentos de objeto como objetos discretos o como JSON insertado.

Por ejemplo, supongamos que tiene un objeto `ResourceManagementClient` mediante el cual se crea un grupo de recursos con el método `create_or_update`. El segundo argumento para este método es de tipo `ResourceGroup`.

Para llamar al `create_or_update` método, puede crear una instancia discreta de `ResourceGroup` directamente con sus argumentos necesarios (`location` en este caso):

Python

```
# Provision the resource group.
rg_result = resource_client.resource_groups.create_or_update(
    "PythonSDKExample-rg",
    ResourceGroup(location="centralus")
)
```

Como alternativa, puede pasar los mismos parámetros como JSON insertado:

Python

```
# Provision the resource group.  
rg_result = resource_client.resource_groups.create_or_update(  
    "PythonAzureExample-rg", {"location": "centralus"}  
)
```

Cuando se usa JSON insertado, las bibliotecas de Azure convierten automáticamente el JSON insertado en el tipo de objeto adecuado para el argumento en cuestión.

Los objetos también pueden tener argumentos de objeto anidados, en cuyo caso también puede usar JSON anidado.

Por ejemplo, supongamos que tiene una instancia del objeto [KeyVaultManagementClient](#) y llama a [create_or_update](#). En este caso, el tercer argumento es de tipo [VaultCreateOrUpdateParameters](#), que contiene un argumento de tipo [VaultProperties](#). [VaultProperties](#), a su vez, contiene argumentos de objeto de tipo [Sku](#) y [list\[AccessPolicyEntry\]](#). Un [Sku](#) objeto contiene un [SkuName](#) objeto y cada [AccessPolicyEntry](#) uno contiene un [Permissions](#) objeto .

Para llamar a [begin_create_or_update](#) con objetos incrustados, use código como el siguiente (suponiendo [tenant_id](#) que ya están definidos , [object_id](#) y [LOCATION](#)). También puede crear los objetos necesarios antes de la llamada de función.

Python

```
# Provision a Key Vault using inline parameters  
poller = keyvault_client.vaults.begin_create_or_update(  
    RESOURCE_GROUP_NAME,  
    KEY_VAULT_NAME_A,  
    VaultCreateOrUpdateParameters(  
        location = LOCATION,  
        properties = VaultProperties(  
            tenant_id = tenant_id,  
            sku = Sku(  
                name="standard",  
                family="A"  
            ),  
            access_policies = [  
                AccessPolicyEntry(  
                    tenant_id = tenant_id,  
                    object_id = object_id,  
                    permissions = Permissions(  
                        keys = ['all'],  
                        secrets = ['all'])  
                )  
            ]  
        )
```

```
)  
)  
  
key_vault1 = poller.result()
```

La misma llamada que usa JSON en línea aparece de la siguiente manera:

Python

```
# Provision a Key Vault using inline JSON  
poller = keyvault_client.vaults.begin_create_or_update(  
    RESOURCE_GROUP_NAME,  
    KEY_VAULT_NAME_B,  
    {  
        'location': LOCATION,  
        'properties': {  
            'sku': {  
                'name': 'standard',  
                'family': 'A'  
            },  
            'tenant_id': tenant_id,  
            'access_policies': [{  
                'tenant_id': tenant_id,  
                'object_id': object_id,  
                'permissions': {  
                    'keys': ['all'],  
                    'secrets': ['all']  
                }  
            }]  
        }  
    }  
)  
  
key_vault2 = poller.result()
```

Dado que ambos formularios son equivalentes, puede elegir lo que prefiera e incluso mezclarlos. (El código completo de estos ejemplos se puede encontrar en [GitHub](#)).

Si el JSON no se ha formado correctamente, normalmente aparece el error "DeserializationError: No se puede deserializar en el objeto: type, AttributeError: 'str' object no tiene el atributo 'get'". Una causa común de este error es que se proporciona una sola cadena para una propiedad cuando la biblioteca espera un objeto JSON anidado. Por ejemplo, si se usa `'sku': 'standard'` en el ejemplo anterior, se genera este error porque el parámetro `sku` es un objeto `Sku` que espera el objeto JSON insertado, en este caso `{'name': 'standard'}`, que se asigna al tipo `SkuName` esperado.

Pasos siguientes

Ahora que comprende los patrones comunes para usar las bibliotecas de Azure para Python, consulte los siguientes ejemplos independientes para explorar escenarios de biblioteca cliente y administración específicos. Puede probar estos ejemplos en cualquier orden, ya que no son secuenciales o interdependientes.

- ejemplo de : [creación de un grupo de recursos](#)
- [Ejemplo: Uso de Azure Storage](#)
- [Ejemplo: Creación de una aplicación web e implementación de código](#)
- [Ejemplo: Creación y consulta de una base de datos](#)
- ejemplo de : [creación de una máquina virtual](#)
- [Uso de Azure Managed Disks con máquinas virtuales](#)
- [Completar una breve encuesta sobre el SDK de Azure para Python ↗](#)

Autenticación de aplicaciones de Python en los servicios de Azure mediante el SDK de Azure para Python

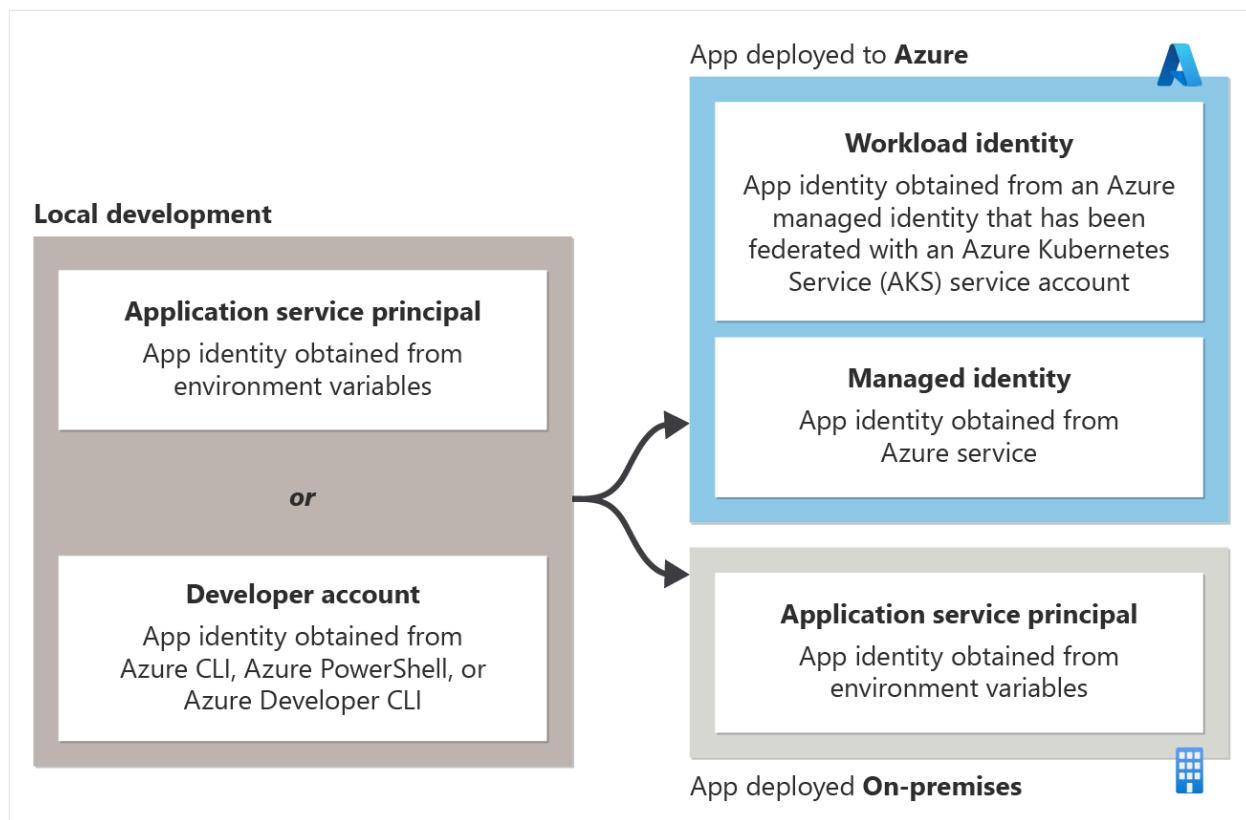
Artículo • 23/09/2024

Cuando una app necesita acceder a un recurso de Azure, como Azure Storage, Azure Key Vault o servicios de Azure AI, la app debe autenticarse en Azure. Este requisito se cumple para todas las apps, tanto si se implementan en Azure, como si se implementan en el entorno local o se están desarrollando en una estación de trabajo de desarrollador local. En este artículo se describen los enfoques recomendados para autenticar una aplicación en Azure cuando se usa el SDK de Azure para Python.

Enfoque de autenticación de aplicaciones recomendado

Use la autenticación basada en tokens en lugar de cadenas de conexión para las aplicaciones cuando se autentiquen en los recursos de Azure. La [biblioteca de clientes de Azure Identity para Python](#) proporciona clases que admiten la autenticación basada en tokens y permiten que las aplicaciones se autentiquen sin problemas en los recursos de Azure, tanto si la aplicación está en desarrollo local como si está implementada en Azure o en un servidor local.

El tipo específico de autenticación basada en tokens que usa una aplicación para autenticarse en los recursos de Azure depende de dónde se ejecute la aplicación. Los tipos de autenticación basada en tokens se muestran en el diagrama siguiente.



- **Cuando un desarrollador ejecuta una aplicación durante el desarrollo local:** la aplicación se autentica en Azure mediante una entidad de servicio de aplicación para el desarrollo local o las credenciales de Azure del desarrollador. Estas opciones se describen en la sección [Autenticación durante el desarrollo local](#).
- **Cuando una aplicación se hospeda en Azure:** la aplicación se autentica en los recursos de Azure mediante una identidad administrada. Esta opción se describe en la sección [Autenticación en entornos de servidor](#).
- **Cuando una aplicación se hospeda e implementa localmente:** la aplicación se autentica en los recursos de Azure mediante una entidad de servicio de aplicación. Esta opción se describe en la sección [Autenticación en entornos de servidor](#).

DefaultAzureCredential

La clase `DefaultAzureCredential` que proporciona la biblioteca cliente Azure Identity de Azure permite a las aplicaciones usar diferentes métodos de autenticación en función del entorno en el que se ejecuten. De esta manera, las aplicaciones se pueden promover desde el desarrollo local hasta los entornos de prueba y la producción sin cambios en el código.

Debe configurar el método de autenticación adecuado para cada entorno y `DefaultAzureCredential` detecta y utiliza automáticamente ese método de autenticación. Es preferible usar `DefaultAzureCredential` frente a la codificación manual de lógica condicional o marcas de características para utilizar diferentes métodos de autenticación en diferentes entornos.

Los detalles sobre el uso de la clase `DefaultAzureCredential` se describen en la sección [Uso de DefaultAzureCredential en una aplicación](#).

Ventajas de la autenticación basadas en token

Use la autenticación basada en tokens en lugar de usar cadenas de conexión al compilar aplicaciones para Azure. La autenticación basada en tokens ofrece las siguientes ventajas sobre la autenticación con cadenas de conexión:

- Los métodos de autenticación basados en tokens descritos en este artículo permiten establecer los permisos específicos que necesita la aplicación en el recurso de Azure. Esta práctica sigue el [principio de privilegios mínimos](#). Por el contrario, una cadena de conexión concede derechos completos al recurso de Azure.
- Cualquier persona o aplicación con una cadena de conexión puede conectarse a un recurso de Azure, pero los métodos de autenticación basados en tokens limitan el acceso al recurso solo a las aplicaciones destinadas a acceder al recurso.
- Con una identidad administrada, no hay ningún secreto de aplicación que almacenar. La aplicación es más segura porque no hay ninguna cadena de conexión o secreto de aplicación que pueda verse comprometido.
- El paquete [azure-identity](#) adquiere y administra los tokens de Microsoft Entra por ti. Esto hace que el uso de la autenticación basada en tokens sea tan fácil de usar como una cadena de conexión.

Límite el uso de cadenas de conexión a aplicaciones de prueba de concepto iniciales o prototipos de desarrollo que no accedan a datos confidenciales o de producción. De lo contrario, siempre se prefieren las clases de autenticación basadas en tokens disponibles en la biblioteca cliente Azure Identity al autenticarse en los recursos de Azure.

Autenticación en entornos de servidor

Cuando va a hospedar en un entorno de servidor, a cada app se le asigna una *identidad de aplicación* única por entorno en el que se ejecuta la app. En Azure, una identidad de app se representa mediante una *entidad de servicio*. Este tipo especial de entidad de seguridad identifica y autentica las aplicaciones en Azure. El tipo de entidad de servicio que se va a usar para la aplicación depende de dónde se esté ejecutando la aplicación:

 Expandir tabla

Método de autenticación	Descripción
Aplicaciones hospedadas en Azure	<p>Las aplicaciones hospedadas en Azure deben usar una <i>entidad de servicio de identidad administrada</i>. Las identidades administradas están diseñadas para representar la identidad de una aplicación hospedada en Azure y solo se pueden usar con aplicaciones hospedadas en Azure.</p> <p>Por ejemplo, a una aplicación web de Django hospedada en Azure App Service se le asignaría una identidad administrada. La identidad administrada asignada a la aplicación se usaría para autenticar la aplicación en otros servicios de Azure.</p> <p>Las aplicaciones que se ejecutan en Azure Kubernetes Service (AKS) pueden usar una credencial de identidad de carga de trabajo. Esta credencial se basa en una identidad administrada que tiene una relación de confianza con una cuenta de servicio de AKS.</p>
Aplicaciones hospedadas fuera de Azure (por ejemplo, aplicaciones locales)	<p>Las aplicaciones hospedadas fuera de Azure (por ejemplo, aplicaciones locales) que necesitan conectarse a los servicios de Azure deben usar una <i>entidad de servicio de aplicación</i>. Una entidad de servicio de aplicación representa la identidad de la aplicación en Azure y se crea a través del proceso de registro de la aplicación.</p> <p>Por ejemplo, considere una aplicación web de Django hospedada localmente que usa Azure Blob Storage. Crearía una entidad de servicio de aplicación para la aplicación mediante el proceso de registro de la aplicación. <code>AZURE_CLIENT_ID</code>, <code>AZURE_TENANT_ID</code> y <code>AZURE_CLIENT_SECRET</code> se almacenarían como variables de entorno para que la aplicación las lea en tiempo de ejecución y permitan que la aplicación se autentique en Azure mediante la entidad de servicio de la aplicación.</p>

[Más información sobre la autenticación de aplicaciones hospedadas en Azure](#)

[Más información sobre la autenticación de aplicaciones hospedadas fuera de Azure](#)

Autenticación durante el desarrollo local

Cuando una app se ejecuta en la estación de trabajo de un desarrollador durante el desarrollo local, debe autenticarse en los servicios de Azure que usa la aplicación. Hay dos estrategias principales para autenticar aplicaciones en Azure durante el desarrollo local:

Método de autenticación	Descripción
Crear objetos de entidad de servicio de aplicación dedicados que se usarán durante el desarrollo local.	<p>En este método, los objetos de <i>entidad de servicio de aplicación</i> dedicados se configuran mediante el proceso de registro de la aplicación para su uso durante el desarrollo local. A continuación, la identidad de la entidad de servicio se almacena como variables de entorno a las que va a acceder la aplicación cuando se ejecuta en el desarrollo local.</p> <p>Este método permite asignar los permisos de recursos específicos necesarios para la aplicación a los objetos de entidad de servicio utilizados por los desarrolladores durante el desarrollo local. Esta práctica garantiza que la aplicación solo tenga acceso a los recursos específicos que necesita y replica los permisos que tendrá la aplicación en producción.</p> <p>La desventaja de este enfoque es la necesidad de crear objetos de entidad de servicio independientes para cada desarrollador que trabaja en una aplicación.</p> <p style="text-align: center;">Obtenga información sobre la autenticación mediante entidades de servicio para desarrolladores</p>
Autenticar la aplicación en Azure con las credenciales del desarrollador durante el desarrollo local.	<p>En este método, un desarrollador debe iniciar sesión en Azure desde la CLI de Azure, Azure PowerShell o la CLI para desarrolladores de Azure en su estación de trabajo local. A continuación, la aplicación puede acceder a las credenciales del desarrollador desde el almacén de credenciales y usar esas credenciales para acceder a los recursos de Azure desde la aplicación.</p> <p>Este método tiene la ventaja de una configuración más sencilla porque un desarrollador solo necesita iniciar sesión en su cuenta de Azure a través de una de las herramientas de desarrollo antes mencionadas. El inconveniente de este enfoque es que es probable que la cuenta del desarrollador tenga más permisos de los que necesita la aplicación. Como resultado, la aplicación no replica con precisión los permisos con los que se ejecutará en producción.</p> <p style="text-align: center;">Obtenga información sobre la autenticación mediante cuentas de desarrollador</p>

Uso de DefaultAzureCredential en una aplicación

`DefaultAzureCredential` es una secuencia ordenada de mecanismos para autenticarse en Microsoft Entra ID. Cada mecanismo de autenticación es una clase que implementa el protocolo `TokenCredential` y se conoce como *credencial*. En tiempo de ejecución, `DefaultAzureCredential` intenta autenticarse con la primera credencial. Si esa credencial no puede adquirir un token de acceso, se intenta utilizar la siguiente credencial de la secuencia, y así sucesivamente hasta que se obtenga correctamente un token de acceso. De esta manera, la aplicación puede usar diferentes credenciales en distintos entornos sin implementar código específico del entorno.

Para usar `DefaultAzureCredential` en una app de Python, agregue el paquete [azure-identity](#) a la aplicación.

terminal

```
pip install azure-identity
```

Se accede a los servicios de Azure mediante clases de cliente especializadas de las distintas bibliotecas cliente del SDK de Azure. En el ejemplo de código siguiente se muestra cómo crear instancias de un objeto `DefaultAzureCredential` y usarlo con una clase de cliente del SDK de Azure. En este caso, es un objeto `BlobServiceClient` que se usa para acceder a Azure Blob Storage.

Python

```
from azure.identity import DefaultAzureCredential
from azure.storage.blob import BlobServiceClient

# Acquire a credential object
credential = DefaultAzureCredential()

blob_service_client = BlobServiceClient(
    account_url="https://<my_account_name>.blob.core.windows.net",
    credential=credential)
```

Cuando el código anterior se ejecuta en su estación de trabajo de desarrollo local, busca en las variables de entorno un principal de servicio de aplicación o en las herramientas de desarrollo instaladas localmente, como la CLI de Azure, un conjunto de credenciales de desarrollador. Se puede usar cualquier enfoque para autenticar la aplicación en los recursos de Azure durante el desarrollo local.

Cuando se implementa en Azure, este mismo código también puede autenticar la aplicación en los recursos de Azure. `DefaultAzureCredential` puede recuperar la configuración del entorno y las configuraciones de identidad gestionada para autenticarse automáticamente en los servicios de Azure.

Contenido relacionado

- Archivo LÉAME de la biblioteca de cliente de Azure Identity para Python en GitHub ↗
-

Comentarios

¿Le ha resultado útil esta página?

 Sí

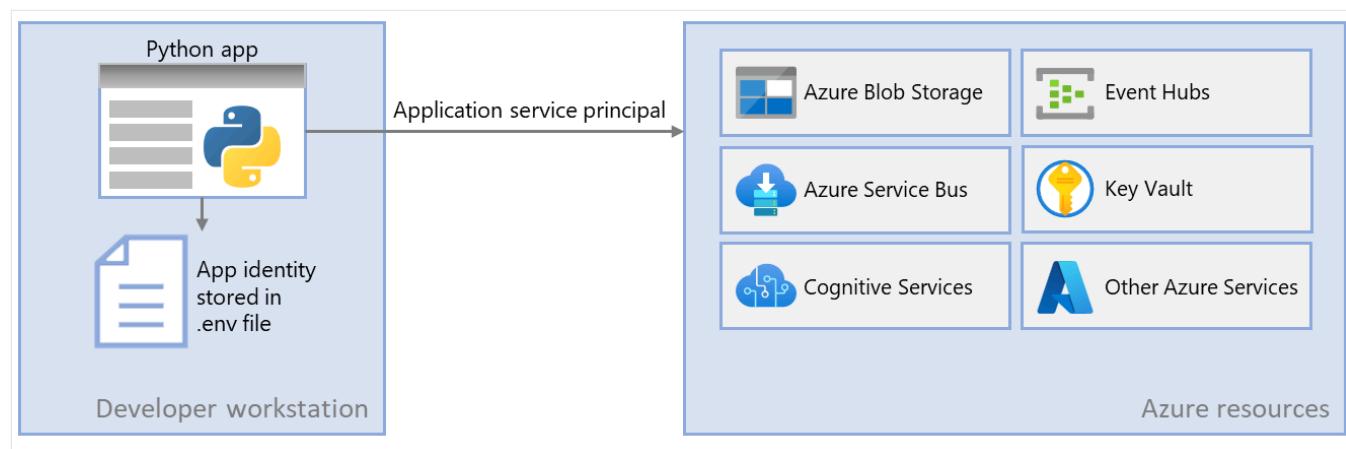
 No

[Proporcionar comentarios sobre el producto ↗](#) | [Obtener ayuda en Microsoft Q&A](#)

Autenticación de aplicaciones Python en servicios de Azure durante el desarrollo local mediante entidades de servicio

02/06/2025

Al compilar aplicaciones en la nube, los desarrolladores a menudo necesitan ejecutar y probar sus aplicaciones localmente. Incluso durante el desarrollo local, la aplicación debe autenticarse en los servicios de Azure con los que interactúa. En este artículo se explica cómo configurar identidades de entidad de servicio dedicadas específicamente para su uso durante el desarrollo local.



Las entidades de servicio de aplicaciones dedicadas para el desarrollo local admiten el principio de privilegios mínimos limitando el acceso solo a los recursos de Azure necesarios para la aplicación durante el desarrollo. El uso de una entidad de servicio de aplicación dedicada reduce el riesgo de acceso no deseado a otros recursos y ayuda a evitar errores relacionados con los permisos al realizar la transición a producción, donde los permisos más amplios podrían provocar problemas.

Al registrar aplicaciones para el desarrollo local en Azure, se recomienda:

- Crear registros de aplicaciones independientes para cada desarrollador: proporciona a cada desarrollador su propia entidad de servicio, evitando la necesidad de compartir credenciales y habilitando un control de acceso más granular.
- Crear registros de aplicaciones independientes para cada aplicación: esto garantiza que cada aplicación solo tenga los permisos que necesita, lo que reduce la posible superficie expuesta a ataques.

Para habilitar la autenticación durante el desarrollo local, establezca variables de entorno con las credenciales de la entidad de servicio de la aplicación. El SDK de Azure para Python detecta estas variables y las usa para autenticar las solicitudes en los servicios de Azure.

1: Registro de la aplicación en Azure

Los objetos principal de servicio de la aplicación son creados al registrar una aplicación en Azure. Este registro se puede realizar mediante Azure Portal o la CLI de Azure. El proceso de registro crea un registro de aplicación en Microsoft Entra ID (anteriormente Azure Active Directory) y genera un objeto de entidad de servicio para la aplicación. El objeto de principal de servicio se usa para autenticar la aplicación en los servicios de Azure. El proceso de registro de aplicaciones también genera un secreto de cliente (contraseña) para la aplicación. Este secreto se usa para autenticar la aplicación en los servicios de Azure. El secreto de cliente nunca se almacena en el control de código fuente, sino en un `.env` archivo del directorio de la aplicación. La `.env` aplicación lee el archivo en tiempo de ejecución para establecer variables de entorno que el SDK de Azure para Python usa para autenticar la aplicación. En los pasos siguientes se muestra cómo registrar una aplicación en Azure y crear una entidad de servicio para la aplicación. Los pasos se muestran para la CLI de Azure y Azure Portal.

CLI de Azure

Los comandos de la CLI de Azure se pueden ejecutar en [Azure Cloud Shell](#) o en una estación de trabajo con la [CLI de Azure instalada](#).

En primer lugar, use el comando `az ad sp create-for-rbac` para crear una nueva entidad de servicio para la aplicación. El comando también crea el registro de aplicación para la aplicación al mismo tiempo.

Azure CLI

```
SERVICE_PRINCIPAL_NAME=<service-principal-name>
az ad sp create-for-rbac --name $SERVICE_PRINCIPAL_NAME
```

La salida de este comando es similar a la siguiente. Anote estos valores o mantenga esta ventana abierta, ya que necesitará estos valores en los pasos siguientes y no podrá volver a ver el valor de la contraseña (secreto del cliente). Sin embargo, si es necesario, se puede agregar una nueva contraseña posteriormente sin invalidar la entidad de servicio ni las contraseñas existentes.

JSON

```
{
  "appId": "00001111-aaaa-2222-bbbb-3333cccc4444",
  "displayName": "<service-principal-name>",
  "password": "Ee5Ff~6Gg7.-Hh8Ii9Jj0Kk1Ll2Mm3_Nn40o5Pp6",
  "tenant": "aaaabbbb-0000-c000-1111-dddd2222eeee"
}
```

A continuación, debe obtener el `appId` valor y almacenarlo en una variable. Este valor se usa para establecer variables de entorno en el entorno de desarrollo local para que el SDK de Azure para Python pueda autenticarse en Azure mediante la entidad de servicio.

Azure CLI

```
APP_ID=$(az ad sp list \
--all \
--query "[?displayName=='$SERVICE_PRINCIPAL_NAME'].appId | [0]" \
--output tsv)
```

2 - Creación de un grupo de seguridad de Microsoft Entra para el desarrollo local

Dado que normalmente hay varios desarrolladores que trabajan en una aplicación, se recomienda crear un grupo de seguridad de Microsoft Entra para encapsular los roles (permisos) que necesita la aplicación en el desarrollo local, en lugar de asignar los roles a objetos de entidad de servicio individuales. Esto ofrece las ventajas que se indican a continuación:

- Todos los desarrolladores están seguros de tener asignados los mismos roles, ya que los roles se asignan en el nivel de grupo.
- Si se necesita un nuevo rol para la aplicación, solo es necesario agregarlo al grupo Microsoft Entra para la aplicación.
- Si un nuevo desarrollador se une al equipo, se crea una nueva entidad de servicio de aplicación para el desarrollador y esta se agrega al grupo, lo que garantiza que el desarrollador tiene los permisos adecuados para trabajar en la aplicación.

CLI de Azure

El comando `az ad group create` se usa para crear grupos de seguridad en Microsoft Entra ID. Los parámetros `--display-name` y `--main nickname` son obligatorios. El nombre proporcionado al grupo debe basarse en el nombre de la aplicación. También resulta útil incluir una cadena como "local-dev" en el nombre del grupo para indicar el propósito del grupo.

Azure CLI

```
GROUP_DISPLAY_NAME=<group-name>
GROUP_MAIL_NICKNAME=<group-mail-nickname>
GROUP_DESCRIPTION=<group-description>
az ad group create \
```

```
--display-name $GROUP_DISPLAY_NAME \
--mail-nickname $GROUP_MAIL_NICKNAME \
--description $GROUP_DESCRIPTION
```

Para agregar miembros al grupo, necesitará el id. de objeto de la entidad de servicio de la aplicación, que es diferente al id. de la aplicación. Use [az ad sp list](#) para enumerar las entidades de servicio disponibles. El comando de parámetro `--filter` acepta filtros de estilo de OData y se puede usar para filtrar la lista como se muestra. El parámetro `--query` limita las columnas solo a aquellas de interés.

Azure CLI

```
SP_OBJECT_ID=$(az ad sp list \
--filter "startswith(displayName, '$GROUP_DISPLAY_NAME')" \
--query "[0].id" \
--output tsv)
```

El comando [az ad group member add](#) se puede usar para agregar miembros a grupos.

Azure CLI

```
az ad group member add \
--group $GROUP_DISPLAY_NAME \
--member-id $SP_OBJECT_ID
```

! Nota

De forma predeterminada, la creación de grupos de seguridad de Microsoft Entra se limita a determinados roles con privilegios en un directorio. Si no puede crear un grupo, póngase en contacto con un administrador del directorio. Si no puede agregar miembros a un grupo existente, póngase en contacto con el propietario del grupo o con un administrador de directorios. Para obtener más información, consulte [Administración de grupos de Microsoft Entra y pertenencia a grupos](#).

3: Asignación de roles a la aplicación

A continuación, debe determinar qué roles (permisos) necesita la aplicación y en qué recursos para, a continuación, asignar dichos roles a la aplicación. En este ejemplo, los roles se asignan al grupo Microsoft Entra creado en el paso 2. Los roles se pueden asignar en el ámbito de recurso, grupo de recursos o suscripción. En este ejemplo se muestra cómo asignar roles en el

ámbito del grupo de recursos, ya que la mayoría de las aplicaciones agrupan todos sus recursos de Azure en un único grupo de recursos.

CLI de Azure

A un usuario, grupo o entidad de servicio de aplicación se le asigna el rol en Azure con el comando [az role assignment create](#). Puede especificar un grupo con su identificador de objeto. Puede especificar una entidad de servicio de aplicación con su appId.

Azure CLI

```
RESOURCE_GROUP_NAME=<resource-group-name>
SUBSCRIPTION_ID=$(az account show --query id --output tsv)
ROLE_NAME=<role-name>
az role assignment create \
--assignee "$APP_ID" \
--scope
"./subscriptions/$SUBSCRIPTION_ID/resourceGroups/$RESOURCE_GROUP_NAME" \
--role "$ROLE_NAME"
```

! [! NOTA] Para evitar que Git Bash trate /subscriptions/... como ruta de acceso de archivo, anteponga ./ a la cadena para el `scope` parámetro y use comillas dobles alrededor de toda la cadena.

Para obtener los nombres de roles que se pueden asignar, use el comando [az role definition list](#).

Azure CLI

```
az role definition list \
--query "sort_by([].{roleName:roleName, description:description}, \
&roleName)" \
--output table
```

Por ejemplo, para permitir que la entidad de servicio de aplicación con el appId de `00001111-aaaa-2222-bbbb-3333cccc4444` lea, escriba y elimine el acceso a los contenedores y datos de blobs de Azure Storage para todas las cuentas de almacenamiento del grupo de recursos `msdocs-python-sdk-auth-example` en la suscripción con ID `aaaa0a0a-bb1b-cc2c-dd3d-eeeeeee4e4e4e`, asignaría el rol de *Colaborador de datos de blobs de almacenamiento* a la entidad de servicio de la aplicación mediante el siguiente comando.

Azure CLI

```
az role assignment create --assignee 00001111-aaaa-2222-bbbb-3333cccc4444 \
--scope "./subscriptions/aaaa0a0a-bb1b-cc2c-dd3d-
eeeeee4e4e/resourceGroups/msdocs-python-sdk-auth-example" \
--role "Storage Blob Data Contributor"
```

Para obtener información sobre cómo asignar permisos en el nivel de recurso o suscripción mediante la CLI de Azure, consulte el artículo [Asignación de roles de Azure mediante la CLI de Azure](#).

4 - Configuración de variables del entorno de desarrollo local

El objeto `DefaultAzureCredential` buscará información sobre la entidad de servicio en un conjunto de variables de entorno en tiempo de ejecución. Dado que la mayoría de los desarrolladores trabajan en varias aplicaciones, se recomienda usar un paquete como [python-dotenv](#) para acceder al entorno desde un archivo `.env` almacenado en el directorio de la aplicación durante el desarrollo. Esto limita las variables de entorno que se usan para autenticar la aplicación en Azure de modo que esta aplicación solo las pueda usar.

El archivo `.env` nunca se registra en el control de código fuente, ya que contiene la clave secreta de aplicación para Azure. El archivo [.gitignore](#) estándar para Python excluye automáticamente el archivo `.env` de la protección.

Para usar el paquete `python-dotenv`, instale primero el paquete en la aplicación.

terminal

```
pip install python-dotenv
```

A continuación, cree un archivo `.env` en el directorio raíz de la aplicación. Establezca los valores de las variables de entorno con los valores obtenidos del proceso de registro de aplicaciones de la siguiente manera:

- `AZURE_CLIENT_ID` → Valor del id. de la aplicación.
- `AZURE_TENANT_ID` → Valor del id. de inquilino.
- `AZURE_CLIENT_SECRET` → Contraseña o credencial generada para la aplicación.

Bash

```
AZURE_CLIENT_ID=00001111-aaaa-2222-bbbb-3333cccc4444
AZURE_TENANT_ID=aaaabbbb-0000-cccc-1111-dddd2222eeee
```

```
AZURE_CLIENT_SECRET=Ee5Ff~6Gg7.-Hh8Ii9Jj0Kk1Ll2Mm3_Nn4Oo5Pp6
```

Por último, en el código de inicio de la aplicación, use la biblioteca `python-dotenv` para leer las variables de entorno del archivo `.env` al iniciarse.

Python

```
from dotenv import load_dotenv

if ( os.environ['ENVIRONMENT'] == 'development'):
    print("Loading environment variables from .env file")
    load_dotenv(".env")
```

5: Implementación de DefaultAzureCredential en la aplicación

Para autenticar objetos de cliente del SDK de Azure en Azure, la aplicación debe usar la clase `DefaultAzureCredential` del paquete `azure.identity`. En este escenario, `DefaultAzureCredential` detectará que las variables de entorno `AZURE_CLIENT_ID`, `AZURE_TENANT_ID` y `AZURE_CLIENT_SECRET` están establecidas y las leerá para obtener la información de la entidad de servicio de la aplicación con la que conectarse a Azure.

Empiece agregando el paquete `azure.identity` a la aplicación.

terminal

```
pip install azure-identity
```

A continuación, para cualquier código Python que cree un objeto cliente del SDK de Azure en su aplicación, querrá:

1. Importar la clase `DefaultAzureCredential` desde el módulo `azure.identity`.
2. Crear un objeto `DefaultAzureCredential`.
3. Pasar el objeto `DefaultAzureCredential` al constructor de objetos de cliente del SDK de Azure.

En el segmento de código siguiente se muestra un ejemplo de esto.

Python

```
from azure.identity import DefaultAzureCredential
from azure.storage.blob import BlobServiceClient
```

```
# Acquire a credential object
token_credential = DefaultAzureCredential()

blob_service_client = BlobServiceClient(
    account_url="https://<my_account_name>.blob.core.windows.net",
    credential=token_credential)
```

Autenticación de aplicaciones Python en servicios de Azure durante el desarrollo local mediante cuentas de desarrollador

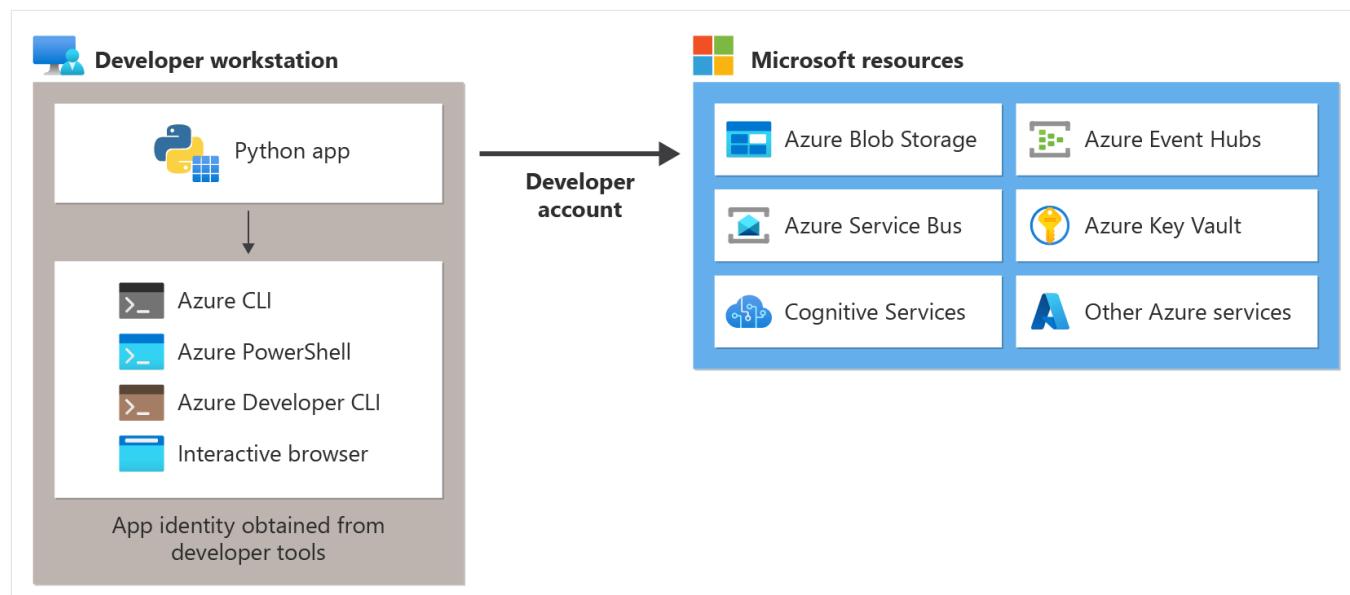
31/05/2025

Al desarrollar aplicaciones en la nube, los desarrolladores suelen compilar, probar y depurar su código localmente antes de implementarlo en Azure. Sin embargo, incluso durante el desarrollo local, la aplicación debe autenticarse con cualquier servicio de Azure con el que interactúe, como Key Vault, Storage o bases de datos.

En este artículo se muestra cómo configurar la aplicación para usar las credenciales de Azure del desarrollador para la autenticación durante el desarrollo local. Este enfoque permite una experiencia de desarrollo sin problemas y segura sin insertar secretos ni escribir lógica específica del entorno.

Introducción a la autenticación de desarrollo local mediante cuentas de desarrollador

Al desarrollar una aplicación que use la biblioteca de identidades de Azure para Python, puede autenticarse en los servicios de Azure durante el desarrollo local mediante la cuenta de Azure del desarrollador. Este enfoque suele ser la manera más sencilla de autenticarse en los servicios de Azure durante el desarrollo local, ya que no requiere crear ni administrar entidades de servicio ni secretos.



Para permitir que una aplicación se autentique en Azure durante el desarrollo local mediante las propias credenciales de Azure del desarrollador, el desarrollador primero debe iniciar sesión

con una de las herramientas de línea de comandos compatibles:

- CLI de Azure (`az login`)
- CLI para desarrolladores de Azure (`azd login`)
- Azure PowerShell (`Connect-AzAccount`)

Una vez iniciada la sesión, la biblioteca de identidades de Azure para Python puede detectar automáticamente la sesión activa y recuperar los tokens necesarios de la memoria caché de credenciales. Esta funcionalidad permite que la aplicación se autentique en los servicios de Azure como el usuario que ha iniciado sesión, sin necesidad de ninguna configuración adicional ni secretos codificados.

Este comportamiento se habilita cuando se usa `DefaultAzureCredential`, que recurre de forma transparente a las credenciales basadas en CLI en entornos locales.

El uso de las credenciales de Azure de un desarrollador que ha iniciado sesión es la configuración más sencilla para el desarrollo local. Aprovecha la cuenta de Azure existente de cada miembro del equipo, lo que permite el acceso sin problemas a los servicios de Azure sin necesidad de configuración adicional.

Sin embargo, las cuentas de desarrollador suelen tener permisos más amplios que la aplicación en producción. Estos permisos más amplios pueden provocar incoherencias en las pruebas o permitir accidentalmente las operaciones que la aplicación no estaría autorizada para realizar en un entorno de producción. Para reflejar estrechamente los permisos de producción y mejorar la posición de seguridad, puede crear entidades de servicio específicas de la aplicación para el desarrollo local. Estas identidades:

- Solo se pueden asignar los roles y permisos que necesita la aplicación.
- Principio de privilegios mínimos
- Ofrecer pruebas coherentes del comportamiento relacionado con el acceso entre entornos

Los desarrolladores pueden configurar el entorno local para usar el principal de servicio a través de variables de entorno y `DefaultAzureCredential` lo detecta automáticamente. Para más información, consulte el artículo [Autenticación de aplicaciones de Python en servicios de Azure durante el desarrollo local mediante entidades de servicio](#).

1 - Creación de un grupo de seguridad de Microsoft Entra para el desarrollo local

En la mayoría de los escenarios de desarrollo, varios desarrolladores contribuyen a la misma aplicación. Para simplificar el control de acceso y garantizar permisos coherentes en todo el

equipo, se recomienda crear primero un grupo de seguridad de Microsoft Entra específicamente para las necesidades de desarrollo local de la aplicación.

La asignación de roles de Azure en el nivel de grupo, en lugar de a usuarios individuales, ofrece varias ventajas clave:

- Asignaciones de roles coherentes

Todos los desarrolladores del grupo heredan automáticamente los mismos roles y permisos, lo que garantiza un entorno de desarrollo uniforme.

- Administración simplificada de roles

Cuando la aplicación requiere un nuevo rol, solo tiene que agregarlo una vez al grupo. No es necesario actualizar los permisos de usuario individuales.

- Incorporación sencilla

A los nuevos desarrolladores se les pueden conceder los permisos necesarios simplemente agregándolos al grupo. No se requieren asignaciones de roles manuales.

Si su organización ya tiene un grupo de seguridad de Microsoft Entra adecuado para el equipo de desarrollo, puede reutilizarlo. De lo contrario, puede crear un nuevo grupo específicamente para la aplicación.

CLI de Azure

Para crear un grupo de seguridad en Microsoft Entra ID, use el comando [az ad group create](#) en Azure CLI.

Este comando requiere los parámetros siguientes:

`--display-name`: un nombre fácil de usar para el grupo.

`--mail-nickname`: un identificador único usado para el correo electrónico y la referencia interna

Se recomienda basar el nombre del grupo en el nombre de la aplicación e incluir un sufijo como `-local-dev` para indicar claramente su propósito.

Bash

```
#!/bin/bash
az ad group create \
--display-name MyDisplay \
```

```
--mail-nickname MyDisplay \
--description "<group-description>"
```

PowerShell

```
# PowerShell syntax
az ad group create ` 
    --display-name MyDisplay ` 
    --mail-nickname MyDisplay ` 
    --description "<group-description>"
```

Después de ejecutar el `az ad group create` comando, copie el valor de la `id` propiedad desde la salida del comando. Necesita el `Object ID` del grupo de seguridad de Microsoft Entra para asignar roles en pasos posteriores de este artículo. Para recuperar de nuevo `Object ID` más tarde, use el siguiente comando `az ad group show`: `az ad group show --group "my-app-local-dev" --query id --output tsv`.

Para agregar un usuario al grupo, primero debe obtener la `Object ID` de la cuenta de usuario de Azure que desea agregar. Use el comando `az ad user list` con el parámetro `--filter` para buscar un usuario específico por nombre visible. El `--query` parámetro ayuda a limitar la salida a los campos pertinentes:

Bash

```
#!/bin/bash
az ad user list \
--filter "startswith(displayName, 'Bob')" \
--query "[].{objectId:id, displayName:displayName}" \
--output table
```

PowerShell

```
# PowerShell syntax
az ad user list ` 
    --filter "startswith(displayName, 'Bob')" ` 
    --query "[].{objectId:id, displayName:displayName}" ` 
    --output table
```

Una vez que tenga el `Object ID` del usuario, puede agregarlos al grupo mediante el comando `az ad group member add`.

Bash

```
#!/bin/bash
az ad group member add \
```

```
--group <group-name> \
--member-id <object-id>
```

PowerShell

```
# PowerShell syntax
az ad group member add `

  --group <group-name> `

  --member-id <object-id>
```

⚠ Nota

De forma predeterminada, la creación de grupos de seguridad de Microsoft Entra se limita a determinados roles con privilegios en un directorio. Si no puede crear un grupo, póngase en contacto con un administrador del directorio. Si no puede agregar miembros a un grupo existente, póngase en contacto con el propietario del grupo o con un administrador de directorios. Para obtener más información, consulte [Administración de grupos de Microsoft Entra y pertenencia a grupos](#).

2: Asignación de roles al grupo de Microsoft Entra

Después de crear el grupo de seguridad de Microsoft Entra y agregar miembros, el siguiente paso es determinar qué roles (permisos) requiere la aplicación y asignar esos roles al grupo en el ámbito adecuado.

- Determinación de los roles necesarios

Identifique los roles que la aplicación necesita para funcionar. Algunos ejemplos comunes son:

- Usuario de Azure Key Vault Secrets: para leer secretos de Azure Key Vault
- Colaborador de datos de cola de Storage: para enviar mensajes a Azure Queue Storage

Consulte las definiciones de roles integradas para obtener más opciones.

- Elegir un ámbito para la asignación de roles

Los roles se pueden asignar en distintos ámbitos:

- Nivel de recurso (por ejemplo, una sola bóveda de claves o cuenta de almacenamiento)
- Nivel de grupo de recursos (recomendado para la mayoría de las aplicaciones)
- Nivel de suscripción (use con precaución: acceso más amplio)

En este ejemplo, asignamos roles en el ámbito del grupo de recursos, que es típico cuando todos los recursos de aplicación se agrupan en un grupo de recursos.

CLI de Azure

A un usuario, grupo o entidad de servicio de aplicación se le asigna el rol en Azure con el comando [az role assignment create](#). Puede especificar un grupo con su `Object ID`.

Bash

```
#!/bin/bash
az role assignment create --assignee <objectId> \
    --scope /subscriptions/<subscriptionId>/resourceGroups/<resourceGroupName>
\
    --role "<roleName>"
```

PowerShell

```
# PowerShell syntax
az role assignment create ` 
    --assignee <objectId> ` 
    --scope /subscriptions/<subscriptionId>/resourceGroups/<resourceGroupName>
` 
    --role "<roleName>"
```

Para obtener los nombres de roles que se pueden asignar, use el comando [az role definition list](#).

Bash

```
#!/bin/bash
az role definition list --query "sort_by([].{roleName:roleName,
description:description}, &roleName)" --output table
```

Powershell

```
# PowerShell syntax
az role definition list --query "sort_by([].{roleName:roleName,
description:description}, &roleName)" --output table
```

Para conceder acceso de lectura, escritura y eliminación a contenedores y datos de blobs de Azure Storage para todas las cuentas de almacenamiento de un grupo de recursos específico, asigne el rol Colaborador de datos de Storage Blob al grupo de seguridad de Microsoft Entra.

Bash

```
#!/bin/bash
az role assignment create --assignee bbbbbbbb-1111-2222-3333-ccccccccccc \
    --scope /subscriptions/aaaa0a0a-bb1b-cc2c-dd3d-
eeeeee4e4e/resourceGroups/msdocs-python-sdk-auth-example \
    --role "Storage Blob Data Contributor"
```

Powershell

```
# PowerShell syntax
az role assignment create --assignee bbbbbbbb-1111-2222-3333-ccccccccccc ` \
    --scope /subscriptions/aaaa0a0a-bb1b-cc2c-dd3d-
eeeeee4e4e/resourceGroups/msdocs-python-sdk-auth-example ` \
    --role "Storage Blob Data Contributor"
```

Para obtener información sobre cómo asignar permisos en el nivel de recurso o suscripción mediante la CLI de Azure, consulte el artículo [Asignación de roles de Azure mediante la CLI de Azure](#).

3 - Inicio de sesión en Azure mediante la CLI de Azure, Azure PowerShell, la CLI para desarrolladores de Azure o en un explorador

Para autenticarse con su cuenta de Azure, elija uno de los métodos siguientes:

CLI de Azure

Abra un terminal en la estación de trabajo del desarrollador e inicie sesión en Azure desde la [CLI de Azure](#).

Azure CLI

```
az login
```

4: Implementación de DefaultAzureCredential en la aplicación

Para autenticar objetos de cliente del SDK de Azure con Azure, la aplicación debe usar la `DefaultAzureCredential` clase del `azure-identity` paquete. Este es el método de autenticación recomendado para las implementaciones de desarrollo y producción locales.

En un escenario de desarrollo local, `DefaultAzureCredential` funciona comprobando secuencialmente los orígenes de autenticación disponibles. En concreto, busca sesiones activas en las siguientes herramientas:

- CLI de Azure (`az login`)
- Azure PowerShell (`Connect-AzAccount`)
- Azure Developer CLI (`azd auth login`)

Si el desarrollador ha iniciado sesión en Azure con cualquiera de estas herramientas, `DefaultAzureCredential` detecta automáticamente la sesión y usa esas credenciales para autenticar la aplicación con los servicios de Azure. Esto permite a los desarrolladores autenticarse de forma segura sin almacenar secretos ni modificar código para distintos entornos.

Empiece agregando el paquete [azure.identity](#) a la aplicación.

Consola

```
pip install azure-identity
```

A continuación, para cualquier código de Python que cree un objeto cliente del Azure SDK en tu aplicación, quieres:

1. Importar la clase `DefaultAzureCredential` desde el módulo `azure.identity`.
2. Crear un objeto `DefaultAzureCredential`.
3. Pasar el objeto `DefaultAzureCredential` al constructor de objetos de cliente del SDK de Azure.

En el segmento de código siguiente se muestra un ejemplo de estos pasos.

Python

```
from azure.identity import DefaultAzureCredential
from azure.storage.blob import BlobServiceClient

# Acquire a credential object
token_credential = DefaultAzureCredential()

blob_service_client = BlobServiceClient(
    account_url="https://<my_account_name>.blob.core.windows.net",
    credential=token_credential)
```


Autenticación de aplicaciones hospedadas en recursos de Azure con el SDK de Azure para Python

Artículo • 15/10/2024

Al hospedar una aplicación en Azure mediante servicios como Azure App Service, Azure Virtual Machines o Azure Container Instances, el enfoque recomendado para autenticar una aplicación en los recursos de Azure es con la [identidad administrada](#).

Una identidad administrada proporciona una identidad para la aplicación para que pueda conectarse a otros recursos de Azure sin necesidad de usar una clave secreta u otro secreto de aplicación. Internamente, Azure conoce la identidad de la aplicación y a qué recursos se le permite conectarse. Azure usa esta información para obtener automáticamente tokens de Microsoft Entra para que la aplicación permita la conexión a otros recursos de Azure, todo ello sin tener que administrar ningún secreto de aplicación.

ⓘ Nota

Las aplicaciones que se ejecutan en Azure Kubernetes Service (AKS) pueden usar una identidad de carga de trabajo para autenticarse con recursos de Azure. En AKS, una identidad de carga de trabajo representa una relación de confianza entre una identidad administrada y una cuenta de servicio de Kubernetes. Si una aplicación implementada en AKS está configurada con una cuenta de servicio de Kubernetes en dicha relación,

`DefaultAzureCredential` autentica la aplicación en Azure mediante la identidad administrada. La autenticación mediante una identidad de carga de trabajo se describe en [Uso de ID de carga de trabajo de Microsoft Entra con Azure Kubernetes Service](#). Para conocer los pasos sobre cómo configurar la identidad de carga de trabajo, consulte [Implementación y configuración de la identidad de carga de trabajo en un clúster de Azure Kubernetes Service \(AKS\)](#).

Tipos de identidad administrada

Hay dos tipos de identidades administradas:

- **Identidades administradas asignadas por el sistema:** este tipo de identidad administrada se proporciona mediante un recurso de Azure y está vinculado a dicho recurso. Al habilitar la identidad administrada en un recurso de Azure, obtendrá una identidad administrada asignada por el sistema para ese recurso. Una identidad administrada asignada por el sistema está vinculada al ciclo de vida del recurso de Azure al que está

asociada. Por tanto, cuando se elimina el recurso, Azure elimina automáticamente la identidad. Dado que todo lo que tiene que hacer es habilitar la identidad administrada para el recurso de Azure que hospeda el código, este enfoque es el tipo de identidad administrada más fácil de usar.

- **Identidades administradas asignadas por el usuario:** también puede crear una identidad administrada como un recurso independiente de Azure. Este enfoque se usa con mayor frecuencia cuando la solución tiene varias cargas de trabajo que se ejecutan en varios recursos de Azure que deben compartir la misma identidad y los mismos permisos. Por ejemplo, si la solución tiene componentes que se ejecutan en varias instancias de App Service y de máquina virtual que necesitan acceso al mismo conjunto de recursos de Azure, tiene sentido una identidad administrada asignada por el usuario que se use en esos recursos.

En este artículo se describen los pasos para habilitar y usar una identidad administrada asignada por el sistema para una aplicación. Si necesita usar una identidad administrada asignada por el usuario, consulte el artículo [Administración de identidades administradas asignadas por el usuario](#) para obtener información sobre cómo crear una identidad administrada asignada por el usuario.

1: Activación de la identidad administrada en el recurso de Azure que hospeda la aplicación

El primer paso es habilitar la identidad administrada en el recurso de Azure que hospeda la aplicación. Por ejemplo, si hospeda una aplicación de Django mediante Azure App Service, debe habilitar la identidad administrada para la aplicación web de App Service que hospeda la aplicación. Si usas una máquina virtual para hospedar la aplicación, debe permitir que la máquina virtual use la identidad administrada.

Puede habilitar la identidad administrada para que se use para un recurso de Azure mediante Azure Portal o la CLI de Azure.

CLI de Azure

Los comandos de la CLI de Azure se pueden ejecutar en [Azure Cloud Shell](#) o en una estación de trabajo con la [CLI de Azure instalada](#).

Los comandos de la CLI de Azure que se usan para habilitar la identidad administrada para un recurso de Azure tienen el formato `az <command-group> identity --resource-group <resource-group-name> --name <resource-name>`. A continuación se muestran comandos específicos para los servicios populares de Azure.

Azure App Service

Azure CLI

```
az webapp identity assign --resource-group <resource-group-name> --name  
<web-app-name>
```

El resultado tendrá un aspecto similar al siguiente.

JSON

```
{  
  "principalId": "aaaaaaaa-bbbb-cccc-1111-222222222222",  
  "tenantId": "aaaabbbb-0000-cccc-1111-dddd2222eeee",  
  "type": "SystemAssigned",  
  "userAssignedIdentities": null  
}
```

El valor `principalId` es el identificador único de la identidad administrada. Mantenga una copia de este resultado, ya que necesitará estos valores en el paso siguiente.

2: Asignación de roles a la identidad administrada

A continuación, debe determinar qué roles (permisos) necesita la aplicación y asignar la identidad administrada a dichos roles en Azure. Se pueden asignar roles a una identidad administrada en un ámbito de recurso, grupo de recursos o suscripción. En este ejemplo se muestra cómo asignar roles en el ámbito del grupo de recursos, ya que la mayoría de las aplicaciones agrupan todos sus recursos de Azure en un único grupo de recursos.

CLI de Azure

En Azure, los roles se asignan a una identidad administrada mediante el comando [az role assignment create](#). Para el usuario asignado, use el `principalId` que copió en el paso 1.

Azure CLI

```
az role assignment create --assignee <managedIdentityprincipalId> \  
  --scope /subscriptions/<subscriptionId>/resourceGroups/<resourceGroupName>  
  \  
  --role "<rolename>"
```

Para obtener los nombres de roles a los que se puede asignar una entidad de servicio, use el comando [az role definition list](#).

Azure CLI

```
az role definition list \
--query "sort_by([].{roleName:roleName, description:description},
&roleName)" \
--output table
```

Por ejemplo, para permitir que la identidad administrada con ID de `aaaaaaaa-bbbb-cccc-1111-222222222222` lea, escriba y elimine el acceso a los contenedores y datos de blobs de Azure Storage para todas las cuentas de almacenamiento del grupo de recursos `msdocs-python-sdk-auth-example` en la suscripción con ID `aaaa0a0a-bb1b-cc2c-dd3d-eeeeee4e4e4e`, asignaría el rol de *Colaborador de datos de blobs de almacenamiento* a la entidad de servicio de la aplicación mediante el siguiente comando.

Azure CLI

```
az role assignment create --assignee aaaaaaaaa-bbbb-cccc-1111-222222222222 \
--scope /subscriptions/aaaa0a0a-bb1b-cc2c-dd3d-
eeeeee4e4e4e/resourceGroups/msdocs-python-sdk-auth-example \
--role "Storage Blob Data Contributor"
```

Para obtener información sobre cómo asignar permisos en el nivel de recurso o suscripción mediante la CLI de Azure, consulte el artículo [Asignación de roles de Azure mediante la CLI de Azure](#).

3: Implementación de DefaultAzureCredential en la aplicación

Cuando el código se ejecuta en Azure y la identidad administrada se ha habilitado en el recurso de Azure que hospeda la aplicación, `DefaultAzureCredential` determina las credenciales que se usarán en el orden siguiente:

1. Comprueba el entorno de una entidad de servicio tal como se define en las variables de entorno `AZURE_CLIENT_ID`, `AZURE_TENANT_ID` y `AZURE_CLIENT_SECRET` o `AZURE_CLIENT_CERTIFICATE_PATH` y (opcionalmente) `AZURE_CLIENT_CERTIFICATE_PASSWORD`.
2. Comprueba los parámetros de palabra clave de una identidad administrada asignada por el usuario. Puede pasar una identidad administrada asignada por el usuario especificando su identificador de cliente en el parámetro `managed_identity_client_id`.

3. Comprueba la variable de entorno `AZURE_CLIENT_ID` para el identificador de cliente de una identidad administrada asignada por el usuario.
4. Usa la identidad administrada asignada por el sistema para el recurso de Azure si está habilitada.

Puede excluir las identidades administradas de la credencial estableciendo el parámetro de palabra clave `exclude_managed_identity_credential` `True`.

En este artículo, se usa la identidad administrada asignada por el sistema para una aplicación web de App de Azure Service, por lo que no es necesario configurar una identidad administrada en el entorno ni pasarlala como parámetro. En los pasos siguientes se muestra cómo usar `DefaultAzureCredential`.

Primero, agregue el paquete `azure.identity` a la aplicación.

terminal

```
pip install azure-identity
```

A continuación, para cualquier código Python que cree un objeto de cliente del SDK de Azure en la aplicación, querrá:

1. Importar la clase `DefaultAzureCredential` desde el módulo `azure.identity`.
2. Crear un objeto `DefaultAzureCredential`.
3. Pasar el objeto `DefaultAzureCredential` al constructor de objetos de cliente del SDK de Azure.

En el segmento de código siguiente se muestra un ejemplo de estos pasos.

Python

```
from azure.identity import DefaultAzureCredential
from azure.storage.blob import BlobServiceClient

# Acquire a credential object
token_credential = DefaultAzureCredential()

blob_service_client = BlobServiceClient(
    account_url="https://<my_account_name>.blob.core.windows.net",
    credential=token_credential)
```

Como se describe en el artículo [Información general sobre la autenticación del SDK de Azure para Python](#), `DefaultAzureCredential` admite varios métodos de autenticación y determina el método de autenticación que se usa en tiempo de ejecución. La ventaja de este enfoque es que la aplicación puede usar diferentes métodos de autenticación en distintos entornos sin

implementar código específico del entorno. Cuando el código anterior se ejecute en la estación de trabajo durante el desarrollo local, `DefaultAzureCredential` usará una entidad de servicio de aplicación, según lo determinado por la configuración del entorno o las credenciales de la herramienta de desarrollador para autenticarse con otros recursos de Azure. Por lo tanto, se puede usar el mismo código para autenticar la aplicación en los recursos de Azure durante el desarrollo local y cuando se implementa en Azure.

Autenticación en recursos de Azure desde aplicaciones Python hospedadas en el entorno local

05/06/2025

Las aplicaciones hospedadas fuera de Azure (por ejemplo, locales o en un centro de datos de terceros) deben usar una entidad de servicio de aplicación para autenticarse en Azure al acceder a los recursos de Azure. Los objetos de entidad de servicio de aplicación se crean mediante el proceso de registro de aplicaciones en Azure. Cuando se crea una entidad de servicio de aplicación, se generará un id. de cliente y un secreto de cliente para la aplicación. El identificador de cliente, el secreto de cliente y el identificador de inquilino se almacenan en variables de entorno para que el SDK de Azure para Python las pueda usar para autenticar la aplicación en Azure en tiempo de ejecución.

Debe crear un registro de aplicación diferente para cada entorno en el que se hospeda la aplicación. Esto permite configurar permisos de recursos específicos del entorno para cada entidad de servicio y asegurarse de que una aplicación implementada en un entorno no se comunique con los recursos de Azure que forman parte de otro entorno.

1: Registro de la aplicación en Azure AD

Una aplicación se puede registrar en Azure mediante Azure Portal o la CLI de Azure.

```
CLI de Azure
Azure CLI
APP_NAME=<app-name>
az ad sp create-for-rbac --name $APP_NAME
```

La salida del comando es similar a la siguiente. Anote estos valores o mantenga esta ventana abierta, ya que necesitará estos valores en los pasos siguientes y no podrá volver a ver el valor de la contraseña (secreto del cliente).

```
JSON
{
  "appId": "00001111-aaaa-2222-bbbb-3333cccc4444",
  "displayName": "msdocs-python-sdk-auth-prod",
  "password": "Ee5FF~6Gg7.-Hh8Ii9Jj0Kk1Ll2Mm3_Nn40o5Pp6",
```

```
        "tenant": "aaaabbbb-0000-cccc-1111-dddd2222eeee"  
    }
```

A continuación, debe obtener el `appId` valor y almacenarlo en una variable. Este valor se usa para establecer variables de entorno en el entorno de desarrollo local para que el SDK de Azure para Python pueda autenticarse en Azure mediante la entidad de servicio.

Azure CLI

```
APP_ID=$(az ad sp create-for-rbac \  
--name $APP_NAME --query appId --output tsv)
```

2: Asignación de roles a la entidad de servicio de la aplicación

A continuación, debe determinar qué roles (permisos) necesita la aplicación y en qué recursos y asignar dichos roles a la aplicación. Los roles se pueden asignar a un rol en el ámbito de recurso, grupo de recursos o suscripción. En este ejemplo se muestra cómo asignar roles a la entidad de servicio en el ámbito del grupo de recursos, ya que la mayoría de las aplicaciones agrupan todos sus recursos de Azure en un único grupo de recursos.

CLI de Azure

Los roles se asignan a las entidades de servicio mediante el comando [az role assignment create](#).

Azure CLI

```
RESOURCE_GROUP_NAME=<resource-group-name>  
SUBSCRIPTION_ID=$(az account show --query id --output tsv)  
ROLE_NAME=<role-name>  
  
az role assignment create \  
--assignee "$APP_ID" \  
--scope  
"./subscriptions/$SUBSCRIPTION_ID/resourceGroups/$RESOURCE_GROUP_NAME" \  
--role "$ROLE_NAME"
```

[! NOTA] Para evitar que Git Bash trate /subscriptions/... como ruta de acceso de archivo, anteponga ./ a la cadena para el `scope` parámetro y use comillas dobles alrededor de toda la cadena.

Para obtener los nombres de roles a los que se puede asignar una entidad de servicio, use el comando [az role definition list](#).

Azure CLI

```
az role definition list \
--query "sort_by([].{roleName:roleName, description:description},
&roleName)" \
--output table
```

Por ejemplo, para permitir que la entidad de servicio con el appId de `00001111-aaaa-2222-bbbb-3333cccc4444` lea, escriba y elimine el acceso a los contenedores y datos de blobs de Azure Storage para todas las cuentas de almacenamiento del grupo de recursos `msdocs-python-sdk-auth-example` en la suscripción con ID `aaaa0a0a-bb1b-cc2c-dd3d-eeeeee4e4e4e`, asignaría el rol de *Colaborador de datos de blobs de almacenamiento* a la entidad de servicio de la aplicación mediante el siguiente comando.

Azure CLI

```
az role assignment create --assignee 00001111-aaaa-2222-bbbb-3333cccc4444 \
--scope "./subscriptions/aaaa0a0a-bb1b-cc2c-dd3d-
eeeeee4e4e4e/resourceGroups/msdocs-python-sdk-auth-example" \
--role "Storage Blob Data Contributor"
```

Para obtener información sobre cómo asignar permisos en el nivel de recurso o suscripción mediante la CLI de Azure, consulte el artículo [Asignación de roles de Azure mediante la CLI de Azure](#).

3: Configuración de variables de entorno para la aplicación

Debe establecer las variables de entorno `AZURE_CLIENT_ID`, `AZURE_TENANT_ID` y `AZURE_CLIENT_SECRET` para el proceso que ejecuta la aplicación de Python para que las credenciales de la entidad de servicio de la aplicación estén disponibles para la aplicación en tiempo de ejecución. El objeto `DefaultAzureCredential` busca la información de la entidad de servicio en estas variables de entorno.

Cuando se usa [Gunicorn](#) para ejecutar aplicaciones web de Python en un entorno de servidor UNIX, se pueden especificar variables de entorno para una aplicación mediante la directiva `EnvironmentFile` del archivo `unicorn.server`, como se muestra a continuación.

```
unicorn.server
```

```
[Unit]
Description=gunicorn daemon
After=network.target

[Service]
User=www-user
Group=www-data
WorkingDirectory=/path/to/python-app
EnvironmentFile=/path/to/python-app/py-env/app-environment-variables
ExecStart=/path/to/python-app/py-env/gunicorn --config config.py wsgi:app

[Install]
WantedBy=multi-user.target
```

El archivo especificado en la directiva `EnvironmentFile` debe contener una lista de variables de entorno con sus valores, como se muestra a continuación.

Bash

```
AZURE_CLIENT_ID=<value>
AZURE_TENANT_ID=<value>
AZURE_CLIENT_SECRET=<value>
```

4: Implementación de DefaultAzureCredential en su aplicación

Para autenticar objetos de cliente del SDK de Azure en Azure, la aplicación debe usar la clase `DefaultAzureCredential` del paquete `azure.identity`.

Empiece agregando el paquete `azure.identity` a la aplicación.

terminal

```
pip install azure-identity
```

A continuación, para cualquier código Python que cree un objeto de cliente del SDK de Azure en la aplicación, querrá:

1. Importar la clase `DefaultAzureCredential` desde el módulo `azure.identity`.
2. Crear un objeto `DefaultAzureCredential`.
3. Pasar el objeto `DefaultAzureCredential` al constructor de objetos de cliente del SDK de Azure.

En el segmento de código siguiente se muestra un ejemplo de esto.

Python

```
from azure.identity import DefaultAzureCredential
from azure.storage.blob import BlobServiceClient

# Acquire a credential object
token_credential = DefaultAzureCredential()

blob_service_client = BlobServiceClient(
    account_url="https://<my_account_name>.blob.core.windows.net",
    credential=token_credential)
```

Cuando el código anterior crea una instancia del objeto `DefaultAzureCredential`, `DefaultAzureCredential` lee las variables de entorno `AZURE_TENANT_ID`, `AZURE_CLIENT_ID` y `AZURE_CLIENT_SECRET` para que la información de la entidad de servicio de la aplicación se conecte a Azure.

Métodos adicionales para autenticarse en recursos de Azure desde aplicaciones de Python

Artículo • 10/04/2025

En este artículo se enumeran métodos adicionales que las aplicaciones pueden usar para autenticarse en recursos de Azure. Los métodos de este artículo se usan con menos frecuencia; siempre que sea posible, le recomendamos que use uno de los métodos descritos en [autenticación de aplicaciones de Python en Azure mediante la introducción al SDK de Azure](#).

Autenticación interactiva del explorador

Este método autentica de manera interactiva una aplicación a través de [InteractiveBrowserCredential](#), recopilando credenciales de usuario en el sistema predeterminado.

La autenticación interactiva del explorador habilita la aplicación para todas las operaciones permitidas por las credenciales de inicio de sesión interactivas. Como resultado, si es el propietario o administrador de la suscripción, el código tiene acceso inherente a la mayoría de los recursos de esa suscripción sin tener que asignar permisos específicos. Por este motivo, no se recomienda el uso de la autenticación interactiva del explorador excepto para la experimentación.

Habilitación de aplicaciones para la autenticación interactiva del explorador

Realice los pasos siguientes para permitir que la aplicación se autentique a través del flujo de explorador interactivo. Estos pasos también funcionan para la [autenticación de código de dispositivo](#) descrita más adelante. Seguir este proceso es necesario solo si se usa [InteractiveBrowserCredential](#) en el código.

1. En [Azure Portal](#), vaya a Microsoft Entra ID y seleccione **Registros de aplicaciones** en el menú izquierdo.
2. Seleccione el registro de la aplicación y, a continuación, seleccione **Autenticación**.
3. En **Configuración avanzada**, seleccione **Sí** para **Permitir flujos de cliente públicos**.
4. Seleccione **Guardar** para aplicar los cambios.

5. Para autorizar la aplicación para recursos específicos, vaya al recurso en cuestión, seleccione **Permisos de API** y habilite **Microsoft Graph** y otros recursos a los que quiera acceder. Microsoft Graph normalmente está habilitado de forma predeterminada.

ⓘ Importante

También debe ser el administrador del inquilino para conceder el consentimiento a la aplicación al iniciar sesión por primera vez.

Si no puede configurar la opción de flujo de código de dispositivo en Active Directory, es posible que la aplicación tenga que ser multiinquilino. Para realizar este cambio, vaya al panel **Autenticación**, seleccione **Cuentas en cualquier directorio organizativo** (en **Tipos de cuenta admitidos**) y, a continuación, seleccione **Sí** para **Permitir flujos de cliente públicos**.

Ejemplo de uso de InteractiveBrowserCredential

En el ejemplo siguiente se muestra el uso de [InteractiveBrowserCredential](#) para autenticarse con [SubscriptionClient](#).

Python

```
# Show Azure subscription information

import os
from azure.identity import InteractiveBrowserCredential
from azure.mgmt.resource import SubscriptionClient

credential = InteractiveBrowserCredential()
subscription_client = SubscriptionClient(credential)

subscription = next(subscription_client.subscriptions.list())
print(subscription.subscription_id)
```

Para un control más exacto, como establecer URI de redirección, puede proporcionar argumentos específicos a `InteractiveBrowserCredential` como `redirect_uri`.

Autenticación interactiva con agente

Este método autentica de forma interactiva una aplicación mediante [InteractiveBrowserBrokerCredential](#) con recopilación de credenciales de usuario mediante el agente de autenticación del sistema. Este tipo de credencial se proporciona en el complemento Azure Identity Broker, [azure-identity-broker](#).

Un agente de autenticación del sistema es una aplicación que se ejecuta en la máquina de un usuario que administra los protocolos de enlace de autenticación y el mantenimiento de tokens para todas las cuentas conectadas. Actualmente, solo se admite el agente de autenticación de Windows, el Administrador de cuentas web (WAM). Los usuarios de macOS y Linux se autenticarán a través de un explorador.

Se admiten cuentas personales de Microsoft y cuentas profesionales o educativas. Si se usa una versión compatible de Windows, la interfaz de usuario predeterminada basada en explorador se reemplaza por una experiencia de autenticación más fluida, similar a las aplicaciones integradas de Windows.

La autenticación interactiva con agente habilita la aplicación para todas las operaciones permitidas por las credenciales de inicio de sesión interactivas. Como resultado, si es el propietario o administrador de la suscripción, el código tiene acceso inherente a la mayoría de los recursos de esa suscripción sin tener que asignar permisos específicos.

Habilitar aplicaciones para la autenticación intermediada interactiva

Realice los pasos siguientes para permitir que la aplicación se autentique a través del flujo de agente interactivo.

1. En [Azure Portal](#), vaya a Microsoft Entra ID y seleccione **Registros de aplicaciones** en el menú izquierdo.
2. Seleccione el registro de la aplicación y, a continuación, seleccione **Autenticación**.
3. Agregue el URI de redirección de WAM al registro de la aplicación a través de una configuración de plataforma:
 - a. En **Configuraciones de plataforma**, seleccione + **Agregar una plataforma**.
 - b. En **Configurar plataformas**, seleccione el ícono del tipo de aplicación (plataforma) para configurar sus opciones; Por ejemplo, **aplicaciones móviles y de escritorio**.
 - c. En **URI de redirección personalizados**, escriba el URI de redirección de WAM:

text
ms-appx-web://microsoft.aad.brokerplugin/{client_id}

El `{client_id}` debe reemplazarse por el ID de la aplicación (cliente) que aparece en el panel Información general del registro de la aplicación.

d. Seleccione **Configurar**.

Para más información, consulte [Adición de un URI de redirección a un registro de aplicación](#).

4. De nuevo en el panel **Autenticación**, en **Configuración avanzada**, seleccione **Sí** para **Permitir flujos de cliente públicos**.

5. Seleccione **Guardar** para aplicar los cambios.

6. Para autorizar la aplicación para recursos específicos, vaya al recurso en cuestión, seleccione **Permisos de API** y habilite **Microsoft Graph** y otros recursos a los que quiera acceder. Microsoft Graph normalmente está habilitado de forma predeterminada.

ⓘ Importante

También debe ser el administrador del inquilino para conceder el consentimiento a la aplicación al iniciar sesión por primera vez.

Ejemplo de uso de InteractiveBrowserBrokerCredential

En el ejemplo siguiente se muestra el uso de un [InteractiveBrowserBrokerCredential](#) para autenticarse con el [BlobServiceClient](#):

Python

```
import win32gui
from azure.identity.broker import InteractiveBrowserBrokerCredential
from azure.storage.blob import BlobServiceClient

# Get the handle of the current window
current_window_handle = win32gui.GetForegroundWindow()

# To authenticate and authorize with an app, use the following line to get a
# credential and
# substitute the <app_id> and <tenant_id> placeholders with the values for your
# app and tenant.
# credential =
InteractiveBrowserBrokerCredential(parent_window_handle=current_window_handle,
client_id=<app_id>, tenant_id=<tenant_id>)
credential =
InteractiveBrowserBrokerCredential(parent_window_handle=current_window_handle)
client = BlobServiceClient("https://<storage-account-
name>.blob.core.windows.net/", credential=credential)

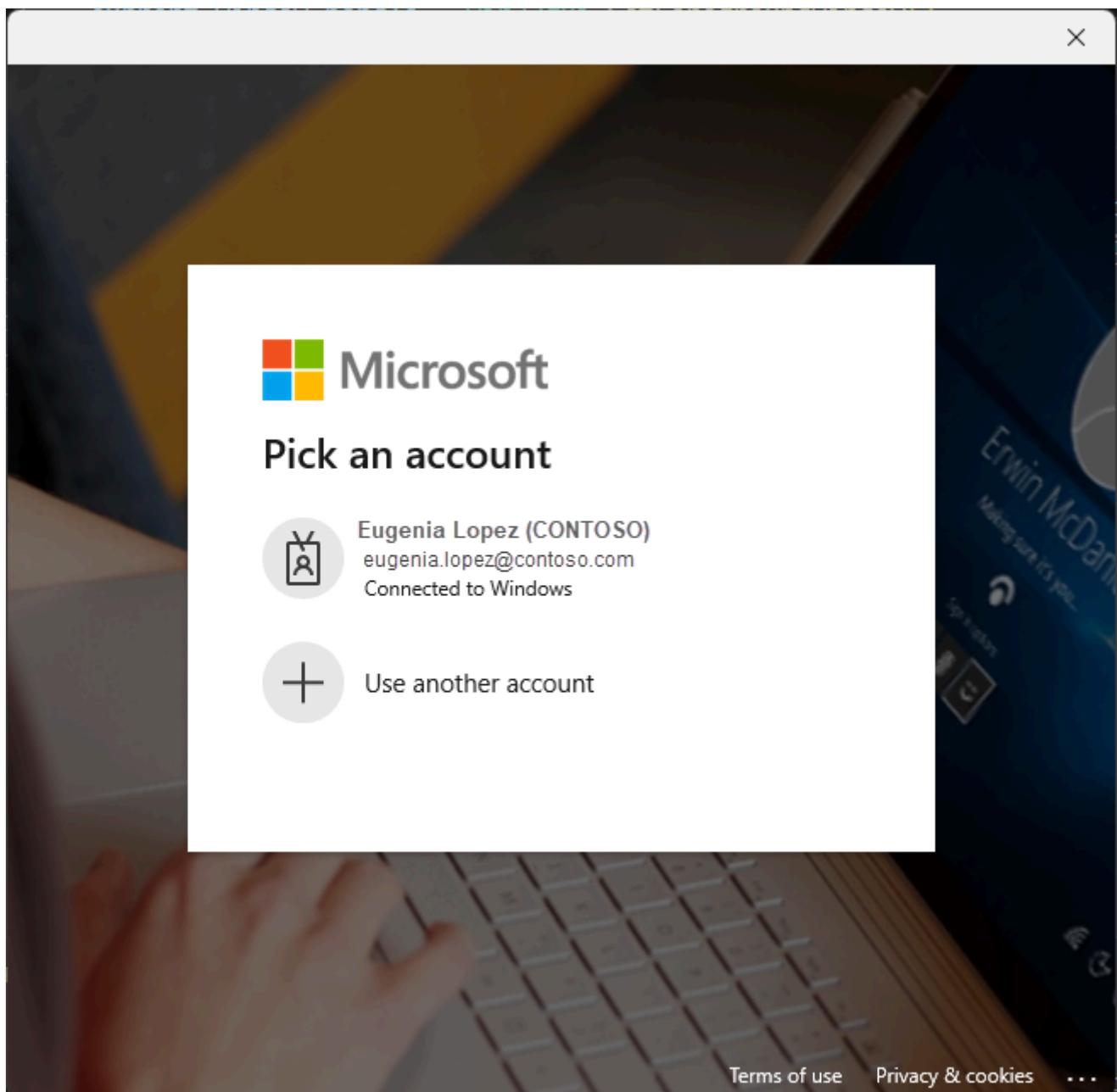
# Prompt for credentials appears on first use of the client
```

```
for container in client.list_containers():
    print(container.name)
```

Para un control más exacto, como establecer un tiempo de espera, puede proporcionar argumentos específicos a `InteractiveBrowserBrokerCredential` como `timeout`.

Para que el código se ejecute correctamente, a la cuenta de usuario se le debe asignar un rol de Azure en la cuenta de almacenamiento que permita el acceso a contenedores de blobs como "Colaborador de datos de la cuenta de almacenamiento". Si se especifica una aplicación, debe tener permisos de API establecidos para `user_impersonation` **Acceso a Azure Storage** (paso 6 de la sección anterior). Este permiso de API permite que la aplicación acceda a Azure Storage en nombre del usuario que inició sesión después de conceder el consentimiento durante el inicio de sesión.

En la captura de pantalla siguiente se muestra la experiencia de inicio de sesión del usuario:



Autenticación de la cuenta del sistema predeterminada a través de WAM

Muchas personas siempre inician sesión en Windows con la misma cuenta de usuario y, por lo tanto, solo quieren autenticarse con esa cuenta. Obligar a estas personas a seleccionar repetidamente su única cuenta en un selector de cuentas puede resultar molesto.

Afortunadamente, `InteractiveBrowserBrokerCredential` ofrece una manera de permitir que estos usuarios inicien sesión silenciosamente con la cuenta del sistema predeterminada, que, en Windows, es el usuario que ha iniciado sesión.

Para habilitar el inicio de sesión con la cuenta del sistema predeterminada:

1. Asegúrese de usar `azure-identity-broker` la versión 1.1.0 o posterior.
2. Establezca el argumento `use_default_broker_account` en `True` al crear una instancia de `InteractiveBrowserBrokerCredential`.

En el ejemplo siguiente se muestra cómo habilitar el inicio de sesión con la cuenta del sistema predeterminada:

Python

```
import win32gui
from azure.identity.broker import InteractiveBrowserBrokerCredential

# code omitted for brevity

window_handle = win32gui.GetForegroundWindow()

credential = InteractiveBrowserBrokerCredential(
    parent_window_handle=window_handle,
    use_default_broker_account=True
)
```

Una vez que opte por este comportamiento, el tipo de credencial intenta iniciar sesión pidiendo a la Biblioteca de autenticación de Microsoft subyacente (MSAL) que realice el inicio de sesión para la cuenta del sistema predeterminada. Si se produce un error en el inicio de sesión, el tipo de credencial vuelve a mostrar el cuadro de diálogo del selector de cuentas, desde el que el usuario puede seleccionar la cuenta adecuada.

Autenticación con código de dispositivo

Este método autentica interactivamente a un usuario en dispositivos con una interfaz de usuario limitada (normalmente dispositivos sin teclado):

1. Cuando la aplicación intenta autenticarse, la credencial solicita al usuario una dirección URL y un código de autenticación.
2. El usuario visita la dirección URL en un dispositivo independiente habilitado para el explorador (un equipo, smartphone, etc.) y escribe el código.
3. El usuario sigue un proceso de autenticación normal en el explorador.
4. Tras la autenticación correcta, la aplicación se autentica en el dispositivo.

Para obtener más información, consulte [Plataforma de identidad de Microsoft y flujo de concesión de autorización de dispositivos de OAuth 2.0](#).

La autenticación de código de dispositivo en un entorno de desarrollo habilita la aplicación para todas las operaciones permitidas por las credenciales de inicio de sesión interactivas. Como resultado, si es el propietario o administrador de la suscripción, el código tiene acceso inherente a la mayoría de los recursos de esa suscripción sin tener que asignar permisos específicos. Sin embargo, puede usar este método con un identificador de cliente específico, en lugar del valor predeterminado, para el que puede asignar permisos específicos.

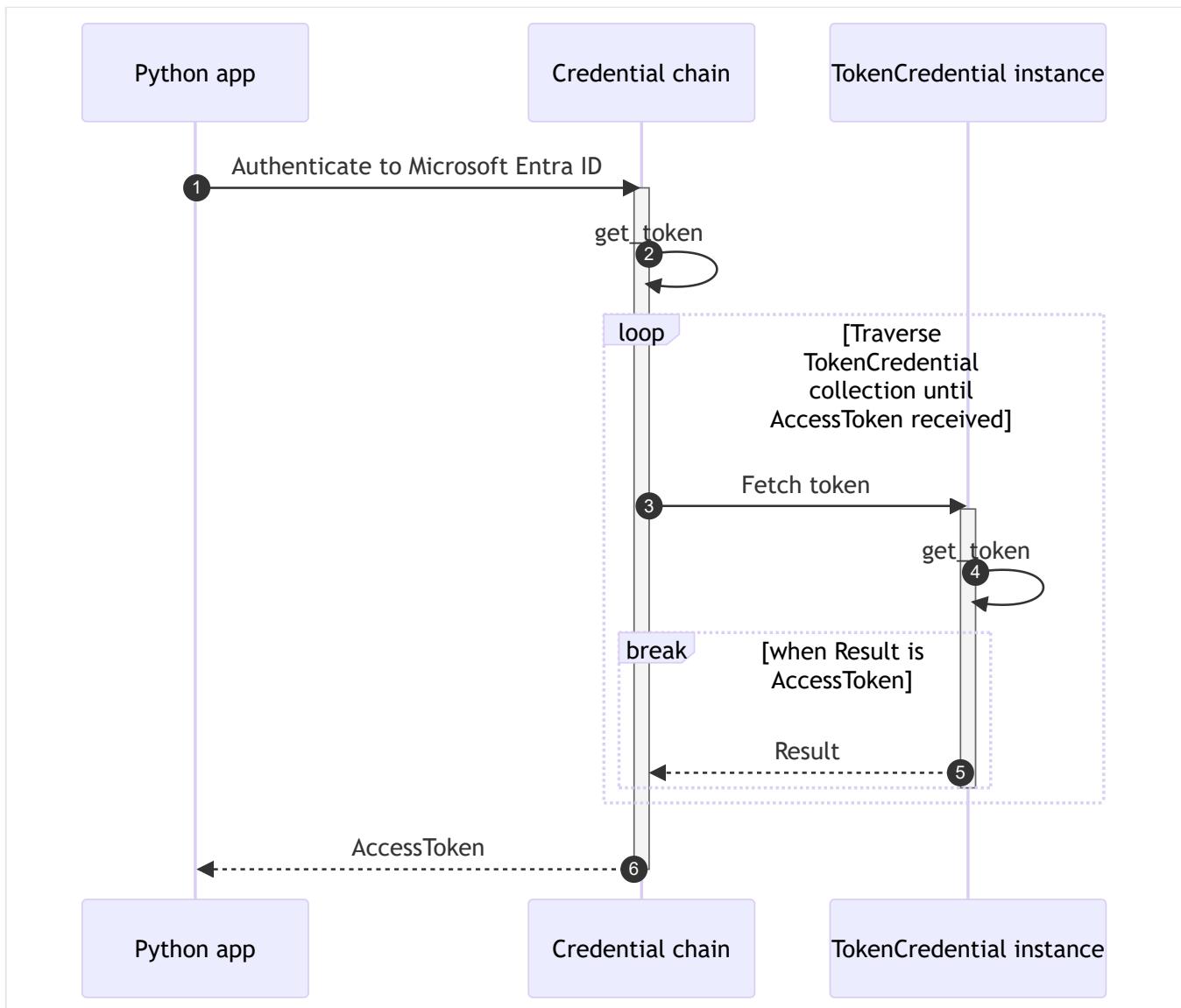
Cadenas de credenciales en la biblioteca de identidades de Azure para Python

16/08/2025

La biblioteca de identidades de Azure proporciona *credenciales*, clases públicas que implementan el protocolo [TokenCredential](#) de la biblioteca de Azure Core. Una credencial representa un flujo de autenticación distinto para adquirir un token de acceso de Microsoft Entra ID. Estas credenciales se pueden encadenar para formar una secuencia ordenada de mecanismos de autenticación que se van a intentar.

Funcionamiento de una credencial encadenada

En tiempo de ejecución, una cadena de credenciales intenta autenticarse mediante la primera credencial de la secuencia. Si esa credencial no puede adquirir un token de acceso, se intenta utilizar la siguiente credencial de la secuencia, y así sucesivamente hasta que se obtenga correctamente un token de acceso. En el siguiente diagrama de secuencia se ilustra este comportamiento:



¿Por qué usar cadenas de credenciales?

Una credencial encadenada puede ofrecer las siguientes ventajas:

- **Reconocimiento del entorno:** selecciona automáticamente la credencial más adecuada en función del entorno en el que se ejecuta la aplicación. Sin ella, tendría que escribir código como este:

Python

```

# Set up credential based on environment (Azure or local development)
if os.getenv("WEBSITE_HOSTNAME"):
    credential = ManagedIdentityCredential(client_id=user_assigned_client_id)
else:
    credential = AzureCliCredential()

```

- **Transiciones fluidas:** la aplicación puede pasar del desarrollo local al entorno de ensayo o producción sin cambiar el código de autenticación.

- **Resiliencia mejorada:** incluye un mecanismo de respaldo que cambia a la siguiente credencial cuando la anterior no puede adquirir un token de acceso.

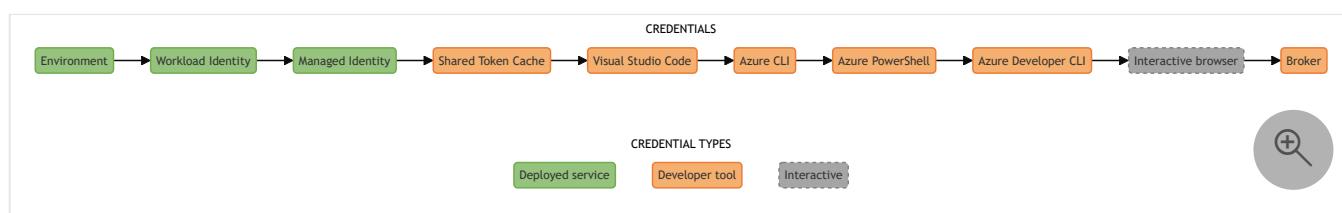
Cómo elegir una credencial encadenada

Hay dos filosofías dispares para encadenar credenciales:

- “**Desmontar**” una cadena: comience con una cadena preconfigurada y excluya lo que no necesita. Para este enfoque, consulte la sección [Información general sobre DefaultAzureCredential](#).
- “**Crear**” una cadena: comience con una cadena vacía e incluya solo lo que necesita. Para este enfoque, consulte la sección [Información general sobre ChainedTokenCredential](#).

Información general sobre DefaultAzureCredential

`DefaultAzureCredential` es una cadena preconfigurada de credenciales fundamentada. Está diseñada para admitir muchos entornos, junto con los flujos de autenticación y las herramientas de desarrollo más comunes. En forma gráfica, la cadena subyacente tiene este aspecto:



El orden en el que `DefaultAzureCredential` intenta las credenciales es el siguiente.

[] Expandir tabla

Pedido	Credencial	Descripción	¿Habilitado de forma predeterminada?
1	Entorno	Lee una colección de variables de entorno para determinar si un principal de servicio de aplicación (usuario de aplicación) está configurado para la aplicación. Si es así, <code>DefaultAzureCredential</code> usa estos valores para autenticar la aplicación en Azure. Este método se usa con más frecuencia en entornos de servidor, pero también se puede usar al desarrollar localmente.	Sí
2	Identidad de carga de	Si la aplicación se implementa en un host de Azure con la identidad de carga de trabajo habilitada, autentica esa	Sí

Pedido de	Credencial	Descripción	¿Habilitado de forma predeterminada?
	trabajo	cuenta.	
3	Identidad administrada	Si la aplicación se implementa en un host de Azure con la identidad administrada habilitada, se autentica la aplicación en Azure usando esa identidad administrada.	Sí
4	Caché de tokens compartidos	Solo en Windows, si el desarrollador se autentica en Azure iniciando sesión en Visual Studio, autentique la aplicación en Azure con esa misma cuenta.	Sí
5	Visual Studio Code	Si el desarrollador se ha autenticado a través de la extensión Azure Resources de Visual Studio Code y el paquete azure-identity-broker está instalado, autentique esa cuenta.	Sí
6	CLI de Azure	Si el desarrollador se autenticó en Azure mediante el comando <code>az login</code> de la CLI de Azure, se autentica la aplicación en Azure con esa misma cuenta.	Sí
7	Azure PowerShell	Si el desarrollador se autenticó en Azure mediante el cmdlet <code>Connect-AzAccount</code> de Azure PowerShell, se autentica la aplicación en Azure con esa misma cuenta.	Sí
8	CLI de desarrollo de Azure	Si el desarrollador se autenticó en Azure mediante el comando <code>azd auth login</code> de la CLI para desarrolladores de Azure, se autentica con esa cuenta.	Sí
9	Navegador interactivo	Si está habilitado, autenticará de forma interactiva al desarrollador mediante el explorador predeterminado del sistema actual.	No
10	broker	Se autentica mediante la cuenta predeterminada que ha iniciado sesión en el sistema operativo a través de un intermediario. Requiere que se instale el paquete azure-identity-broker , ya que se usa una instancia de <code>InteractiveBrowserBrokerCredential</code> .	Sí

En su forma más sencilla, puede usar la versión sin parámetros de `DefaultAzureCredential` de la siguiente manera:

Python

```
from azure.identity import DefaultAzureCredential
from azure.storage.blob import BlobServiceClient

# Acquire a credential object
credential = DefaultAzureCredential()
```

```
blob_service_client = BlobServiceClient(  
    account_url="https://<my_account_name>.blob.core.windows.net",  
    credential=credential  
)
```

Personalización de DefaultAzureCredential

En las secciones siguientes se describen las estrategias para controlar qué credenciales se incluyen en la cadena.

Excluir una credencial individual

Para excluir una credencial individual de `DefaultAzureCredential`, utilice el parámetro correspondiente `exclude`-prefijo `palabra clave`. Por ejemplo:

Python

```
credential = DefaultAzureCredential(  
    exclude_environment_credential=True,  
    exclude_workload_identity_credential=True,  
    managed_identity_client_id=user_assigned_client_id  
)
```

En el ejemplo de código anterior, `EnvironmentCredential` y `WorkloadIdentityCredential` se quitan de la cadena de credenciales. Como resultado, la primera credencial que se va a probar es `ManagedIdentityCredential`. La cadena modificada tiene este aspecto:



Nota

`InteractiveBrowserCredential` se excluye de forma predeterminada y, por tanto, no se muestra en el diagrama anterior. Para incluir `InteractiveBrowserCredential`, establezca el parámetro de palabra clave `exclude_interactive_browser_credential` en `False` al llamar al constructor de `DefaultAzureCredential`.

A medida que se establecen más parámetros de palabra clave con el prefijo `exclude` en `True` (se configuran exclusiones de credenciales), las ventajas de usar `DefaultAzureCredential` disminuyen. En tales casos, `ChainedTokenCredential` es una mejor opción y requiere menos código. Para ilustrarlo, estos dos ejemplos de código se comportan de la misma manera:

DefaultAzureCredential

Python

```
credential = DefaultAzureCredential(  
    exclude_environment_credential=True,  
    exclude_workload_identity_credential=True,  
    exclude_shared_token_cache_credential=True,  
    exclude_visual_studio_code_credential=True,  
    exclude_azure_powershell_credential=True,  
    exclude_azure_developer_cli_credential=True,  
    exclude_broker_credential=True,  
    managed_identity_client_id=user_assigned_client_id  
)
```

Excluir una categoría de tipo de credencial

Para excluir todas las credenciales `Developer tool` o `Deployed service`, establezca la variable de entorno `AZURE_TOKEN_CREDENTIALS` a `prod` o `dev`, respectivamente. Cuando se usa un valor de `prod`, la cadena de credenciales subyacente tiene el siguiente aspecto:



Cuando se usa un valor de `dev`, la cadena tiene el siguiente aspecto:



ⓘ Importante

La `AZURE_TOKEN_CREDENTIALS` variable de entorno se admite en `azure-identity` las versiones del paquete 1.23.0 y versiones posteriores.

Uso de una credencial específica

Para excluir todas las credenciales excepto para una, establezca la variable `AZURE_TOKEN_CREDENTIALS` de entorno en el nombre de la credencial. Por ejemplo, puede reducir la `DefaultAzureCredential` cadena a `AzureCliCredential` estableciendo `AZURE_TOKEN_CREDENTIALS` en `AzureCliCredential`. La comparación de cadenas se realiza de

forma que no distingue mayúsculas de minúsculas. Entre los valores de cadena válidos para la variable de entorno se incluyen:

- `AzureCliCredential`
- `AzureDeveloperCliCredential`
- `AzurePowerShellCredential`
- `EnvironmentCredential`
- `InteractiveBrowserCredential`
- `ManagedIdentityCredential`
- `VisualStudioCodeCredential`
- `WorkloadIdentityCredential`

 **Importante**

La `AZURE_TOKEN_CREDENTIALS` variable de entorno admite nombres de credenciales individuales en `azure-identity` las versiones del paquete 1.24.0 y versiones posteriores.

Información general sobre ChainedTokenCredential

`ChainedTokenCredential` es una cadena vacía a la que agrega credenciales para satisfacer las necesidades de la aplicación. Por ejemplo:

Python

```
credential = ChainedTokenCredential(  
    AzureCliCredential(),  
    AzureDeveloperCliCredential()  
)
```

El ejemplo de código anterior crea una cadena de credenciales adaptada formada por dos credenciales en tiempo de desarrollo. Primero se intenta `AzureCliCredential`, seguido de `AzureDeveloperCliCredential` si es necesario. En forma gráfica, la cadena tiene este aspecto:



 **Sugerencia**

Para mejorar el rendimiento, optimice el orden de las credenciales en `ChainedTokenCredential` de la mayoría a las credenciales menos usadas.

Guía de uso para DefaultAzureCredential

`DefaultAzureCredential` es sin duda la manera más fácil de empezar a trabajar con la biblioteca de identidad de Azure, pero esa comodidad conlleva compromisos. Una vez que implemente la aplicación en Azure, debe comprender los requisitos de autenticación de la aplicación. Por ese motivo, reemplace `DefaultAzureCredential` por una implementación de `TokenCredential` específica, como `ManagedIdentityCredential`.

Aquí se detallan los motivos:

- **Desafíos de depuración:** cuando se produce un error en la autenticación, puede resultar difícil depurar e identificar las credenciales incorrectas. Debe habilitar el registro para ver la progresión de una credencial a la siguiente y el estado de éxito o error de cada una. Para obtener más información, consulte [Depuración de una credencial encadenada](#).
- **Sobrecarga de rendimiento:** el proceso de probar secuencialmente varias credenciales puede suponer una sobrecarga de rendimiento. Por ejemplo, cuando se ejecuta en una máquina de desarrollo local, la identidad administrada no está disponible. Por lo tanto, `ManagedIdentityCredential` siempre falla en el entorno de desarrollo local, a menos que se des habilite explícitamente utilizando la propiedad correspondiente con el prefijo `exclude`.
- **Comportamiento imprevisible:** `DefaultAzureCredential` comprueba la presencia de determinadas [variables de entorno](#). Es posible que alguien pueda agregar o modificar estas variables de entorno en el nivel de sistema en el equipo host. Esos cambios se aplican globalmente y, por tanto, modifican el comportamiento de `DefaultAzureCredential` en tiempo de ejecución en cualquier aplicación que se ejecute en esa máquina.

Depuración de una credencial encadenada

Para diagnosticar un problema inesperado o comprender lo que hace una credencial encadenada, [habilite el registro](#) en la aplicación. Opcionalmente, filtre los registros solo a esos eventos emitidos desde la biblioteca cliente de identidades de Azure. Por ejemplo:

Python

```
import logging
from azure.identity import DefaultAzureCredential
```

```

# Set the logging level for the Azure Identity library
logger = logging.getLogger("azure.identity")
logger.setLevel(logging.DEBUG)

# Direct logging output to stdout. Without adding a handler,
# no logging output is visible.
handler = logging.StreamHandler(stream=sys.stdout)
logger.addHandler(handler)

# Optional: Output logging levels to the console.
print(
    f"Logger enabled for ERROR={logger.isEnabledFor(logging.ERROR)}, "
    f"WARNING={logger.isEnabledFor(logging.WARNING)}, "
    f"INFO={logger.isEnabledFor(logging.INFO)}, "
    f"DEBUG={logger.isEnabledFor(logging.DEBUG)}"
)

```

Para fines ilustrativos, supongamos que se usa la forma sin parámetros de `DefaultAzureCredential` para autenticar una solicitud en una cuenta ed Blob Storage. La aplicación se ejecuta en el entorno de desarrollo local y el desarrollador se autentica en Azure mediante la CLI de Azure. Supongamos también que el nivel de registro está establecido en `logging.DEBUG`. Cuando se ejecuta la aplicación, aparecen las siguientes entradas pertinentes en la salida:

Resultados

```

Logger enabled for ERROR=True, WARNING=True, INFO=True, DEBUG=True
No environment configuration found.
ManagedIdentityCredential will use IMDS
EnvironmentCredential.get_token failed: EnvironmentCredential authentication
unavailable. Environment variables are not fully configured.
Visit https://aka.ms/azsdk/python/identity/environmentcredential/troubleshoot to
troubleshoot this issue.
ManagedIdentityCredential.get_token failed: ManagedIdentityCredential
authentication unavailable, no response from the IMDS endpoint.
SharedTokenCacheCredential.get_token failed: SharedTokenCacheCredential
authentication unavailable. No accounts were found in the cache.
VisualStudioCodeCredential.get_token failed: VisualStudioCodeCredential
authentication unavailable. No Azure account information found in Visual Studio
Code.
AzureCliCredential.get_token succeeded
[Authenticated account] Client ID: 00001111-aaaa-2222-bbbb-3333cccc4444. Tenant
ID: aaaabbbb-0000-cccc-1111-dddd2222eeee. User Principal Name: unavailableUpn.
Object ID (user): aaaaaaaaaa-0000-1111-2222-bbbbbbbbbbb
DefaultAzureCredential acquired a token from AzureCliCredential

```

En la salida anterior, puede ver lo siguiente:

- `EnvironmentCredential`, `ManagedIdentityCredential`, `SharedTokenCacheCredential` y `VisualStudioCodeCredential` cada uno no pudo adquirir un token de acceso de Microsoft Entra, en ese orden.
- La llamada a `AzureCliCredential.get_token` se realiza correctamente y la salida también indica que `DefaultAzureCredential` adquirió un token de `AzureCliCredential`. Dado que `AzureCliCredential` se realizó correctamente, no se intentaron más credenciales.

! Nota

En el ejemplo anterior, el nivel de registro está establecido en `logging.DEBUG`. Tenga cuidado al usar este nivel de registro, ya que puede generar información confidencial. Por ejemplo, en este caso, el identificador de cliente, el identificador de inquilino y el identificador de objeto de la entidad de usuario del desarrollador en Azure. Toda la información de seguimiento se ha quitado de la salida para garantizar la claridad.

Tutorial: Autenticación integrada para aplicaciones de Python con servicios de Azure

28/05/2025

Microsoft Entra ID, cuando se usa con Azure Key Vault, proporciona un enfoque sólido y seguro para autenticar aplicaciones tanto en servicios de Azure como en plataformas de terceros que requieren claves de acceso o credenciales. Esta combinación elimina la necesidad de codificar los secretos de forma rígida en el código de la aplicación, y en su lugar, se basa en identidades administradas, control de acceso basado en roles (RBAC) y administración centralizada de secretos a través de Key Vault. Este enfoque simplifica la administración de identidades y mejora la posición de seguridad en entornos en la nube.

En este tutorial se exploran estos mecanismos de autenticación mediante un ejemplo práctico proporcionado en el repositorio de GitHub: [github.com/Azure-Samples/python-integrated-authentication ↗](https://github.com/Azure-Samples/python-integrated-authentication).

En el ejemplo se muestra cómo una aplicación de Python puede:

- Autenticación con Azure mediante DefaultAzureCredential
- Acceso a secretos de Azure Key Vault sin almacenar credenciales
- Comunicación segura con otros servicios de Azure, como Azure Storage, Cosmos DB y mucho más

Este artículo forma parte de una serie que proporciona un tutorial detallado sobre cómo autenticar una aplicación de Python con el identificador de Microsoft Entra, Azure Key Vault y Azure Queue Storage mediante la biblioteca del SDK `azure-identity` de Python de Azure.

Parte 1: Contexto

Aunque muchos servicios de Azure dependen exclusivamente del control de acceso basado en rol (RBAC), mientras que otros requieren acceso a través de secretos o claves. Estos servicios incluyen Azure Storage, bases de datos, servicios de Azure AI, Key Vault y Event Hubs.

Al compilar aplicaciones en la nube que interactúan con estos servicios, los desarrolladores pueden usar Azure Portal, la CLI o PowerShell para generar y configurar claves de acceso específicas del servicio. Estas claves están vinculadas a directivas de acceso concretas para evitar el acceso no autorizado. Sin embargo, este modelo requiere que la aplicación administre

las claves explícitamente y autentíquese por separado con cada servicio, un proceso que sea tedioso y propenso a errores.

Insertar secretos directamente en el código o almacenarlos en máquinas de desarrollador corre el riesgo de exponerlos en:

- Control de código fuente
- Entornos locales no seguros
- Exportaciones accidentales de registros o configuraciones

Azure ofrece dos servicios clave para mejorar la seguridad y simplificar la autenticación:

- **Azure Key Vault** Azure Key Vault proporciona un almacén seguro basado en la nube para secretos, incluidas las claves de acceso, las cadenas de conexión y los certificados. Al recuperar secretos de Key Vault solo en tiempo de ejecución, las aplicaciones evitan exponer datos confidenciales en el código fuente o los archivos de configuración.
- Con [las identidades administradas de Microsoft Entra](#), la aplicación puede autenticarse una vez con el identificador de Microsoft Entra. Desde allí, puede acceder a otros servicios de Azure (incluido Key Vault) sin administrar las credenciales directamente.

Este enfoque proporciona:

- Código sin credenciales (sin secretos en el control de código fuente)
- Integración sin problemas con los servicios de Azure
- Coherencia del entorno: el mismo código se ejecuta localmente y en la nube con una configuración mínima

En este tutorial se muestra cómo usar la identidad administrada de Microsoft Entra y Key Vault juntas en la misma aplicación. Con el identificador de Entra de Microsoft y Key Vault juntos, la aplicación nunca necesita autenticarse con servicios individuales de Azure y puede acceder de forma sencilla y segura a las claves necesarias para los servicios de terceros.

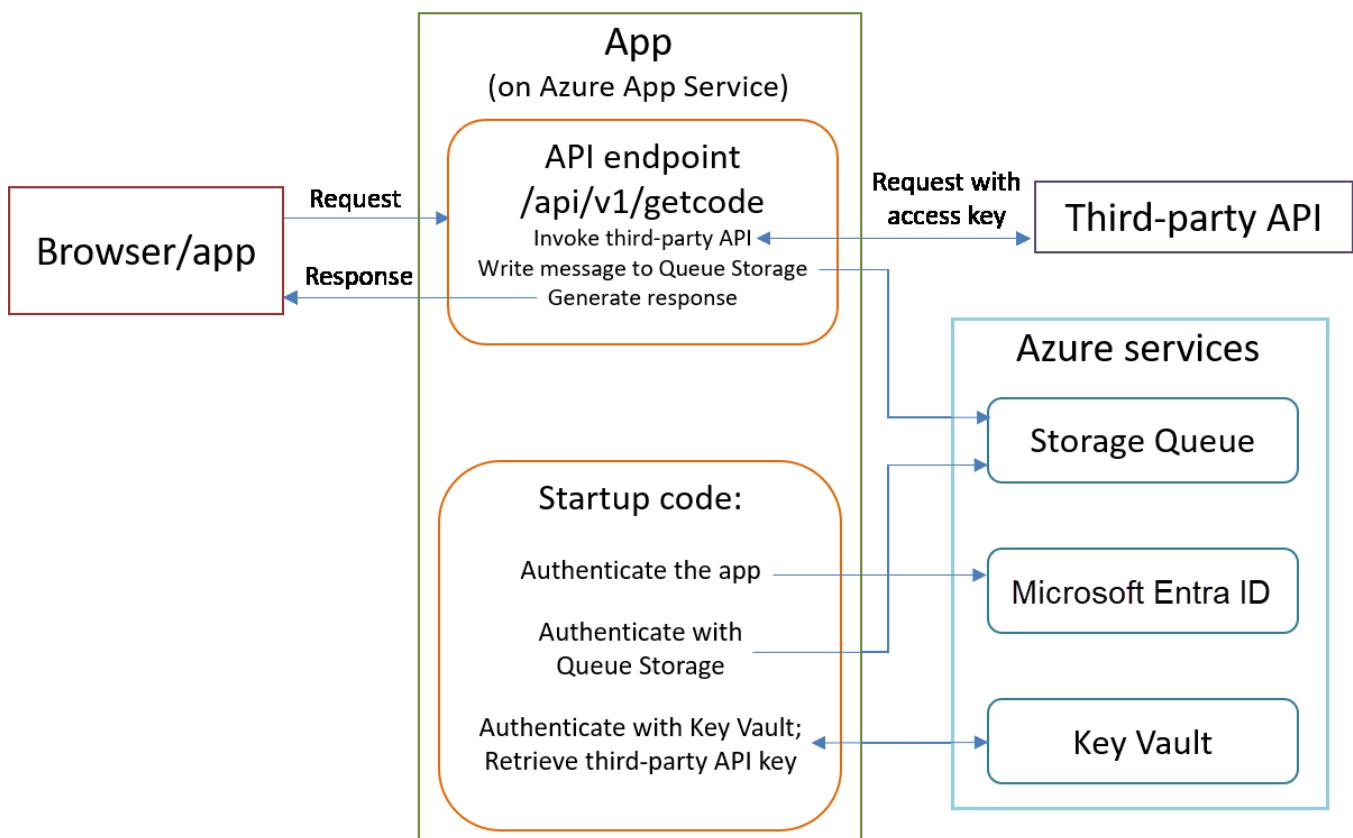
Importante

En este artículo se usa el término genérico común "clave" para hacer referencia a lo que se almacenan como "secretos" en Azure Key Vault, como una clave de acceso para una API REST. Este uso no debe confundirse con la administración de claves *criptográficas* de Key Vault, que es una característica independiente de los *secretos* de Key Vault.

Escenario de aplicación en la nube de ejemplo

Para comprender el proceso de autenticación de Azure más profundamente, tenga en cuenta el siguiente escenario: una aplicación web de Flask se implementa en Azure App Service. Expone un punto de conexión de API público y no autenticado que devuelve datos JSON en respuesta a solicitudes HTTP.

- Para generar su respuesta, la API invoca una API de terceros que requiere una clave de acceso. En lugar de almacenar esta clave en archivos de código o configuración, la aplicación la recupera de forma segura en tiempo de ejecución desde Azure Key Vault mediante la identidad administrada de Microsoft Entra.
- Antes de devolver su respuesta al cliente, la aplicación escribe un mensaje en una cola de Azure Storage para el procesamiento asíncrono. El mensaje podría representar una tarea, un registro o una señal, aunque el procesamiento de bajada no es el foco de este escenario.



! Nota

Aunque los puntos de conexión de API públicos normalmente están protegidos por sus propias claves de acceso o mecanismos de autenticación, en este tutorial se supone que el punto de conexión está abierto y no autenticado.

Esta simplificación ayuda a aislar los requisitos de autenticación internos de la aplicación (como el acceso a Azure Key Vault y Azure Storage) desde cualquier problema de autenticación relacionado con los clientes externos.

El escenario se centra únicamente en el comportamiento de la aplicación y no demuestra ni implica a un llamador externo autenticándose con el punto de conexión.

Parte 2: Requisitos de autenticación >>>

Parte 2: Necesidades de autenticación en este escenario de ejemplo

29/05/2025

Parte anterior: Introducción y antecedentes

En este escenario de ejemplo, la aplicación principal tiene tres requisitos de autenticación distintos:

- Azure Key Vault

La aplicación debe autenticarse con Azure Key Vault para recuperar una clave de API almacenada de forma segura necesaria para llamar a un servicio de terceros.

- API de terceros

Una vez recuperada la clave de API, la aplicación la usa para autenticarse con la API externa de terceros.

- Azure Queue Storage

Después de procesar la solicitud, la aplicación debe autenticarse con Azure Queue Storage para poner en cola un mensaje para el procesamiento asíncrono o diferido.

Estas tareas requieren que la aplicación administre tres conjuntos de credenciales:

- Dos para los recursos de Azure (Key Vault y Storage)
- Uno para un servicio externo (API de terceros)

Desafíos de autenticación clave

La creación de aplicaciones en la nube seguras requiere un control cuidadoso de las credenciales, especialmente cuando hay varios servicios implicados. En este escenario de ejemplo se presentan varios desafíos críticos:

- Dependencia circular en Key Vault

La aplicación usa Azure Key Vault para almacenar de forma segura secretos, como claves de API de terceros o credenciales de Azure Storage. Sin embargo, para recuperar esos secretos, la aplicación debe autenticarse primero con Key Vault. Esto crea un problema circular: la aplicación necesita credenciales para acceder a Key Vault, pero esas credenciales deben almacenarse de forma segura. Sin una solución segura, esto podría

provocar credenciales codificadas de forma segura o configuraciones no seguras en entornos de desarrollo.

- Control seguro de claves de API de terceros

Después de recuperar la clave de API de Key Vault, la aplicación debe usarla para llamar a un servicio externo de terceros. Esta clave debe manipularse con cuidado extremo.

- Nunca se codifica de forma rígida en archivos de configuración o código fuente
- Nunca se registró en stdout, stderr ni en los registros de la aplicación
- Solo se mantiene en la memoria y se accede a él en tiempo de ejecución, justo antes de usar
- Se gestiona rápidamente una vez completada la solicitud

Si no se siguen estas prácticas, aumenta el riesgo de pérdida de credenciales o uso no autorizado.

- Protección de las credenciales de Azure Queue Storage

Para escribir mensajes en Azure Queue Storage, la aplicación normalmente necesita una cadena de conexión o un token de acceso compartido. Estas credenciales:

- Debe almacenarse en una ubicación segura, como Key Vault.
- No debe aparecer en registros, trazas ni herramientas de desarrollo
- Solo se debe tener acceso a ellos a través de mecanismos de tiempo de ejecución seguros
- Requerir una configuración de RBAC adecuada si se usa la identidad administrada

- Flexibilidad del entorno

La aplicación debe ejecutarse de forma confiable en entornos de desarrollo local y producción en la nube, con el mismo código base y la misma lógica condicional mínima.

Esto significa lo siguiente:

- No hay secretos específicos del entorno insertados en el código
- No es necesario alternar manualmente las credenciales ni las rutas de acceso lógicas
- Uso coherente de la autenticación basada en identidades entre entornos

autenticación de Azure-First con el identificador de Entra de Microsoft

A medida que las aplicaciones en la nube se escalan en complejidad e integran con más servicios, la autenticación segura y simplificada se convierte en esencial. Azure ofrece un modelo de identidad "primero en Azure" a través de Microsoft Entra ID, lo que permite la

administración unificada de identidades e integración sin problemas con los servicios de Azure para la autenticación segura y sin credenciales.

En lugar de administrar manualmente secretos o incrustar credenciales en el código de aplicación, una práctica propensa a riesgos de seguridad, el identificador de Entra de Microsoft permite que las aplicaciones se autentiquen de forma segura mediante identidades administradas.

Las principales ventajas de las identidades administradas de Microsoft Entra son:

- Sin secretos en el código

Las aplicaciones ya no requieren cadenas de conexión codificadas de forma codificada, secretos de cliente ni claves.

- Identidad integrada para aplicaciones

Azure puede asignar automáticamente una identidad administrada a la aplicación, lo que permite el acceso seguro a los servicios, como Key Vault, Storage y SQL sin credenciales adicionales.

- Coherencia del entorno

El mismo código y modelo de identidad funcionan tanto en entornos de desarrollo local como hospedados en Azure mediante DefaultAzureCredential del SDK de Azure.

Flujo de identidad específico del entorno

Las aplicaciones que usan microsoft Entra ID para la autenticación se benefician de un modelo de identidad flexible que funciona perfectamente en entornos de desarrollo locales y hospedados en Azure. Esta coherencia se logra mediante el SDK de

`DefaultAzureCredential` Azure, que selecciona automáticamente el método de identidad adecuado en función del entorno.

Entorno de Azure

Cuando la aplicación se implementa en Azure:

- Una identidad administrada se asigna automáticamente a la aplicación.
- Azure controla el ciclo de vida de emisión de tokens y credenciales internamente, sin secretos manuales necesarios.
- La aplicación usa el control de acceso basado en roles (RBAC) Role-Based o las directivas de acceso de Key Vault para acceder a los servicios.

Entorno de desarrollo locales

Durante el desarrollo local:

- Una entidad de servicio actúa como identidad de la aplicación.
- Los desarrolladores se autentican mediante la CLI de Azure (az login), las variables de entorno o las integraciones de Visual Studio/VS Code.
- El mismo código de aplicación se ejecuta sin modificaciones; solo cambia el origen de la identidad.

En ambos entornos, los SDK de Azure usan el `DefaultAzureCredential`, que abstrae el origen de identidad y selecciona automáticamente el método correcto.

Procedimientos recomendados para el desarrollo seguro

Aunque es posible establecer secretos como variables de entorno (por ejemplo, a través de Azure App Settings), este enfoque tiene desventajas:

- Debe replicar manualmente secretos en entornos locales.
- Existe el riesgo de que los secretos se filtren hacia el control de código fuente.
- Es posible que se requiera lógica adicional para diferenciar entre entornos.

En su lugar, el enfoque recomendado es:

- Use Key Vault para almacenar claves de API de terceros y otros secretos.
- Asigne una identidad administrada a la aplicación implementada.
- Use una entidad de servicio para el desarrollo local y asígnele los mismos derechos de acceso.
- Use `DefaultAzureCredential` en el código para abstraer la lógica de autenticación.
- Evite almacenar o registrar credenciales.

Flujo de autenticación en la práctica

Este es el funcionamiento de la autenticación en tiempo de ejecución:

- El código crea una `DefaultAzureCredential` instancia.
- Utiliza esta credencial para instanciar un cliente (por ejemplo, `SecretClient`, `QueueServiceClient`).
- Cuando la aplicación invoca un método (por ejemplo, `get_secret()`), el cliente usa la credencial para autenticar la solicitud.

- Azure comprueba la identidad y comprueba si tiene el rol o la directiva correctos para realizar la operación.

Este flujo garantiza que la aplicación pueda acceder de forma segura a los servicios de Azure sin insertar secretos en archivos de código o configuración. También permite cambiar sin problemas entre el desarrollo local y la implementación en la nube sin cambiar la lógica de autenticación.

[parte 3: implementación de api de terceros >>>](#)

Parte 3: Implementación de API de terceros de ejemplo

29/05/2025

Parte anterior: Requisitos de autenticación

En nuestro escenario de ejemplo, la aplicación principal consume una API de terceros protegida con una clave de acceso. En esta sección se muestra la API mediante Azure Functions, pero se aplican los mismos principios independientemente de cómo o dónde se implemente la API, tanto si hospeda la aplicación en otro proveedor de nube como en un servidor web tradicional.

El aspecto clave es que las solicitudes de cliente al punto de conexión protegido deben incluir la clave de acceso, que la aplicación debe administrar de forma segura. En esta sección se proporciona información general sobre cómo implementar dicha API mediante Azure Functions, pero puede adaptar los principios a sus necesidades específicas.

Implementación de API de terceros de ejemplo

La API de terceros de ejemplo es un punto de conexión simple que devuelve un número aleatorio entre 1 y 999. La API está protegida con una clave de acceso, que se debe proporcionar en la solicitud para acceder al punto de conexión. Con fines de demostración, esta API se implementa en el punto de conexión, <https://msdocs-example-api.azurewebsites.net/api/RandomNumber>. Sin embargo, para llamar a la API, debe proporcionar la clave `d0c5atM1cr0s0ft` de acceso en un `?code=` parámetro de dirección URL o en una `'x-functions-key'` propiedad del encabezado HTTP. Por ejemplo, después de implementar la aplicación y la API, pruebe esta dirección URL en un explorador o curl: <https://msdocs-example-api.azurewebsites.net/api/RandomNumber?code=d0c5atM1cr0s0ft>.

Si la clave de acceso es válida, el punto de conexión devuelve una respuesta JSON que contiene una sola propiedad, "value", cuyo valor es un número comprendido entre 1 y 999, como `{"value": 959}`.

El punto de conexión se implementa en Python y se implementa en Azure Functions. El código es el siguiente:

Python

```
import logging
import random
import json
```

```
import azure.functions as func

def main(req: func.HttpRequest) -> func.HttpResponse:
    logging.info('RandomNumber invoked via HTTP trigger.')

    random_value = random.randint(1, 1000)
    dict = { "value" : random_value }
    return func.HttpResponse(json.dumps(dict))
```

En el repositorio de ejemplo, este código se encuentra en *third_party_api/RandomNumber/_init_.py*. La carpeta *RandomNumber* proporciona el nombre de la función y *_init_.py* contiene el código. Otro archivo de la carpeta, *function.json*, describe cuándo se desencadena la función. Otros archivos de la carpeta *principal third_party_api* proporcionan detalles para la aplicación de funciones de Azure que hospeda la propia función.

Para implementar el código, el script de aprovisionamiento del ejemplo realiza los pasos siguientes:

1. Cree una cuenta de almacenamiento de respaldo para Azure Functions utilizando el comando de la CLI de Azure, destinado a administrar el estado y las operaciones internas.
2. Cree una aplicación de Azure Functions con el comando de la CLI de Azure, [az function app create](#).
3. Después de esperar 60 segundos para que el host se aprovisione por completo, implemente el código mediante el comando [Azure Functions Core Tools , func azure functionapp publish](#).
4. Asigne la clave de acceso, `d0c5atM1cr0s0ft`, a la función . (Consulte [Aseguramiento de Azure Functions](#) para obtener información sobre las claves de función).

En el script de aprovisionamiento, este paso se realiza mediante el comando [az functionapp function keys set](#) de la CLI de Azure.

Los comentarios se incluyen para mostrar cómo realizar este paso a través de una llamada api REST a [Functions Key Management API](#) si lo desea. Para llamar a esa API REST, primero debe realizarse otra llamada a la API REST para recuperar la clave maestra de la aplicación de funciones.

También puede asignar claves de acceso a través de [Azure Portal](#). En la página de la aplicación de Funciones, seleccione **Funciones** y, a continuación, seleccione la función específica que desea proteger, la cual está denominada `RandomNumber` en este ejemplo. En la página de la función, seleccione **Claves** de función para abrir la página donde puede crear y administrar estas claves.

Parte 4: Implementación de la aplicación principal >>>

Parte 4: Implementación principal de la aplicación de ejemplo

18/06/2025

Parte anterior: [Implementación de API de terceros](#)

La aplicación principal de nuestro escenario es una aplicación de Flask sencilla que se implementa en Azure App Service. La aplicación proporciona un punto de conexión de API público denominado `/api/v1/getcode`, que genera un código para algún otro propósito en la aplicación (por ejemplo, con autenticación en dos fases para usuarios humanos). La aplicación principal también proporciona una página principal sencilla que muestra un vínculo al punto de conexión de API.

El script de aprovisionamiento del ejemplo realiza los pasos siguientes:

1. Cree el host de App Service e implemente el código con el comando de la CLI de Azure, [az webapp up](#).
2. Cree una cuenta de Azure Storage para la aplicación principal (mediante [az storage account create](#)).
3. Cree una cola en la cuenta de almacenamiento denominada "code-requests" (mediante [az storage queue create](#)).
4. Para asegurarse de que la aplicación puede escribir en la cola, use [az role assignment create](#) para asignar el rol "Colaborador de datos de la cola de Storage" a la aplicación. Para más información sobre los roles, consulte [Asignación de permisos de rol mediante la CLI de Azure](#).

El código de aplicación principal es el siguiente; Se proporcionan explicaciones de detalles importantes en las siguientes partes de esta serie.

Python

```
from flask import Flask, request, jsonify
import requests, random, string, os
from datetime import datetime
from azure.keyvault.secrets import SecretClient
from azure.identity import DefaultAzureCredential
from azure.storage.queue import QueueClient

app = Flask(__name__)
app.config["DEBUG"] = True

number_url = os.environ["THIRD_PARTY_API_ENDPOINT"]
```

```

# Authenticate with Azure. First, obtain the DefaultAzureCredential
credential = DefaultAzureCredential()

# Next, get the client for the Key Vault. You must have first enabled managed
identity
# on the App Service for the credential to authenticate with Key Vault.
key_vault_url = os.environ["KEY_VAULT_URL"]
keyvault_client = SecretClient(vault_url=key_vault_url, credential=credential)

# Obtain the secret: for this step to work you must add the app's service
principal to
# the key vault's access policies for secret management.
api_secret_name = os.environ["THIRD_PARTY_API_SECRET_NAME"]
vault_secret = keyvault_client.get_secret(api_secret_name)

# The "secret" from Key Vault is an object with multiple properties. The key we
# want for the third-party API is in the value property.
access_key = vault_secret.value

# Set up the Storage queue client to which we write messages
queue_url = os.environ["STORAGE_QUEUE_URL"]
queue_client = QueueClient.from_queue_url(queue_url=queue_url,
credential=credential)

@app.route('/', methods=['GET'])
def home():
    return f'Home page of the main app. Make a request to <a href="./api/v1/getcode">/api/v1/getcode</a>.'

def random_char(num):
    return ''.join(random.choice(string.ascii_letters) for x in range(num))

@app.route('/api/v1/getcode', methods=['GET'])
def get_code():
    headers = {
        'Content-Type': 'application/json',
        'x-functions-key': access_key
    }

    r = requests.get(url = number_url, headers = headers)

    if (r.status_code != 200):
        return "Could not get you a code.", r.status_code

    data = r.json()
    chars1 = random_char(3)
    chars2 = random_char(3)
    code_value = f'{chars1}-{data['value']}-{chars2}'
    code = { "code": code_value, "timestamp" : str(datetime.utcnow()) }

    # Log a queue message with the code for, say, a process that invalidates

```

```
# the code after a certain period of time.  
queue_client.send_message(code)  
  
return jsonify(code)  
  
if __name__ == '__main__':  
    app.run()
```

Parte 5: Dependencias y variables de entorno >>>

Parte 5: Dependencias principales de la aplicación, instrucciones de importación y variables de entorno

29/05/2025

[Parte anterior:](#) Implementación de la aplicación principal

En esta sección se revisan las bibliotecas de Python importadas por la aplicación principal y las variables de entorno de las que depende. Cuando la aplicación se implementa en Azure, estas variables de entorno se proporcionan a través de la configuración de la aplicación en Azure App Service.

Dependencias e declaraciones de importación

La aplicación se basa en las bibliotecas siguientes:

- Flask: para definir la API web
- requests: el cliente HTTP estándar de Python para realizar llamadas de API externas
- azure.identity – para gestionar la autenticación basada en tokens de Microsoft Entra ID
- azure.keyvault.secrets: para recuperar de forma segura los secretos de Azure Key Vault
- azure.storage.queue: para interactuar con Azure Queue Storage

Estas dependencias se incluyen en el archivo *requirements.txt* de la aplicación y se instalan durante la implementación o la configuración local.

txt

```
flask
requests
azure.identity
azure.keyvault.secrets
azure.storage.queue
```

Al implementar la aplicación en Azure App Service, Azure instala automáticamente estos requisitos en el servidor host. Cuando se ejecuta localmente, se instalan en el entorno con `pip install -r requirements.txt`.

El archivo de código comienza con las instrucciones de importación necesarias para las partes de las bibliotecas usadas en el código:

Python

```

from flask import Flask, request, jsonify
import requests, random, string, os
from datetime import datetime
from azure.keyvault.secrets import SecretClient
from azure.identity import DefaultAzureCredential
from azure.storage.queue import QueueClient

```

Variables de entorno

El código de la aplicación depende de estas cuatro variables de entorno:

 Expandir tabla

Variable	Valor
THIRD_PARTY_API_ENDPOINT	Dirección URL de la API de terceros, como <code>https://msdocs-example-api.azurewebsites.net/api/RandomNumber</code> , que se describe en la Parte 3 .
KEY_VAULT_URL	Dirección URL de Azure Key Vault en la que almacenó la clave de acceso para la API de terceros.
THIRD_PARTY_API_SECRET_NAME	Nombre del secreto en Key Vault que contiene la clave de acceso de la API de terceros.
STORAGE_QUEUE_URL	La dirección URL de una cola de almacenamiento de Azure que configure en Azure, como <code>https://msdocsexamplemainapp.queue.core.windows.net/code-requests</code> (consulte la parte 4). Dado que el nombre de la cola se incluye al final de la dirección URL, no verá el nombre en ninguna parte del código.

La forma de establecer estas variables depende de dónde se ejecuta el código:

- Al ejecutar el código localmente, cree estas variables en el shell de comandos que use (como PowerShell, Bash o CMD). (Si implementa la aplicación en una máquina virtual, crearía variables del lado servidor similares). También puede usar una biblioteca como [python-dotenv](#), que lee pares clave-valor de un archivo `.env` y los establece como variables de entorno.
- Cuando el código se implementa en Azure App Service, como se muestra en este tutorial, no tiene acceso al propio servidor. En su lugar, debe definir los [ajustes de la aplicación](#) con los mismos nombres en la configuración del servicio App Service. Esta configuración se expone automáticamente a la aplicación como variables de entorno.

Los scripts de aprovisionamiento crean esta configuración mediante el comando de la CLI de Azure, [az webapp config appsettings set](#). Las cuatro variables se establecen con un solo comando.

Para crear configuraciones a través del portal de Azure, consulte [Configurar una aplicación de servicio de aplicaciones en el portal de Azure](#).

Al ejecutar el código localmente, también debe especificar variables de entorno que contengan información sobre la entidad de servicio local. `DefaultAzureCredential` busca estos valores.

Cuando se implementa en App Service, no es necesario establecer estos valores, ya que la identidad administrada asignada por el sistema de la aplicación se usa en su lugar para autenticarse.

 Expandir tabla

Variable	Valor
AZURE_TENANT_ID	Id. de inquilino de Microsoft Entra (directorio).
AZURE_CLIENT_ID	ID de cliente (aplicación) de un registro de aplicación en el inquilino.
AZURE_CLIENT_SECRET	Secreto de cliente que se generó para el registro de aplicaciones.

Para obtener más información, consulte [Autenticación de aplicaciones de Python en servicios de Azure durante el desarrollo local mediante entidades de servicio](#).

parte 6: código de inicio de la aplicación principal >>>

Parte 6: Código de inicio de la aplicación principal

29/05/2025

Parte anterior: Dependencias y variables de entorno

Inmediatamente después de las `import` instrucciones , el código de inicio de la aplicación inicializa las variables clave usadas en las funciones de control de solicitudes.

En primer lugar, la aplicación crea el objeto de aplicación flask, que sirve como base para definir rutas y controlar las solicitudes HTTP entrantes. A continuación, recupera la dirección URL del punto de conexión de API de terceros de una variable de entorno. Esto permite configurar fácilmente el punto de conexión sin modificar el código base:

Python

```
app = Flask(__name__)
app.config["DEBUG"] = True

number_url = os.environ["THIRD_PARTY_API_ENDPOINT"]
```

A continuación, obtiene el `DefaultAzureCredential` objeto , que es la credencial recomendada que se usará al autenticarse con los servicios de Azure. Consulte [Autenticación de aplicaciones hospedadas en Azure con DefaultAzureCredential](#).

Python

```
credential = DefaultAzureCredential()
```

Cuando se ejecuta localmente, `DefaultAzureCredential` busca las variables de entorno

`AZURE_TENANT_ID`, `AZURE_CLIENT_ID` y `AZURE_CLIENT_SECRET` que contienen información para el principal de servicio que está usando para el desarrollo local. Cuando se ejecuta en Azure, `DefaultAzureCredential` el valor predeterminado es usar la identidad administrada asignada por el sistema habilitada en la aplicación. Es posible invalidar el comportamiento predeterminado con la configuración de la aplicación, pero en este escenario de ejemplo, se usa el comportamiento predeterminado.

El código siguiente recupera la clave de acceso de la API de terceros de Azure Key Vault. En el script de aprovisionamiento, key Vault se crea mediante [az keyvault create](#)y el secreto se almacena con [az keyvault secret set](#).

Se accede al propio recurso de Key Vault a través de una dirección URL, que se carga desde la `KEY_VAULT_URL` variable de entorno.

Python

```
key_vault_url = os.environ["KEY_VAULT_URL"]
```

Para recuperar un secreto de Azure Key Vault, la aplicación debe crear un objeto de cliente que se comunique con el servicio Key Vault. Dado que el objetivo es leer un secreto, la aplicación usa la `SecretClient` clase de la `azure.keyvault.secrets` biblioteca. Este cliente requiere dos entradas:

- Dirección URL de Key Vault: normalmente se recupera de una variable de entorno.
- Objeto de credencial, como la `DefaultAzureCredential` instancia creada anteriormente, que representa la identidad en la que se ejecuta la aplicación.

Python

```
keyvault_client = SecretClient(vault_url=key_vault_url, credential=credential)
```

La creación de un `SecretClient` objeto no autentica inmediatamente la aplicación. El cliente es simplemente una construcción local que almacena la dirección URL de Key Vault y el objeto de credencial. La autenticación y la autorización solo se producen cuando se invoca una operación a través del cliente, como `get_secret`, que genera una llamada API REST al recurso de Azure.

Python

```
api_secret_name = os.environ["THIRD_PARTY_API_SECRET_NAME"]
vault_secret = keyvault_client.get_secret(api_secret_name)

# The "secret" from Key Vault is an object with multiple properties. The key we
# want for the third-party API is in the value property.
access_key = vault_secret.value
```

Incluso si la identidad de una aplicación está autorizada para acceder a Azure Key Vault, también debe estar autorizada explícitamente para realizar operaciones específicas, como la lectura de secretos. Sin este permiso, se produce un error en una llamada a `get_secret()`, incluso si la identidad es válida de otro modo. Para solucionar este problema, el script de aprovisionamiento establece una directiva de acceso "obtener secretos" para la aplicación mediante el comando de la CLI de Azure, `az keyvault set-policy`. Para más información, consulte [Autenticación de Key Vault](#) y [Concesión de acceso a la aplicación a Key Vault](#). En este último artículo se muestra cómo establecer una directiva de acceso mediante Azure Portal. (El artículo

también se escribe para la identidad administrada, pero se aplica igualmente a un principio de servicio que se usa en el desarrollo local).

Por último, el código de la aplicación configura el objeto de cliente a través del cual puede escribir mensajes en una cola de Azure Storage. La dirección URL de la cola está en la variable de entorno `STORAGE_QUEUE_URL`.

Python

```
queue_url = os.environ["STORAGE_QUEUE_URL"]
queue_client = QueueClient.from_queue_url(queue_url=queue_url,
                                         credential=credential)
```

Al igual que con Azure Key Vault, la aplicación usa un objeto de cliente específico del SDK de Azure para interactuar con Azure Queue Storage. En este caso, usa la `QueueClient` clase de la biblioteca `azure-storage-queue`.

Para inicializar el cliente, la aplicación usa el método `from_queue_url`, proporcionando la dirección URL completa de la cola y un objeto de credencial. Este objeto de credencial es de nuevo la `DefaultAzureCredential` instancia creada anteriormente, que representa la identidad en la que se ejecuta la aplicación.

Como se indicó anteriormente en esta guía, dicha autorización se concede asignando el rol "Colaborador de datos de almacenamiento en cola" a la identidad de la aplicación, ya sea una identidad administrada en Azure o un principal del servicio durante el desarrollo local. Esta asignación de roles se realiza en el script de aprovisionamiento mediante el comando `az role assignment create` de la CLI de Azure .

Suponiendo que todo este código de inicio se ejecuta correctamente, la aplicación tiene todas sus variables internas en lugar para admitir su punto de conexión API `/api/v1/getcode`.

[Parte 7: Punto de conexión de la aplicación principal >>>](#)

Parte 7: Punto de conexión principal de la API de la aplicación

28/05/2025

parte anterior: código de inicio de la aplicación principal

La ruta URL de la aplicación `/api/v1/getcode` en la API genera una respuesta JSON que contiene un código alfanumérico y un sello de tiempo.

En primer lugar, el decorador `@app.route` indica a Flask que la función `get_code` gestiona las solicitudes a la URL `/api/v1/getcode`.

Python

```
@app.route('/api/v1/getcode', methods=['GET'])
def get_code():
```

A continuación, la aplicación llama a la API de terceros, cuya URL está en `number_url`, proporcionando en la cabecera la clave de acceso que recupera del almacén de claves.

Python

```
headers = {
    'Content-Type': 'application/json',
    'x-functions-key': access_key
}

r = requests.get(url = number_url, headers = headers)

if (r.status_code != 200):
    return "Could not get you a code.", r.status_code
```

La API de terceros de ejemplo se implementa en el entorno sin servidor de Azure Functions. La propiedad `x-functions-key` del encabezado es cómo Azure Functions espera que aparezca una clave de acceso en un encabezado. Para más información, consulte [desencadenador HTTP de Azure Functions: claves de autorización](#). Si se produce un error en la llamada a la API por cualquier motivo, el código devuelve un mensaje de error y el código de estado.

Suponiendo que la llamada API se realiza correctamente y devuelve un valor numérico, la aplicación crea un código más complejo con ese número más algunos caracteres aleatorios (mediante su propia función de `random_char`).

Python

```
data = r.json()
chars1 = random_char(3)
chars2 = random_char(3)
code_value = f"{chars1}-{data['value']}-{chars2}"
code = { "code": code_value, "timestamp" : str(datetime.utcnow()) }
```

La variable `code` aquí contiene la respuesta JSON completa para la API de la aplicación, que incluye el valor de código y una marca de tiempo. Una respuesta de ejemplo sería `{"code":"ojE-161-pTv","timestamp":"2020-04-15 16:54:48.816549"}`.

Sin embargo, antes de devolver esa respuesta, escribe un mensaje en la cola de almacenamiento mediante el método `send_message` del cliente de cola:

```
Python

queue_client.send_message(code)

return jsonify(code)
```

Procesamiento de mensajes de cola

Los mensajes almacenados en la cola se pueden ver y administrar mediante [Azure Portal](#), con el comando `az storage message get` de la CLI de Azure o con el [Explorador de Azure Storage](#). El repositorio de ejemplo incluye un script (`test.cmd` y `test.sh`) para solicitar un código desde el punto de conexión de la aplicación y, a continuación, comprobar la cola de mensajes. También hay un script para borrar la cola mediante el comando `az storage message clear`.

Normalmente, una aplicación como la de este ejemplo tendría otro proceso que extrae de forma asíncrona los mensajes de la cola para su posterior procesamiento. Como se mencionó anteriormente, la respuesta generada por este punto de conexión de API podría usarse en otra parte de la aplicación con autenticación de usuario en dos fases. En ese caso, la aplicación debe invalidar el código después de un período de tiempo determinado, por ejemplo, 10 minutos. Una manera sencilla de realizar esta tarea sería mantener una tabla de códigos de autenticación en dos fases válidos, que usa su procedimiento de inicio de sesión de usuario. A continuación, la aplicación tendría un proceso sencillo de inspección de colas con la siguiente lógica (en pseudocódigo):

```
pseudo

pull a message from the queue and retrieve the code.

if (code is already in the table):
```

```
remove the code from the table, thereby invalidating it
else:
    add the code to the table, making it valid
    call queue_client.send_message(code, visibility_timeout=600)
```

Este pseudocódigo emplea el parámetro opcional [send_message](#) del método [visibility_timeout](#), que especifica el número de segundos antes de que el mensaje se vea en la cola. Dado que el tiempo de espera predeterminado es cero, los mensajes escritos inicialmente por el endpoint de API se vuelven inmediatamente visibles para el proceso de monitoreo de cola. Como resultado, ese proceso los almacena en la tabla de código válida inmediatamente. El proceso pone en cola el mismo mensaje nuevamente con el límite de tiempo de espera, para que vuelva a recibir el código 10 minutos más tarde, momento en el cual elimina el mensaje de la tabla.

Implementación del punto de conexión de API de la aplicación principal en Azure Functions

El código que se muestra anteriormente en este artículo usa el marco web de Flask para crear su punto de conexión de API. Dado que Flask debe ejecutarse con un servidor web, este código debe implementarse en Azure App Service o en una máquina virtual.

Una opción de implementación alternativa es el entorno sin servidor de Azure Functions. En este caso, todo el código de inicio y el código de punto de conexión de API se incluirán en la misma función enlazada a un desencadenador HTTP. Al igual que con App Service, se usa la [configuración de la aplicación de funciones](#) para crear las variables de entorno para el código.

Una parte de la implementación que se vuelve más fácil es autenticarse con Queue Storage. En lugar de obtener un objeto [QueueClient](#) mediante la dirección URL de la cola y un objeto de credencial, se crea un *enlace de Queue Storage* para la función. El enlace controla toda la autenticación en segundo plano. Con este tipo de enlace, a la función se le asigna un objeto de cliente listo para usar como parámetro. Para obtener más información y ejemplos de código, consulte [Funciones de Azure conectadas a Azure Queue Storage](#).

Pasos siguientes

A través de este tutorial, ha aprendido cómo las aplicaciones se autentican con otros servicios de Azure mediante identidad administrada y cómo las aplicaciones pueden usar Azure Key Vault para almacenar cualquier otro secreto necesario para las API de terceros.

El mismo patrón que se muestra aquí con Azure Key Vault y Azure Storage se aplica a todos los demás servicios de Azure. El paso fundamental es asignar el rol correcto para la aplicación

dentro de la página de ese servicio en Azure Portal o a través de la CLI de Azure. (Consulte [Cómo asignar roles de Azure](#)). Asegúrese de comprobar la documentación del servicio para ver si necesita configurar otras directivas de acceso.

Recuerde siempre que debe asignar los mismos roles y directivas de acceso a cualquier entidad de servicio que use para el desarrollo local.

En resumen, una vez completado este tutorial, puede aplicar sus conocimientos a cualquier número de otros servicios de Azure y a cualquier número de otros servicios externos.

Un tema que no hemos tocado en este tutorial es la autenticación de usuarios . Para explorar esta área para aplicaciones web, empiece con [Autenticar y autorizar a los usuarios de extremo a extremo en Azure App Service](#).

Consulte también

- [Autenticación y autorización de aplicaciones de Python en Azure](#)
- Ejemplo del tutorial: [github.com/Azure-Samples/python-integrated-authentication ↗](https://github.com/Azure-Samples/python-integrated-authentication)
- [Documentación de Microsoft Entra](#)
- [Documentación de Azure Key Vault](#)

Authorization in the Azure SDK libraries for Python

11/07/2025

Authorization in Azure determines what actions authenticated users or services can perform on resources. This article explores how to implement authorization using the [Azure SDK for Python](#), covering models, implementation, troubleshooting, and best practices. For detailed authentication setup, refer to [Authenticate Python apps to Azure](#).

Introduction

Authentication (AuthN) verifies the identity of a user or service, while **authorization (AuthZ)** defines what they can do. In Azure, authorization ensures secure access to resources, critical for protecting applications and data. Developers can implement robust authorization with the Azure SDK for Python to control access in various workflows, from managing resources to accessing service-specific data.

Azure authorization models

Azure provides multiple authorization mechanisms to manage access. Understanding these models is essential for effective access control.

Azure Role-Based Access Control

[Azure Role-Based Access Control \(RBAC\)](#) assigns roles to identities at scopes like subscriptions or resource groups. Built-in roles include Owner, Contributor, and Reader, while custom roles allow tailored permissions. When you assign a role at a specific scope, the identity (like a user or service) gets permissions for all resources within that scope and its child scopes. For example, assigning the Contributor role at the subscription level allows management of all resources in that subscription. See [Understand scope for Azure RBAC](#).

Service-specific mechanisms

Some Azure services offer unique authorization methods:

- **Azure Storage:** Uses Shared Access Signatures (SAS) and Access Control Lists (ACLs) for data access. See [Service-Specific Authorization Notes for Azure Storage](#).
- **Azure Key Vault:** Recommends RBAC over legacy access policies. See [Service-Specific Authorization Notes for Azure Key Vault](#).

- Microsoft Graph: Employs OAuth 2.0 scopes and application permissions. See [Service-Specific Authorization Notes for Microsoft Graph](#).

Use authorization in Azure SDK for Python

The Azure SDK for Python provides built-in support for handling authentication and authorization through the `azure-identity` package. The `DefaultAzureCredential` class supports various authentication mechanisms, such as managed identities and service principals, adapting to different environments.

Example: List resource groups

This example shows how the Azure SDK for Python uses a credential (via `DefaultAzureCredential`) to authenticate, and how authorization determines whether the identity can successfully list resource groups. If the identity lacks the Reader or higher role on the subscription or resource group scope, this call returns a 403 Forbidden error.

Python

```
from azure.identity import DefaultAzureCredential
from azure.mgmt.resource import ResourceManagementClient

credential = DefaultAzureCredential()
client = ResourceManagementClient(credential, "<subscription-id>")
resource_groups = client.resource_groups.list()
for rg in resource_groups:
    print(rg.name)
```

Replace `<subscription-id>` with your Azure subscription ID, which is usually in the form of `00000000-0000-0000-0000-000000000000`.

Microsoft Graph with scopes

To access Microsoft Graph, use the official [Microsoft Graph SDK for Python](#), which supports both delegated and application permissions.

This example demonstrates how the SDK uses a credential to request an access token with the required authorization scope `https://graph.microsoft.com/.default` and access Microsoft Graph resources. The identity must be authorized in Microsoft Entra ID with appropriate application permissions (such as `User.Read.All`) to retrieve user data; otherwise, the request fails with a 403 Forbidden.

Importante

Ensure your app or identity has the `User.Read.All` or other required permissions granted in Microsoft Entra ID.

Python

```
from azure.identity import DefaultAzureCredential
from msgraph.core import GraphClient

credential = DefaultAzureCredential()
client = GraphClient(credential=credential, scopes=[
    "https://graph.microsoft.com/.default"])

response = client.get("/users")
users = response.json().get("value", [])
for user in users:
    print(user["displayName"])
```

Learn more about this SDK in the official [Build Python apps with Microsoft Graph](#) tutorial.

Diagnose authorization errors

Authorization issues often result in HTTP 403 Forbidden errors, indicating insufficient permissions. To diagnose:

- **Check Error Messages:** Review the response for details on missing permissions.
- **Enable Logging:** Use Python logging to inspect requests and responses:

Python

```
import logging
logging.basicConfig(level=logging.DEBUG)
```

- **Verify Access:** Use [Azure CLI](#) or the Azure portal to check role assignments:

Azure CLI

```
az role assignment list --assignee <principal-id> --scope <scope>
```

Replace `<principal-id>` with the object ID of your user, service principal, or managed identity. Replace `<scope>` with an Azure resource scope, such as a subscription ID, a resource group name, or a resource. See [Work with scopes](#).

Work with scopes

Often you'll need to provide a scope, like in the example:

Azure CLI

```
az role assignment list \
--assignee <principal-id> \
--scope <scope>
```

Here are some common scope formats:

 Expandir tabla

Scope Level	Example Value
Subscription	/subscriptions/<subscription-id>
Resource Group	/subscriptions/<subscription-id>/resourceGroups/<resource-group-name>
Resource Name	/subscriptions/<subscription-id>/resourceGroups/<resource-group-name>/providers/<provider-namespace>/<resource-type>/<resource-name>

For example, to list all role assignments for a managed identity at the resource group level:

Azure CLI

```
az role assignment list \
--assignee 12345678-90ab-cdef-1234-567890abcdef \
--scope /subscriptions/<subscription-id>/resourceGroups/my-resource-group
```

Or at the subscription level:

Azure CLI

```
az role assignment list \
--assignee 12345678-90ab-cdef-1234-567890abcdef \
--scope /subscriptions/<subscription-id>
```

You can retrieve the object ID (<principal-id>) of a user or managed identity using:

Azure CLI

```
az ad user show --id <user-email> --query objectId
az identity show --name <identity-name> --resource-group <rg-name> --query
```

principalId

Manage access

Manage access through role assignments using:

- **Azure portal:** Add roles via the "Access control (IAM)" service menu
- **Azure CLI:**

Azure CLI

```
az role assignment create --assignee <principal-id> --role <role-name> --  
scope <scope>
```

Replace `<principal-id>` with the object ID of your user, service principal, or managed identity. Replace `<scope>` with an Azure resource scope, such as a subscription ID, a resource group name, or a resource name. See [Work with scopes](#).

- **ARM Templates:** For declarative management.

For managed identities, assign roles to the identity associated with resources like virtual machines. Use the Azure portal's "Check access" feature or the Azure CLI to verify effective permissions.

Service-specific authorization notes

In the section [Service-specific mechanisms](#), it was noted that some Azure services offer unique authorization methods. This section provides more detail for each of the three Azure services mentioned.

Azure Storage

- **RBAC:** Manages control plane operations.
- **SAS and ACLs:** Control data plane access, with Microsoft Entra authentication also supported.

Example: Access blobs with Microsoft Entra ID

Python

```
from azure.identity import DefaultAzureCredential
from azure.storage.blob import BlobServiceClient

credential = DefaultAzureCredential()
client = BlobServiceClient(account_url="https://<account-name>.blob.core.windows.net", credential=credential)
containers = client.list_containers()
for container in containers:
    print(container.name)
```

Replace `<account-name>` with your Azure Storage account name.

Azure Key Vault

RBAC is recommended over legacy access policies for consistency. Access policies are still supported but not preferred.

Example: Retrieve a secret

Python

```
from azure.identity import DefaultAzureCredential
from azure.keyvault.secrets import SecretClient

credential = DefaultAzureCredential()
client = SecretClient(vault_url="https://<vault-name>.vault.azure.net",
                      credential=credential)
secret = client.get_secret("my-secret")
print(secret.value)
```

Replace `<vault-name>` with your Key Vault resource name.

Microsoft Graph

Uses OAuth 2.0 scopes for delegated permissions and application permissions for daemon apps. Specify scopes as shown in the earlier example.

Best practices

- **Least privilege:** Assign only necessary permissions (for example, Reader instead of Contributor).
- **Prefer RBAC:** Especially for Key Vault, for unified access control.
- **Use Managed Identities:** Avoid managing credentials in code.

- **Limit Graph permissions:** Request specific scopes to minimize risks.

Next steps

- [Azure Role-Based Access Control \(RBAC\)](#)
- [Azure CLI Reference](#)
- [Azure SDK for Python Overview](#)
- [Azure Key Vault RBAC Guide](#)
- [Microsoft Graph Permissions Reference](#)

Instalación de paquetes de biblioteca de Azure para Python

22/06/2025

El SDK de Azure para Python se compone de muchas bibliotecas individuales que se pueden instalar en entornos estándar de [Python](#) o [conda](#).

Las bibliotecas para entornos estándar de Python se muestran en el índice del paquete de .

Los paquetes para entornos de Conda se muestran en el [canal de Microsoft en anaconda.org](#). Los paquetes de Azure tienen nombres que comienzan por `azure-`.

Con estas bibliotecas de Azure, puede crear y administrar recursos en los servicios de Azure (mediante las bibliotecas de administración, cuyos nombres de paquete comienzan por `azure-mgmt`) y conectarse con esos recursos desde el código de la aplicación (mediante las bibliotecas cliente, cuyos nombres de paquete comienzan por solo `azure-`).

Instalación de la versión más reciente de un paquete

pepita

Símbolo del sistema de Windows

```
pip install <package>
```

`pip install` recupera la versión más reciente de un paquete en el entorno de Python actual.

En los sistemas Linux, debe instalar un paquete para cada usuario por separado. No se admite la instalación de paquetes para todos los usuarios con `sudo pip install`.

Puede usar cualquier nombre de paquete que aparezca en el índice de paquete . En la página de índice, busque en la columna **Nombre** la funcionalidad que necesita y luego encuentre y seleccione el enlace PyPI en la columna **Paquete**.

Instalación de versiones de paquetes específicas

pepita

Especifique la versión deseada en la línea de comandos con `pip install`.

Símbolo del sistema de Windows

```
pip install <package>==<version>
```

Puede encontrar números de versión en el índice del paquete . En la página de índice, busque en la columna **Nombre** la funcionalidad que necesita y luego encuentre y seleccione el enlace PyPI en la columna **Paquete**. Por ejemplo, para instalar una versión del paquete de `azure-storage-blob` puede usar: `pip install azure-storage-blob==12.19.0`.

Instalación de paquetes de versión preliminar

pepita

Para instalar la versión preliminar más reciente de un paquete, incluya la marca `--pre` en la línea de comandos.

Símbolo del sistema de Windows

```
pip install --pre <package>
```

Microsoft publica periódicamente paquetes de versión preliminar que admiten próximas características. Los paquetes en versión preliminar incluyen la advertencia de que el paquete está sujeto a cambios y no se debe usar en proyectos de producción.

Puede usar cualquier nombre de paquete que aparezca en el índice de paquete .

Comprobación de una instalación de paquetes

pepita

Para comprobar la instalación de un paquete:

Símbolo del sistema de Windows

```
pip show <package>
```

Si el paquete está instalado, `pip show` muestra la versión y otra información de resumen; de lo contrario, el comando no muestra nada.

También puede usar `pip freeze` o `pip list` para ver todos los paquetes instalados en el entorno de Python actual.

Puede usar cualquier nombre de paquete que aparezca en el índice de paquete .

Desinstalación de un paquete

pepita

Para desinstalar un paquete:

Símbolo del sistema de Windows

```
pip uninstall <package>
```

Puede usar cualquier nombre de paquete que aparezca en el índice de paquete .

Ejemplo: Uso de las bibliotecas de Azure para crear un grupo de recursos

30/05/2025

En este ejemplo se muestra cómo usar las bibliotecas de administración del SDK de Azure en un script de Python para crear un grupo de recursos. (El [comando equivalente](#) de la CLI de Azure se proporciona más adelante en este artículo. Si prefiere usar Azure Portal, consulte [Creación de grupos de recursos](#)).

Todos los comandos de este artículo funcionan iguales en los shells de comandos de Linux/macOS y Windows, a menos que se indique.

1: Configuración del entorno de desarrollo local

Si aún no lo ha hecho, configure un entorno en el que pueda ejecutar este código. Estas son algunas opciones:

- Configure un entorno virtual de Python mediante `venv` o la herramienta que prefiera. Para empezar a usar el entorno virtual, asegúrese de activarlo. Para instalar Python, consulte [Instalación de Python](#).



Azure CLI

```
#!/bin/bash
# Create a virtual environment
python -m venv .venv
# Activate the virtual environment
source .venv/Scripts/activate # only required for Windows (Git Bash)
```

- Usa un entorno conda. Para instalar Conda, consulte [Instalación de Miniconda](#).
- Usa un contenedor de desarrollo en Visual Studio Code o en GitHub Codespaces.

2: Instalación de los paquetes de biblioteca de Azure

- En la consola, cree un archivo `requirements.txt` que muestre las bibliotecas de administración que se usan en este ejemplo:

Azure CLI

```
azure-mgmt-resource  
azure-identity
```

2. En la consola con el entorno virtual activado, instale los requisitos:

Consola

```
pip install -r requirements.txt
```

3. Establecer variables de entorno

En este paso, establecerá las variables de entorno para usarlas en el código de este artículo. El código usa el `os.environ` método para recuperar los valores.

Juerga

Azure CLI

```
#!/bin/bash  
export AZURE_RESOURCE_GROUP_NAME=<ResourceGroupName> # Change to your  
preferred resource group name  
export LOCATION=<Location> # Change to your preferred region  
export AZURE_SUBSCRIPTION_ID=$(az account show --query id --output tsv)
```

4: Escribir código para crear un grupo de recursos

En este paso, creará un archivo de Python denominado *provision_blob.py* con el código siguiente. Este script de Python usa el SDK de Azure para bibliotecas de administración de Python para crear un grupo de recursos en la suscripción de Azure.

Cree un archivo de Python denominado *provision_rg.py* con el código siguiente. Los comentarios explican los detalles:

Python

```
# Import the needed credential and management objects from the libraries.  
import os  
  
from azure.identity import DefaultAzureCredential  
from azure.mgmt.resource import ResourceManagementClient
```

```
# Acquire a credential object using DefaultAzureCredential.
credential = DefaultAzureCredential()

# Retrieve subscription ID from environment variable.
subscription_id = os.environ["AZURE_SUBSCRIPTION_ID"]

# Retrieve resource group name and location from environment variables
RESOURCE_GROUP_NAME = os.environ["AZURE_RESOURCE_GROUP_NAME"]
LOCATION = os.environ["LOCATION"]

# Obtain the management object for resources.
resource_client = ResourceManagementClient(credential, subscription_id)

# Provision the resource group.
rg_result = resource_client.resource_groups.create_or_update(RESOURCE_GROUP_NAME,
    { "location": LOCATION })

print(f"Provisioned resource group {rg_result.name}")

# Within the ResourceManagementClient is an object named resource_groups,
# which is of class ResourceGroupsOperations, which contains methods like
# create_or_update.
#
# The second parameter to create_or_update here is technically a ResourceGroup
# object. You can create the object directly using ResourceGroup(location=
# LOCATION) or you can express the object as inline JSON as shown here. For
# details, see Inline JSON pattern for object arguments at
# https://learn.microsoft.com/azure/developer/python/sdk
# /azure-sdk-library-usage-patterns#inline-json-pattern-for-object-arguments

print(
    f"Provisioned resource group {rg_result.name} in the {rg_result.location}
region"
)

# The return value is another ResourceGroup object with all the details of the
# new group. In this case the call is synchronous: the resource group has been
# provisioned by the time the call returns.

# To update the resource group, repeat the call with different properties, such
# as tags:
rg_result = resource_client.resource_groups.create_or_update(
    RESOURCE_GROUP_NAME,
    {
        "location": LOCATION,
        "tags": {"environment": "test", "department": "tech"},
    },
)
print(f"Updated resource group {rg_result.name} with tags")

# Optional lines to delete the resource group. begin_delete is asynchronous.
# poller = resource_client.resource_groups.begin_delete(rg_result.name)
# result = poller.result()
```

Autenticación en el código

Más adelante en este artículo, inicia sesión en Azure mediante la CLI de Azure para ejecutar el código de ejemplo. Si la cuenta tiene permisos suficientes para crear grupos de recursos y recursos de almacenamiento en la suscripción de Azure, el script debe ejecutarse correctamente sin configuración adicional.

Para usar este código en un entorno de producción, autentíquese mediante una entidad de servicio estableciendo variables de entorno. Este enfoque permite el acceso seguro y automatizado sin depender del inicio de sesión interactivo. Para obtener instrucciones detalladas, consulte [Autenticación de aplicaciones de Python con servicios de Azure](#).

Asegúrese de que a la entidad de servicio se le asigna un rol con permisos suficientes para crear grupos de recursos y cuentas de almacenamiento. Por ejemplo, asignar el rol Colaborador en el nivel de suscripción proporciona el acceso necesario. Para más información sobre las asignaciones de roles, consulte [Control de acceso basado en rol \(RBAC\) en Azure](#).

Vínculos de referencia para las clases usadas en el código

- [DefaultAzureCredential \(azure.identity\)](#)
- [ResourceManagementClient \(azure.mgmt.resource\)](#)

5: Ejecutar el script

1. Si aún no lo ha hecho, inicie sesión en Azure mediante la CLI de Azure:

```
Azure CLI
```

```
az login
```

2. Ejecuta el script:

```
Símbolo del sistema de Windows
```

```
python provision_rg.py
```

6: Comprobación del grupo de recursos

Puede comprobar que el grupo de recursos existe a través de Azure Portal o la CLI de Azure.

- Azure Portal: abra [Azure Portal](#), seleccione **Grupos de recursos** y compruebe que aparece el grupo. Si es necesario, use el comando **Actualizar** para actualizar la lista.
- CLI de Azure: use el comando [az group show](#) :

```
Juerga
```

```
Azure CLI
```

```
#!/bin/bash
az group show -n $AZURE_RESOURCE_GROUP_NAME
```

7: Limpieza de recursos

Ejecute el comando [az group delete](#) si no necesita conservar el grupo de recursos creado en este ejemplo. Los grupos de recursos no incurren en cargos continuos en la suscripción, pero es posible que los recursos del grupo de recursos sigan generando cargos. Se recomienda limpiar cualquier grupo que no se esté usando activamente. El argumento `--no-wait` permite que el comando devuelva inmediatamente en lugar de esperar a que finalice la operación.

```
Juerga
```

```
Azure CLI
```

```
#!/bin/bash
az group delete -n $AZURE_RESOURCE_GROUP_NAME --no-wait
```

También puede usar el método [ResourceManagementClient.resource_groups.begin_delete](#) para eliminar un grupo de recursos del código. El código comentado en la parte inferior del script de este artículo muestra el uso.

Como referencia: comando equivalente de la CLI de Azure

El siguiente comando [az group create](#) de la CLI de Azure crea un grupo de recursos con etiquetas igual que el script de Python:

```
Azure CLI
```

```
az group create -n PythonAzureExample-rg -l centralus --tags "department=tech"
"environment=test"
```

Consulte también

- [Ejemplo: Enumeración de grupos de recursos en una suscripción](#)
- [Ejemplo: Creación de Azure Storage](#)
- [Ejemplo de uso de Azure Storage](#)
- [Ejemplo: Creación de una aplicación web e implementación de código](#)
- [Ejemplo: Creación y consulta de una base de datos](#)
- [Ejemplo: Creación de una máquina virtual](#)
- [Uso de Azure Managed Disks con máquinas virtuales](#)
- [Complete una breve encuesta sobre el SDK de Azure para Python ↗](#)

Ejemplo: Uso de las bibliotecas de Azure para enumerar los grupos de recursos y los recursos

Artículo • 30/04/2025

En este ejemplo se muestra cómo usar las bibliotecas de administración del SDK de Azure en un script de Python para realizar dos tareas:

- Enumere todos los grupos de recursos de una suscripción de Azure.
- Enumera los recursos dentro de un grupo de recursos específico.

Todos los comandos de este artículo funcionan iguales en los shells de comandos de Linux/macOS y Windows, a menos que se indique.

Los [comandos equivalentes](#) de la CLI de Azure se enumeran más adelante en este artículo.

1: Configuración del entorno de desarrollo local

Si aún no lo ha hecho, configure un entorno en el que pueda ejecutar este código. Estas son algunas opciones:

- Configure un entorno virtual de Python mediante `venv` o la herramienta que prefiera. Para empezar a usar el entorno virtual, asegúrese de activarlo. Para instalar Python, consulte [Instalación de Python](#).



```
#!/bin/bash
# Create a virtual environment
python -m venv .venv
# Activate the virtual environment
source .venv/Scripts/activate # only required for Windows (Git Bash)
```

- Usa un entorno conda. Para instalar Conda, consulte [Instalación de Miniconda](#).
- Usa un contenedor de desarrollo en Visual Studio Code o en GitHub Codespaces.

2: Instalación de los paquetes de biblioteca de Azure

Cree un archivo denominado *requirements.txt* con el siguiente contenido:

```
txt  
  
azure-mgmt-resource  
azure-identity
```

En un terminal o línea de comandos con el entorno virtual activado, instale los requisitos necesarios:

```
Consola  
  
pip install -r requirements.txt
```

3: Escribir código para trabajar con grupos de recursos

3a. Enumeración de grupos de recursos en una suscripción

Cree un archivo de Python denominado *list_groups.py* con el código siguiente. Los comentarios explican los detalles:

```
Python  
  
# Import the needed credential and management objects from the libraries.  
from azure.identity import DefaultAzureCredential  
from azure.mgmt.resource import ResourceManagementClient  
import os  
  
# Acquire a credential object.  
credential = DefaultAzureCredential()  
  
# Retrieve subscription ID from environment variable.  
subscription_id = os.environ["AZURE_SUBSCRIPTION_ID"]  
  
# Obtain the management object for resources.  
resource_client = ResourceManagementClient(credential, subscription_id)  
  
# Retrieve the list of resource groups  
group_list = resource_client.resource_groups.list()
```

```

# Show the groups in formatted output
column_width = 40

print("Resource Group".ljust(column_width) + "Location")
print("-" * (column_width * 2))

for group in list(group_list):
    print(f"{group.name:<{column_width}}{group.location}")

```

3b. Enumeración de recursos dentro de un grupo de recursos específico

Cree un archivo de Python denominado *list_resources.py* con el código siguiente. Los comentarios explican los detalles.

De forma predeterminada, el código enumera los recursos en "myResourceGroup". Para usar otro grupo de recursos, establezca la `RESOURCE_GROUP_NAME` variable de entorno en el nombre del grupo deseado.

Python

```

# Import the needed credential and management objects from the libraries.
from azure.identity import DefaultAzureCredential
from azure.mgmt.resource import ResourceManagementClient
import os

# Acquire a credential object.
credential = DefaultAzureCredential()

# Retrieve subscription ID from environment variable.
subscription_id = os.environ["AZURE_SUBSCRIPTION_ID"]

# Retrieve the resource group to use, defaulting to "myResourceGroup".
resource_group = os.getenv("RESOURCE_GROUP_NAME", "myResourceGroup")

# Obtain the management object for resources.
resource_client = ResourceManagementClient(credential, subscription_id)

# Retrieve the list of resources in "myResourceGroup" (change to any name
# desired).
# The expand argument includes additional properties in the output.
resource_list = resource_client.resources.list_by_resource_group(
    resource_group, expand = "createdTime,changedTime")

# Show the groups in formatted output
column_width = 36

print("Resource".ljust(column_width) + "Type".ljust(column_width)
    + "Create date".ljust(column_width) + "Change date".ljust(column_width))
print("-" * (column_width * 4))

```

```
for resource in list(resource_list):
    print(f'{resource.name:<{column_width}'}{resource.type:<{column_width}'''
          f'{str(resource.created_time):<{column_width}'}{str(resource.changed_time):'
          '<{column_width}}')
```

Autenticación en el código

Más adelante en este artículo, inicia sesión en Azure con la CLI de Azure para ejecutar el código de ejemplo. Si la cuenta tiene permisos para crear y enumerar grupos de recursos en la suscripción de Azure, el código se ejecutará correctamente.

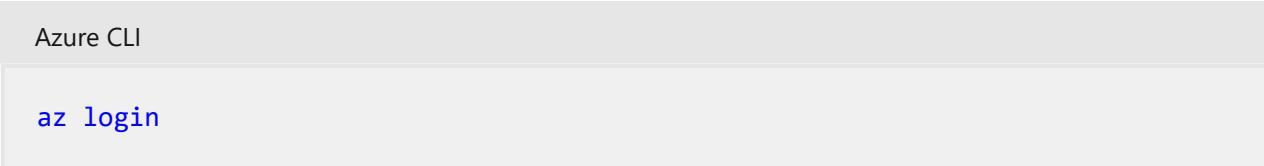
Para usar este código en un script de producción, puede establecer variables de entorno para usar un método basado en la entidad de servicio para la autenticación. Para más información, consulte [Autenticación de aplicaciones de Python con servicios de Azure](#). Debe asegurarse de que el principal de servicio tiene permisos suficientes para crear y enumerar grupos de recursos en tu suscripción mediante la asignación de un [rol adecuado en Azure](#); por ejemplo, el rol *Colaborador* en tu suscripción.

Vínculos de referencia para las clases usadas en el código

- [DefaultAzureCredential \(azure.identity\)](#)
- [ResourceManagementClient \(azure.mgmt.resource\)](#)

4: Ejecutar los scripts

1. Si aún no lo ha hecho, inicie sesión en Azure mediante la CLI de Azure:



Azure CLI

```
az login
```

2. Establezca la variable de entorno `AZURE_SUBSCRIPTION_ID` a su ID de suscripción. (Puede ejecutar el comando `az account show` y obtener el ID de suscripción de la propiedad `id` en la salida):



Bash

```
Bash
```

```
export AZURE_SUBSCRIPTION_ID=00000000-0000-0000-0000-000000000000
```

3. Enumerar todos los grupos de recursos de la suscripción:

Consola

```
python list_groups.py
```

4. Enumerar todos los recursos de un grupo de recursos:

Consola

```
python list_resources.py
```

De forma predeterminada, el código enumera los recursos en "myResourceGroup". Para usar otro grupo de recursos, establezca la `RESOURCE_GROUP_NAME` variable de entorno en el nombre del grupo deseado.

Como referencia: comandos equivalentes de la CLI de Azure

El siguiente comando de la CLI de Azure enumera los grupos de recursos de una suscripción:

Azure CLI

```
az group list
```

El siguiente comando enumera los recursos dentro de "myResourceGroup" en la región centralus (el `location` argumento es necesario para identificar un centro de datos específico):

Azure CLI

```
az resource list --resource-group myResourceGroup --location centralus
```

Consulte también

- [Ejemplo: Aprovisionamiento de un grupo de recursos](#)
- [Ejemplo: Aprovisionamiento de Azure Storage](#)
- [Ejemplo de uso de Azure Storage](#)
- [Ejemplo: Aprovisionamiento de una aplicación web e implementación de código](#)
- [Ejemplo: Aprovisionamiento y consulta de una base de datos](#)

- Ejemplo: Aprovisionamiento de una máquina virtual
- Uso de Azure Managed Disks con máquinas virtuales
- Complete una breve encuesta sobre el SDK de Azure para Python ↗

Ejemplo: Creación de Azure Storage mediante las bibliotecas de Azure para Python

30/05/2025

En este artículo, aprenderá a usar las bibliotecas de administración de Azure para Python para crear un grupo de recursos, junto con una cuenta de Azure Storage y un contenedor de Blob Storage.

Después de aprovisionar estos recursos, consulte la sección [Ejemplo: Uso de Azure Storage](#) para ver cómo usar las bibliotecas cliente de Azure en Python para cargar un archivo en el contenedor de blobs.

Los [comandos equivalentes](#) de la CLI de Azure para Bash y PowerShell se enumeran más adelante en este artículo. Si prefiere usar Azure Portal, consulte [Creación de una cuenta de Azure Storage](#) y [Creación de un contenedor de blobs](#).

1: Configuración del entorno de desarrollo local

Si aún no lo ha hecho, configure un entorno en el que pueda ejecutar el código. Estas son algunas opciones:

- Configure un entorno virtual de Python mediante `venv` o la herramienta que prefiera. Para empezar a usar el entorno virtual, asegúrese de activarlo. Para instalar Python, consulte [Instalación de Python](#).



```
Juerga

Azure CLI

#!/bin/bash
# Create a virtual environment
python -m venv .venv
# Activate the virtual environment
source .venv/Scripts/activate # only required for Windows (Git Bash)
```

- Use un [entorno de Conda](#). Para instalar Conda, consulte [Instalación de Miniconda](#).
- Usa un contenedor de desarrollo en Visual Studio Code o en GitHub Codespaces.

2: Instalación de los paquetes de biblioteca de Azure necesarios

1. En la consola, cree un archivo *requirements.txt* que muestre las bibliotecas de administración que se usan en este ejemplo:

```
Azure CLI
```

```
azure-mgmt-resource  
azure-mgmt-storage  
azure-identity
```

2. En la consola con el entorno virtual activado, instale los requisitos:

```
Consola
```

```
pip install -r requirements.txt
```

3. Establecer variables de entorno

En este paso, establecerá las variables de entorno para usarlas en el código de este artículo. El código usa el `os.environ` método para recuperar los valores.

```
Juerga
```

```
Azure CLI
```

```
#!/bin/bash  
export AZURE_RESOURCE_GROUP_NAME=<ResourceGroupName> # Change to your  
preferred resource group name  
export LOCATION=<Location> # Change to your preferred region  
export AZURE_SUBSCRIPTION_ID=$(az account show --query id --output tsv)  
export STORAGE_ACCOUNT_NAME=<StorageAccountName> # Change to your preferred  
storage account name  
export CONTAINER_NAME=<ContainerName> # Change to your preferred container  
name
```

4: Escribir código para crear una cuenta de almacenamiento y un contenedor de blobs

En este paso, creará un archivo de Python denominado *provision_blob.py* con el código siguiente. Este script de Python usa el SDK de Azure para bibliotecas de administración de Python para crear un grupo de recursos, una cuenta de Azure Storage y un contenedor de blobs mediante el SDK de Azure para Python.

Python

```
import os, random

# Import the needed management objects from the libraries. The azure.common
library
# is installed automatically with the other libraries.
from azure.identity import DefaultAzureCredential
from azure.mgmt.resource import ResourceManagementClient
from azure.mgmt.storage import StorageManagementClient
from azure.mgmt.storage.models import BlobContainer

# Acquire a credential object.
credential = DefaultAzureCredential()

# Retrieve subscription ID from environment variable.
subscription_id = os.environ["AZURE_SUBSCRIPTION_ID"]

# Retrieve resource group name and location from environment variables
RESOURCE_GROUP_NAME = os.environ["AZURE_RESOURCE_GROUP_NAME"]
LOCATION = os.environ["LOCATION"]

# Step 1: Provision the resource group.
resource_client = ResourceManagementClient(credential, subscription_id)

rg_result = resource_client.resource_groups.create_or_update(RESOURCE_GROUP_NAME,
    { "location": LOCATION })

print(f"Provisioned resource group {rg_result.name}")

# For details on the previous code, see Example: Provision a resource group
# at https://docs.microsoft.com/azure/developer/python/azure-sdk-example-resource-
group

# Step 2: Provision the storage account, starting with a management object.

storage_client = StorageManagementClient(credential, subscription_id)

STORAGE_ACCOUNT_NAME = os.environ["STORAGE_ACCOUNT_NAME"]

# Check if the account name is available. Storage account names must be unique
across
# Azure because they're used in URLs.
availability_result = storage_client.storage_accounts.check_name_availability(
    { "name": STORAGE_ACCOUNT_NAME }
)
```

```

if not availability_result.name_available:
    print(f"Storage name {STORAGE_ACCOUNT_NAME} is already in use. Try another
name.")
    exit()

# The name is available, so provision the account
poller = storage_client.storage_accounts.begin_create(RESOURCE_GROUP_NAME,
STORAGE_ACCOUNT_NAME,
{
    "location" : LOCATION,
    "kind": "StorageV2",
    "sku": {"name": "Standard_LRS"}
}
)

# Long-running operations return a poller object; calling poller.result()
# waits for completion.
account_result = poller.result()
print(f"Provisioned storage account {account_result.name}")

# Step 3: Retrieve the account's primary access key and generate a connection
# string.
keys = storage_client.storage_accounts.list_keys(RESOURCE_GROUP_NAME,
STORAGE_ACCOUNT_NAME)

print(f"Primary key for storage account: {keys.keys[0].value}")

conn_string =
f"DefaultEndpointsProtocol=https;EndpointSuffix=core.windows.net;AccountName=
{STORAGE_ACCOUNT_NAME};AccountKey={keys.keys[0].value}"

# print(f"Connection string: {conn_string}")

# Step 4: Provision the blob container in the account (this call is synchronous)
CONTAINER_NAME = os.environ["CONTAINER_NAME"]
container = storage_client.blob_containers.create(RESOURCE_GROUP_NAME,
STORAGE_ACCOUNT_NAME, CONTAINER_NAME, BlobContainer())

print(f"Provisioned blob container {container.name}")

```

Autenticación en el código

Más adelante en este artículo, inicia sesión en Azure mediante la CLI de Azure para ejecutar el código de ejemplo. Si la cuenta tiene permisos suficientes para crear grupos de recursos y recursos de almacenamiento en la suscripción de Azure, el script debe ejecutarse correctamente sin configuración adicional.

Para usar este código en un entorno de producción, auténtíquese mediante una entidad de servicio estableciendo variables de entorno. Este enfoque permite el acceso seguro y

automatizado sin depender del inicio de sesión interactivo. Para obtener instrucciones detalladas, consulte [Autenticación de aplicaciones de Python con servicios de Azure](#).

Asegúrese de que a la entidad de servicio se le asigna un rol con permisos suficientes para crear grupos de recursos y cuentas de almacenamiento. Por ejemplo, asignar el rol Colaborador en el nivel de suscripción proporciona el acceso necesario. Para más información sobre las asignaciones de roles, consulte [Control de acceso basado en rol \(RBAC\) en Azure](#).

Vínculos de referencia para las clases usadas en el código

- [DefaultAzureCredential \(azure.identity\)](#)
- [ResourceManagementClient \(azure.mgmt.resource\)](#)
- [StorageManagementClient \(azure.mgmt.storage\)](#)

5. Ejecutar el script

1. Si aún no lo ha hecho, inicie sesión en Azure mediante la CLI de Azure:

```
Azure CLI
```

```
az login
```

2. Ejecuta el script:

```
Consola
```

```
python provision_blob.py
```

El script tarda un minuto o dos en completarse.

6: Comprobación de los recursos

1. Abra [Azure Portal](#) para comprobar que el grupo de recursos y la cuenta de almacenamiento se crearon según lo previsto. Es posible que tenga que esperar un minuto y seleccionar También **Mostrar tipos ocultos** en el grupo de recursos.

The screenshot shows the Azure portal interface for a resource group named 'PythonAzureExample-Storage-rg'. On the left, there's a navigation menu with options like Overview, Activity log, Access control (IAM), Tags, Events, Settings, Deployments, Security, Policies, Properties, and Locks. The main content area is titled 'Essentials' and displays information about the subscription (Primary), including the Subscription ID and Location (Central US). It also shows tags and deployment details. A table lists one record, with a checkbox labeled 'Show hidden types' highlighted with a red box. The table includes columns for Name, Type, and Storage account, with 'blob-container-01' listed under Name and 'Storage account' under Type.

2. Seleccione la cuenta de almacenamiento y, a continuación, seleccione **Contenedores**> datos en el menú izquierdo para comprobar que aparece "blob-container-01":

The screenshot shows the Azure Storage account 'pythonazurorestorage99737'. The left sidebar has a 'Data storage' section with 'Containers' selected, which is highlighted with a red box. The main pane lists 'Containers' with a search bar at the top. One container, 'blob-container-01', is listed and highlighted with a red box. Other options like 'File shares', 'Queues', and 'Tables' are also visible in the sidebar.

3. Si quiere intentar usar estos recursos desde el código de aplicación, continúe con [Ejemplo: Uso de Azure Storage](#).

Para obtener otro ejemplo de uso de la biblioteca de administración de Azure Storage, consulte el [ejemplo Administración del almacenamiento de Python](#).

7: Limpieza de recursos

Deje los recursos en su lugar si desea seguir el artículo [Ejemplo: Uso de Azure Storage](#) para usar estos recursos en el código de la aplicación. De lo contrario, ejecute el comando `az group delete` si no necesita mantener el grupo de recursos y los recursos de almacenamiento creados en este ejemplo.

Los grupos de recursos no incurren en cargos continuos en la suscripción, pero los recursos, como las cuentas de almacenamiento, en el grupo de recursos pueden incurrir en cargos. Se recomienda limpiar cualquier grupo que no se esté usando activamente. El argumento `--no-wait` permite que el comando devuelva inmediatamente en lugar de esperar a que finalice la operación.

Juerga

```
Azure CLI
#!/bin/bash
az group delete -n $AZURE_RESOURCE_GROUP_NAME --no-wait
```

Como referencia: comandos equivalentes de la CLI de Azure

Los siguientes comandos de la CLI de Azure completan los mismos pasos de creación que el script de Python:

Juerga

```
Azure CLI
#!/bin/bash
#!/bin/bash

# Set variables
export LOCATION=<Location> # Change to your preferred region
export AZURE_RESOURCE_GROUP_NAME=<ResourceGroupName> # Change to your
preferred resource group name
export STORAGE_ACCOUNT_NAME=<StorageAccountName> # Change to your preferred
storage account name
export CONTAINER_NAME=<ContainerName> # Change to your preferred container
name

# Provision the resource group
echo "Creating resource group: $AZURE_RESOURCE_GROUP_NAME"
az group create \
    --location "$LOCATION" \
    --name "$AZURE_RESOURCE_GROUP_NAME"

# Provision the storage account
az storage account create -g $AZURE_RESOURCE_GROUP_NAME -l $LOCATION -n
$STORAGE_ACCOUNT_NAME --kind StorageV2 --sku Standard_LRS

echo Storage account name is $STORAGE_ACCOUNT_NAME
```

```
# Retrieve the connection string
CONNECTION_STRING=$(az storage account show--connection-string -g
$AZURE_RESOURCE_GROUP_NAME -n $STORAGE_ACCOUNT_NAME --query connectionString)

# Provision the blob container
az storage container create --name $CONTAINER_NAME --account-name
$STORAGE_ACCOUNT_NAME --connection-string $CONNECTION_STRING
```

Consulte también

- [Ejemplo: Uso de Azure Storage](#)
- [ejemplo de : creación de un grupo de recursos](#)
- [ejemplo: Enumeración de grupos de recursos en una suscripción](#)
- [Ejemplo: Creación de una aplicación web e implementación de código](#)
- [Ejemplo: Creación y consulta de una base de datos](#)
- [ejemplo de : creación de una máquina virtual](#)
- [Uso de Azure Managed Disks con máquinas virtuales](#)
- [Completar una breve encuesta sobre el SDK de Azure para Python↗](#)

Ejemplo: Acceso a Azure Storage con las bibliotecas de Azure para Python

Artículo • 06/10/2024

En este artículo, aprenderá a usar las bibliotecas cliente de Azure SDK en el código de la aplicación de Python para cargar un archivo en un contenedor de Azure Blob Storage.

En el artículo se da por supuesto que ha creado los recursos que se muestran en [Ejemplo: Creación de Azure Storage](#).

Todos los comandos de este artículo funcionan igual en el bash de Linux o macOS y en los shells de comandos de Windows, a menos que se indique lo contrario.

1. Configuración del entorno de desarrollo local

Si aún no lo ha hecho, configure un entorno en el que pueda ejecutar este código. Estas son algunas opciones:

- Configure un entorno virtual de Python mediante `venv` o la herramienta que prefiera. Puede crear el entorno virtual localmente o en [Azure Cloud Shell](#) y ejecutar el código allí. Asegúrese de activar el entorno virtual para empezar a usarlo.
- Use un [entorno de conda](#).
- Use un [contenedor de desarrollo](#) en [Visual Studio Code](#) o [GitHub Codespaces](#).

2. Instalación de paquetes de biblioteca

En el archivo `requirements.txt`, agregue líneas para los paquetes de biblioteca cliente que necesita y guarde el archivo.

```
txt  
  
azure-storage-blob  
azure-identity
```

Luego, en el terminal o el símbolo del sistema, instale los requisitos.

```
Consola
```

```
pip install -r requirements.txt
```

3. Creación de un archivo para cargar

Cree un archivo de origen denominado *sample-source.txt*. Este nombre de archivo es lo que el código espera.

txt

Hello there, Azure Storage. I'm a friendly file ready to be stored in a blob.

4. Uso de Blob Storage en el código de la aplicación

Esta sección demuestra dos formas de acceder a los datos en el contenedor blob que creaste en [Ejemplo: Creación de Azure Storage](#). Para acceder a los datos del contenedor de blobs, su aplicación debe ser capaz de autenticarse con Azure y estar autorizada para acceder a los datos del contenedor. Esta sección presenta dos formas de hacerlo:

- El método **sin contraseña (recomendado)** autentica la app usando [DefaultAzureCredential](#). `DefaultAzureCredential` es una credencial encadenada que puede autenticar una aplicación (o un usuario) usando una secuencia de credenciales diferentes, incluidas las credenciales de la herramienta de desarrollo, las principales del servicio de aplicaciones y las identidades gestionadas.
- El método de **Cadena de conexión** usa una cadena de conexión para acceder directamente a la cuenta de almacenamiento.

Por las siguientes razones, entre otras, recomendamos usar el método sin contraseña siempre que sea posible:

- Una cadena de conexión autentica al agente que se conecta con la *cuenta* de almacenamiento en lugar de con recursos individuales dentro de esa cuenta. En consecuencia, una cadena de conexión otorga una autorización más amplia de la que podría ser necesaria. Con `DefaultAzureCredential` puede conceder permisos más granulares y menos privilegiados sobre sus recursos de almacenamiento a la identidad bajo la que se ejecuta su aplicación usando [Azure RBAC](#).

- Una cadena de conexión contiene información de acceso en texto sin formato y, por lo tanto, presenta posibles vulnerabilidades si no está correctamente construida o protegida. Si dicha cadena de conexión quedara expuesta, podría usarse para acceder a una amplia variedad de recursos dentro de la cuenta de almacenamiento.
- Una cadena de conexión normalmente se almacena en una variable de entorno, lo que la hace vulnerable si un atacante obtiene acceso a su entorno. Muchos de los tipos de credenciales compatibles con `DefaultAzureCredential` no requieren almacenar secretos en su entorno.

Sin contraseña (recomendado)

`DefaultAzureCredential` es una cadena de credenciales preconfigurada. Está diseñada para admitir muchos entornos, junto con los flujos de autenticación y las herramientas de desarrollo más comunes. Una instancia de `DefaultAzureCredential` determina para qué tipos de credenciales intentar obtener un token basándose en una combinación de su entorno en tiempo de ejecución, el valor de ciertas variables de entorno bien conocidas y, opcionalmente, los parámetros pasados a su constructor.

En los siguientes pasos, se configura una entidad de seguridad de servicio de aplicación como identidad de la aplicación. Las entidades de seguridad del servicio de aplicaciones son adecuadas para su uso tanto durante el desarrollo local como para aplicaciones alojadas en las instalaciones. Para configurar

`DefaultAzureCredential` para usar la entidad de seguridad del servicio de aplicaciones, debe establecer las siguientes variables de entorno: `AZURE_CLIENT_ID`, `AZURE_TENANT_ID` y `AZURE_CLIENT_SECRET`.

Observe que se ha configurado un secreto de cliente. Esto es necesario para una entidad de seguridad de servicio de aplicación, pero, dependiendo de su escenario, también puede configurar `DefaultAzureCredential` para usar de credenciales que no requieran establecer un secreto o contraseña en una variable de entorno.

Por ejemplo, en el desarrollo local, si `DefaultAzureCredential` no puede obtener un token usando variables de entorno configuradas, se intenta obtener uno usando el usuario (ya) registrado en herramientas de desarrollo como Azure CLI; para una aplicación alojada en Azure, se puede configurar `DefaultAzureCredential` para usar una identidad administrada. En todos los casos, el código de su aplicación sigue siendo el mismo, solo cambia la configuración y/o el entorno de ejecución.

1. Cree un archivo llamado `use_blob_auth.py` con el código siguiente. Los pasos se explican en los comentarios.

```
Python

import os
import uuid

from azure.identity import DefaultAzureCredential

# Import the client object from the SDK library
from azure.storage.blob import BlobClient

credential = DefaultAzureCredential()

# Retrieve the storage blob service URL, which is of the form
# https://<your-storage-account-name>.blob.core.windows.net/
storage_url = os.environ["AZURE_STORAGE_BLOB_URL"]

# Create the client object using the storage URL and the credential
blob_client = BlobClient(
    storage_url,
    container_name="blob-container-01",
    blob_name=f"sample-blob-{str(uuid.uuid4())[0:5]}.txt",
    credential=credential,
)

# Open a local file and upload its contents to Blob Storage
with open("./sample-source.txt", "rb") as data:
    blob_client.upload_blob(data)
    print(f"Uploaded sample-source.txt to {blob_client.url}")
```

Vínculos de referencia:

- `DefaultAzureCredential` (`azure.identity`)
- `BlobClient` (`azure.storage.blob`)

2. Cree una variable de entorno llamada `AZURE_STORAGE_BLOB_URL`:

cmd

Símbolo del sistema de Windows

```
set
AZURE_STORAGE_BLOB_URL=https://pythonazurestorage12345.blob.cor
e.windows.net
```

Reemplace "pythonazurestorage12345" por el nombre de su cuenta de almacenamiento.

En este ejemplo solo se usa la variable de entorno `AZURE_STORAGE_BLOB_URL`. Las bibliotecas de Azure no la usan.

3. Use el comando `az ad sp create-for-rbac` para crear una entidad de servicio nueva para la aplicación. El comando crea el registro de aplicación para la aplicación al mismo tiempo. Asigne a la entidad de servicio un nombre de su elección.

Azure CLI

```
az ad sp create-for-rbac --name <service-principal-name>
```

El resultado de este comando tendrá un aspecto similar al siguiente. Anote estos valores o mantenga esta ventana abierta, ya que necesitará estos valores en el paso siguiente y no podrá volver a ver el valor de la contraseña (secreto del cliente). Sin embargo, si es necesario, se puede agregar una nueva contraseña posteriormente sin invalidar la entidad de servicio ni las contraseñas existentes.

JSON

```
{  
    "appId": "00001111-aaaa-2222-bbbb-3333cccc4444",  
    "displayName": "<service-principal-name>",  
    "password": "Aa1Bb~2Cc3.-Dd4Ee5Ff6Gg7Hh8Ii9_Jj0Kk1Ll2",  
    "tenant": "aaaabbbb-0000-cccc-1111-dddd2222eeee"  
}
```

Los comandos de la CLI de Azure se pueden ejecutar en [Azure Cloud Shell](#) o en una estación de trabajo que tenga la [CLI de Azure instalada](#).

4. Cree variables de entorno para la entidad de servicio de la aplicación:

Cree las siguientes variables de entorno con los valores de la salida del comando anterior. Estas variables indican a `DefaultAzureCredential` que debe usar la entidad de servicio de la aplicación.

- `AZURE_CLIENT_ID` → Valor del id. de la aplicación.
- `AZURE_TENANT_ID` → Valor del id. de inquilino.
- `AZURE_CLIENT_SECRET` → Contraseña o credencial generada para la aplicación.

cmd

Símbolo del sistema de Windows

```
set AZURE_CLIENT_ID=00001111-aaaa-2222-bbbb-3333cccc4444
set AZURE_TENANT_ID=aaaabbbb-0000-cccc-1111-dddd2222eeee
set AZURE_CLIENT_SECRET=Aa1Bb~2Cc3.-.
Dd4Ee5Ff6Gg7Hh8Ii9_Jj0Kk1Ll2
```

5. Intente ejecutar el código (que produce un error de forma intencionada):

Consola

```
python use_blob_auth.py
```

6. Observe el error "Esta solicitud no está autorizada para realizar esta operación mediante este permiso". Este error es esperado, porque la entidad de servicio local que se está usando aún no tiene permiso para acceder al contenedor de blobs.

7. Conceda permisos de [Colaborador de Storage Blob Data](#) en el contenedor blob al service principal usando el comando [az role assignment create](#) de Azure CLI:

Azure CLI

```
az role assignment create --assignee <AZURE_CLIENT_ID> \
    --role "Storage Blob Data Contributor" \
    --scope
"/subscriptions/<AZURE_SUBSCRIPTION_ID>/resourceGroups/PythonAzureE
xample-Storage-
rg/providers/Microsoft.Storage/storageAccounts/pythonazurestorage12
345/blobServices/default/containers/blob-container-01"
```

El argumento `--assignee` identifica la entidad de servicio. Reemplace el marcador de posición `<AZURE_CLIENT_ID>` por el identificador de aplicación de la entidad de servicio.

El argumento `--scope` identifica dónde se aplica esta asignación de roles. En este ejemplo, concederá el rol "Colaborador de datos de blobs de almacenamiento" a la entidad de servicio para el contenedor específico denominado "blob-container-01".

- Reemplace `PythonAzureExample-Storage-rg` y `pythonazurorestorage12345` por el grupo de recursos que contiene la cuenta de almacenamiento y el nombre exacto de la cuenta de almacenamiento. Además, ajuste el nombre del contenedor de blobs, si es necesario. Si usa un nombre incorrecto, aparecerá el error "No se puede realizar la operación solicitada en el recurso anidado. No se encontró el recurso primario 'pythonazurorestorage12345'".
- Reemplace el marcador de posición `<AZURE_SUBSCRIPTION_ID>` por su identificador de suscripción de Azure. (Puede ejecutar el comando `az account show` y obtener el identificador de suscripción de la propiedad `id` en la salida).

Sugerencia

Si el comando de asignación de roles devuelve un error "No se encontraron adaptadores de conexión" al usar el shell de Bash, intente establecer `export MSYS_NO_PATHCONV=1` para evitar la traducción de rutas de acceso. Para más información, consulta [esta propuesta](#).

8. **Espere un minuto o dos para que los permisos se propaguen** y, después, vuelva a ejecutar el código para comprobar que ahora funciona. Si vuelve a ver el error de permisos, espere un poco más y vuelva a probar el código.

Para más información sobre las asignaciones de roles, consulte [Incorporación o eliminación de asignaciones de roles de Azure mediante la CLI de Azure](#).

Importante

En los pasos anteriores, su aplicación se ejecutó bajo una entidad de seguridad de servicio de aplicación. Una entidad de seguridad de servicio de aplicación requiere un secreto de cliente en su configuración. Sin embargo, puede usar el mismo código para ejecutar la aplicación bajo diferentes tipos de credenciales que no requieren que configure explícitamente una contraseña o secreto en el entorno. Por ejemplo, durante el desarrollo, `DefaultAzureCredential` puede usar credenciales de la herramienta para desarrolladores como las credenciales que usa para iniciar sesión a través de la CLI de Azure; o bien, para las aplicaciones alojadas en Azure, puede usar una [identidad administrada](#). Para obtener más información, consulte [Autenticar aplicaciones Python en servicios Azure mediante el SDK de Azure para Python](#).

5. Comprobación de la creación de blobs

Después de ejecutar el código de cualquiera de los métodos, vaya a [Azure Portal](#), vaya al contenedor de blobs para comprobar que existe un nuevo blob llamado `sample-blob-{random}.txt` con el mismo contenido que el archivo `sample-source.txt`:

The screenshot shows the Azure Storage Blob Container Overview page for 'blob-container-01'. On the left, there's a sidebar with options like Overview, Diagnose and solve problems, Access Control (IAM), Settings, Shared access tokens, Access policy, Properties, Metadata, and Editor (preview). The 'Overview' tab is selected. At the top right, there are buttons for Upload, Change access level, Refresh, Delete, and Change. Below the top bar, it says 'Authentication method: Access key (Switch to Microsoft Entra user account)' and 'Location: blob-container-01'. A search bar says 'Search blobs by prefix (case-sensitive)'. The main area shows a table with a single row: 'Name' followed by a checkbox and a file icon next to 'sample-blob.txt'. This row is highlighted with a red box.

Si creó una variable de entorno denominada `AZURE_STORAGE_CONNECTION_STRING`, también Blob Storage puede usar la CLI de Azure para comprobar que el blob existe mediante el comando [az storage blob list](#):

```
Azure CLI
az storage blob list --container-name blob-container-01
```

Si ha seguido las instrucciones para usar la autenticación sin contraseña, puede agregar el parámetro `--connection-string` al comando anterior con la cadena de conexión de la cuenta de almacenamiento. Para obtener la cadena de conexión, use el comando [az storage account show-connection-string](#).

```
Azure CLI
az storage account show-connection-string --resource-group PythonAzureExample-Storage-rg --name pythonazurestorage12345 --output tsv
```

Use la cadena de conexión completa como valor para el parámetro `--connection-string`.

⚠️ Nota

Si su cuenta de usuario Azure tiene el rol "Storage Blob Data Contributor" en el contenedor, puede usar el siguiente comando para listar los blobs en el contenedor:

Azure CLI

```
az storage blob list --container-name blob-container-01 --account-name pythonazurestorage12345 --auth-mode login
```

6. Limpieza de recursos

Ejecute el comando [az group delete](#) si no necesita conservar el grupo de recursos y los recursos de almacenamiento usados en este ejemplo. Los grupos de recursos no incurren en cargos continuos en la suscripción, pero los recursos, como las cuentas de almacenamiento en el grupo de recursos pueden seguir incurriendo en cargos. Es recomendable limpiar cualquier grupo que no esté usando activamente. El argumento `-no-wait` permite que el comando devuelva el control inmediatamente en lugar de esperar a que finalice la operación.

Azure CLI

```
az group delete -n PythonAzureExample-Storage-rg --no-wait
```

También puede usar el método

[ResourceManagementClient.resource_groups.begin_delete](#) para eliminar un grupo de recursos del código. El código de [Ejemplo: Creación de un grupo de recursos](#) muestra el uso.

Si ha seguido las instrucciones para usar la autenticación sin contraseña, es recomendable eliminar la entidad de servicio de la aplicación que creó. Puede usar el comando [az ad app delete](#). Reemplace el marcador de posición <AZURE_CLIENT_ID> por el identificador de aplicación de la entidad de servicio.

Azure CLI

```
az ad app delete --id <AZURE_CLIENT_ID>
```

Consulte también

- [Inicio rápido: Biblioteca cliente de Azure Blob Storage para Python](#)
 - [Ejemplo: Creación de un grupo de recursos](#)
 - [Ejemplo: Enumeración de los grupos de recursos de una suscripción](#)
 - [Ejemplo: Creación de una aplicación web e implementación de código](#)
 - [Ejemplo: Creación de Azure Storage](#)
 - [Ejemplo: Creación y consulta de una base de datos](#)
 - [Ejemplo: Creación de una máquina virtual](#)
 - [Uso de Azure Managed Disks con máquinas virtuales](#)
 - [Realización de una breve encuesta sobre el SDK de Azure para Python ↗](#)
-

Comentarios

¿Le ha resultado útil esta página?



Sí



No

[Proporcionar comentarios sobre el producto ↗](#) | [Obtener ayuda en Microsoft Q&A](#)

Ejemplo: Uso de las bibliotecas de Azure para crear e implementar una aplicación web

21/05/2025

En este ejemplo se muestra cómo usar las bibliotecas de administración del SDK de Azure en un script de Python para crear e implementar una aplicación web en Azure App Service, con el código de la aplicación extraído de un repositorio de GitHub.

El SDK de Azure para Python incluye bibliotecas de administración (espacios de nombres que comienzan con `azure-mgmt`) que permiten automatizar la configuración y la implementación de recursos, de forma similar a lo que puede hacer con el portal de Azure, la CLI de Azure o las plantillas de ARM. Para obtener ejemplos, consulte [Inicio rápido: Implementación de una aplicación web de Python \(Django o Flask\) en Azure App Service](#).

1: Configuración del entorno de desarrollo local

Si aún no lo ha hecho, configure un entorno en el que pueda ejecutar este código. Estas son algunas opciones:

- Configure un entorno virtual de Python mediante `venv` o la herramienta que prefiera. Para empezar a usar el entorno virtual, asegúrese de activarlo. Para instalar Python, consulte [Instalación de Python](#).



Azure CLI

```
#!/bin/bash
# Create a virtual environment
python -m venv .venv
# Activate the virtual environment
source .venv/Scripts/activate # only required for Windows (Git Bash)
```

- Usa un entorno conda. Para instalar Conda, consulte [Instalación de Miniconda](#).
- Usa un contenedor de desarrollo en Visual Studio Code o en GitHub Codespaces.

2: Instalación de los paquetes de biblioteca de Azure necesarios

Cree un archivo denominado *requirements.txt* con el siguiente contenido:

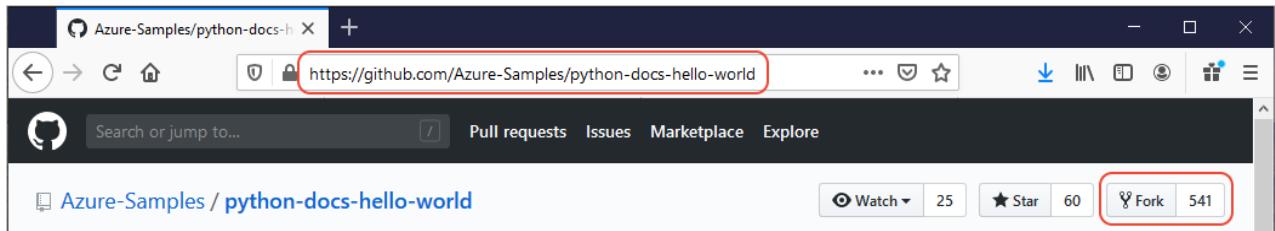
```
txt  
  
azure-mgmt-resource  
azure-mgmt-web  
azure-identity
```

En el entorno de desarrollo local, instale los requisitos mediante el código siguiente:

```
Consola  
  
pip install -r requirements.txt
```

3: Bifurcar el repositorio de ejemplo

1. Visite <https://github.com/Azure-Samples/python-docs-hello-world> y haga un fork del repositorio en su cuenta de GitHub. El uso de una bifurcación garantiza que dispone de los permisos necesarios para implementar la aplicación en Azure.



2. A continuación, cree una variable de entorno denominada `REPO_URL` y establezcala en la dirección URL del repositorio bifurcada. El código de ejemplo de la sección siguiente requiere esta variable.

```
bash  
  
Bash  
  
export REPO_URL=<url_of_your_fork>  
export AZURE_SUBSCRIPTION_ID=<subscription_id>
```

4: Escribir código para crear e implementar una aplicación web

Cree un archivo de Python denominado `provision_deploy_web_app.py` y agregue el código siguiente. Los comentarios en línea explican lo que hace cada parte del script. Las `REPO_URL` variables de entorno y `AZURE_SUBSCRIPTION_ID` ya deben establecerse en el paso anterior.

Python

```
import random, os
from azure.identity import AzureCliCredential
from azure.mgmt.resource import ResourceManagementClient
from azure.mgmt.web import WebSiteManagementClient

# Acquire a credential object using CLI-based authentication.
credential = AzureCliCredential()

# Retrieve subscription ID from environment variable
subscription_id = os.environ["AZURE_SUBSCRIPTION_ID"]

# Constants we need in multiple places: the resource group name and the region
# in which we provision resources. You can change these values however you want.
RESOURCE_GROUP_NAME = 'PythonAzureExample-WebApp-rg'
LOCATION = "centralus"

# Step 1: Provision the resource group.
resource_client = ResourceManagementClient(credential, subscription_id)

rg_result = resource_client.resource_groups.create_or_update(RESOURCE_GROUP_NAME,
    { "location": LOCATION })

print(f"Provisioned resource group {rg_result.name}")

# For details on the previous code, see Example: Provision a resource group
# at https://docs.microsoft.com/azure/developer/python/azure-sdk-example-resource-group

#Step 2: Provision the App Service plan, which defines the underlying VM for the
# web app.

# Names for the App Service plan and App Service. We use a random number with the
# latter to create a reasonably unique name. If you've already provisioned a
# web app and need to re-run the script, set the WEB_APP_NAME environment
# variable to that name instead.
SERVICE_PLAN_NAME = 'PythonAzureExample-WebApp-plan'
WEB_APP_NAME = os.environ.get("WEB_APP_NAME", f"PythonAzureExample-WebApp-{random.randint(1,100000):05}")

# Obtain the client object
app_service_client = WebSiteManagementClient(credential, subscription_id)
```

```

# Provision the plan; Linux is the default
poller =
app_service_client.app_service_plans.begin_create_or_update(RESOURCE_GROUP_NAME,
    SERVICE_PLAN_NAME,
{
    "location": LOCATION,
    "reserved": True,
    "sku" : {"name" : "B1"}
}
)

plan_result = poller.result()

print(f"Provisioned App Service plan {plan_result.name}")

# Step 3: With the plan in place, provision the web app itself, which is the
process that can host
# whatever code we want to deploy to it.

poller = app_service_client.web_apps.begin_create_or_update(RESOURCE_GROUP_NAME,
    WEB_APP_NAME,
{
    "location": LOCATION,
    "server_farm_id": plan_result.id,
    "site_config": {
        "linux_fx_version": "python|3.8"
    }
}
)

web_app_result = poller.result()

print(f"Provisioned web app {web_app_result.name} at
{web_app_result.default_host_name}")

# Step 4: deploy code from a GitHub repository. For Python code, App Service on
Linux runs
# the code inside a container that makes certain assumptions about the structure
of the code.
# For more information, see How to configure Python apps,
# https://docs.microsoft.com/azure/app-service/containers/how-to-configure-python.
#
# The create_or_update_source_control method doesn't provision a web app. It only
sets the
# source control configuration for the app. In this case we're simply pointing to
# a GitHub repository.
#
# You can call this method again to change the repo.

REPO_URL = os.environ["REPO_URL"]

poller =
app_service_client.web_apps.begin_create_or_update_source_control(RESOURCE_GROUP_N

```

```

AME,
    WEB_APP_NAME,
    {
        "location": "GitHub",
        "repo_url": REPO_URL,
        "branch": "master",
        "is_manual_integration": True
    }
)

sc_result = poller.result()

print(f"Set source control on web app to {sc_result.branch} branch of
{sc_result.repo_url}")

# Step 5: Deploy the code using the repository and branch configured in the
previous step.
#
# If you push subsequent code changes to the repo and branch, you must call this
method again
# or use another Azure tool like the Azure CLI or Azure portal to redeploy.
# Note: By default, the method returns None.

app_service_client.web_apps.sync_repository(RESOURCE_GROUP_NAME, WEB_APP_NAME)

print("Deploy code")

```

Este código usa la autenticación basada en la CLI (mediante `AzureCliCredential`) porque muestra las acciones que podría hacer directamente con la CLI de Azure. En ambos casos, usa la misma identidad para la autenticación. En función de su entorno, es posible que tenga que ejecutar `az login` primero para autenticarse.

Para usar este código en un script de producción (por ejemplo, para automatizar la administración de máquinas virtuales), use `DefaultAzureCredential` (recomendado) con un método basado en una entidad de servicio, tal como se describe en [Autenticación de aplicaciones de Python con servicios de Azure](#).

Vínculos de referencia para las clases usadas en el código

- [AzureCliCredential \(azure.identity\)](#)
- [ResourceManagementClient \(azure.mgmt.resource\)](#)
- [WebSiteManagementClient \(azure.mgmt.web import\)](#)

5: Ejecutar el script

Consola

```
python provision_deploy_web_app.py
```

6: Comprobación de la implementación de la aplicación web

Para ver el sitio web implementado, ejecute el siguiente comando:

Azure CLI

```
az webapp browse --name <PythonAzureExample-WebApp-12345> --resource-group  
PythonAzureExample-WebApp-rg
```

Reemplace el nombre de la aplicación web (`--name`) por el valor generado por el script. No es necesario cambiar el nombre del grupo de recursos (`--resource-group`) a menos que lo haya cambiado en el script. Al abrir el sitio, debería ver "Hello, World!" en el explorador.

Sugerencia

Si no ve la salida esperada, espere unos minutos e inténtelo de nuevo.

Si sigue sin ver el resultado esperado:

1. Vaya a [Azure Portal](#).
2. Vaya a **Grupos de recursos** y busque el grupo de recursos que creó.
3. Seleccione el grupo de recursos para ver sus recursos. Asegúrese de que incluye tanto un Plan de Servicio de Aplicaciones como un Servicio de Aplicaciones.
4. Seleccione **App Service** y, a continuación, vaya al **Centro de implementación**.
5. Abra la pestaña **registros** para comprobar los registros de implementación en busca de errores o actualizaciones de estado.

7: Volver a implementar el código de la aplicación web (opcional)

El script provisoria todos los recursos necesarios para alojar tu aplicación web y configura el origen de implementación para usar tu repositorio bifurcado mediante integración manual. Con la integración manual, debe iniciar manualmente la aplicación web para obtener actualizaciones del repositorio y la rama especificados.

El script usa el método `WebSiteManagementClient.web_apps.sync_repository` para desencadenar la aplicación web para extraer código del repositorio. Si realiza más cambios en el código, puede volver a implementar mediante una llamada a esta API de nuevo o mediante otras herramientas de Azure, como la CLI de Azure o Azure Portal.

Para volver a implementar el código mediante la CLI de Azure, ejecute el comando `az webapp deployment source sync`:

Azure CLI

```
az webapp deployment source sync --name <PythonAzureExample-WebApp-12345> --resource-group PythonAzureExample-WebApp-rg
```

No es necesario cambiar el nombre del grupo de recursos (`--resource-group`) a menos que lo haya cambiado en el script.

Para implementar el código desde Azure Portal:

1. Vaya a [Azure Portal](#).
2. Vaya a **Grupos de recursos** y busque el grupo de recursos que creó.
3. Seleccione el nombre del grupo de recursos para ver sus recursos. Asegúrese de que incluye tanto un Plan de Servicio de Aplicaciones como un Servicio de Aplicaciones.
4. Seleccione **App Service** y, a continuación, vaya al **Centro de implementación**.
5. En el menú superior, seleccione **Sincronizar** para desencadenar la implementación del código.

8: Limpieza de recursos

Azure CLI

```
az group delete --name PythonAzureExample-WebApp-rg --no-wait
```

No es necesario cambiar el nombre del grupo de recursos (`--resource-group` opción) a menos que lo haya cambiado en el script.

Si ya no necesita el grupo de recursos creado en este ejemplo, puede eliminarlo ejecutando el comando `az group delete`. Aunque los grupos de recursos no incurren en cargos continuos, se recomienda limpiar los recursos no utilizados. Use el `--no-wait` argumento para devolver inmediatamente el control a la línea de comandos sin esperar a que se complete la eliminación.

También puede eliminar un grupo de recursos mediante programación mediante el `ResourceManagementClient.resource_groups.begin_delete` método .

Consulte también

- [Ejemplo: Creación de un grupo de recursos](#)
- [Ejemplo: Enumeración de grupos de recursos en una suscripción](#)
- [Ejemplo: Creación de Azure Storage](#)
- [Ejemplo de uso de Azure Storage](#)
- [Ejemplo: Creación y consulta de una base de datos MySQL](#)
- [Ejemplo: Creación de una máquina virtual](#)
- [Uso de Azure Managed Disks con máquinas virtuales](#)
- [Complete una breve encuesta sobre el SDK de Azure para Python ↗](#)

Ejemplo: Uso de las bibliotecas de Azure para crear una base de datos

28/05/2025

En este ejemplo se muestra cómo usar las bibliotecas de administración de Azure SDK para Python para crear mediante programación un servidor flexible de Azure Database for MySQL y una base de datos correspondiente. También incluye un script básico que usa la biblioteca mysql-connector-python (no forma parte del SDK de Azure) para conectarse a la base de datos y consultarla.

Puede adaptar este ejemplo para crear un servidor flexible de Azure Database for PostgreSQL modificando las importaciones de SDK pertinentes y las llamadas API.

Si prefiere usar la CLI de Azure, los [comandos equivalentes](#) de la CLI de Azure se proporcionan más adelante en este artículo. Para obtener una experiencia gráfica, consulte la documentación de Azure Portal:

- [Creación de un servidor MySQL](#)
- [Creación de un servidor PostgreSQL](#)

A menos que se especifique lo contrario, todos los ejemplos y comandos funcionan de forma coherente en los shells de comandos de Linux/macOS y Windows.

1: Configuración del entorno de desarrollo local

Si aún no lo ha hecho, configure un entorno en el que pueda ejecutar el código. Estas son algunas opciones:

- Configure un entorno virtual de Python mediante `venv` o la herramienta que prefiera. Para empezar a usar el entorno virtual, asegúrese de activarlo. Para instalar Python, consulte [Instalación de Python](#).



The screenshot shows a terminal window with a light gray background. At the top, there is a blue header bar with the text "Juerga". Below the header, the terminal window has a title bar labeled "Azure CLI". The main area of the terminal contains the following text:

```
#!/bin/bash
# Create a virtual environment
python -m venv .venv
# Activate the virtual environment
source .venv/Scripts/activate # only required for Windows (Git Bash)
```

- Use un [entorno de Conda](#). Para instalar Conda, consulte [Instalación de Miniconda](#).
- Usa un contenedor de desarrollo en Visual Studio Code o en GitHub Codespaces.

2: Instalación de los paquetes de biblioteca de Azure necesarios

En este paso, instalará las bibliotecas del SDK de Azure necesarias para crear la base de datos.

1. En la consola, cree un archivo `requirements.txt` que muestre las bibliotecas de administración que se usan en este ejemplo:

Azure CLI

```
azure-mgmt-resource  
azure-mgmt-rdbms  
azure-identity  
mysql-connector-python
```

ⓘ Nota

La `mysql-connector-python` biblioteca no forma parte del SDK de Azure. Es una biblioteca de terceros que puede usar para conectarse a bases de datos MySQL. También puede usar otras bibliotecas, como `PyMySQL` o `SQLAlchemy`, para conectarse a bases de datos MySQL.

2. En la consola con el entorno virtual activado, instale los requisitos:

Consola

```
pip install -r requirements.txt
```

ⓘ Nota

En Windows, al intentar instalar la biblioteca mysql en una biblioteca de Python de 32 bits se produce un error sobre el archivo `mysql.h`. En este caso, instale una versión de 64 bits de Python e inténtelo de nuevo.

3. Establecer variables de entorno

En este paso, establecerá las variables de entorno para usarlas en el código de este artículo. El código usa el `os.environ` método para recuperar los valores.

Juerga

Azure CLI

```
#!/bin/bash
export AZURE_RESOURCE_GROUP_NAME=<ResourceGroupName> # Change to your
preferred resource group name
export LOCATION=<Location> # Change to your preferred region
export AZURE_SUBSCRIPTION_ID=$(az account show --query id --output tsv)
export PUBLIC_IP_ADDRESS=$(curl -s https://api.ipify.org)
export DB_SERVER_NAME=<DB_Server_Name> # Change to your preferred DB server
name
export DB_ADMIN_NAME=<DB_Admin_Name> # Change to your preferred admin name
export DB_ADMIN_PASSWORD=<DB_Admin_Passwrod> # Change to your preferred admin
password
export DB_NAME=<DB_Name> # Change to your preferred database name
export DB_PORT=3306
export version=ServerVersion.EIGHT0_21
```

4: Escribir código para crear y configurar un servidor flexible de MySQL con una base de datos

En este paso, creará un archivo de Python denominado `provision_blob.py` con el código siguiente. Este script de Python usa el SDK de Azure para bibliotecas de administración de Python para crear un grupo de recursos, un servidor flexible de MySQL y una base de datos en ese servidor.

Python

```
import random, os
from azure.identity import DefaultAzureCredential
from azure.mgmt.resource import ResourceManagementClient
from azure.mgmt.rdbms.mysql_flexibleServers import MySQLManagementClient
from azure.mgmt.rdbms.mysql_flexibleServers.models import Server, ServerVersion

# Acquire a credential object using CLI-based authentication.
credential = DefaultAzureCredential()

# Retrieve subscription ID from environment variable
subscription_id = os.environ["AZURE_SUBSCRIPTION_ID"]

# Retrieve resource group name and location from environment variables
```

```
RESOURCE_GROUP_NAME = os.environ["AZURE_RESOURCE_GROUP_NAME"]
LOCATION = os.environ["LOCATION"]

# Step 1: Provision the resource group.
resource_client = ResourceManagementClient(credential, subscription_id)

rg_result = resource_client.resource_groups.create_or_update(RESOURCE_GROUP_NAME,
    { "location": LOCATION })

print(f"Provisioned resource group {rg_result.name}")

# For details on the previous code, see Example: Provision a resource group
# at https://docs.microsoft.com/azure/developer/python/azure-sdk-example-resource-
group

# Step 2: Provision the database server

# Retrieve server name, admin name, and admin password from environment variables

db_server_name = os.environ.get("DB_SERVER_NAME")
db_admin_name = os.environ.get("DB_ADMIN_NAME")
db_admin_password = os.environ.get("DB_ADMIN_PASSWORD")

# Obtain the management client object
mysql_client = MySQLManagementClient(credential, subscription_id)

# Provision the server and wait for the result
server_version = os.environ.get("DB_SERVER_VERSION")

poller = mysql_client.servers.begin_create(RESOURCE_GROUP_NAME,
    db_server_name,
    Server(
        location=LOCATION,
        administrator_login=db_admin_name,
        administrator_login_password=db_admin_password,
        version=ServerVersion[server_version] # Note: dictionary-style enum
access
    )
)

server = poller.result()

print(f"Provisioned MySQL server {server.name}")

# Step 3: Provision a firewall rule to allow the local workstation to connect

RULE_NAME = "allow_ip"
ip_address = os.environ["PUBLIC_IP_ADDRESS"]

# Provision the rule and wait for completion
poller = mysql_client.firewall_rules.begin_create_or_update(RESOURCE_GROUP_NAME,
    db_server_name, RULE_NAME,
    { "start_ip_address": ip_address, "end_ip_address": ip_address }
)
```

```

firewall_rule = poller.result()

print(f"Provisioned firewall rule {firewall_rule.name}")

# Step 4: Provision a database on the server

db_name = os.environ.get("DB_NAME", "example-db1")

poller = mysql_client.databases.begin_create_or_update(RESOURCE_GROUP_NAME,
    db_server_name, db_name, {})

db_result = poller.result()

print(f"Provisioned MySQL database {db_result.name} with ID {db_result.id}")

```

Autenticación en el código

Más adelante en este artículo, inicia sesión en Azure mediante la CLI de Azure para ejecutar el código de ejemplo. Si la cuenta tiene permisos suficientes para crear grupos de recursos y recursos de almacenamiento en la suscripción de Azure, el script debe ejecutarse correctamente sin configuración adicional.

Para su uso en entornos de producción, se recomienda autenticarse con una entidad de servicio estableciendo las variables de entorno adecuadas. Este enfoque permite el acceso seguro y no interactivo adecuado para la automatización. Para obtener instrucciones de configuración, consulte [Autenticación de aplicaciones de Python con servicios de Azure](#).

Asegúrese de que a la entidad de servicio se le asigne un rol con permisos adecuados, como por ejemplo el rol de Colaborador a nivel de suscripción o grupo de recursos. Para más información sobre la asignación de roles, consulte [Control de acceso basado en rol \(RBAC\) en Azure](#).

Vínculos de referencia para las clases usadas en el código

- [ResourceManagementClient](#) (`azure.mgmt.resource`)
- [MySQLManagementClient](#) (`azure.mgmt.rdbms.mysql_flexibleServers`)
- [Servidor](#) (`azure.mgmt.rdbms.mysql_flexibleServers.models`)
- [ServerVersion](#) (`azure.mgmt.rdbms.mysql_flexibleServers.models`)

Para el servidor de bases de datos PostgreSQL, consulte:

- [PostgreSQLManagementClient](#) (`azure.mgmt.rdbms.postgresql_flexibleServers`)

5: Ejecutar el script

1. Si aún no lo ha hecho, inicie sesión en Azure mediante la CLI de Azure:

```
Azure CLI
```

```
az login
```

2. Ejecuta el script:

```
Consola
```

```
python provision_db.py
```

El script tarda un minuto o dos en completarse.

6: Insertar un registro y consultar la base de datos

En este paso, creará una tabla en la base de datos e insertará un registro. Puede usar la biblioteca mysql-connector para conectarse a la base de datos y ejecutar comandos SQL.

1. Cree un archivo denominado *use_db.py* con el código siguiente.

Este código solo funciona para MySQL; se usan bibliotecas diferentes para PostgreSQL.

```
Python
```

```
import os
import mysql.connector

db_server_name = os.environ["DB_SERVER_NAME"]
db_admin_name = os.getenv("DB_ADMIN_NAME")
db_admin_password = os.getenv("DB_ADMIN_PASSWORD")

db_name = os.getenv("DB_NAME")
db_port = os.getenv("DB_PORT")

connection = mysql.connector.connect(user=db_admin_name,
                                      password=db_admin_password, host=f"{db_server_name}.mysql.database.azure.com",
                                      port=db_port, database=db_name,
                                      ssl_ca='./BaltimoreCyberTrustRoot.crt.pem')

cursor = connection.cursor()

"""

# Alternate pyodbc connection; include pyodbc in requirements.txt
import pyodbc
```

```

driver = "{MySQL ODBC 5.3 UNICODE Driver}"

connect_string = f"DRIVER={driver};PORT=3306;SERVER=
{db_server_name}.mysql.database.azure.com;" \
                f"DATABASE={DB_NAME};UID={db_admin_name};PWD=
{db_admin_password}"

connection = pyodbc.connect(connect_string)
"""

table_name = "ExampleTable1"

sql_create = f"CREATE TABLE {table_name} (name varchar(255), code int)"

cursor.execute(sql_create)
print(f"Successfully created table {table_name}")

sql_insert = f"INSERT INTO {table_name} (name, code) VALUES ('Azure', 1)"
insert_data = "('Azure', 1)"

cursor.execute(sql_insert)
print("Successfully inserted data into table")

sql_select_values= f"SELECT * FROM {table_name}"

cursor.execute(sql_select_values)
row = cursor.fetchone()

while row:
    print(str(row[0]) + " " + str(row[1]))
    row = cursor.fetchone()

connection.commit()

```

Todo este código usa la API mysql.connector. La única parte específica de Azure es el dominio de host completo para el servidor MySQL (mysql.database.azure.com).

2. A continuación, descargue el certificado necesario para comunicarse a través de TSL/SSL con el servidor de Azure Database for MySQL. Para más información, consulte [Obtención de un certificado SSL](#) en la documentación de Azure Database for MySQL.

Juerga

Azure CLI

```

#!/bin/bash
# Download Baltimore CyberTrust Root certificate required for Azure MySQL
SSL connections
CERT_URL="https://www.digicert.com/CACerts/BaltimoreCyberTrustRoot.crt.pe
m"

```

```
CERT_FILE="BaltimoreCyberTrustRoot.crt.pem"
echo "Downloading SSL certificate..."
curl -o "$CERT_FILE" "$CERT_URL"
```

3. Por último, ejecute el código:

Consola

```
python use_db.py
```

Si ve un error que indica que la dirección IP del cliente no está permitida, compruebe que ha definido correctamente la variable `PUBLIC_IP_ADDRESS` de entorno. Si ya ha creado el servidor MySQL con la dirección IP incorrecta, puede agregar otro en [Azure Portal](#). En el portal, seleccione el servidor MySQL y, a continuación, seleccione **Seguridad de conexión**. Agregue la dirección IP de la estación de trabajo a la lista de direcciones IP permitidas.

7: Limpieza de recursos

Ejecute el comando `az group delete` si no es necesario mantener el grupo de recursos y los recursos de almacenamiento creados en este ejemplo.

Los grupos de recursos no incurren en cargos continuos en la suscripción, pero los recursos en el grupo de recursos, como las cuentas de almacenamiento, pueden seguir incurriendo en cargos. Se recomienda limpiar cualquier grupo que no se esté usando activamente. El argumento `--no-wait` permite que el comando devuelva inmediatamente en lugar de esperar a que finalice la operación.

Juerga

Azure CLI

```
#!/bin/bash
az group delete -n $AZURE_RESOURCE_GROUP_NAME --no-wait
```

También puede usar el método `ResourceManagementClient.resource_groups.begin_delete` para eliminar un grupo de recursos del código. El código de [Ejemplo: Crear un grupo de recursos](#) muestra el uso.

Como referencia: comandos equivalentes de la CLI de Azure

Los siguientes comandos de la CLI de Azure completan los mismos pasos de aprovisionamiento que el script de Python. Para una base de datos PostgreSQL, use [az postgres flexible-server](#) comandos.

Juerga

Azure CLI

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

# Set variables
export LOCATION=<Location> # Change to your preferred region
export AZURE_RESOURCE_GROUP_NAME=<ResourceGroupName> # Change to your
preferred resource group name
export DB_SERVER_NAME=<DB_Server_Name> # Change to your preferred DB server
name
export DB_ADMIN_NAME=<DB_Admin_Name> # Change to your preferred admin name
export DB_ADMIN_PASSWORD=<DB_Admin_Password> # Change to your preferred admin
password
export DB_NAME=<DB_Name> # Change to your preferred database name
export DB_SERVER_VERSION="5.7"

# Get public IP address
export PUBLIC_IP_ADDRESS=$(curl -s https://api.ipify.org)

# Provision the resource group
echo "Creating resource group: $AZURE_RESOURCE_GROUP_NAME"
az group create \
    --location "$LOCATION" \
    --name "$AZURE_RESOURCE_GROUP_NAME"

# Provision the MySQL Flexible Server
echo "Creating MySQL Flexible Server: $DB_SERVER_NAME"
az mysql flexible-server create \
    --location "$LOCATION" \
    --resource-group "$AZURE_RESOURCE_GROUP_NAME" \
    --name "$DB_SERVER_NAME" \
    --admin-user "$DB_ADMIN_NAME" \
    --admin-password "$DB_ADMIN_PASSWORD" \
    --sku-name Standard_B1ms \
    --version "$DB_SERVER_VERSION" \
    --yes

# Provision a firewall rule to allow access from the public IP address
echo "Creating firewall rule for public IP: $PUBLIC_IP_ADDRESS"
az mysql flexible-server firewall-rule create \
    --resource-group "$AZURE_RESOURCE_GROUP_NAME" \
    --name "$DB_SERVER_NAME" \
    --rule-name allow_ip \
    --start-ip-address "$PUBLIC_IP_ADDRESS" \
    --end-ip-address "$PUBLIC_IP_ADDRESS"
```

```
# Provision the database
echo "Creating database: $DB_NAME"
az mysql flexible-server db create \
    --resource-group "$AZURE_RESOURCE_GROUP_NAME" \
    --server-name "$DB_SERVER_NAME" \
    --database-name "$DB_NAME"

echo "MySQL Flexible Server and database created successfully."
```

Consulte también

- ejemplo de : [creación de un grupo de recursos](#)
- [ejemplo: Enumeración de grupos de recursos en una suscripción](#)
- [Ejemplo : Creación de Azure Storage](#)
- [Ejemplo: Uso de Azure Storage](#)
- [Ejemplo: Creación e implementación de una aplicación web](#)
- ejemplo de : [creación de una máquina virtual](#)
- [Uso de Azure Managed Disks con máquinas virtuales](#)
- [Completar una breve encuesta sobre el SDK de Azure para Python ↗](#)

Ejemplo: Uso de las bibliotecas de Azure para crear una máquina virtual

11/06/2025

En este artículo, aprenderá a usar las bibliotecas de administración del SDK de Azure en un script de Python para crear un grupo de recursos que contenga una máquina virtual Linux.

Los [comandos equivalentes](#) de la CLI de Azure se enumeran más adelante en este artículo. Si prefiere usar Azure Portal, consulte [Creación de una máquina virtual Linux](#) y [Creación de una máquina virtual Windows](#).

! Nota

La creación de una máquina virtual mediante código es un proceso de varios pasos que implica el aprovisionamiento de una serie de otros recursos que requiere la máquina virtual. Si simplemente ejecuta este código desde la línea de comandos, es mucho más fácil usar el [az vm create](#) comando , que aprovisiona automáticamente estos recursos secundarios con valores predeterminados para cualquier configuración que elija omitir. Los únicos argumentos necesarios son un grupo de recursos, un nombre de máquina virtual, un nombre de imagen y credenciales de inicio de sesión. Para más información, consulte [Creación rápida de una máquina virtual con la CLI de Azure](#).

1: Configuración del entorno de desarrollo local

Si aún no lo ha hecho, configure un entorno en el que pueda ejecutar este código. Estas son algunas opciones:

Azure CLI

```
#!/bin/bash
# Create a virtual environment
python -m venv .venv
# Activate the virtual environment
source .venv/Scripts/activate # only required for Windows (Git Bash)
```

- Use un [entorno de Conda](#). Para instalar Conda, consulte [Instalación de Miniconda](#).
- Usa un contenedor de desarrollo en Visual Studio Code o en GitHub Codespaces .

2: Instalación de los paquetes de biblioteca de Azure necesarios

Cree un archivo *requirements.txt* que especifique los paquetes de administración de Azure SDK necesarios para este script.

```
txt  
  
azure-mgmt-resource  
azure-mgmt-compute  
azure-mgmt-network  
azure-identity
```

A continuación, instale las bibliotecas de administración especificadas en *requirements.txt*:

```
Consola  
  
pip install -r requirements.txt
```

3: Escribir código para crear una máquina virtual

Cree un archivo de Python denominado *provision_vm.py* con el código siguiente. Los comentarios explican los detalles:

```
Python  
  
# Import the needed credential and management objects from the libraries.  
import os  
  
from azure.identity import DefaultAzureCredential  
from azure.mgmt.compute import ComputeManagementClient  
from azure.mgmt.network import NetworkManagementClient  
from azure.mgmt.resource import ResourceManagementClient  
  
print(  
    "Provisioning a virtual machine...some operations might take a \  
    minute or two."  
)  
  
# Acquire a credential object.  
credential = DefaultAzureCredential()  
  
# Retrieve subscription ID from environment variable.  
subscription_id = os.environ["AZURE_SUBSCRIPTION_ID"]  
  
# Step 1: Provision a resource group
```

```
# Obtain the management object for resources.
resource_client = ResourceManagementClient(credential, subscription_id)

# Constants we need in multiple places: the resource group name and
# the region in which we provision resources. You can change these
# values however you want.
RESOURCE_GROUP_NAME = "PythonAzureExample-VM-rg"
LOCATION = "westus2"

# Provision the resource group.
rg_result = resource_client.resource_groups.create_or_update(
    RESOURCE_GROUP_NAME, {"location": LOCATION}
)

print(
    f"Provisioned resource group {rg_result.name} in the \
{rg_result.location} region"
)

# For details on the previous code, see Example: Provision a resource
# group at https://learn.microsoft.com/azure/developer/python/
# azure-sdk-example-resource-group

# Step 2: provision a virtual network

# A virtual machine requires a network interface client (NIC). A NIC
# requires a virtual network and subnet along with an IP address.
# Therefore we must provision these downstream components first, then
# provision the NIC, after which we can provision the VM.

# Network and IP address names
VNET_NAME = "python-example-vnet"
SUBNET_NAME = "python-example-subnet"
IP_NAME = "python-example-ip"
IP_CONFIG_NAME = "python-example-ip-config"
NIC_NAME = "python-example-nic"

# Obtain the management object for networks
network_client = NetworkManagementClient(credential, subscription_id)

# Provision the virtual network and wait for completion
poller = network_client.virtual_networks.begin_create_or_update(
    RESOURCE_GROUP_NAME,
    VNET_NAME,
    {
        "location": LOCATION,
        "address_space": {"address_prefixes": ["10.0.0.0/16"]},
    },
)
vnet_result = poller.result()

print(
    f"Provisioned virtual network {vnet_result.name} with address \\"
```

```

prefixes {vnet_result.address_space.address_prefixes}"
)

# Step 3: Provision the subnet and wait for completion
poller = network_client.subnets.begin_create_or_update(
    RESOURCE_GROUP_NAME,
    VNET_NAME,
    SUBNET_NAME,
    {"address_prefix": "10.0.0.0/24"},
)
subnet_result = poller.result()

print(
    f"Provisioned virtual subnet {subnet_result.name} with address \
prefix {subnet_result.address_prefix}"
)

# Step 4: Provision an IP address and wait for completion
poller = network_client.public_ip_addresses.begin_create_or_update(
    RESOURCE_GROUP_NAME,
    IP_NAME,
    {
        "location": LOCATION,
        "sku": {"name": "Standard"},
        "public_ip_allocation_method": "Static",
        "public_ip_address_version": "IPV4",
    },
)
ip_address_result = poller.result()

print(
    f"Provisioned public IP address {ip_address_result.name} \
with address {ip_address_result.ip_address}"
)

# Step 5: Provision the network interface client
poller = network_client.network_interfaces.begin_create_or_update(
    RESOURCE_GROUP_NAME,
    NIC_NAME,
    {
        "location": LOCATION,
        "ip_configurations": [
            {
                "name": IP_CONFIG_NAME,
                "subnet": {"id": subnet_result.id},
                "public_ip_address": {"id": ip_address_result.id},
            }
        ],
    },
)
nic_result = poller.result()

print(f"Provisioned network interface client {nic_result.name}")

```

```

# Step 6: Provision the virtual machine

# Obtain the management object for virtual machines
compute_client = ComputeManagementClient(credential, subscription_id)

VM_NAME = "ExampleVM"
USERNAME = "azureuser"
PASSWORD = "ChangePa$$w0rd24"

print(
    f"Provisioning virtual machine {VM_NAME}; this operation might \
take a few minutes."
)

# Provision the VM specifying only minimal arguments, which defaults
# to an Ubuntu 18.04 VM on a Standard_DS1_v2 plan with a public IP address
# and a default virtual network/subnet.

poller = compute_client.virtual_machines.begin_create_or_update(
    RESOURCE_GROUP_NAME,
    VM_NAME,
    {
        "location": LOCATION,
        "storage_profile": {
            "image_reference": {
                "publisher": "Canonical",
                "offer": "UbuntuServer",
                "sku": "16.04.0-LTS",
                "version": "latest",
            }
        },
        "hardware_profile": {"vm_size": "Standard_DS1_v2"},
        "os_profile": {
            "computer_name": VM_NAME,
            "admin_username": USERNAME,
            "admin_password": PASSWORD,
        },
        "network_profile": {
            "network_interfaces": [
                {
                    "id": nic_result.id,
                }
            ]
        },
    },
)
vm_result = poller.result()

print(f"Provisioned virtual machine {vm_result.name}")

```

Autenticación en el código

Más adelante en este artículo, inicia sesión en Azure mediante la CLI de Azure para ejecutar el código de ejemplo. Si la cuenta tiene permisos suficientes para crear grupos de recursos y recursos de almacenamiento en la suscripción de Azure, el script debe ejecutarse correctamente sin configuración adicional.

Para usar este código en un entorno de producción, auténtíquese mediante una entidad de servicio estableciendo variables de entorno. Este enfoque permite el acceso seguro y automatizado sin depender del inicio de sesión interactivo. Para obtener instrucciones detalladas, consulte [Autenticación de aplicaciones de Python con servicios de Azure](#).

Asegúrese de que a la entidad de servicio se le asigna un rol con permisos suficientes para crear grupos de recursos y cuentas de almacenamiento. Por ejemplo, asignar el rol Colaborador en el nivel de suscripción proporciona el acceso necesario. Para más información sobre las asignaciones de roles, consulte [Control de acceso basado en rol \(RBAC\) en Azure](#).

Vínculos de referencia para las clases usadas en el código

- [DefaultCredential \(azure.identity\)](#)
- [ResourceManagementClient \(azure.mgmt.resource\)](#)
- [NetworkManagementClient \(azure.mgmt.network\)](#)
- [ComputeManagementClient \(azure.mgmt.compute\)](#)

4. Ejecutar el script

1. Si aún no lo ha hecho, inicie sesión en Azure mediante la CLI de Azure:

```
Azure CLI
```

```
az login
```

2. Establezca la variable de entorno `AZURE_SUBSCRIPTION_ID` en su identificador de suscripción. (Puede ejecutar el comando `az account show` y obtener el identificador de suscripción de la propiedad `id` en la salida):

```
Azure CLI
```

```
export AZURE_SUBSCRIPTION_ID=$(az account show --query id -o tsv)
```

3. Ejecuta el script:

```
Consola
```

```
python provision_vm.py
```

El proceso de aprovisionamiento tarda unos minutos en completarse.

5. Comprobación de los recursos

Abra [Azure Portal](#), vaya al grupo de recursos "PythonAzureExample-VM-rg" y anote la máquina virtual, el disco virtual, el grupo de seguridad de red, la dirección IP pública, la interfaz de red y la red virtual.

The screenshot shows the Azure Portal interface for the 'PythonAzureExample-VM-rg' resource group. The left sidebar has sections for Overview, Activity log, Access control (IAM), Tags, Events, Settings (Deployments, Security, Policies, Properties, Locks), Cost Management (Cost analysis, Cost alerts (preview)), and a bottom section for Log Analytics. The main area shows the 'Essentials' blade with subscription information (Primary, Subscription ID) and a 'Tags' section. Below that is a search bar and filter options (Type == all, Location == all, Add filter, No grouping). A table lists five resources: ExampleVM, ExampleVM_disk1_f1eda1ebf9f84a11983a1b550f37b865, python-example-ip, python-example-nic, and python-example-vnet. The last three items are highlighted with a red box.

También puede usar la CLI de Azure para comprobar que la máquina virtual existe con el comando [az vm list](#) :

```
Azure CLI
```

```
az vm list --resource-group PythonAzureExample-VM-rg
```

Comandos equivalentes de la CLI de Azure

```
Azure CLI
```

```
# Provision the resource group
```

```

az group create -n PythonAzureExample-VM-rg -l westus2

# Provision a virtual network and subnet

az network vnet create -g PythonAzureExample-VM-rg -n python-example-vnet \
--address-prefix 10.0.0.0/16 --subnet-name python-example-subnet \
--subnet-prefix 10.0.0.0/24

# Provision a public IP address

az network public-ip create -g PythonAzureExample-VM-rg -n python-example-ip \
--allocation-method Dynamic --version IPv4

# Provision a network interface client

az network nic create -g PythonAzureExample-VM-rg --vnet-name python-example-vnet \
--subnet python-example-subnet -n python-example-nic \
--public-ip-address python-example-ip

# Provision the virtual machine

az vm create -g PythonAzureExample-VM-rg -n ExampleVM -l "westus2" \
--nics python-example-nic --image UbuntuLTS --public-ip-sku Standard \
--admin-username azureuser --admin-password ChangePa$$w0rd24

```

Si recibe un error sobre las restricciones de capacidad, puede probar otro tamaño o región. Para obtener más información, consulte [Solución de errores para SKU no disponible](#).

6: Limpieza de recursos

Deje los recursos en su lugar si desea seguir usando la máquina virtual y la red que creó en este artículo. De lo contrario, ejecute el comando [az group delete](#) para eliminar el grupo de recursos.

Los grupos de recursos no incurren en cargos continuos en la suscripción, pero los recursos contenidos en el grupo, como las máquinas virtuales, pueden seguir incurriendo en cargos. Se recomienda limpiar cualquier grupo que no se esté usando activamente. El argumento `--no-wait` permite que el comando devuelva inmediatamente en lugar de esperar a que finalice la operación.

Azure CLI

```
az group delete -n PythonAzureExample-VM-rg --no-wait
```

También puede usar el método [ResourceManagementClient.resource_groups.begin_delete](#) para eliminar un grupo de recursos del código. El código de Ejemplo: Crear un grupo de recursos

muestra el uso.

Consulte también

- ejemplo de : [creación de un grupo de recursos](#)
- [ejemplo: Enumeración de grupos de recursos en una suscripción](#)
- Ejemplo : [Creación de Azure Storage](#)
- [Ejemplo: Uso de Azure Storage](#)
- [Ejemplo: Creación de una aplicación web e implementación de código](#)
- [Ejemplo: Creación y consulta de una base de datos](#)
- [Uso de Azure Managed Disks con máquinas virtuales](#)
- [Completar una breve encuesta sobre el SDK de Azure para Python ↗](#)

Los siguientes recursos contienen ejemplos más completos con Python para crear una máquina virtual:

- [Ejemplos de administración de Azure Virtual Machines: Python ↗](#) (GitHub). En el ejemplo se muestran más operaciones de administración, como iniciar y reiniciar una máquina virtual, detener y eliminar una máquina virtual, aumentar el tamaño del disco y administrar discos de datos.

Uso de Azure Managed Disks con las bibliotecas de Azure (SDK) para Python

12/06/2025

Azure Managed Disks es un almacenamiento en bloques duradero y de alto rendimiento diseñado para su uso con Azure Virtual Machines y Azure VMware Solution. Simplifican la administración de discos, ofrecen mayor escalabilidad, mejoran la seguridad y eliminan la necesidad de administrar las cuentas de almacenamiento directamente. Para más información, consulte [Azure Managed Disks](#).

Para las operaciones en Managed Disks asociadas a una máquina virtual existente, use la `azure-mgmt-compute` biblioteca .

En los ejemplos de código de este artículo se muestran las operaciones comunes con Managed Disks mediante la `azure-mgmt-compute` biblioteca. Estos ejemplos no están diseñados para ejecutarse como scripts independientes, sino para integrarse en su propio código. Para obtener información sobre cómo crear una `ComputeManagementClient` instancia desde `azure.mgmt.compute` en el script, consulte [Ejemplo: Creación de una máquina virtual](#).

Para obtener ejemplos más completos de cómo usar la `azure-mgmt-compute` biblioteca, consulte [Ejemplos de Azure SDK para Python relacionados con cómputo](#) en GitHub.

Discos administrados independientes

En los ejemplos siguientes se muestran diferentes formas de aprovisionar Managed Disks independientes.

Creación de un disco administrado vacío

En este ejemplo se muestra cómo crear un disco administrado vacío. Puede usarlo como un disco en blanco para conectarse a una máquina virtual o como punto de partida para crear instantáneas o imágenes.

Python

```
from azure.mgmt.compute.models import DiskCreateOption

poller = compute_client.disks.begin_create_or_update(
    'my_resource_group',
    'my_disk_name',
    {
        'location': 'eastus',
```

```
        'disk_size_gb': 20,
        'creation_data': {
            'create_option': DiskCreateOption.empty
        }
    }
)
disk_resource = poller.result()
```

Creación de un disco administrado a partir de Blob Storage

En este ejemplo se muestra cómo crear un disco administrado a partir de un archivo VHD almacenado en Azure Blob Storage. Esto resulta útil cuando desea reutilizar o mover un disco duro virtual existente a Azure.

Python

```
from azure.mgmt.compute.models import DiskCreateOption

poller = compute_client.disks.begin_create_or_update(
    'my_resource_group',
    'my_disk_name',
    {
        'location': 'eastus',
        'creation_data': {
            'create_option': DiskCreateOption.IMPORT,
            'storage_account_id': '/subscriptions/<subscription-id>/resourceGroups/<resource-group-name>/providers/Microsoft.Storage/storageAccounts/<storage-account-name>',
            'source_uri': 'https://<storage-account-name>.blob.core.windows.net/vm-images/test.vhd'
        }
    }
)
disk_resource = poller.result()
```

Creación de una imagen de Managed Disk a partir de Blob Storage

En este ejemplo se muestra cómo crear una imagen de Managed Disk a partir de un archivo VHD almacenado en Azure Blob Storage. Esto resulta útil cuando desea crear una imagen reutilizable que se pueda usar para crear nuevas máquinas virtuales.

Python

```
from azure.mgmt.compute.models import OperatingSystemStateTypes, HyperVGeneration

poller = compute_client.images.begin_create_or_update(
```

```

'my_resource_group',
'my_image_name',
{
    'location': 'eastus',
    'storage_profile': {
        'os_disk': {
            'os_type': 'Linux',
            'os_state': OperatingSystemStateTypes.GENERALIZED,
            'blob_uri': 'https://<storage-account-
name>.blob.core.windows.net/vm-images/test.vhd',
            'caching': "ReadWrite",
        },
    },
    'hyper_v_generation': HyperVGeneration.V2,
}
)
image_resource = poller.result()

```

Creación de un disco administrado a partir de su propia imagen

En este ejemplo se muestra cómo crear un disco administrado mediante la copia de uno existente. Esto resulta útil cuando desea realizar una copia de seguridad o usar la misma configuración de disco en otra máquina virtual.

Python

```

from azure.mgmt.compute.models import DiskCreateOption

# If you don't know the id, do a 'get' like this to obtain it
managed_disk = compute_client.disks.get(self.group_name, 'myImageDisk')

poller = compute_client.disks.begin_create_or_update(
    'my_resource_group',
    'my_disk_name',
    {
        'location': 'eastus',
        'creation_data': {
            'create_option': DiskCreateOption.COPY,
            'source_resource_id': managed_disk.id
        }
    }
)

disk_resource = poller.result()

```

Máquina virtual con Managed Disks

Puede crear una máquina virtual con un disco administrado creado implícitamente en función de una imagen de disco específica, lo que elimina la necesidad de definir manualmente todos los detalles del disco.

Un disco administrado se crea implícitamente al crear una máquina virtual a partir de una imagen del sistema operativo en Azure. Azure controla automáticamente la cuenta de almacenamiento, por lo que no es necesario especificar `storage_profile.os_disk` ni crear manualmente una cuenta de almacenamiento.

Python

```
storage_profile = azure.mgmt.compute.models.StorageProfile(  
    image_reference = azure.mgmt.compute.models.ImageReference(  
        publisher='Canonical',  
        offer='UbuntuServer',  
        sku='16.04-LTS',  
        version='latest'  
    )  
)
```

Para obtener un ejemplo completo en el que se muestra cómo crear una máquina virtual mediante las bibliotecas de administración de Azure para Python, consulte [Ejemplo: Creación de una máquina virtual](#). En este ejemplo se muestra cómo usar el `storage_profile` parámetro .

También puede crear un `storage_profile` a partir de su propia imagen.

Python

```
# If you don't know the id, do a 'get' like this to obtain it  
image = compute_client.images.get(self.group_name, 'myImageDisk')  
  
storage_profile = azure.mgmt.compute.models.StorageProfile(  
    image_reference = azure.mgmt.compute.models.ImageReference(  
        id = image.id  
    )  
)
```

Puede conectar fácilmente un disco administrado aprovisionado anteriormente:

Python

```
vm = compute_client.virtual_machines.get(  
    'my_resource_group',  
    'my_vm'  
)  
managed_disk = compute_client.disks.get('my_resource_group', 'myDisk')  
  
vm.storage_profile.data_disks.append({
```

```

'lun': 12, # You choose the value, depending of what is available for you
'name': managed_disk.name,
'create_option': DiskCreateOptionTypes.attach,
'managed_disk': {
    'id': managed_disk.id
}
})

async_update = compute_client.virtual_machines.begin_create_or_update(
    'my_resource_group',
    vm.name,
    vm,
)
async_update.wait()

```

Conjuntos de escalado de máquinas virtuales con discos administrados

Antes de Azure Managed Disks, tenías que crear manualmente una cuenta de almacenamiento para cada máquina virtual en tu Conjunto de Escala de Máquinas Virtuales y usar el parámetro `vhd_containers` para especificar esas cuentas de almacenamiento en la API REST del conjunto de escala.

Con Azure Managed Disks, la administración de cuentas de almacenamiento ya no es necesaria. Como resultado, los `storage_profile` conjuntos de escalado de máquinas virtuales usados para los conjuntos de escalado de máquinas virtuales ahora pueden coincidir con el usado para la creación de máquinas virtuales individuales:

Python

```

'storage_profile': {
    'image_reference': {
        "publisher": "Canonical",
        "offer": "UbuntuServer",
        "sku": "16.04-LTS",
        "version": "latest"
    }
},

```

El ejemplo completo es el siguiente:

Python

```

naming_infix = "PyTestInfix"

vmss_parameters = {
    'location': self.region,

```

```

    "overprovision": True,
    "upgrade_policy": {
        "mode": "Manual"
    },
    'sku': {
        'name': 'Standard_A1',
        'tier': 'Standard',
        'capacity': 5
    },
    'virtual_machine_profile': {
        'storage_profile': {
            'image_reference': {
                "publisher": "Canonical",
                "offer": "UbuntuServer",
                "sku": "16.04-LTS",
                "version": "latest"
            }
        },
        'os_profile': {
            'computer_name_prefix': naming_infix,
            'admin_username': 'Foo12',
            'admin_password': 'Bar@123!!!!',
        },
        'network_profile': {
            'network_interface_configurations' : [
                {
                    'name': naming_infix + 'nic',
                    "primary": True,
                    'ip_configurations': [
                        {
                            'name': naming_infix + 'ipconfig',
                            'subnet': {
                                'id': subnet.id
                            }
                        }
                    ]
                }
            ]
        }
    }
}

# Create VMSS test
result_create = compute_client.virtual_machine_scale_sets.begin_create_or_update(
    'my_resource_group',
    'my_scale_set',
    vmss_parameters,
)
vmss_result = result_create.result()

```

Otras operaciones con Managed Disks

Cambio de tamaño de un disco administrado

En este ejemplo se muestra cómo hacer que un disco administrado existente sea mayor. Esto resulta útil cuando necesita más espacio para los datos o las aplicaciones.

Python

```
managed_disk = compute_client.disks.get('my_resource_group', 'myDisk')
managed_disk.disk_size_gb = 25

async_update = self.compute_client.disks.begin_create_or_update(
    'my_resource_group',
    'myDisk',
    managed_disk
)
async_update.wait()
```

Actualización del tipo de cuenta de almacenamiento de Managed Disks

En este ejemplo se muestra cómo cambiar el tipo de almacenamiento de un disco administrado y aumentar su tamaño. Esto resulta útil cuando necesita más espacio o un mejor rendimiento para los datos o las aplicaciones.

Python

```
from azure.mgmt.compute.models import StorageAccountTypes

managed_disk = compute_client.disks.get('my_resource_group', 'myDisk')
managed_disk.account_type = StorageAccountTypes.STANDARD_LRS

async_update = self.compute_client.disks.begin_create_or_update(
    'my_resource_group',
    'myDisk',
    managed_disk
)
async_update.wait()
```

Creación de una imagen a partir de Blob Storage

En este ejemplo se muestra cómo crear una imagen de Managed Disk a partir de un archivo VHD almacenado en Azure Blob Storage. Esto resulta útil cuando desea crear una imagen reutilizable que puede usar para crear nuevas máquinas virtuales.

Python

```
async_create_image = compute_client.images.create_or_update(
    'my_resource_group',
```

```
'myImage',
{
    'location': 'eastus',
    'storage_profile': {
        'os_disk': {
            'os_type': 'Linux',
            'os_state': "Generalized",
            'blob_uri': 'https://<storage-account-name>.blob.core.windows.net/vm-images/test.vhd',
            'caching': "ReadWrite",
        }
    }
}
)
image = async_create_image.result()
```

Creación de una instantánea de un disco administrado que está conectado actualmente a una máquina virtual

En este ejemplo se muestra cómo tomar una instantánea de un disco administrado que está conectado a una máquina virtual. Puede usar la instantánea para hacer una copia de seguridad del disco o restaurarlo más adelante si es necesario.

Python

```
managed_disk = compute_client.disks.get('my_resource_group', 'myDisk')

async_snapshot_creation = self.compute_client.snapshots.begin_create_or_update(
    'my_resource_group',
    'mySnapshot',
    {
        'location': 'eastus',
        'creation_data': {
            'create_option': 'Copy',
            'source_uri': managed_disk.id
        }
    }
)
snapshot = async_snapshot_creation.result()
```

Consulte también

- [Ejemplo: Creación de una máquina virtual](#)
- [Ejemplo: Creación de un grupo de recursos](#)
- [Ejemplo: Enumeración de grupos de recursos en una suscripción](#)
- [Ejemplo: Creación de Azure Storage](#)

- Ejemplo de uso de Azure Storage
- Ejemplo: Creación y uso de una base de datos MySQL
- Complete una breve encuesta sobre el SDK de Azure para Python ↗

Configuración del registro en las bibliotecas de Azure para Python

Artículo • 14/11/2024

Las bibliotecas de Azure para Python que están [basadas en azure.core](#) proporcionan la salida del registro mediante la biblioteca de [registro](#) de Python estándar.

Este es el proceso general para trabajar con el registro:

1. Adquiera el objeto de registro de la biblioteca deseada y establezca el nivel de registro.
2. Registre un controlador para el flujo de registro.
3. Para incluir información de HTTP, pase un parámetro `logging_enable=True` a un constructor de objetos de cliente, un constructor de objetos de credencial o un método específico.

En las restantes secciones del artículo encontrará más información al respecto.

Como norma general, el mejor recurso para comprender el uso del registro dentro de las bibliotecas es examinar el código fuente del SDK en [github.com/Azure/azure-sdk-for-python](#). Se recomienda clonar este repositorio de forma local para que pueda buscar fácilmente los detalles cuando sea necesario, como se sugiere en las siguientes secciones.

Establecimiento de los niveles de registro

Python

```
import logging

# ...

# Acquire the logger for a library (azure.mgmt.resource in this example)
logger = logging.getLogger('azure.mgmt.resource')

# Set the desired logging level
logger.setLevel(logging.DEBUG)
```

- En este ejemplo se adquiere el registrador de la biblioteca `azure.mgmt.resource` y, a continuación, se establece el nivel de registro en `logging.DEBUG`.
- Puede llamar a `logger.setLevel` en cualquier momento para cambiar el nivel de registro de los distintos segmentos de código.

Para establecer un nivel para una biblioteca diferente, utilice el nombre de esa biblioteca en la llamada a `logging.getLogger`. Por ejemplo, la biblioteca `azure-eventhubs` proporciona un registrador llamado `azure.eventhubs`, la biblioteca `azure-storage-queue` proporciona un registrador llamado `azure.storage.queue`, etc. (El código fuente del SDK usa con frecuencia la instrucción `logging.getLogger(__name__)`, que adquiere un registrador con el nombre del módulo contenedor).

También puede utilizar espacios de nombres más generales. Por ejemplo,

Python

```
import logging

# Set the logging level for all azure-storage-* libraries
logger = logging.getLogger('azure.storage')
logger.setLevel(logging.INFO)

# Set the logging level for all azure-* libraries
logger = logging.getLogger('azure')
logger.setLevel(logging.ERROR)
```

Algunas bibliotecas utilizan el registrador `azure` en lugar de un registrador específico.

Por ejemplo, la biblioteca `azure-storage-blob` usa el registrador `azure`.

puede usar el método `logger.isEnabledFor` para comprobar si está habilitado un nivel de registro determinado:

Python

```
print(
    f"Logger enabled for ERROR={logger.isEnabledFor(logging.ERROR)}, "
    f"WARNING={logger.isEnabledFor(logging.WARNING)}, "
    f"INFO={logger.isEnabledFor(logging.INFO)}, "
    f"DEBUG={logger.isEnabledFor(logging.DEBUG)}"
)
```

Los niveles de registro son los mismos que los [niveles de la biblioteca de registro estándar](#). En la tabla siguiente se describe el uso general de estos niveles de registro en las bibliotecas de Azure para Python:

[] Expandir tabla

Nivel de registro	Uso típico
<code>logging.ERROR</code>	Errores en los que no es probable que la aplicación se recupere (por ejemplo, memoria insuficiente).

Nivel de registro	Uso típico
logging.WARNING (predeterminado)	Una función no puede realizar la tarea deseada (pero no cuando la función se puede recuperar, como al volver a intentar una llamada a la API REST). Normalmente, las funciones registran una advertencia al generar excepciones. El nivel de advertencia habilita automáticamente el nivel de error.
logging.INFO	La función opera con normalidad o se cancela una llamada de servicio. Los eventos de información suelen incluir solicitudes, respuestas y encabezados. El nivel de información habilita automáticamente los niveles de error y de advertencia.
logging.DEBUG	Información detallada que se usa normalmente para solucionar problemas e incluye un seguimiento de la pila para las excepciones. El nivel de depuración habilita automáticamente los niveles de información, de advertencia y de error. PRECAUCIÓN: Si también se establece <code>logging_enable=True</code> , el nivel de depuración incluye información confidencial, como las claves de cuenta, en encabezados y otras credenciales. Asegúrese de proteger estos registros para evitar poner en peligro la seguridad.
logging.NOTSET	Deshabilita todos los registros.

Comportamiento del nivel de registro específico de la biblioteca

El comportamiento de registro exacto en cada nivel depende de la biblioteca en cuestión. Algunas bibliotecas, como `azure.eventhub`, realizan un registro extensivo, mientras que otras registran muy poco.

La mejor manera de examinar el registro exacto de una biblioteca es buscar los niveles de registro en el [código fuente del SDK de Azure para Python](#) ↗:

1. En la carpeta del repositorio, vaya a la carpeta `sdk` y, a continuación, vaya a la carpeta del servicio específico de su interés.
2. En esa carpeta, busque cualquiera de las siguientes cadenas:
 - `_LOGGER.error`
 - `_LOGGER.warning`
 - `_LOGGER.info`
 - `_LOGGER.debug`

Registro de un controlador del flujo de registros

Para capturar la salida del registro, debe registrar al menos un controlador del flujo de registros en el código:

Python

```
import logging
# Direct logging output to stdout. Without adding a handler,
# no logging output is visible.
handler = logging.StreamHandler(stream=sys.stdout)
logger.addHandler(handler)
```

En este ejemplo se registra un controlador que dirige la salida del registro a stdout. Puede usar otros tipos de controladores, como se describe en [logging.handlers](#) en la documentación de Python, o bien usar el método [logging.basicConfig](#).

Habilitación del registro HTTP para un objeto o una operación de cliente

De forma predeterminada, el registro en las bibliotecas de Azure no incluye información de HTTP. Para incluir información HTTP en la salida del registro, debe pasar `logging_enable=True` explícitamente a un constructor de objetos de cliente o credencial o a un método específico.

⊗ Precaución

El registro HTTP puede incluir información confidencial, como las claves de cuenta, en encabezados y otras credenciales. Asegúrese de proteger estos registros para evitar poner en peligro la seguridad.

Habilitación del registro HTTP para un objeto de cliente

Python

```
from azure.storage.blob import BlobClient
from azure.identity import DefaultAzureCredential

# Enable HTTP logging on the client object when using DEBUG level
```

```
# endpoint is the Blob storage URL.  
client = BlobClient(endpoint, DefaultAzureCredential(), logging_enable=True)
```

La habilitación del registro HTTP para un objeto de cliente habilita el registro de todas las operaciones invocadas mediante ese objeto.

Habilitación del registro HTTP para un objeto de credencial

Python

```
from azure.storage.blob import BlobClient  
from azure.identity import DefaultAzureCredential  
  
# Enable HTTP logging on the credential object when using DEBUG level  
credential = DefaultAzureCredential(logging_enable=True)  
  
# endpoint is the Blob storage URL.  
client = BlobClient(endpoint, credential)
```

Al habilitar el registro HTTP para un objeto de credencial, se habilita el registro de todas las operaciones invocadas a través de ese objeto, pero no de las operaciones de un objeto de cliente que no impliquen la autenticación.

Habilitación del registro para un método individual

Python

```
from azure.storage.blob import BlobClient  
from azure.identity import DefaultAzureCredential  
  
# endpoint is the Blob storage URL.  
client = BlobClient(endpoint, DefaultAzureCredential())  
  
# Enable HTTP logging for only this operation when using DEBUG level  
client.create_container("container01", logging_enable=True)
```

Ejemplo de salida del registro

El siguiente código es el que se muestra en [Ejemplo: Uso de una cuenta de almacenamiento](#), al que se ha agregado la habilitación del registro de DEBUG y HTTP:

Python

```

import logging
import os
import sys
import uuid

from azure.core import exceptions
from azure.identity import DefaultAzureCredential
from azure.storage.blob import BlobClient

logger = logging.getLogger("azure")
logger.setLevel(logging.DEBUG)

# Set the logging level for the azure.storage.blob library
logger = logging.getLogger("azure.storage.blob")
logger.setLevel(logging.DEBUG)

# Direct logging output to stdout. Without adding a handler,
# no logging output is visible.
handler = logging.StreamHandler(stream=sys.stdout)
logger.addHandler(handler)

print(
    f"Logger enabled for ERROR={logger.isEnabledFor(logging.ERROR)}, "
    f"WARNING={logger.isEnabledFor(logging.WARNING)}, "
    f"INFO={logger.isEnabledFor(logging.INFO)}, "
    f"DEBUG={logger.isEnabledFor(logging.DEBUG)}"
)

try:
    credential = DefaultAzureCredential()
    storage_url = os.environ["AZURE_STORAGE_BLOB_URL"]
    unique_str = str(uuid.uuid4())[0:5]

    # Enable logging on the client object
    blob_client = BlobClient(
        storage_url,
        container_name="blob-container-01",
        blob_name=f"sample-blob-{unique_str}.txt",
        credential=credential,
    )

    with open("./sample-source.txt", "rb") as data:
        blob_client.upload_blob(data, logging_body=True,
logging_enable=True)

except (
    exceptions.ClientAuthenticationError,
    exceptions.HttpResponseError
) as e:
    print(e.message)

```

La salida es como sigue:

Output

```
Logger enabled for ERROR=True, WARNING=True, INFO=True, DEBUG=True
Request URL: 'https://pythonazurestorage12345.blob.core.windows.net/blob-
container-01/sample-blob-5588e.txt'
Request method: 'PUT'
Request headers:
  'Content-Length': '77'
  'x-ms-blob-type': 'BlockBlob'
  'If-None-Match': '*'
  'x-ms-version': '2023-11-03'
  'Content-Type': 'application/octet-stream'
  'Accept': 'application/xml'
  'User-Agent': 'azsdk-python-storage-blob/12.19.0 Python/3.10.11
(Windows-10-10.0.22631-SP0)'
  'x-ms-date': 'Fri, 19 Jan 2024 19:25:53 GMT'
  'x-ms-client-request-id': '8f7b1b0b-b700-11ee-b391-782b46f5c56b'
  'Authorization': '*****'

Request body:
b"Hello there, Azure Storage. I'm a friendly file ready to be stored in a
blob."
Response status: 201
Response headers:
  'Content-Length': '0'
  'Content-MD5': 'SUytm0872jZh+KYqtgjbTA=='
  'Last-Modified': 'Fri, 19 Jan 2024 19:25:54 GMT'
  'ETag': '"0x8DC1924749AE3C3"'
  'Server': 'Windows-Azure-Blob/1.0 Microsoft-HTTPAPI/2.0'
  'x-ms-request-id': '7ac499fa-601e-006d-3f0d-4bdf28000000'
  'x-ms-client-request-id': '8f7b1b0b-b700-11ee-b391-782b46f5c56b'
  'x-ms-version': '2023-11-03'
  'x-ms-content-crc64': 'rtHLUlztgxc='
  'x-ms-request-server-encrypted': 'true'
  'Date': 'Fri, 19 Jan 2024 19:25:53 GMT'
Response content:
b''
```

⚠ Nota

Si obtiene un error de autorización, asegúrese de que la identidad con la que se está ejecutando tiene asignada la función "Colaborador de datos de blob de almacenamiento" en su contenedor de blob. Para obtener más información, consulte [Uso del almacenamiento blob desde el código de la aplicación \(pestana Sin contraseña\)](#).

Comentarios

¿Le ha resultado útil esta página?

 Sí

 No

[Proporcionar comentarios sobre el producto](#) | [Obtener ayuda en Microsoft Q&A](#)

Configuración de servidores proxy para El SDK de Azure para Python

22/05/2025

A menudo se necesita un proxy si:

- Está detrás de un firewall corporativo
- El tráfico de red debe pasar por un dispositivo de seguridad
- Quieres usar un proxy personalizado para depurar o enrutar

Si su organización requiere el uso de un servidor proxy para acceder a los recursos de Internet, debe establecer una variable de entorno con la información del servidor proxy para usar el SDK de Azure para Python. Establecer las variables de entorno (`HTTP_PROXY` y `HTTPS_PROXY`) hace que Azure SDK para Python use el servidor proxy en tiempo de ejecución.

Una dirección URL del servidor proxy tiene el formato `http[s]://[username:password@]<ip_address_or_domain>:<port>/` en el que la combinación de nombre de usuario y contraseña es opcional.

Puede obtener la información de proxy del equipo de TI o red, desde el explorador o desde las utilidades de red.

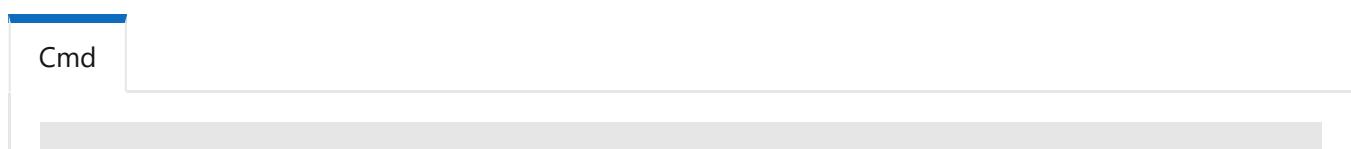
A continuación, puede configurar un proxy globalmente mediante variables de entorno o puede especificar un proxy pasando un argumento denominado `proxies` a un constructor de cliente individual o método de operación.

Configuración global

Para configurar un proxy globalmente para su script o aplicación, defina las variables de entorno `HTTP_PROXY` o `HTTPS_PROXY` con la dirección URL del servidor. Estas variables funcionan con cualquier versión de las bibliotecas de Azure. Tenga en cuenta que `HTTPS_PROXY` no significa `HTTPS` proxy, sino el proxy para las solicitudes de `https://`.

Estas variables de entorno se omiten si se pasa el parámetro `use_env_settings=False` a un constructor de objetos de cliente o método de operación.

Establecer desde la línea de comandos



Símbolo del sistema de Windows

```
rem Non-authenticated HTTP server:  
set HTTP_PROXY=http://10.10.1.10:1180  
  
rem Authenticated HTTP server:  
set HTTP_PROXY=http://username:password@10.10.1.10:1180  
  
rem Non-authenticated HTTPS server:  
set HTTPS_PROXY=http://10.10.1.10:1180  
  
rem Authenticated HTTPS server:  
set HTTPS_PROXY=http://username:password@10.10.1.10:1180
```

Configurado en código de Python

Puede establecer la configuración de proxy mediante variables de entorno, sin necesidad de configuración personalizada.

Python

```
import os  
os.environ["HTTP_PROXY"] = "http://10.10.1.10:1180"  
  
# Alternate URL and variable forms:  
# os.environ["HTTP_PROXY"] = "http://username:password@10.10.1.10:1180"  
# os.environ["HTTPS_PROXY"] = "http://10.10.1.10:1180"  
# os.environ["HTTPS_PROXY"] = "http://username:password@10.10.1.10:1180"
```

Configuración personalizada

Establecer en código de Python por cada cliente o por cada método

Para la configuración personalizada, puede especificar un proxy para un objeto de cliente específico o un método de operación. Especifique un servidor proxy con un argumento denominado `proxies`.

Por ejemplo, el código siguiente del artículo [Ejemplo: uso de Azure Storage](#) especifica un proxy HTTPS con credenciales de usuario con el `BlobClient` constructor. En este caso, el objeto procede de la biblioteca `azure.storage.blob`, que se basa en `azure.core`.

Python

```
from azure.identity import DefaultAzureCredential

# Import the client object from the SDK library
from azure.storage.blob import BlobClient

credential = DefaultAzureCredential()

storage_url = "https://<storageaccountname>.blob.core.windows.net"

blob_client = BlobClient(storage_url, container_name="blob-container-01",
    blob_name="sample-blob.txt", credential=credential,
    proxies={"https": "https://username:password@10.10.1.10:1180" })
)

# Other forms that the proxy URL might take:
# proxies={"http": "http://10.10.1.10:1180" }
# proxies={"http": "http://username:password@10.10.1.10:1180" }
# proxies={"https": "https://10.10.1.10:1180" }
```

Multinube: conexión a todas las regiones con las bibliotecas de Azure para Python

25/08/2025

Puede usar las bibliotecas de Azure para Python para conectarse a todas las regiones en las que Azure está [disponible ↗](#).

De forma predeterminada, las bibliotecas de Azure están configuradas para conectarse a la nube global de Azure.

Uso de constantes de nube soberana predefinidas

El módulo `AzureAuthorityHosts` de la biblioteca `azure.identity` proporciona constantes predefinidas de nube soberana.

- `AZURE_CHINA`
- `AZURE_GOVERNMENT`
- `AZURE_PUBLIC_CLOUD`

Para usar una definición, importe la constante adecuada de

`azure.identity.AzureAuthorityHosts` y aplíquela al crear objetos de cliente.

Al usar `DefaultAzureCredential`, como se muestra en el ejemplo siguiente, puede especificar la nube mediante el valor adecuado de `azure.identity.AzureAuthorityHosts`.

Python

```
import os
from azure.mgmt.resource import ResourceManagementClient, SubscriptionClient
from azure.identity import DefaultAzureCredential, AzureAuthorityHosts
from azure.core import AzureClouds

authority = AzureAuthorityHosts.AZURE_CHINA

# Set environment variable AZURE_SUBSCRIPTION_ID as well as environment variables
# for DefaultAzureCredential. For combinations of environment variables, see
# https://github.com/Azure/azure-sdk-for-python/tree/main/sdk/identity/azure-
# identity#environment-variables
subscription_id = os.environ["AZURE_SUBSCRIPTION_ID"]

# When using sovereign domains (that is, any cloud other than AZURE_PUBLIC_CLOUD),
# you must use an authority with DefaultAzureCredential.
credential = DefaultAzureCredential(authority=authority)

resource_client = ResourceManagementClient(
```

```
        credential, subscription_id, cloud_setting=AzureClouds.AZURE_CHINA_CLOUD
    )

subscription_client = SubscriptionClient(
    credential, cloud_setting=AzureClouds.AZURE_CHINA_CLOUD
)
```

! Nota

La `cloud_setting` característica se agrega recientemente y se implementa en las bibliotecas de administración del SDK de Azure. Durante este período, algunos clientes lo admiten mientras que otros no lo admiten. Para comprobar la compatibilidad, busque un `cloud_setting` parámetro en el constructor de cliente. Si el cliente del servicio aún no expone `cloud_setting`, puede seguir apuntando a nubes soberanas utilizando el enfoque anterior que se muestra en los siguientes ejemplos.

Python

```
import os
from azure.mgmt.resource import ResourceManagementClient, SubscriptionClient
from azure.identity import DefaultAzureCredential, AzureAuthorityHosts

authority = AzureAuthorityHosts.AZURE_CHINA
resource_manager = "https://management.chinacloudapi.cn"

# Set environment variable AZURE_SUBSCRIPTION_ID as well as environment variables
# for DefaultAzureCredential. For combinations of environment variables, see
# https://github.com/Azure/azure-sdk-for-python/tree/main/sdk/identity/azure-
# identity#environment-variables
subscription_id = os.environ["AZURE_SUBSCRIPTION_ID"]

# When using sovereign domains (that is, any cloud other than AZURE_PUBLIC_CLOUD),
# you must use an authority with DefaultAzureCredential.
credential = DefaultAzureCredential(authority=authority)

resource_client = ResourceManagementClient(
    credential,
    subscription_id,
    base_url=resource_manager,
    credential_scopes=[resource_manager + "/.default"],
)

subscription_client = SubscriptionClient(
    credential,
    base_url=resource_manager,
    credential_scopes=[resource_manager + "/.default"],
)
```

Uso de su propia definición de nube

En el código siguiente, reemplace los valores de las `authority`, `endpoint` y `audience` por los valores adecuados para la nube privada.

Python

```
import os
from azure.mgmt.resource import ResourceManagementClient, SubscriptionClient
from azure.identity import DefaultAzureCredential
from azure.profiles import KnownProfiles

# Set environment variable AZURE_SUBSCRIPTION_ID as well as environment variables
# for DefaultAzureCredential. For combinations of environment variables, see
# https://github.com/Azure/azure-sdk-for-python/tree/main/sdk/identity/azure-
# identity#environment-variables
subscription_id = os.environ["AZURE_SUBSCRIPTION_ID"]

authority = "<your authority>"
endpoint = "<your endpoint>"
audience = "<your audience>

# When using a private cloud, you must use an authority with
DefaultAzureCredential.
# The active_directory endpoint should be a URL like
https://login.microsoftonline.com.
credential = DefaultAzureCredential(authority=authority)

resource_client = ResourceManagementClient(
    credential, subscription_id,
    base_url=endpoint,
    profile=KnownProfiles.v2019_03_01_hybrid,
    credential_scopes=[audience])

subscription_client = SubscriptionClient(
    credential,
    base_url=endpoint,
    profile=KnownProfiles.v2019_03_01_hybrid,
    credential_scopes=[audience])
```

Por ejemplo, para Azure Stack, puede usar el comando `az cloud show` cli para devolver los detalles de una nube registrada. En la salida siguiente se muestran los valores devueltos para la nube pública de Azure, pero la salida de una nube privada de Azure Stack debe ser similar.

Resultados

```
{
  "endpoints": {
    "activeDirectory": "https://login.microsoftonline.com",
    "activeDirectoryDataLakeResourceId": "https://datalake.azure.net/",
    "activeDirectoryGraphResourceId": "https://graph.windows.net/",
```

```

    "activeDirectoryResourceId": "https://management.core.windows.net/",
    "appInsightsResourceId": "https://api.applicationinsights.io",
    "appInsightsTelemetryChannelResourceId":
    "https://dc.applicationinsights.azure.com/v2/track",
    "attestationResourceId": "https://attest.azure.net",
    "azmirrorStorageAccountResourceId": null,
    "batchResourceId": "https://batch.core.windows.net/",
    "gallery": "https://gallery.azure.com/",
    "logAnalyticsResourceId": "https://api.loganalytics.io",
    "management": "https://management.core.windows.net/",
    "mediaResourceId": "https://rest.media.azure.net",
    "microsoftGraphResourceId": "https://graph.microsoft.com/",
    "osssrdbmsResourceId": "https://osssrdbms-aad.database.windows.net",
    "portal": "https://portal.azure.com",
    "resourceManager": "https://management.azure.com/",
    "sqlManagement": "https://management.core.windows.net:8443/",
    "synapseAnalyticsResourceId": "https://dev.azuresynapse.net",
    "vmImageAliasDoc": "https://raw.githubusercontent.com/Azure/azure-rest-api-
specs/main/arm-compute/quickstart-templates/aliases.json"
},
"isActive": true,
"name": "AzureCloud",
"profile": "latest",
"suffixes": {
    "acrLoginServerEndpoint": ".azurecr.io",
    "attestationEndpoint": ".attest.azure.net",
    "azureDatalakeAnalyticsCatalogAndJobEndpoint": "azuredatalakeanalytics.net",
    "azureDatalakeStoreFileSystemEndpoint": "azuredatalakestore.net",
    "keyvaultDns": ".vault.azure.net",
    "mariadbServerEndpoint": ".mariadb.database.azure.com",
    "mhsmDns": ".managedhsm.azure.net",
    "mysqlServerEndpoint": ".mysql.database.azure.com",
    "postgresqlServerEndpoint": ".postgres.database.azure.com",
    "sqlServerHostname": ".database.windows.net",
    "storageEndpoint": "core.windows.net",
    "storageSyncEndpoint": "afs.azure.net",
    "synapseAnalyticsEndpoint": ".dev.azuresynapse.net"
}
}

```

En el código anterior, puede establecer `authority` en el valor de la propiedad `endpoints.activeDirectory`, `endpoint` en el valor de la propiedad `endpoints.resourceManager` y `audience` en el valor de la propiedad `endpoints.activeDirectoryResourceId + ".default"`.

Para más información, consulte [Uso de la CLI de Azure con Azure Stack Hub](#) y [Obtención de información de autenticación para Azure Stack Hub](#).

Introducción a los aspectos básicos del SDK de Azure para Python

27/08/2025

La sección Aspectos básicos del SDK de Python para Azure equipa a los desarrolladores con los conceptos básicos y los comportamientos básicos que respaldan todas las bibliotecas cliente del SDK de Azure para Python. Estos temas se consideran "fundamentales" porque establecen los bloques de creación esenciales para el desarrollo eficaz, idiomático y resistente de aplicaciones en todos los servicios de Azure.

Independientemente de si está trabajando con reintentos HTTP, manejo de errores, comprensión de los tipos de respuesta del SDK o siguiendo las pautas de diseño lingüístico coherentes, estos artículos proporcionan el conocimiento de línea base necesario para navegar y ampliar con confianza el uso del SDK de Azure. Dominar estos aspectos básicos garantiza que no solo está escribiendo código funcional, sino también escribiendo código que sea fácil de mantener, sólido y alineado con los procedimientos recomendados en todo el ecosistema de Azure.

 Expandir tabla

Título del artículo	Propósito
Control de errores generados por el SDK de Azure para Python	Describe el modelo de errores completo del SDK, incluidos los procedimientos recomendados para controlar tipos de excepciones específicos e implementar estrategias resistentes de control de errores.
Canalización HTTP y reintentos en las bibliotecas de Azure SDK para Python	Proporciona una profundización en la canalización HTTP interna del SDK, en la que se muestra cómo las directivas como reintentos, registro y autenticación se superponen para administrar solicitudes y respuestas.
Descripción de los tipos de respuesta comunes en el SDK de Azure para Python	Explica cómo los métodos del SDK devuelven objetos de Python intuitivos y fuertemente tipados, lo que simplifica cómo se trabaja con las respuestas de Azure y las operaciones de larga duración.
Directrices de diseño de lenguaje del SDK de Azure para Python	Describe las convenciones y los patrones de diseño que se usan en el SDK para garantizar la coherencia, la facilidad de uso y la alineación con los procedimientos recomendados de Python.

Control de errores generados por el SDK de Azure para Python

27/08/2025

La creación de aplicaciones en la nube confiables requiere más que implementar características, lo que requiere estrategias sólidas de control de errores. Al trabajar con sistemas distribuidos y servicios en la nube, la aplicación debe estar preparada para controlar varios escenarios de error correctamente.

El SDK de Azure para Python proporciona un modelo de error completo diseñado para ayudar a los desarrolladores a crear aplicaciones resistentes. Comprender este modelo de error es fundamental para:

- Mejora de la confiabilidad de las aplicaciones anticipando y controlando escenarios de error comunes
- Mejora de la experiencia del usuario a través de mensajes de error significativos y degradación correcta
- Simplificación de la solución de problemas mediante la captura y el registro de información de diagnóstico pertinente

En este artículo se explora la arquitectura de errores del SDK de Azure para Python y se proporcionan instrucciones prácticas para implementar un control eficaz de errores en las aplicaciones.

Cómo se producen errores del SDK de Azure para Python

El SDK de Azure para Python usa un modelo de excepción jerárquico que proporciona funcionalidades generales y específicas de control de errores. En el núcleo de este modelo se encuentra `AzureError`, que actúa como clase de excepción base para todos los errores relacionados con el SDK de Azure.

Jerarquía de excepciones

```
AzureError
├── ClientAuthenticationError
├── ResourceNotFoundError
├── ResourceExistsError
└── ResourceModifiedError
```

ResourceNotModifiedError
ServiceRequestError
ServiceResponseError
HttpErrorResponse

Tipos de excepciones clave

 Expandir tabla

Error	Description
AzureError	Clase de excepción base para todos los errores del SDK de Azure. Úselo como un punto de referencia cuando necesite controlar cualquier error relacionado con Azure.
ClientAuthenticationError	Se genera cuando se produce un error en la autenticación. Entre las causas comunes se incluyen credenciales no válidas, tokens expirados y configuraciones de autenticación mal configuradas.
ResourceNotFoundError	Se genera al intentar acceder a un recurso que no existe. Normalmente corresponde a respuestas HTTP 404.
ResourceExistsError	Se genera al intentar crear un recurso que ya existe. Esto ayuda a evitar sobrescrituras accidentales.
ServiceRequestError	Se genera cuando el SDK no puede enviar una solicitud al servicio. Entre las causas comunes se incluyen problemas de conectividad de red, errores de resolución de DNS y puntos de conexión de servicio no válidos.
ServiceResponseError	Se genera cuando el servicio devuelve una respuesta inesperada que el SDK no puede procesar.
HttpErrorResponse	Se genera para las respuestas de error HTTP (códigos de estado 4xx y 5xx). Esta excepción proporciona acceso a los detalles de respuesta HTTP subyacentes.

Escenarios de error comunes

Comprender los escenarios de error típicos le ayuda a implementar estrategias de control adecuadas para cada situación.

Errores de autenticación y autorización

Los errores de autenticación se producen cuando el SDK no puede comprobar la identidad:

Python

```

from azure.core.exceptions import ClientAuthenticationError
from azure.identity import DefaultAzureCredential
from azure.storage.blob import BlobServiceClient

try:
    credential = DefaultAzureCredential()
    blob_service = BlobServiceClient(
        account_url="https://myaccount.blob.core.windows.net",
        credential=credential
    )
    # Attempt to list containers
    containers = blob_service.list_containers()
except ClientAuthenticationError as e:
    print(f"Authentication failed: {e.message}")
    # Don't retry - fix credentials first

```

Los errores de autorización (normalmente `HttpErrorResponse` con el estado 403) se producen cuando faltan permisos:

Python

```

from azure.core.exceptions import HttpResponseError

try:
    blob_client.upload_blob(data)
except HttpResponseError as e:
    if e.status_code == 403:
        print("Access denied. Check your permissions.")
    else:
        raise

```

Errores de recursos

Controle los recursos que faltan correctamente:

Python

```

from azure.core.exceptions import ResourceNotFoundError

try:
    blob_client = container_client.get_blob_client("myblob.txt")
    content = blob_client.download_blob().readall()
except ResourceNotFoundError:
    print("Blob not found. Using default content.")
    content = b"default"

```

Impedir la creación de recursos duplicados:

Python

```
from azure.core.exceptions import ResourceExistsError

try:
    container_client.create_container()
except ResourceExistsError:
    print("Container already exists.")
    # Continue with existing container
```

Errores de servidor

Controle correctamente los errores del lado servidor:

Python

```
from azure.core.exceptions import HttpResponseError

try:
    result = client.process_data(large_dataset)
except HttpResponseError as e:
    if 500 <= e.status_code < 600:
        print(f"Server error ({e.status_code}). The service may be temporarily
unavailable.")
        # Consider retry logic here
    else:
        raise
```

Procedimientos recomendados para el control de errores

- **Usar el control de excepciones específico** : detectar siempre excepciones específicas antes de revertir a las generales:

Python

```
from azure.core.exceptions import (
    AzureError,
    ClientAuthenticationError,
    ResourceNotFoundError,
    HttpResponseError
)

try:
    # Azure SDK operation
    result = client.get_resource()
except ClientAuthenticationError:
```

```

# Handle authentication issues
print("Please check your credentials")
except ResourceNotFoundError:
    # Handle missing resources
    print("Resource not found")
except HttpResponseError as e:
    # Handle specific HTTP errors
    if e.status_code == 429:
        print("Rate limited. Please retry later.")
    else:
        print(f"HTTP error {e.status_code}: {e.message}")
except AzureError as e:
    # Catch-all for other Azure errors
    print(f"Azure operation failed: {e}")

```

- Implementar estrategias de reintento adecuadas : algunos errores garantizan reintentos, mientras que otros no.

No vuelva a intentarlo:

- 401 No autorizado (errores de autenticación)
- 403 Prohibido (errores de autorización)
- 400 Solicitud incorrecta (errores de cliente)
- 404 No encontrado (a menos que espere que aparezca el recurso)

Considere la posibilidad de volver a intentarlo:

- 408 Tiempo de espera de solicitud
- 429 Demasiadas solicitudes (con el retroceso adecuado)
- Error interno del servidor 500
- 502 Puerta de enlace incorrecta
- 503 Servicio no disponible
- Tiempo de espera de puerta de enlace 504

- Extracción de información de error significativa

Python

```

from azure.core.exceptions import HttpResponseError

try:
    client.perform_operation()
except HttpResponseError as e:
    # Extract detailed error information
    print(f"Status code: {e.status_code}")
    print(f"Error message: {e.message}")
    print(f"Error code: {e.error.code if e.error else 'N/A'}")

    # Request ID is crucial for Azure support
    if hasattr(e, 'response') and e.response:

```

```
request_id = e.response.headers.get('x-ms-request-id')
print(f"Request ID: {request_id}")
```

Directivas de reintento y resistencia

El SDK de Azure incluye mecanismos de reintento integrados que controlan los errores transitorios automáticamente.

Comportamiento de reintento predeterminado

La mayoría de los clientes del SDK de Azure incluyen directivas de reintento predeterminadas que:

- Reintento en errores de conexión y códigos de estado HTTP específicos
- Uso del retroceso exponencial con vibración
- Limitar el número de reintentos

Personalización de directivas de reintento

Si el comportamiento predeterminado no se ajusta a su caso de uso, puede personalizar la directiva de reintento como en el ejemplo siguiente:

Python

```
from azure.storage.blob import BlobServiceClient
from azure.core.pipeline.policies import RetryPolicy

# Create a custom retry policy
retry_policy = RetryPolicy(
    retry_total=5, # Maximum retry attempts
    retry_backoff_factor=2, # Exponential backoff factor
    retry_backoff_max=60, # Maximum backoff time in seconds
    retry_on_status_codes=[408, 429, 500, 502, 503, 504]
)

# Apply to client
blob_service = BlobServiceClient(
    account_url="https://myaccount.blob.core.windows.net",
    credential=credential,
    retry_policy=retry_policy
)
```

Evitar el control de errores de red y tiempo de espera con bucles personalizados

Debe intentar usar reintentos integrados para errores de red y tiempo de espera antes de implementar su propia lógica personalizada.

Python

```
from azure.core.exceptions import ServiceRequestError
import time

# Avoid this approach if possible
max_retries = 3
retry_count = 0

while retry_count < max_retries:
    try:
        response = client.get_secret("mysecret")
        break
    except ServiceRequestError as e:
        retry_count += 1
        if retry_count >= max_retries:
            raise
        print(f"Network error. Retrying... ({retry_count}/{max_retries})")
        time.sleep(2 ** retry_count) # Exponential backoff
```

Implementación de patrones de disyuntor

Para las operaciones críticas, considere la posibilidad de implementar patrones de disyuntor:

Python

```
class CircuitBreaker:
    def __init__(self, failure_threshold=5, recovery_timeout=60):
        self.failure_threshold = failure_threshold
        self.recovery_timeout = recovery_timeout
        self.failure_count = 0
        self.last_failure_time = None
        self.state = 'closed' # closed, open, half-open

    def call(self, func, *args, **kwargs):
        if self.state == 'open':
            if time.time() - self.last_failure_time > self.recovery_timeout:
                self.state = 'half-open'
            else:
                raise Exception("Circuit breaker is open")

        try:
            result = func(*args, **kwargs)
            if self.state == 'half-open':
                self.state = 'closed'
                self.failure_count = 0
            return result
        except Exception as e:
```

```

        self.failure_count += 1
        self.last_failure_time = time.time()

    if self.failure_count >= self.failure_threshold:
        self.state = 'open'

    raise e

```

Descripción de los códigos y mensajes de error

Los servicios de Azure devuelven respuestas de error estructuradas que proporcionan información de depuración valiosa.

- Análisis de respuestas de error

Python

```

from azure.core.exceptions import HttpResponseError
import json

try:
    client.create_resource(resource_data)
except HttpResponseError as e:
    # Many Azure services return JSON error details
    if e.response and e.response.text():
        try:
            error_detail = json.loads(e.response.text())
            print(f"Error code: {error_detail.get('error', {}).get('code')}")
            print(f"Error message: {error_detail.get('error',
{}).get('message')}")

            # Some services provide additional details
            if 'details' in error_detail.get('error', {}):
                for detail in error_detail['error']['details']:
                    print(f" - {detail.get('code')}:"
{detail.get('message')}")
        except json.JSONDecodeError:
            print(f"Raw error: {e.response.text()}")

```

- Capturar información de diagnóstico : capture siempre la información de diagnóstico clave para solucionar problemas:

Python

```

import logging
from azure.core.exceptions import AzureError

logger = logging.getLogger(__name__)

```

```

try:
    result = client.perform_operation()
except AzureError as e:
    # Log comprehensive error information
    logger.error(
        "Azure operation failed",
        extra={
            'error_type': type(e).__name__,
            'error_message': str(e),
            'operation': 'perform_operation',
            'timestamp': datetime.utcnow().isoformat(),
            'request_id': getattr(e.response, 'headers', {}).get('x-ms-request-id') if hasattr(e, 'response') else None
        }
    )
    raise

```

- **Registro y diagnóstico:** habilite el registro de nivel de SDK para solucionar problemas detallados:

Python

```

import logging
import sys

# Configure logging for Azure SDKs
logging.basicConfig(level=logging.DEBUG)

# Enable HTTP request/response logging
logging.getLogger('azure.core.pipeline.policies.http_logging_policy').setLevel(logging.DEBUG)

# For specific services
logging.getLogger('azure.storage.blob').setLevel(logging.DEBUG)
logging.getLogger('azure.identity').setLevel(logging.DEBUG)

```

Para más información sobre el registro, consulte [Configuración del registro en las bibliotecas de Azure para Python](#).

- **Uso del seguimiento de red :** para la depuración profunda, habilite el seguimiento de nivel de red:

i Importante

El registro HTTP puede incluir información confidencial, como claves de cuenta en encabezados y otras credenciales. Asegúrese de proteger estos registros para evitar poner en peligro la seguridad.

Python

```
from azure.storage.blob import BlobServiceClient

# Enable network tracing
blob_service = BlobServiceClient(
    account_url="https://myaccount.blob.core.windows.net",
    credential=credential,
    logging_enable=True, # Enable logging
    logging_body=True     # Log request/response bodies (careful with
sensitive data)
)
```

Consideraciones especiales para la programación asincrónica

Cuando se usan clientes asincrónicos, el control de errores requiere especial atención.

- Control básico de errores asincrónicos

Python

```
import asyncio
from azure.core.exceptions import AzureError

async def get_secret_async(client, secret_name):
    try:
        secret = await client.get_secret(secret_name)
        return secret.value
    except ResourceNotFoundError:
        print(f"Secret '{secret_name}' not found")
        return None
    except AzureError as e:
        print(f"Error retrieving secret: {e}")
        raise
```

- Control de cancelaciones

Python

```
async def long_running_operation(client):
    try:
        result = await client.start_long_operation()
        # Wait for completion
        final_result = await result.result()
        return final_result
    except asyncio.CancelledError:
        print("Operation cancelled")
        # Cleanup if necessary
        if hasattr(result, 'cancel'):
            await result.cancel()
```

```
        raise
    except AzureError as e:
        print(f"Operation failed: {e}")
        raise
```

- Control simultáneo de errores

Python

```
async def process_multiple_resources(client, resource_ids):
    tasks = []
    for resource_id in resource_ids:
        task = client.get_resource(resource_id)
        tasks.append(task)

    results = []
    errors = []

    # Use gather with return_exceptions to handle partial failures
    outcomes = await asyncio.gather(*tasks, return_exceptions=True)

    for resource_id, outcome in zip(resource_ids, outcomes):
        if isinstance(outcome, Exception):
            errors.append((resource_id, outcome))
        else:
            results.append(outcome)

    # Process successful results and errors appropriately
    if errors:
        print(f"Failed to process {len(errors)} resources")
        for resource_id, error in errors:
            print(f" - {resource_id}: {error}")

    return results
```

Resumen de los procedimientos recomendados

El control de errores efectivo en el SDK de Azure para aplicaciones de Python requiere:

- **Anticipación de errores:** las aplicaciones en la nube deben esperar y controlar los errores parciales correctamente.
- **Use un control de excepciones específico:** capture excepciones específicas como ResourceNotFoundError y ClientAuthenticationError antes de revertir al control general de AzureError.
- **Implementar lógica de reintento inteligente:** use directivas de reintento integradas o personalícelas en función de sus necesidades. Recuerde que no todos los errores deben desencadenar reintentos.

- **Capturar información de diagnóstico:** los identificadores de solicitud de registro siempre, los códigos de error y las marcas de tiempo para una solución de problemas eficaz.
- **Proporcionar comentarios significativos sobre los usuarios:** transforme los errores técnicos en mensajes fáciles de usar al tiempo que conserva los detalles técnicos para obtener soporte técnico.
- **Escenarios de error** de prueba: incluya el control de errores en la cobertura de pruebas para asegurarse de que la aplicación se comporta correctamente en condiciones de error.

Pasos siguientes

- Referencia del módulo de excepciones de Azure Core
- Más información sobre [la solución de problemas de autenticación y autorización](#)
- Exploración de [OpenTelemetry](#) de Azure Monitor para una supervisión completa de aplicaciones

Descripción de la canalización HTTP y los reintentos en el SDK de Azure para Python

27/08/2025

Al realizar una llamada a cualquier servicio de Azure mediante el SDK de Azure para Python (ya sea Blob Storage, Key Vault, Cosmos DB o cualquier otro servicio basado en HTTP), la solicitud no va directamente al servicio de Azure. En su lugar, fluye a través de una canalización HTTP sofisticada que controla los problemas críticos transversales automáticamente.

Comprender cómo funciona la canalización HTTP es esencial para crear aplicaciones sólidas y eficaces. La canalización gestiona los reintentos para fallos transitorios, maneja la autenticación, ofrece capacidad de registro y permite agregar un comportamiento personalizado cuando sea necesario. Este conocimiento le ayuda a depurar problemas de rendimiento, optimizar la resistencia y personalizar la interacción de la aplicación con los servicios de Azure.

¿Qué es la canalización HTTP?

El SDK de Azure para Python usa una arquitectura de canalización HTTP interna para procesar todas las solicitudes y respuestas. Esta canalización consta de una serie de directivas que se ejecutan en secuencia, cada una de las cuales es responsable de un aspecto específico de la comunicación HTTP. Piense en la canalización como una cadena de pasos de procesamiento:

ascii

```
Client Request → Retry Policy → Authentication Policy → Logging Policy → HTTP  
Transport → Azure Service
```

↓

```
Client Response ← Retry Policy ← Authentication Policy ← Logging Policy ← HTTP  
Transport ← Response
```

Cada política de la canalización puede:

- Modificar la solicitud antes de enviarla
- Procesar la respuesta después de recibirla
- Realizar acciones como reintentar solicitudes con errores
- Adición de encabezados, información de registro o implementación de lógica personalizada

Políticas clave en proceso

El SDK de Azure para Python incluye varias directivas integradas que controlan escenarios comunes:

- **RetryPolicy**: vuelven a intentarse automáticamente las solicitudes que han fallado debido a errores transitorios. Esta directiva implementa lógica de reintento inteligente con retroceso exponencial para evitar sobrecargar los servicios durante las interrupciones.
- **BearerTokenCredentialPolicy**: administra la autenticación mediante la adquisición y actualización automática de tokens de acceso. Esta directiva garantiza que las solicitudes incluyan credenciales de autenticación válidas sin administración manual de tokens.
- **NetworkTraceLoggingPolicy**: captura información detallada sobre las solicitudes y respuestas HTTP con fines de depuración. Esta directiva es inestimable al solucionar problemas de comunicación.
- **HttpTransport**: la capa más baja de la canalización que realmente envía solicitudes HTTP a través de la red. En el SDK de Azure para Python, normalmente se implementa mediante solicitudes o aiohttp para operaciones asincrónicas.

Directivas adicionales

- **RedirectPolicy**: controla automáticamente las redirecciones HTTP.
- **DistributedTracingPolicy**: se integra con sistemas de seguimiento distribuidos para la supervisión
- **ProxyPolicy**: enruta las solicitudes a través de servidores proxy HTTP cuando se configuran
- **UserAgentPolicy**: agrega información de versión del SDK para solicitar encabezados

Comportamiento de reintento

El SDK de Azure para Python implementa lógica de reintento inteligente para controlar los errores transitorios automáticamente. Comprender este comportamiento le ayuda a crear aplicaciones más resistentes.

Condiciones que se reintentan automáticamente

El SDK reintenta las solicitudes de estos códigos de estado HTTP:

- **408 Tiempo de espera de solicitud**: el servidor agota el tiempo de espera de la solicitud.
- **429 Demasiadas solicitudes**: la limitación de velocidad está en vigor
- **500 Error interno del servidor**: problema temporal del servidor
- **502 Puerta de enlace incorrecta**: problema de red temporal
- **503 Servicio no disponible**: Servicio temporalmente no disponible

- 504 Tiempo de espera de la puerta de enlace: tiempo de espera de la puerta de enlace o del proxy

Configuración de reinicio predeterminada

La configuración de reinicio predeterminada proporciona un buen equilibrio entre resistencia y rendimiento:

- Número máximo de reinicios: 3
- Modo de reinicio: retroceso exponencial
- Retraso base: 0,8 segundos
- Retraso máximo: 60 segundos
- Tiempo de reinicio total máximo: 120 segundos

El cálculo de retroceso exponencial sigue este patrón:

Python

```
delay = min(base_delay * (2 ** retry_attempt), max_delay)
```

Personalización de reinicios

Puede personalizar el comportamiento de reinicio al crear clientes del SDK para que coincidan con los requisitos específicos de la aplicación.

Python

```
from azure.storage.blob import BlobServiceClient
from azure.core.pipeline.policies import RetryPolicy

# Custom retry configuration
retry_policy = RetryPolicy(
    retry_total=5,                      # Maximum number of retry attempts
    retry_backoff_factor=0.5,            # Base backoff time in seconds
    retry_backoff_max=120,              # Maximum backoff time in seconds
    retry_on_status_codes=[429, 500, 502, 503, 504] # HTTP status codes to retry
)

# Apply custom retry policy to client
client = BlobServiceClient(
    account_url="https://myaccount.blob.core.windows.net",
    credential=credential,
    retry_policy=retry_policy
)
```

Deshabilitar reintentos

En escenarios en los que los reintentos no son adecuados:

Python

```
from azure.core.pipeline.policies import RetryPolicy

# Disable retries completely
no_retry_policy = RetryPolicy(retry_total=0)

client = BlobServiceClient(
    account_url="https://myaccount.blob.core.windows.net",
    credential=credential,
    retry_policy=no_retry_policy
)
```

Análisis y depuración del comportamiento de reintento

Comprender cuándo y por qué se producen los reintentos es fundamental para solucionar problemas de rendimiento.

Habilitación del registro del SDK

El SDK de Azure para Python usa el marco de registro estándar de Python:

Python

```
import logging
import sys

# Configure logging to see retry attempts
logging.basicConfig(
    level=logging.DEBUG,
    format='%(asctime)s - %(name)s - %(levelname)s - %(message)s',
    stream=sys.stdout
)

# Enable specific Azure SDK loggers
azure_logger = logging.getLogger('azure')
azure_logger.setLevel(logging.DEBUG)

# Now SDK operations will log retry attempts
```

Identificación de patrones de reintento

Busque entradas de registro como:

Resultados

```
Retry attempt 1 for request [GET]
https://myaccount.blob.core.windows.net/container/blob
Waiting 0.8 seconds before retry
```

Problemas comunes de reintento

- Reintentar errores no transitorios:** el SDK no reintenta los errores de cliente (4xx), excepto 408 y 429.
- Omitir la latencia de reintento:** recuerde que los reintentos agregan latencia a las operaciones con errores.
- Tiempo de espera insuficiente:** asegúrate de que el tiempo de espera general de la operación tenga en cuenta los retrasos de reintento.

Avanzado: Adición de directivas personalizadas

Puede ampliar el flujo de trabajo con políticas personalizadas para escenarios especializados.

Creación de una directiva personalizada

Python

```
from azure.core.pipeline import PipelineRequest, PipelineResponse
from azure.core.pipeline.policies import HTTPPolicy
from typing import Any, Optional

class CustomTelemetryPolicy(HTTPPolicy):
    """Custom policy to add telemetry headers"""

    def send(self, request: PipelineRequest) -> PipelineResponse:
        # Add custom header before sending request
        request.http_request.headers['X-Custom-Telemetry'] = 'my-app-v1.0'

        # Continue with the pipeline
        response = self.next.send(request)

        # Log response time
        print(f"Request to {request.http_request.url} completed")
```

```
    return response
```

Aplicación de directivas personalizadas

Python

```
from azure.storage.blob import BlobServiceClient

# Create client with custom policy
client = BlobServiceClient(
    account_url="https://myaccount.blob.core.windows.net",
    credential=credential,
    per_call_policies=[CustomTelemetryPolicy()], # Policies that run per request
    per_retry_policies=[] # Policies that run per retry attempt
)
```

Ordenación de directivas

Las directivas se ejecutan en un orden específico:

- Directivas por llamada (se ejecutan una vez por operación)
- Directiva de reinicio
- Directivas por reinicio (se ejecutan en cada intento)
- Directiva de autenticación
- Transporte HTTP

procedimientos recomendados

Usar la configuración predeterminada siempre que sea posible

La configuración de reinicio predeterminada funciona bien para la mayoría de los escenarios. Personalice solo cuando tenga requisitos específicos.

Directrices de personalización

Al personalizar el comportamiento de reinicio:

- **Emplear retroceso exponencial:** Previene que los servicios se saturen durante la recuperación

- **Establecer límites razonables:** Límite total del tiempo de reintento para evitar la espera indefinida
- **Monitoreo de métricas de reintento:** Monitorear las tasas de reintento en producción para identificar problemas.
- **Considere los cortacircuitos:** para escenarios de gran volumen, implemente patrones de cortacircuito

Qué no volver a intentar

Evite reintentar estos tipos de errores:

- **Errores de autenticación (401, 403):** los errores de autenticación requieren corregir credenciales, no reintentar
- **Errores de cliente (400, 404):** los errores de cliente indican problemas con la propia solicitud
- **Errores de lógica de negocios:** errores específicos de la aplicación que no se resuelven con reintentos

Excelencia operativa

- **Identificadores de correlación de registro:** incluir `x-ms-client-request-id` en los registros para la compatibilidad con Azure
- **Establecer tiempos de espera adecuados:** equilibrio entre la confiabilidad y la experiencia del usuario
- **Comportamiento de reintento de prueba:** verifique que su aplicación maneja los reintentos correctamente.
- **Supervisar el rendimiento:** haga seguimiento de las latencias de P95/P99 (métricas de latencia basadas en percentiles), incluido el coste de reintento.

Pasos siguientes

- [Implementación de aplicaciones resistentes](#)

Descripción de los tipos de respuesta comunes en el SDK de Azure para Python

27/08/2025

El SDK de Azure para Python abstrae las llamadas al protocolo de comunicación del servicio de Azure subyacente, tanto si ese protocolo es HTTP como AMQP (que se usa para los SDK de mensajería como ServiceBus, EventHubs, etc.). Por ejemplo, si usa una de las bibliotecas que usa HTTP, El SDK de Azure para Python realiza solicitudes HTTP y recibe respuestas HTTP en segundo plano. El SDK abstrae esta complejidad, lo que le permite trabajar con objetos intuitivos de Python en lugar de respuestas HTTP sin procesar o cargas JSON.

Comprender los tipos de objetos que recibe de las operaciones del SDK es esencial para escribir aplicaciones eficaces de Azure. En este artículo se explican los tipos de respuesta comunes que se encuentran y cómo se relacionan con la comunicación HTTP subyacente.

⚠ Nota

En este artículo solo se examina el escenario HTTP, no el escenario de AMQP.

Objetos de Python deserializados

El SDK de Azure para Python prioriza la productividad del desarrollador devolviendo objetos de Python fuertemente tipados a partir de operaciones de servicio. En lugar de analizar JSON o controlar directamente los códigos de estado HTTP, se trabaja con modelos de recursos que representan recursos de Azure como objetos de Python.

Por ejemplo, al recuperar un blob de Azure Storage, recibirá un objeto BlobProperties con atributos como nombre, tamaño y last_modified, en lugar de un diccionario JSON sin formato:

Python

```
from azure.storage.blob import BlobServiceClient

# Connect to storage account
blob_service_client = BlobServiceClient.from_connection_string(connection_string)
container_client = blob_service_client.get_container_client("mycontainer")

# Get blob properties - returns a BlobProperties object
blob_client = container_client.get_blob_client("myblob.txt")
properties = blob_client.get_blob_properties()

# Access properties as Python attributes
print(f"Blob name: {properties.name}")
```

```
print(f"Blob size: {properties.size} bytes")
print(f"Last modified: {properties.last_modified}")
```

Dónde proceden los datos

Comprender el flujo de datos le ayuda a apreciar lo que hace el SDK en segundo plano:

- El código llama a un método sdk: se invoca un método como `get_blob_properties()`
- El SDK crea una solicitud HTTP: el SDK compila la solicitud HTTP adecuada con encabezados, autenticación y parámetros de consulta.
- Respuesta del servicio de Azure: el servicio devuelve una respuesta HTTP, normalmente con una carga JSON en el cuerpo de la respuesta.
- El SDK procesa la respuesta: el SDK:
 - Comprueba el código de estado HTTP.
 - Analiza el cuerpo de la respuesta (normalmente JSON)
 - Valida los datos con los esquemas esperados.
 - Asigna los datos a objetos de modelo de Python
- El código recibe objetos de Python: trabaja con los objetos deserializados, no con datos HTTP sin procesar.

Esta abstracción le permite centrarse en la lógica de la aplicación en lugar de en los detalles del protocolo HTTP.

Tipos de respuesta comunes

El SDK de Azure para Python usa varios tipos de respuesta estándar en todos los servicios. Comprender estos tipos le ayuda a trabajar de forma eficaz con cualquier servicio de Azure.

Modelos de recursos

La mayoría de las operaciones del SDK devuelven modelos de recursos: objetos de Python que representan recursos de Azure. Estos modelos son específicos del servicio, pero siguen patrones coherentes:

Python

```
# Azure Key Vault example
from azure.keyvault.secrets import SecretClient

secret_client = SecretClient(vault_url=vault_url, credential=credential)
secret = secret_client.get_secret("mysecret") # Returns KeyVaultSecret

print(f"Secret name: {secret.name}")
```

```

print(f"Secret value: {secret.value}")
print(f"Secret version: {secret.properties.version}")

# Azure Cosmos DB example
from azure.cosmos import CosmosClient

cosmos_client = CosmosClient(url=cosmos_url, credential=credential)
database = cosmos_client.get_database_client("mydatabase")
container = database.get_container_client("mycontainer")
item = container.read_item(item="item-id", partition_key="partition-value") # Returns dict

print(f"Item ID: {item['id']}")

```

ItemPaged para los resultados de la colección

Al enumerar recursos, el SDK devuelve `ItemPaged` objetos que controlan la paginación de forma transparente:

Python

```

from azure.storage.blob import BlobServiceClient
from azure.core.paging import ItemPaged

blob_service_client = BlobServiceClient.from_connection_string(connection_string)
container_client = blob_service_client.get_container_client("mycontainer")

# list_blobs returns ItemPaged[BlobProperties]
blobs: ItemPaged[BlobProperties] = container_client.list_blobs()

# Iterate naturally - SDK handles pagination
for blob in blobs:
    print(f"Blob: {blob.name}, Size: {blob.size}")

```

Acceso a la respuesta HTTP sin procesar

Aunque las abstracciones de alto nivel del SDK satisfacen la mayoría de las necesidades, a veces necesita acceso a la respuesta HTTP subyacente. Entre los escenarios habituales se incluyen los siguientes:

- Depuración de solicitudes con errores
- Acceso a encabezados de respuesta personalizados
- Implementación de lógica de reintento personalizada
- Trabajar con formatos de respuesta no estándar

La mayoría de los métodos del SDK aceptan un `raw_response_hook` parámetro:

Python

```
from azure.keyvault.secrets import SecretClient

secret_client = SecretClient(vault_url=vault_url, credential=credential)

def inspect_response(response):
    # Access the raw HTTP response
    print(f"Request URL: {response.http_request.url}")
    print(f"Status code: {response.http_response.status_code}")
    print(f"Response headers: {dict(response.http_response.headers)}")

    # Access custom headers
    request_id = response.http_response.headers.get('x-ms-request-id')
    print(f"Request ID: {request_id}")

    # Must return the response
    return response

# Hook is called before deserialization
secret = secret_client.get_secret("mysecret", raw_response_hook=inspect_response)
```

Paginación e iteradores

Los servicios de Azure suelen devolver grandes colecciones de recursos. El SDK usa ItemPaged para controlar estas colecciones de forma eficaz sin cargar todo en memoria a la vez.

Paginación automática

El SDK captura automáticamente las páginas nuevas a medida que recorre en iteración:

Python

```
# List all blobs - could be thousands
blobs = container_client.list_blobs()

# SDK fetches pages as needed during iteration
for blob in blobs:
    process_blob(blob) # Pages loaded on-demand
```

Trabajar con páginas explícitamente

También puede trabajar con páginas directamente cuando sea necesario:

Python

```
blobs = container_client.list_blobs()

# Process by page
for page in blobs.by_page():
    print(f"Processing page with {len(list(page))} items")
    for blob in page:
        process_blob(blob)
```

Control del tamaño de página

Muchas operaciones de lista aceptan un parámetro results_per_page:

Python

```
# Fetch 100 items per page instead of the default
blobs = container_client.list_blobs(results_per_page=100)
```

Algunos métodos para algunos servicios de Azure tienen otros mecanismos para controlar el tamaño de página. Por ejemplo, KeyVault y Azure Search usan `top` el kwarg para limitar los resultados por llamada. Consulte el [código fuente](#) ↗ del método de `search()` Azure Search como ejemplo.

Caso especial: operaciones y sondeos de larga duración

Algunas operaciones de Azure no se pueden completar inmediatamente. Algunos ejemplos son:

- Creación o eliminación de máquinas virtuales
- Implementación de plantillas de ARM
- Entrenamiento de modelos de Aprendizaje automático
- Copia de blobs grandes

Estas operaciones devuelven objetos de sondeo que realizan un seguimiento del progreso de la operación.

Trabajar con sondeos

Python

```
from azure.mgmt.storage import StorageManagementClient
```

```

storage_client = StorageManagementClient(credential, subscription_id)

# Start storage account creation
poller = storage_client.storage_accounts.begin_create(
    resource_group_name="myresourcegroup",
    account_name="mystorageaccount",
    parameters=storage_parameters
)

# Option 1: Wait for completion (blocking)
storage_account = poller.result()

# Option 2: Check status periodically
while not poller.done():
    print(f"Status: {poller.status()}")
    time.sleep(5)

storage_account = poller.result()

```

Sondeos asincrónicos

Cuando se usan patrones asincrónicos o await, se trabaja con `AsyncLROPoller`:

Python

```

from azure.storage.blob.aio import BlobServiceClient

async with BlobServiceClient.from_connection_string(connection_string) as client:
    container_client = client.get_container_client("mycontainer")

    # Start async copy operation
    blob_client = container_client.get_blob_client("large-blob.vhd")
    poller = await blob_client.begin_copy_from_url(source_url)

    # Wait for async completion
    copy_properties = await poller.result()

```

Ejemplo de sondeo de objetos para operaciones de larga duración: Máquinas virtuales

Implementación de máquinas virtuales en un ejemplo de una operación que tarda tiempo en completarse y controlarla devolviendo objetos de sondeo (LROPoller para código sincrónico, AsyncLROPoller para código asincrónico):

Python

```

from azure.mgmt.compute import ComputeManagementClient
from azure.core.polling import LROPoller

```

```

compute_client = ComputeManagementClient(credential, subscription_id)

# Start VM creation - returns immediately with a poller
poller: LROPoller = compute_client.virtual_machines.begin_create_or_update(
    resource_group_name="myresourcegroup",
    vm_name="myvm",
    parameters=vm_parameters
)

# Wait for completion and get the result
vm = poller.result() # Blocks until operation completes
print(f"VM {vm.name} provisioned successfully")

```

Acceso a la respuesta para los resultados paginados

Para los resultados paginados, use el `by_page()` método con `raw_response_hook`:

Python

```

def page_response_hook(response):
    continuation_token = response.http_response.headers.get('x-ms-continuation')
    print(f"Continuation token: {continuation_token}")
    return response

blobs = container_client.list_blobs()
for page in blobs.by_page(raw_response_hook=page_response_hook):
    for blob in page:
        print(blob.name)

```

procedimientos recomendados

- Preferir abstracciones de alto nivel
- Trabaje con los modelos de recursos del SDK en lugar de las respuestas sin procesar siempre que sea posible y evite tener acceso a cualquier método con un carácter de subrayado `_`, ya que, por convención, son privados en Python. No hay ninguna garantía sobre los cambios importantes, etc. en comparación con las API públicas:

Python

```

# Preferred: Work with typed objects
secret = secret_client.get_secret("mysecret")
if secret.properties.enabled:
    use_secret(secret.value)

# Avoid: Manual JSON parsing (unless necessary) ...

```

```

# AND avoid accessing any objects or methods that start with `_
response = secret_client._client.get(...) # Don't access internal clients
data = json.loads(response.text)
if data['attributes']['enabled']:
    use_secret(data['value'])

```

- **Controlar correctamente la paginación** : siempre recorre en iteración los resultados paginados en lugar de convertir en una lista:

Python

```

# Good: Memory-efficient iteration
for blob in container_client.list_blobs():
    process_blob(blob)

# Avoid: Loading everything into memory
all_blobs = list(container_client.list_blobs()) # Could consume excessive
memory

```

- **Uso `poller.result()` para operaciones de larga duración** : use siempre el `result()` método para asegurarse de que las operaciones se completen correctamente:

Python

```

# Correct: Wait for operation completion
poller = compute_client.virtual_machines.begin_delete(
    resource_group_name="myresourcegroup",
    vm_name="myvm"
)
poller.result() # Ensures deletion completes
print("VM deleted successfully")

# Wrong: Assuming immediate completion
poller = compute_client.virtual_machines.begin_delete(...)
print("VM deleted successfully") # Deletion might still be in progress!

```

- **Acceso a respuestas sin procesar solo cuando sea necesario** : use el acceso a la respuesta sin procesar con moderación y solo para requisitos específicos:

Python

```

# Good use case: Debugging or logging
def log_request_id(response):
    request_id = response.http_response.headers.get('x-ms-request-id')
    logger.info(f"Operation request ID: {request_id}")
    return response

blob_client.upload_blob(data, raw_response_hook=log_request_id)

```

```
# Good use case: Custom error handling
def check_custom_header(response):
    if response.http_response.headers.get('x-custom-error'):
        raise CustomApplicationError("Custom error condition detected")
    return response
```

Directrices de diseño de lenguaje del SDK de Azure para Python

27/08/2025

Las directrices de diseño del SDK de Azure son estándares completos que garantizan la coherencia, la previsibilidad y la facilidad de uso en todos los SDK de Azure. Estas directrices ayudan a los desarrolladores a trabajar de forma eficaz con los servicios de Azure proporcionando patrones y comportamientos conocidos en distintos servicios y lenguajes de programación.

Las directrices constan de dos categorías:

- **Directrices generales:** principios básicos que se aplican a todos los SDK de Azure independientemente del lenguaje de programación
- **Language-Specific Directrices:** detalles de implementación optimizados para cada lenguaje admitido, incluido [Python](#), [.NET](#), [Java](#), [TypeScript](#) y muchos más (consulte la tabla de contenido a partir de la página [Directrices generales: Introducción](#)).

Estas directrices se desarrollan abiertamente en GitHub, lo que permite la revisión y la contribución de la comunidad.

Principios generales de diseño

Todos los SDK de Azure siguen estos principios fundamentales:

 Expandir tabla

Principio	Description
Uso idiomático	Los SDK siguen convenciones y patrones específicos del lenguaje
Uniformidad	Comportamientos uniformes en distintos servicios de Azure
Simplicidad	Las tareas comunes requieren código mínimo
Divulgación progresiva	Las características avanzadas están disponibles, pero no complican el uso básico
Robustez	Control integrado de errores, reintentos y tiempos de espera

Directrices específicas de Python

El resto de este documento se centrará en las [directrices de Python](#).

Convenciones de nomenclatura

Los SDK de Azure para Python siguen las convenciones de nomenclatura estándar de Python:

- **Métodos** : use snake_case.

```
Python
```

```
list_containers()  
get_secret()  
create_database()
```

- **Variables** : use snake_case.

```
Python
```

```
connection_string = "..."  
retry_count = 3
```

- **Clases** : use PascalCase.

```
Python
```

```
BlobServiceClient  
SecretClient  
CosmosClient
```

- **Constantes** : use UPPER_CASE.

```
Python
```

```
DEFAULT_CHUNK_SIZE  
MAX_RETRIES
```

Estructura del paquete

Los paquetes del SDK de Azure siguen una estructura coherente:

```
ascii
```

```
azure-<service>-<feature>  
|__ azure/  
|   |__ <service>/
```

```
|   └── __init__.py  
|   └── _client.py  
|   └── _models.py  
└── aio/           # Async implementations  
    └── __init__.py
```

Creación de instancias de cliente

Los clientes proporcionan varios métodos de creación de instancias:

Python

```
from azure.storage.blob import BlobServiceClient  
from azure.identity import DefaultAzureCredential  
  
# Note: do not use connection string if you can possibly avoid it!  
  
# Using account URL and credential  
credential = DefaultAzureCredential()  
client = BlobServiceClient(account_url="https://account.blob.core.windows.net",  
                           credential=credential)
```

Autenticación

Los SDK de Azure usan patrones de autenticación coherentes:

Python

```
from azure.identity import DefaultAzureCredential, ClientSecretCredential  
  
# Default credential chain  
credential = DefaultAzureCredential()  
  
# Explicit credential  
credential = ClientSecretCredential(  
    tenant_id="tenant-id",  
    client_id="client-id",  
    client_secret="secret"  
)
```

Administradores de contexto

La mayoría de los clientes del SDK de Azure implementan protocolos de administrador de contexto para la limpieza automática de recursos:

Python

```
from azure.storage.blob import BlobServiceClient

# Automatic cleanup with context manager
with BlobServiceClient.from_connection_string(conn_str) as client:
    container_client = client.get_container_client("mycontainer")
    blob_list = container_client.list_blobs()
```

! Nota

Aunque la mayoría de los clientes admiten administradores de contexto, compruebe la documentación específica del cliente para obtener disponibilidad.

Operaciones asincrónicas

Los clientes asincrónicos se proporcionan en módulos independientes `.aio` :

Python

```
from azure.storage.blob.aio import BlobServiceClient
import asyncio

async def list_blobs_async():
    async with BlobServiceClient.from_connection_string(conn_str) as client:
        container_client = client.get_container_client("mycontainer")
        async for blob in container_client.list_blobs():
            print(blob.name)

# Run async function
asyncio.run(list_blobs_async())
```

Operaciones de larga duración

Las operaciones de larga duración usan el prefijo `begin_` y devuelven objetos de sondeo:

Python

```
from azure.storage.blob import BlobServiceClient

client = BlobServiceClient.from_connection_string(conn_str)
container_client = client.get_container_client("mycontainer")

# Start long-running operation
poller = container_client.begin_copy_blob_from_url(source_url)
```

```
# Wait for completion
result = poller.result()

# Or check status
if poller.done():
    result = poller.result()
```

Paginación

Las operaciones de lista devuelven iterables que controlan la paginación automáticamente:

Python

```
from azure.storage.blob import BlobServiceClient

client = BlobServiceClient.from_connection_string(conn_str)

# Automatic pagination
for container in client.list_containers():
    print(container.name)

# Manual pagination control
containers = client.list_containers(results_per_page=10).by_page()
for page in containers:
    for container in page:
        print(container.name)
```

Tipos de retorno

Los métodos devuelven objetos de modelo fuertemente tipados en lugar de diccionarios:

Python

```
from azure.keyvault.secrets import SecretClient

client = SecretClient(vault_url="...", credential=credential)

# Returns KeyVaultSecret object, not dict
secret = client.get_secret("my-secret")
print(secret.value)
print(secret.properties.created_on)
```

Control de errores

Las excepciones del SDK de Azure heredan de AzureError y proporcionan tipos de excepciones específicos:

Python

```
from azure.core.exceptions import (
    AzureError,
    ResourceNotFoundError,
    ResourceExistsError,
    ClientAuthenticationError,
    HttpResponseError
)

try:
    blob_client.download_blob()
except ResourceNotFoundError:
    # Handle missing resource
    print("Blob not found")
except ClientAuthenticationError:
    # Handle authentication failure
    print("Authentication failed")
except HttpResponseError as e:
    # Handle HTTP errors
    print(f"HTTP {e.status_code}: {e.message}")
except AzureError as e:
    # Handle any other Azure SDK error
    print(f"Azure SDK error: {e}")
```

Opciones de configuración

Los clientes aceptan la configuración mediante argumentos de palabra clave:

Python

```
from azure.storage.blob import BlobServiceClient

client = BlobServiceClient(
    account_url="...",
    credential=credential,
    # Configuration options
    max_single_put_size=64 * 1024 * 1024,
    max_block_size=4 * 1024 * 1024,
    retry_total=3,
    logging_enable=True
)
```

Patrones comunes del SDK

Jerarquía de cliente de servicio

Los SDK de Azure suelen seguir una jerarquía de tres niveles:

- **Cliente de servicio**: punto de entrada para las operaciones de servicio

```
Python
```

```
service_client = BlobServiceClient(...)
```

- **Cliente de recursos**: operaciones en recursos específicos

```
Python
```

```
container_client = service_client.get_container_client("container")
```

- **Métodos de operación**: acciones en recursos

```
Python
```

```
blob_client = container_client.get_blob_client("blob.txt")
blob_client.upload_blob(data)
```

Nomenclatura de método coherente

Los nombres de método siguen patrones de predicción:

 Expandir tabla

Operation	Patrón de método	Example
Create	<code>create_<resource></code>	<code>create_container()</code>
Read	<code>get_<resource></code>	<code>get_blob()</code>
Update	<code>update_<resource></code>	<code>update_secret()</code>
Delete	<code>delete_<resource></code>	<code>delete_container()</code>
List	<code>list_<resources></code>	<code>list_blobs()</code>
Exists	<code>exists()</code>	<code>blob_client.exists()</code>

Uso de SDK de Azure

En el ejemplo siguiente se muestra cómo las directrices de diseño crean coherencia en distintos servicios de Azure:

Python

```
from azure.storage.blob import BlobServiceClient
from azure.keyvault.secrets import SecretClient
from azure.cosmos import CosmosClient
from azure.identity import DefaultAzureCredential

# Consistent authentication
credential = DefaultAzureCredential()

# Consistent client instantiation
blob_service = BlobServiceClient(
    account_url="https://account.blob.core.windows.net",
    credential=credential
)
secret_client = SecretClient(
    vault_url="https://vault.vault.azure.net",
    credential=credential
)
cosmos_client = CosmosClient(
    url="https://account.documents.azure.com",
    credential=credential
)

# Consistent method patterns
containers = blob_service.list_containers()
secrets = secret_client.list_properties_of_secrets()
databases = cosmos_client.list_databases()

# Consistent error handling
from azure.core.exceptions import ResourceNotFoundError

try:
    blob_service.get_container_client("container").get_container_properties()
    secret_client.get_secret("secret")
    cosmos_client.get_database_client("database").read()
except ResourceNotFoundError as e:
    print(f"Resource not found: {e}")
```

Contribución a los SDK de Azure

Al extender o contribuir a los SDK de Azure:

- **Siga las expresiones de Python** : uso de patrones y convenciones de Python
- **Mantener la coherencia** : alineación con los patrones de SDK existentes
- **Escritura de pruebas completas** : incluir pruebas unitarias e de integración
- **Documento a fondo** : proporcione docstrings y ejemplos

- **Directrices de revisión** : consulte la Guía de contribución del SDK de Azure.

Este es un ejemplo de implementación de un método de cliente personalizado que sigue las Directrices de diseño de lenguaje.

Python

```
def list_items_with_prefix(self, prefix: str, **kwargs) ->
    ItemPaged[ItemProperties]:
    """List items that start with the specified prefix.

    :param str prefix: The prefix to filter items
    :return: An iterable of ItemProperties
    :rtype: ~azure.core.paging.ItemPaged[ItemProperties]

    :example:
        items = client.list_items_with_prefix("test-")
        for item in items:
            print(item.name)
    """
    return self.list_items(name_starts_with=prefix, **kwargs)
```

Pasos siguientes

- Revisión de las [directrices de diseño](#) completas del SDK de Azure
- Lea la [página Versiones del SDK de Azure](#).

Índice de paquetes de las bibliotecas de Azure

11/08/2025

Los paquetes del SDK de Python de Azure se publican en [PyPI](#), incluidas las versiones beta marcadas con "b" en el número de versión (por ejemplo, 1.0.0b1). Para conocer las versiones más recientes y el historial de versiones, consulte Las [versiones del SDK de Azure: Python](#).

Si busca ayuda con el uso de un paquete de SDK específico, tiene varias opciones:

- **Código fuente y ejemplos:** en el repositorio de GitHub del SDK de Azure para Python, cada paquete tiene su propia carpeta. Para ver su código, seleccione el vínculo de GitHub en la columna "Origen" del índice del paquete. La mayoría de los repositorios incluyen un archivo de README.md con un uso de ejemplo.
- **Documentación de referencia de API:** para ver la referencia de API, seleccione el vínculo Docs de la misma tabla. Estas páginas proporcionan información general sobre el paquete, sus clases y sus métodos disponibles.
- **Tutoriales e instrucciones para desarrolladores:** para explorar tutoriales, guías paso a paso e instrucciones de instalación, visite la documentación para [desarrolladores de Azure para Python](#). Por ejemplo, para obtener información sobre cómo instalar paquetes de SDK, consulte : [Instalación de paquetes de biblioteca de Azure para Python](#).

! Nota

La columna **Nombre** incluye un nombre descriptivo para cada paquete. Para buscar el nombre que necesita usar para instalar el paquete con [pip](#), use los vínculos de las columnas **Paquete (Package)**, **Documentos (Docs)** u **Origen (Source)**. Por ejemplo, la columna **Nombre (Name)** del paquete de Azure Blob Storage es "Blobs" mientras que el nombre del paquete es *azure-storage-blob*.

En el caso de las bibliotecas de Conda, consulte el [canal de Microsoft en anaconda.org](#).

Bibliotecas que usan azure.core

 Expandir tabla

Name	Package	Docs	Source
Agentes de IA	PyPI 1.1.0 ↗ PyPI 1.2.0b2 ↗	docs	GitHub 1.1.0 ↗ GitHub 1.2.0b2 ↗
Evaluación de IA	PyPI 1.10.0 ↗	docs	GitHub 1.10.0 ↗
IA generativa	PyPI 1.0.0b11 ↗		GitHub 1.0.0b11 ↗
Inferencia de modelos por IA	PyPI 1.0.0b9 ↗	docs	GitHub 1.0.0b9 ↗
Proyectos de IA	PyPI 1.0.0 ↗ PyPI 1.1.0b2 ↗	docs	GitHub 1.0.0 ↗ GitHub 1.1.0b2 ↗
Recursos de IA	PyPI 1.0.0b9 ↗		GitHub 1.0.0b9 ↗
Anomaly Detector	PyPI 3.0.0b6 ↗	docs	GitHub 3.0.0b6 ↗
App Configuration	PyPI 1.7.1 ↗	docs	GitHub 1.7.1 ↗
Proveedor de Configuración de Aplicaciones	PyPI 2.2.0 ↗	docs	GitHub 2.2.0 ↗
Attestation	PyPI 1.0.0 ↗	docs	GitHub 1.0.0 ↗
Azure AI Search	PyPI 11.5.3 ↗ PyPI 11.6.0b12 ↗	docs	GitHub 11.5.3 ↗ GitHub 11.6.0b12 ↗
SDK de Visión de Azure AI	PyPI 0.15.1b1 ↗		GitHub 0.15.1b1 ↗
Almacén de puntos de control de Azure Blob Storage	PyPI 1.2.0 ↗	docs	GitHub 1.2.0 ↗
AIO de almacén de puntos de control de Azure Blob Storage	PyPI 1.2.0 ↗	docs	GitHub 1.2.0 ↗
OpenTelemetry para Azure Monitor	PyPI 1.6.13 ↗	docs	GitHub 1.6.13 ↗
Azure Remote Rendering	PyPI 1.0.0b2 ↗	docs	GitHub 1.0.0b2 ↗
Automatización de llamadas de comunicación	PyPI 1.4.0 ↗	docs	GitHub 1.4.0 ↗
Chat de Comunicación	PyPI 1.3.0 ↗	docs	GitHub 1.3.0 ↗
Correo electrónico de comunicación	PyPI 1.0.0 ↗ PyPI 1.0.1b1 ↗	docs	GitHub 1.0.0 ↗ GitHub 1.0.1b1 ↗
Identidad comunicativa	PyPI 1.5.0 ↗	docs	GitHub 1.5.0 ↗
Comunicación JobRouter	PyPI 1.0.0 ↗ PyPI 1.1.0b1 ↗	docs	GitHub 1.0.0 ↗ GitHub 1.1.0b1 ↗
Mensajes de comunicación	PyPI 1.1.0 ↗ PyPI 1.2.0b1 ↗	docs	GitHub 1.1.0 ↗ GitHub 1.2.0 B1 ↗

Name	Package	Docs	Source
Travesía de la red de comunicaciones	PyPI 1.1.0b2 ↗	docs	GitHub 1.1.0b2 ↗
Números de Teléfono de Comunicación	PyPI 1.3.0 ↗	docs	GitHub 1.3.0 ↗
	PyPI 1.4.0b2 ↗		GitHub 1.4.0b2 ↗
Salas de comunicación	PyPI 1.2.0 ↗	docs	GitHub 1.2.0 ↗
Comunicación SMS	PyPI 1.1.0 ↗	docs	GitHub 1.1.0 ↗
Confidential Ledger	PyPI 1.1.1 ↗	docs	GitHub 1.1.1 ↗
	PyPI 1.2.0b1 ↗		GitHub 1.2.0 B1 ↗
Container Registry	PyPI 1.2.0 ↗	docs	GitHub 1.2.0 ↗
Seguridad del contenido	PyPI 1.0.0 ↗	docs	GitHub 1.0.0 ↗
Comprensión del lenguaje conversacional	PyPI 1.1.0 ↗	docs	GitHub 1.1.0 ↗
Core - Cliente - Core	PyPI 1.35.0 ↗	docs	GitHub 1.35.0 ↗
Core - Cliente - Core HTTP	PyPI 1.0.0b6 ↗	docs	GitHub 1.0.0b6 ↗
Core - Cliente - Experimental	PyPI 1.0.0b4 ↗	docs	GitHub 1.0.0b4 ↗
Core - Cliente - Seguimiento de Opentelemetry	PyPI 1.0.0b12 ↗	docs	GitHub 1.0.0b12 ↗
Seguimiento básico de Opencensus	PyPI 1.0.0b10 ↗	docs	GitHub 1.0.0b10 ↗
Cosmos DB	PyPI 4.9.0 ↗	docs	GitHub 4.9.0 ↗
	PyPI 4.14.0b2 ↗		GitHub 4.14.0b2 ↗
Defender EASM	PyPI 1.0.0b1 ↗	docs	GitHub 1.0.0b1 ↗
Centro de desarrollo	PyPI 1.0.0 ↗	docs	GitHub 1.0.0 ↗
Actualización de dispositivos	PyPI 1.0.0 ↗	docs	GitHub 1.0.0 ↗
Digital Twins	PyPI 1.3.0 ↗	docs	GitHub 1.3.0 ↗
Inteligencia Documental	PyPI 1.0.2 ↗	docs	GitHub 1.0.2 ↗
Traducción de documentos	PyPI 1.1.0 ↗	docs	GitHub 1.1.0 ↗
Event Grid	PyPI 4.22.0 ↗	docs	GitHub 4.22.0 ↗
Event Hubs	PyPI 5.15.0 ↗	docs	GitHub 5.15.0 ↗
Face	PyPI 1.0.0b2 ↗	docs	GitHub 1.0.0b2 ↗
Face	PyPI 0.6.1 ↗	docs	GitHub 0.6.1 ↗

Name	Package	Docs	Source
FarmBeats	PyPI 1.0.0b2	docs	GitHub 1.0.0b2
Form Recognizer	PyPI 3.3.3	docs	GitHub 3.3.3
Anonimización de Salud	PyPI 1.0.0 PyPI 1.1.0b1	docs	GitHub 1.0.0 GitHub 1.1.0b1
Generación de perfiles de cáncer de Health Insights	PyPI 1.0.0b1	docs	GitHub 1.0.0b1
Coincidencia clínica de Health Insights	PyPI 1.0.0b1	docs	GitHub 1.0.0b1
Información de radiología de Health Insights	PyPI 1.1.0	docs	GitHub 1.1.0
Identity	PyPI 1.24.0	docs	GitHub 1.24.0
Agente de identidad	PyPI 1.3.0	docs	GitHub 1.3.0
Análisis de imágenes	PyPI 1.0.0	docs	GitHub 1.0.0
Key Vault: Administración	PyPI 4.6.0	docs	GitHub 4.6.0
Key Vault: certificados	PyPI 4.10.0	docs	GitHub 4.10.0
Key Vault: claves	PyPI 4.11.0	docs	GitHub 4.11.0
Key Vault - Secretos	PyPI 4.10.0	docs	GitHub 4.10.0
Key Vault: dominio de seguridad	PyPI 1.0.0b1	docs	GitHub 1.0.0b1
Load Testing	PyPI 1.0.1 PyPI 1.1.0b1	docs	GitHub 1.0.1 GitHub 1.1.0b1
Machine Learning	PyPI 1.28.1	docs	GitHub 1.28.1
Machine Learning - Almacén de características	PyPI 1.0.1		GitHub 1.0.1
Puntos de conexión privados administrados	PyPI 0.4.0	docs	GitHub 0.4.0
Geolocalización en mapas	PyPI 1.0.0b3	docs	GitHub 1.0.0b3
Representación de mapas	PyPI 2.0.0b2	docs	GitHub 2.0.0b2
Ruta en mapas	PyPI 1.0.0b3	docs	GitHub 1.0.0b3
Búsqueda de mapas	PyPI 2.0.0b2	docs	GitHub 2.0.0b2
Ánálisis multimedia en el Edge	PyPI 1.0.0b2	docs	GitHub 1.0.0b2
Metrics Advisor	PyPI 1.0.1	docs	GitHub 1.0.1
Autenticación de Realidad Mixta	PyPI 1.0.0b1	docs	GitHub 1.0.0b1

Name	Package	Docs	Source
Repositorio de modelos	PyPI 1.0.0b2	docs	GitHub 1.0.0b2
Supervisión de la ingesta	PyPI 1.1.0	docs	GitHub 1.1.0
Supervisión de registros de consultas	PyPI 2.0.0	docs	GitHub 2.0.0
Supervisión de métricas de consulta	PyPI 1.0.0	docs	GitHub 1.0.0
Exportador de OpenTelemetry	PyPI 1.0.0b41	docs	GitHub 1.0.0b41
Personalizer	PyPI 1.0.0b1		GitHub 1.0.0b1
Cuenta de Purview	PyPI 1.0.0b1	docs	GitHub 1.0.0b1
Administración de Purview	PyPI 1.0.0b1	docs	GitHub 1.0.0b1
Catálogo de Purview	PyPI 1.0.0b4	docs	GitHub 1.0.0b4
Mapa de datos de Purview	PyPI 1.0.0b2	docs	GitHub 1.0.0b2
Examen de Purview	PyPI 1.0.0b2	docs	GitHub 1.0.0b2
Uso compartido de Purview	PyPI 1.0.0b3	docs	GitHub 1.0.0b3
Flujo de trabajo de Purview	PyPI 1.0.0b2	docs	GitHub 1.0.0b2
Respuesta a preguntas	PyPI 1.1.0	docs	GitHub 1.1.0
Registro de esquemas	PyPI 1.3.0	docs	GitHub 1.3.0
Registro de esquema: Avro	PyPI 1.0.0	docs	GitHub 1.0.0
Registro de esquema: Avro	PyPI 1.0.0b4		GitHub 1.0.0b4
Service Bus	PyPI 7.14.2	docs	GitHub 7.14.2
Spark	PyPI 0.7.0	docs	GitHub 0.7.0
Almacenamiento: Blobs	PyPI 12.26.0	docs	GitHub 12.26.0
	PyPI 12.27.0b1		GitHub 12.27.0b1
Storage: Blobs Changefeed	PyPI 12.0.0b5	docs	GitHub 12.0.0b5
Storage: lago de datos de archivos	PyPI 12.21.0	docs	GitHub 12.21.0
	PyPI 12.22.0b1		GitHub 12.22.0b1
Storage - Compartición de archivos	PyPI 12.22.0	docs	GitHub 12.22.0
	PyPI 12.23.0b1		GitHub 12.23.0b1
- Colas de almacenamiento	PyPI 12.13.0	docs	GitHub 12.13.0
	PyPI 12.14.0b1		GitHub 12.14.0b1

Name	Package	Docs	Source
Synapse: AccessControl	PyPI 0.7.0 ↗	docs	GitHub 0.7.0 ↗
Synapse – Artifacts	PyPI 0.20.0 ↗	docs	GitHub 0.20.0 ↗
Synapse: supervisión	PyPI 0.2.0 ↗	docs	GitHub 0.2.0 ↗
Tables	PyPI 12.7.0 ↗	docs	GitHub 12.7.0 ↗
Text Analytics	PyPI 5.3.0 ↗	docs	GitHub 5.3.0 ↗
Traducción de texto	PyPI 1.0.1 ↗	docs	GitHub 1.0.1 ↗
TimeZones	PyPI 1.0.0b1 ↗	docs	GitHub 1.0.0b1 ↗
unknown	PyPI 1.0.0b1 ↗	docs	GitHub 1.0.0b1 ↗
Analizador de Video Edge	PyPI 1.0.0b4 ↗	docs	GitHub 1.0.0b4 ↗
Web PubSub	PyPI 1.3.0 ↗	docs	GitHub 1.3.0 ↗
Cliente de Web PubSub	PyPI 1.1.0 ↗	docs	GitHub 1.1.0 ↗
Core - Administración - Core	PyPI 1.6.0 ↗	docs	GitHub 1.6.0 ↗
Administración de recursos - Astro	PyPI 1.0.0b1 ↗	docs	GitHub 1.0.0b1 ↗
Administración de recursos: centro de desarrollo	PyPI 1.1.0 ↗	docs	GitHub 1.1.0 ↗
Administración de recursos - Elastic SAN	PyPI 1.1.0 ↗ PyPI 1.2.0b2 ↗	docs	GitHub 1.1.0 ↗ GitHub 1.2.0b2 ↗
Administración de recursos: centro de seguridad	PyPI 1.0.0b2 ↗	docs	GitHub 1.0.0b2 ↗
Administración de recursos: Advisor	PyPI 9.0.0 ↗ PyPI 10.0.0b1 ↗	docs	GitHub 9.0.0 ↗ GitHub 10.0.0b1 ↗
Administración de recursos: Agrifood	PyPI 1.0.0b2 ↗	docs	GitHub 1.0.0b2 ↗
Administración de recursos: Agrifood	PyPI 1.0.0b3 ↗	docs	GitHub 1.0.0b3 ↗
Administración de recursos - Centro para desarrolladores de AKS	PyPI 1.0.0b1 ↗	docs	GitHub 1.0.0b1 ↗
Administración de recursos: administración de alertas	PyPI 1.0.0 ↗ PyPI 2.0.0b2 ↗	docs	GitHub 1.0.0 ↗ GitHub 2.0.0b2 ↗
Administración de recursos: Centro API	PyPI 1.0.0 ↗	docs	GitHub 1.0.0 ↗
Administración de recursos: API Management	PyPI 5.0.0 ↗	docs	GitHub 5.0.0 ↗
Administración de recursos: automatización del	PyPI 1.0.0 ↗	docs	GitHub 1.0.0 ↗

Name	Package	Docs	Source
cumplimiento de aplicaciones			
Administración de recursos: Configuración de la aplicación	PyPI 5.0.0	docs	GitHub 5.0.0
Administración de recursos: App Platform	PyPI 10.0.1	docs	GitHub 10.0.1
Administración de recursos: App Service	PyPI 9.0.0	docs	GitHub 9.0.0
Administración de recursos: Application Insights	PyPI 4.1.0 PyPI 5.0.0b1	docs	GitHub 4.1.0 GitHub 5.0.0b1
Administración de recursos - Datos de Arc	PyPI 1.0.0 PyPI 2.0.0b1	docs	GitHub 1.0.0 GitHub 2.0.0b1
Administración de recursos: Arize AI Observability Eval	PyPI 1.0.0	docs	GitHub 1.0.0
Administración de recursos: atestación	PyPI 1.0.0 PyPI 2.0.0b1	docs	GitHub 1.0.0 GitHub 2.0.0b1
Administración de recursos: autorización	PyPI 4.0.0 PyPI 5.0.0b1	docs	GitHub 4.0.0 GitHub 5.0.0b1
Administración de recursos: Automanage	PyPI 1.0.0 PyPI 2.0.0b1	docs	GitHub 1.0.0 GitHub 2.0.0b1
Administración de recursos: Automatización	PyPI 1.0.0 PyPI 1.1.0b4	docs	GitHub 1.0.0 GitHub 1.1.0b4
Administración de recursos- Azure AD B2C	PyPI 1.0.0b2	docs	GitHub 1.0.0b2
Administración de recursos: Búsqueda de Azure AI	PyPI 9.2.0	docs	GitHub 9.2.0
Administración de recursos: Azure Stack	PyPI 1.0.0 PyPI 2.0.0b1	docs	GitHub 1.0.0 GitHub 2.0.0b1
Administración de recursos: Azure Stack HCI	PyPI 7.0.0 PyPI 8.0.0b4	docs	GitHub 7.0.0 GitHub 8.0.0b4
Administración de recursos: Azure VMware Solution	PyPI 9.1.0	docs	GitHub 9.1.0
Administración de recursos - Infraestructura BareMetal	PyPI 1.0.0 PyPI 1.1.0b2	docs	GitHub 1.0.0 GitHub 1.1.0b2
Administración de recursos: Batch	PyPI 18.0.0	docs	GitHub 18.0.0
Administración de recursos: Batch AI	PyPI 7.0.0	docs	GitHub 7.0.0
Administración de recursos: facturación	PyPI 7.0.0	docs	GitHub 7.0.0
Administración de recursos: ventajas de la facturación	PyPI 1.0.0b1	docs	GitHub 1.0.0b1

Name	Package	Docs	Source
Administración de recursos: Bot Service	PyPI 2.0.0	docs	GitHub 2.0.0
Administración de recursos: optimización de carbono	PyPI 1.0.0	docs	GitHub 1.0.0
Administración de recursos: Análisis de cambios	PyPI 1.0.0	docs	GitHub 1.0.0
Administración de recursos: Chaos	PyPI 2.0.0	docs	GitHub 2.0.0
Administración de recursos: Cloudhealth	PyPI 1.0.0b1	docs	GitHub 1.0.0b1
Administración de recursos: Cognitive Services	PyPI 13.7.0	docs	GitHub 13.7.0
Administración de recursos: Commerce	PyPI 6.0.0 PyPI 6.1.0b1	docs	GitHub 6.0.0 GitHub 6.1.0b1
Administración de recursos: comunicación	PyPI 2.1.0	docs	GitHub 2.1.0
Administración de recursos: Compute	PyPI 35.0.0	docs	GitHub 35.0.0
Administración de recursos: flota de cómputo	PyPI 1.0.0	docs	GitHub 1.0.0
Administración de recursos: programación de cálculo	PyPI 1.1.0 PyPI 1.2.0b1	docs	GitHub 1.1.0 GitHub 1.2.0 B1
Administración de recursos: Confidential Ledger	PyPI 1.0.0 PyPI 2.0.0b5	docs	GitHub 1.0.0 GitHub 2.0.0b5
Administración de recursos - Confluent	PyPI 2.1.0	docs	GitHub 2.1.0
Administración de recursos: caché conectada	PyPI 1.0.0b1	docs	GitHub 1.0.0b1
Administración de recursos - VMWare conectado	PyPI 1.0.0	docs	GitHub 1.0.0
Administración de recursos: consumo	PyPI 10.0.0 PyPI 11.0.0b1	docs	GitHub 10.0.0 GitHub 11.0.0b1
Administración de recursos: Container Apps	PyPI 3.2.0	docs	GitHub 3.2.0
Administración de recursos: Container Instances	PyPI 10.1.0 PyPI 10.2.0b1	docs	GitHub 10.1.0 GitHub 10.2.0b1
Administración de recursos: entorno de ejecución de Container Orchestrator	PyPI 1.0.0b1	docs	GitHub 1.0.0b1
Administración de recursos: Container Registry	PyPI 14.0.0 PyPI 14.1.0b1	docs	GitHub 14.0.0 GitHub 14.1.0b1
Administración de recursos: servicio de contenedor	PyPI 39.0.0	docs	GitHub 39.0.0
Administración de recursos: flota de servicio de contenedor	PyPI 3.1.0	docs	GitHub 3.1.0

Name	Package	Docs	Source
Administración de recursos - Controles de seguridad de servicios de contenedores	PyPI 1.0.0b1	docs	GitHub 1.0.0b1
Administración de recursos: Content Delivery Network	PyPI 13.1.1	docs	GitHub 13.1.1
Administración de recursos: Cosmos DB	PyPI 9.8.0 PyPI 10.0.0b5	docs	GitHub 9.8.0 GitHub 10.0.0b5
Administración de recursos: Cosmos DB	PyPI 0.1.4		GitHub 0.1.4
Administración de recursos: Cosmos DB para PostgreSQL	PyPI 1.0.0 PyPI 1.1.0b1	docs	GitHub 1.0.0 GitHub 1.1.0b1
Administración de recursos: Gestión de costos	PyPI 4.0.1	docs	GitHub 4.0.1
Administración de recursos: proveedores personalizados	PyPI 1.0.0 PyPI 1.1.0b1	docs	GitHub 1.0.0 GitHub 1.1.0b1
Administración de recursos: Data Box	PyPI 3.1.0	docs	GitHub 3.1.0
Administración de recursos: Data Box Edge	PyPI 2.0.0 PyPI 3.0.0b1	docs	GitHub 2.0.0 GitHub 3.0.0b1
Administración de recursos: Data Factory	PyPI 9.2.0	docs	GitHub 9.2.0
Administración de recursos: Data Lake Analytics	PyPI 1.0.0b2	docs	GitHub 1.0.0b2
Administración de recursos: Data Lake Store	PyPI 1.0.0 PyPI 1.1.0b1	docs	GitHub 1.0.0
Administración de recursos: migración de datos	PyPI 10.0.0 PyPI 10.1.0b2	docs	GitHub 10.0.0 GitHub 10.1.0b2
Administración de recursos: protección de datos	PyPI 1.4.0	docs	GitHub 1.4.0
Administración de recursos: Data Share	PyPI 1.0.0 PyPI 1.1.0b1	docs	GitHub 1.0.0 GitHub 1.1.0b1
Administración de recursos: Monitor de base de datos	PyPI 1.0.0b1	docs	GitHub 1.0.0b1
Administración de recursos: Databricks	PyPI 2.0.0	docs	GitHub 2.0.0
Administración de recursos: Datadog	PyPI 2.1.0	docs	GitHub 2.1.0
Administración de recursos: Defender EASM	PyPI 1.0.0b1	docs	GitHub 1.0.0b1
Administración de recursos: Dellstorage	PyPI 1.0.0b1	docs	GitHub 1.0.0b1
Administración de recursos: Mapa de dependencias	PyPI 1.0.0b1	docs	GitHub 1.0.0b1

Name	Package	Docs	Source
Administración de recursos: Administrador de Despliegue	PyPI 1.0.0 ↗ PyPI 2.0.0b1 ↗	docs	GitHub 1.0.0 ↗ GitHub 2.0.0b1 ↗
Administración de recursos: virtualización de escritorio	PyPI 2.0.0 ↗	docs	GitHub 2.0.0 ↗
Administración de recursos: Dev Spaces	PyPI 1.0.0b3 ↗	docs	GitHub 1.0.0b3 ↗
Administración de recursos - Servicios de Aprovisionamiento de Dispositivos	PyPI 1.1.0 ↗ PyPI 1.2.0b2 ↗	docs	GitHub 1.1.0 ↗
Administración de recursos - Registro de dispositivos	PyPI 1.0.0 ↗	docs	GitHub 1.0.0 ↗
Administración de recursos: actualización de dispositivos	PyPI 1.1.0 ↗	docs	GitHub 1.1.0 ↗
Administración de recursos - Infraestructura de DevOps	PyPI 1.0.0 ↗	docs	GitHub 1.0.0 ↗
Administración de recursos: DevTest Labs	PyPI 9.0.0 ↗ PyPI 10.0.0b2 ↗	docs	GitHub 9.0.0 ↗ GitHub 10.0.0b2 ↗
Administración de recursos: Digital Twins	PyPI 7.0.0 ↗	docs	GitHub 7.0.0 ↗
Administración de recursos: DNS	PyPI 9.0.0 ↗	docs	GitHub 9.0.0 ↗
Administración de recursos: solucionador DNS	PyPI 1.1.0 ↗	docs	GitHub 1.1.0 ↗
Administración de recursos: Durable Task	PyPI 1.0.0b2 ↗	docs	GitHub 1.0.0b2 ↗
Administración de recursos: Dynatrace	PyPI 2.0.0 ↗	docs	GitHub 2.0.0 ↗
Administración de recursos: orden perimetral	PyPI 2.0.0 ↗	docs	GitHub 2.0.0 ↗
Administración de recursos - Edge Zones	PyPI 1.0.0b2 ↗	docs	GitHub 1.0.0b2 ↗
Administración de recursos - Educación	PyPI 1.0.0b2 ↗	docs	GitHub 1.0.0b2 ↗
Administración de recursos: elástico	PyPI 1.0.0 ↗ PyPI 1.1.0b4 ↗	docs	GitHub 1.0.0 ↗ GitHub 1.1.0b4 ↗
Administración de recursos: Event Grid	PyPI 10.4.0 ↗ PyPI 10.5.0b1 ↗	docs	GitHub 10.4.0 ↗ GitHub 10.5.0b1 ↗
Administración de recursos: Event Hubs	PyPI 11.2.0 ↗ PyPI 12.0.0b1 ↗	docs	GitHub 11.2.0 ↗ GitHub 12.0.0b1 ↗
Administración de recursos: ubicación ampliada	PyPI 2.0.0 ↗	docs	GitHub 2.0.0 ↗
Administración de recursos - Fabric	PyPI 1.0.0 ↗	docs	GitHub 1.0.0 ↗
Administración de recursos: Fluid Relay	PyPI 1.0.0 ↗ PyPI 1.1.0b1 ↗	docs	GitHub 1.0.0 ↗ GitHub 1.1.0b1 ↗

Name	Package	Docs	Source
Administración de recursos: Front Door	PyPI 1.2.0	docs	GitHub 1.2.0
Administración de recursos: Servicios de Graph	PyPI 1.0.0	docs	GitHub 1.0.0
Administración de recursos: configuración de invitados	PyPI 1.0.0b2	docs	GitHub 1.0.0b2
Administración de recursos: HANA en Azure	PyPI 1.0.0 PyPI 1.1.0b1	docs	GitHub 1.0.0 GitHub 1.1.0b1
Administración de recursos: módulos de seguridad de hardware	PyPI 1.0.0	docs	GitHub 1.0.0
Administración de recursos: HDInsight	PyPI 9.0.0 PyPI 9.1.0b1	docs	GitHub 9.0.0 GitHub 9.1.0b1
Gestión de recursos - Contenedores HDInsight	PyPI 1.0.0b3	docs	GitHub 1.0.0b3
Administración de recursos: Health Bot	PyPI 1.0.0b2	docs	GitHub 1.0.0b2
Administración de recursos: Health Data AI Services	PyPI 1.0.0	docs	GitHub 1.0.0
Administración de recursos: API de sanidad	PyPI 2.1.0	docs	GitHub 2.1.0
Administración de recursos: computación híbrida	PyPI 9.0.0 PyPI 9.1.0b2	docs	GitHub 9.0.0 GitHub 9.1.0b2
Administración de recursos: conectividad híbrida	PyPI 1.0.0 PyPI 2.0.0b1	docs	GitHub 1.0.0 GitHub 2.0.0b1
Administración de recursos: Servicio de contenedor híbrido	PyPI 1.0.0	docs	GitHub 1.0.0
Administración de recursos: Kubernetes híbrido	PyPI 1.1.0 PyPI 1.2.0b2	docs	GitHub 1.1.0 GitHub 1.2.0b2
Administración de recursos: red híbrida	PyPI 2.0.0	docs	GitHub 2.0.0
Administración de recursos: Image Builder	PyPI 1.4.0	docs	GitHub 1.4.0
Administración de recursos: informes de impacto	PyPI 1.0.0b1	docs	GitHub 1.0.0b1
Administración de recursos - Informatica Data Management	PyPI 1.0.0	docs	GitHub 1.0.0
Administración de recursos: IoT Central	PyPI 9.0.0 PyPI 10.0.0b2	docs	GitHub 9.0.0 GitHub 10.0.0b2
Administración de recursos: IoT Firmware Defense	PyPI 1.0.0 PyPI 2.0.0b1	docs	GitHub 1.0.0 GitHub 2.0.0b1

Name	Package	Docs	Source
Administración de recursos: IoT Hub	PyPI 4.0.0 ↗ PyPI 5.0.0b1 ↗	docs	GitHub 4.0.0 ↗ GitHub 5.0.0b1 ↗
Administración de recursos: operaciones de IoT	PyPI 1.0.0 ↗	docs	GitHub 1.0.0 ↗
Administración de recursos: Key Vault	PyPI 12.0.0 ↗	docs	GitHub 12.0.0 ↗
Administración de recursos: configuración de Kubernetes	PyPI 3.1.0 ↗	docs	GitHub 3.1.0 ↗
Administración de recursos: Kubernetesconfiguration-Extensions	PyPI 1.0.0b1 ↗	docs	GitHub 1.0.0b1 ↗
Administración de recursos: Kubernetesconfiguration-Extensiontypes	PyPI 1.0.0b1 ↗	docs	GitHub 1.0.0b1 ↗
Administración de recursos: Kubernetesconfiguration-Fluxconfigurations	PyPI 1.0.0b1 ↗	docs	GitHub 1.0.0b1 ↗
Administración de recursos: Kubernetesconfiguration-Privatelinkscopes	PyPI 1.0.0b1 ↗	docs	GitHub 1.0.0b1 ↗
Administración de recursos: Kusto	PyPI 3.4.0 ↗	docs	GitHub 3.4.0 ↗
Administración de recursos: servicios de laboratorio	PyPI 2.0.0 ↗ PyPI 2.1.0b1 ↗	docs	GitHub 2.0.0 ↗ GitHub 2.1.0b1 ↗
Administración de recursos - Lambdatesthyperexecute	PyPI 1.0.0 ↗	docs	GitHub 1.0.0 ↗
Administración de recursos: instancia grande	PyPI 1.0.0b1 ↗	docs	GitHub 1.0.0b1 ↗
Administración de recursos: prueba de carga	PyPI 1.0.0 ↗	docs	GitHub 1.0.0 ↗
Administración de recursos: Log Analytics	PyPI 12.0.0 ↗ PyPI 13.0.0b7 ↗	docs	GitHub 12.0.0 ↗ GitHub 13.0.0b7 ↗
Administración de recursos: Logic Apps	PyPI 10.0.0 ↗ PyPI 10.1.0b1 ↗	docs	GitHub 10.0.0 ↗ GitHub 10.1.0b1 ↗
Administración de recursos - Logz	PyPI 1.1.1 ↗	docs	GitHub 1.1.1 ↗
Administración de recursos - Computación para Aprendizaje Automático	PyPI 1.0.0b2 ↗	docs	GitHub 1.0.0b2 ↗
Administración de recursos: Machine Learning Services	PyPI 1.0.0 ↗ PyPI 2.0.0b2 ↗	docs	GitHub 1.0.0 ↗ GitHub 2.0.0b2 ↗
Administración de recursos: mantenimiento	PyPI 2.1.0 ↗ PyPI 2.2.0b2 ↗	docs	GitHub 2.1.0 ↗ GitHub 2.2.0b2 ↗
Administración de recursos: aplicaciones administradas	PyPI 1.0.0b1 ↗	docs	GitHub 1.0.0b1 ↗

Name	Package	Docs	Source
Administración de recursos: Grafana administrado	PyPI 1.1.0 ↗ PyPI 2.0.0b1 ↗	docs	GitHub 1.1.0 ↗ GitHub 2.0.0b1 ↗
Administración de recursos: tejido de red administrada	PyPI 1.0.0 ↗	docs	GitHub 1.0.0 ↗
Administración de recursos: identidad de servicio administrado	PyPI 7.1.0 ↗	docs	GitHub 7.1.0 ↗
Administración de recursos: servicios administrados	PyPI 6.0.0 ↗ PyPI 7.0.0b2 ↗	docs	GitHub 6.0.0 ↗ GitHub 7.0.0b2 ↗
Administración de recursos: Grupos de administración	PyPI 1.0.0 ↗ PyPI 1.1.0b2 ↗	docs	GitHub 1.0.0 ↗ GitHub 1.1.0b2 ↗
Administración de recursos: asociado de administración	PyPI 1.0.0 ↗ PyPI 1.1.0b1 ↗	docs	GitHub 1.0.0 ↗ GitHub 1.1.0b1 ↗
Administración de recursos: Mapas	PyPI 2.1.0 ↗	docs	GitHub 2.1.0 ↗
Administración de recursos: pedidos de Marketplace	PyPI 1.1.0 ↗ PyPI 1.2.0b2 ↗	docs	GitHub 1.1.0 ↗ GitHub 1.2.0b2 ↗
Administración de recursos: Media Services	PyPI 10.2.1 ↗	docs	GitHub 10.2.1 ↗
Administración de recursos: evaluación de la migración	PyPI 1.0.0b1 ↗	docs	GitHub 1.0.0b1 ↗
Administración de recursos - SAP de detección de migración	PyPI 1.0.0b1 ↗	docs	GitHub 1.0.0b1 ↗
Administración de recursos: Mixed Reality	PyPI 1.0.0 ↗ PyPI 1.1.0b1 ↗	docs	GitHub 1.0.0 ↗ GitHub 1.1.0b1 ↗
Administración de recursos: red móvil	PyPI 3.3.0 ↗	docs	GitHub 3.3.0 ↗
Administración de recursos - Clúster de Mongo	PyPI 1.0.0 ↗ PyPI 1.1.0b1 ↗	docs	GitHub 1.0.0 ↗ GitHub 1.1.0b1 ↗
Administración de recursos - Mongodbatlas	PyPI 1.0.0 ↗	docs	GitHub 1.0.0 ↗
Administración de recursos: Monitoreo	PyPI 7.0.0 ↗ PyPI 8.0.0b1 ↗	docs	GitHub 7.0.0 ↗ GitHub 8.0.0b1 ↗
Administración de recursos: servidores flexibles de MySQL	PyPI 1.0.0b3 ↗	docs	GitHub 1.0.0b3 ↗
Administración de recursos: Neon Postgres	PyPI 1.0.0 ↗	docs	GitHub 1.0.0 ↗
Administración de recursos: NetApp Files	PyPI 13.6.0 ↗ PyPI 14.0.0b1 ↗	docs	GitHub 13.6.0 ↗ GitHub 14.0.0b1 ↗
Administración de recursos: Red	PyPI 29.0.0 ↗	docs	GitHub 29.0.0 ↗

Name	Package	Docs	Source
Administración de recursos: análisis de red	PyPI 1.0.0 ↗	docs	GitHub 1.0.0 ↗
Administración de recursos: función de red	PyPI 1.0.0b1 ↗	docs	GitHub 1.0.0b1 ↗
Administración de recursos: Networkcloud	PyPI 2.1.0 ↗	docs	GitHub 2.1.0 ↗
Administración de recursos - New Relic Observability	PyPI 1.1.0 ↗	docs	GitHub 1.1.0 ↗
Administración de recursos: Nginx	PyPI 3.0.0 ↗ PyPI 3.1.0b2 ↗	docs	GitHub 3.0.0 ↗ GitHub 3.1.0b2 ↗
Administración de recursos: Notification Hubs	PyPI 8.0.0 ↗ PyPI 8.1.0b2 ↗	docs	GitHub 8.0.0 ↗ GitHub 8.1.0b2 ↗
Administración de recursos - Oep	PyPI 1.0.0b2 ↗	docs	GitHub 1.0.0b2 ↗
Administración de recursos - Onlineexperimentation	PyPI 1.0.0b1 ↗	docs	GitHub 1.0.0b1 ↗
Administración de recursos: administración de operaciones	PyPI 1.0.0 ↗ PyPI 2.0.0b1 ↗	docs	GitHub 1.0.0 ↗ GitHub 2.0.0b1 ↗
Administración de recursos - Oracle Database	PyPI 2.0.0 ↗	docs	GitHub 2.0.0 ↗
Administración de recursos: Orbital	PyPI 2.0.0 ↗	docs	GitHub 2.0.0 ↗
Administración de recursos: Palo Alto Networks - Firewall de próxima generación	PyPI 1.0.0 ↗ PyPI 2.0.0b1 ↗	docs	GitHub 1.0.0 ↗ GitHub 2.0.0b1 ↗
Administración de recursos: emparejamiento	PyPI 1.0.0 ↗ PyPI 2.0.0b1 ↗	docs	GitHub 1.0.0 ↗ GitHub 2.0.0b1 ↗
Administración de recursos: base de datos vectorial de Pinecone	PyPI 1.0.0b2 ↗	docs	GitHub 1.0.0b2 ↗
Administración de recursos: Planetarycomputer	PyPI 1.0.0b1 ↗	docs	GitHub 1.0.0b1 ↗
Gestión de recursos - Playwright	PyPI 1.0.0b1 ↗	docs	GitHub 1.0.0b1 ↗
Administración de recursos - Playwright Testing	PyPI 1.0.0 ↗	docs	GitHub 1.0.0 ↗
Administración de recursos - Perspectivas de políticas	PyPI 1.0.0 ↗ PyPI 1.1.0b5 ↗	docs	GitHub 1.0.0 ↗ GitHub 1.1.0b5 ↗
Administración de recursos - Portal	PyPI 1.0.0 ↗ PyPI 1.1.0b1 ↗	docs	GitHub 1.0.0 ↗ GitHub 1.1.0b1 ↗
Administración de recursos - Portalservicescopilot	PyPI 1.0.0b1 ↗	docs	GitHub 1.0.0b1 ↗
Administración de recursos: PostgreSQL	PyPI 10.1.0 ↗ PyPI 10.2.0b18 ↗	docs	GitHub 10.1.0 ↗ GitHub 10.2.0b18 ↗

Name	Package	Docs	Source
Administración de recursos: servidores flexibles de PostgreSQL	PyPI 1.1.0 ↗ PyPI 1.2.0b1 ↗	docs	GitHub 1.1.0 ↗ GitHub 1.2.0 B1 ↗
Administración de recursos: Power BI dedicado	PyPI 1.0.0 ↗ PyPI 1.1.0b1 ↗	docs	GitHub 1.0.0 ↗ GitHub 1.1.0b1 ↗
Administración de recursos: DNS privado	PyPI 1.2.0 ↗	docs	GitHub 1.2.0 ↗
Administración de recursos - Purestorageblock	PyPI 1.0.0 ↗	docs	GitHub 1.0.0 ↗
Administración de recursos: Purview	PyPI 1.0.0 ↗ PyPI 1.1.0b1 ↗	docs	GitHub 1.0.0 ↗ GitHub 1.1.0b1 ↗
Administración de recursos: Quantum	PyPI 1.0.0b5 ↗	docs	GitHub 1.0.0b5 ↗
Administración de recursos: Qumulo	PyPI 2.0.0 ↗	docs	GitHub 2.0.0 ↗
Administración de recursos: cuota	PyPI 2.0.0 ↗	docs	GitHub 2.0.0 ↗
Administración de recursos: Recovery Services	PyPI 3.1.0 ↗	docs	GitHub 3.1.0 ↗
Administración de recursos: copia de seguridad de Recovery Services	PyPI 9.2.0 ↗	docs	GitHub 9.2.0 ↗
Administración de recursos: replicación de datos de Recovery Services	PyPI 1.0.0 ↗	docs	GitHub 1.0.0 ↗
Administración de recursos: Site Recovery de Recovery Services	PyPI 1.3.0 ↗	docs	GitHub 1.3.0 ↗
Administración de recursos: Red Hat OpenShift	PyPI 2.0.0 ↗	docs	GitHub 2.0.0 ↗
Administración de recursos: Redis	PyPI 14.5.0 ↗	docs	GitHub 14.5.0 ↗
Administración de recursos: Redis Enterprise	PyPI 3.0.0 ↗ PyPI 3.1.0b4 ↗	docs	GitHub 3.0.0 ↗ GitHub 3.1.0b4 ↗
Administración de recursos: traslado de región	PyPI 1.0.0b1 ↗		GitHub 1.0.0b1 ↗
Administración de recursos: Relay	PyPI 1.1.0 ↗ PyPI 2.0.0b1 ↗	docs	GitHub 1.1.0 ↗ GitHub 2.0.0b1 ↗
Administración de recursos: reservas	PyPI 2.3.0 ↗	docs	GitHub 2.3.0 ↗
Administración de recursos: Conector de recursos	PyPI 1.0.0 ↗	docs	GitHub 1.0.0 ↗
Administración de recursos: Resource Graph	PyPI 8.0.0 ↗ PyPI 8.1.0b3 ↗	docs	GitHub 8.0.0 ↗ GitHub 8.1.0b3 ↗
Administración de recursos - Salud de recursos	PyPI 1.0.0b6 ↗	docs	GitHub 1.0.0b6 ↗

Name	Package	Docs	Source
Administración de recursos: Resource Mover	PyPI 1.1.0 ↗	docs	GitHub 1.1.0 ↗
Administración de recursos: Resource-Bicep	PyPI 1.0.0b1 ↗	docs	GitHub 1.0.0b1 ↗
Administración de recursos: Resource-Deployments	PyPI 1.0.0b1 ↗	docs	GitHub 1.0.0b1 ↗
Administración de recursos: Resource-Deploymentscripts	PyPI 1.0.0b1 ↗	docs	GitHub 1.0.0b1 ↗
Administración de recursos: Resource-Deploymentstacks	PyPI 1.0.0b1 ↗	docs	GitHub 1.0.0b1 ↗
Administración de recursos: Resource-Templatespecs	PyPI 1.0.0b1 ↗	docs	GitHub 1.0.0b1 ↗
Administración de recursos: Recursos	PyPI 24.0.0 ↗ PyPI 25.0.0b1 ↗	docs	GitHub 24.0.0 ↗ GitHub 25.0.0b1 ↗
Administración de recursos: Recursos	PyPI 24.0.0 ↗ PyPI 25.0.0b1 ↗	docs	GitHub 24.0.0 ↗ GitHub 25.0.0b1 ↗
Administración de recursos: Scheduler	PyPI 7.0.0 ↗	docs	GitHub 7.0.0 ↗
Administración de recursos - Scvmm	PyPI 1.0.0 ↗	docs	GitHub 1.0.0 ↗
Administración de recursos - Secretsstoreextension	PyPI 1.0.0b1 ↗	docs	GitHub 1.0.0b1 ↗
Administración de recursos: seguridad	PyPI 7.0.0 ↗	docs	GitHub 7.0.0 ↗
Administración de recursos: Security Insights	PyPI 1.0.0 ↗ PyPI 2.0.0b2 ↗	docs	GitHub 1.0.0 ↗ GitHub 2.0.0b2 ↗
Administración de recursos: autoayuda	PyPI 1.0.0 ↗ PyPI 2.0.0b4 ↗	docs	GitHub 1.0.0 ↗ GitHub 2.0.0b4 ↗
Administración de recursos: consola serie	PyPI 1.0.0 ↗ PyPI 1.1.0b1 ↗	docs	GitHub 1.0.0 ↗ GitHub 1.1.0b1 ↗
Administración de recursos: Administración del servidor	PyPI 2.0.1 ↗		GitHub 2.0.1 ↗
Administración de recursos: Service Bus	PyPI 9.0.0 ↗ PyPI 10.0.0b1 ↗	docs	GitHub 9.0.0 ↗ GitHub 10.0.0b1 ↗
Administración de recursos: Service Fabric	PyPI 2.1.0 ↗ PyPI 2.2.0b1 ↗	docs	GitHub 2.1.0 ↗ GitHub 2.2.0b1 ↗
Administración de recursos: clústeres administrados de Service Fabric	PyPI 2.0.0 ↗ PyPI 2.1.0b3 ↗	docs	GitHub 2.0.0 ↗ GitHub 2.1.0b3 ↗
Administración de recursos: Service Linker	PyPI 1.1.0 ↗ PyPI 1.2.0b3 ↗	docs	GitHub 1.1.0 ↗ GitHub 1.2.0b3 ↗
Administración de recursos: Redes de servicio	PyPI 2.0.0 ↗ PyPI 2.1.0b1 ↗	docs	GitHub 2.0.0 ↗ GitHub 2.1.0b1 ↗

Name	Package	Docs	Source
Administración de recursos: SignalR	PyPI 1.2.0 ↗ PyPI 2.0.0b2 ↗	docs	GitHub 1.2.0 ↗ GitHub 2.0.0b2 ↗
Administración de recursos: Sitemanager	PyPI 1.0.0b1 ↗	docs	GitHub 1.0.0b1 ↗
Administración de recursos: Sphere	PyPI 1.0.0 ↗	docs	GitHub 1.0.0 ↗
Administración de recursos - Detección de aplicaciones de Spring	PyPI 1.0.0b1 ↗	docs	GitHub 1.0.0b1 ↗
Administración de recursos: SQL	PyPI 3.0.1 ↗ PyPI 4.0.0b22 ↗	docs	GitHub 3.0.1 ↗ GitHub 4.0.0b22 ↗
Administración de recursos: máquina virtual de SQL	PyPI 1.0.0b6 ↗	docs	GitHub 1.0.0b6 ↗
Administración de recursos - Grupo en espera	PyPI 2.0.0 ↗	docs	GitHub 2.0.0 ↗
Administración de recursos: Storage	PyPI 23.0.1 ↗	docs	GitHub 23.0.1 ↗
Administración de recursos - Acciones de almacenamiento	PyPI 1.0.0 ↗	docs	GitHub 1.0.0 ↗
Administración de recursos: caché de almacenamiento	PyPI 2.0.0 ↗	docs	GitHub 2.0.0 ↗
Administración de recursos: importación y exportación de almacenamiento	PyPI 1.0.0b2 ↗	docs	GitHub 1.0.0b2 ↗
Administración de recursos: Storage Mover	PyPI 2.1.0 ↗	docs	GitHub 2.1.0 ↗
Administración de recursos: grupo de almacenamiento	PyPI 1.0.0 ↗ PyPI 1.1.0b1 ↗	docs	GitHub 1.0.0 ↗ GitHub 1.1.0b1 ↗
Administración de recursos: sincronización de almacenamiento	PyPI 1.0.0 ↗ PyPI 2.0.0b1 ↗	docs	GitHub 1.0.0 ↗ GitHub 2.0.0b1 ↗
Administración de recursos: Storagediscovery	PyPI 1.0.0b1 ↗	docs	GitHub 1.0.0b1 ↗
Administración de recursos: Stream Analytics	PyPI 1.0.0 ↗ PyPI 2.0.0b2 ↗	docs	GitHub 1.0.0 ↗ GitHub 2.0.0b2 ↗
Administración de recursos: suscripciones	PyPI 3.1.1 ↗ PyPI 3.2.0b1 ↗	docs	GitHub 3.1.1 ↗ GitHub 3.2.0b1 ↗
Administración de recursos: soporte técnico	PyPI 7.0.0 ↗	docs	GitHub 7.0.0 ↗
Administración de recursos: Synapse	PyPI 2.0.0 ↗ PyPI 2.1.0b7 ↗	docs	GitHub 2.0.0 ↗ GitHub 2.1.0b7 ↗
Administración de recursos: Terraform	PyPI 1.0.0b1 ↗	docs	GitHub 1.0.0b1 ↗
Administración de recursos - Base de pruebas	PyPI 1.0.0b2 ↗	docs	GitHub 1.0.0b2 ↗

Name	Package	Docs	Source
Administración de recursos: Time Series Insights	PyPI 1.0.0 ↗ PyPI 2.0.0b1 ↗	docs	GitHub 1.0.0 ↗ GitHub 2.0.0b1 ↗
Administración de recursos: Administrador de Tráfico	PyPI 1.1.0 ↗	docs	GitHub 1.1.0 ↗
Administración de recursos: Trusted Signing	PyPI 1.0.0b1 ↗	docs	GitHub 1.0.0b1 ↗
Administración de recursos: VMware Solution by CloudSimple	PyPI 1.0.0b2 ↗	docs	GitHub 1.0.0b2 ↗
Administración de recursos: Servicios de Voz	PyPI 1.0.0 ↗	docs	GitHub 1.0.0 ↗
Administración de recursos: Web PubSub	PyPI 2.0.0 ↗	docs	GitHub 2.0.0 ↗
Administración de recursos: ponderaciones y sesgos	PyPI 1.0.0b1 ↗	docs	GitHub 1.0.0b1 ↗
Administración de recursos: monitor de carga de trabajo	PyPI 1.0.0b4 ↗	docs	GitHub 1.0.0b4 ↗
Administración de recursos: cargas de trabajo	PyPI 1.0.0 ↗	docs	GitHub 1.0.0 ↗
Administración de recursos - Instancia virtual SAP de gestión de cargas de trabajo	PyPI 1.0.0 ↗	docs	GitHub 1.0.0 ↗

Todas las bibliotecas

[\[+\] Expandir tabla](#)

Name	Package	Docs	Source
Agentes de IA	PyPI 1.1.0 ↗ PyPI 1.2.0b2 ↗	docs	GitHub 1.1.0 ↗ GitHub 1.2.0b2 ↗
Evaluación de IA	PyPI 1.10.0 ↗	docs	GitHub 1.10.0 ↗
Inferencia de modelos por IA	PyPI 1.0.0b9 ↗	docs	GitHub 1.0.0b9 ↗
Proyectos de IA	PyPI 1.0.0 ↗ PyPI 1.1.0b2 ↗	docs	GitHub 1.0.0 ↗ GitHub 1.1.0b2 ↗
Anomaly Detector	PyPI 3.0.0b6 ↗	docs	GitHub 3.0.0b6 ↗
App Configuration	PyPI 1.7.1 ↗	docs	GitHub 1.7.1 ↗
Proveedor de Configuración de Aplicaciones	PyPI 2.2.0 ↗	docs	GitHub 2.2.0 ↗
Attestation	PyPI 1.0.0 ↗	docs	GitHub 1.0.0 ↗

Name	Package	Docs	Source
Azure AI Search	PyPI 11.5.3 ↗ PyPI 11.6.0b12 ↗	docs	GitHub 11.5.3 ↗ GitHub 11.6.0b12 ↗
Almacén de puntos de control de Azure Blob Storage	PyPI 1.2.0 ↗	docs	GitHub 1.2.0 ↗
AIO de almacén de puntos de control de Azure Blob Storage	PyPI 1.2.0 ↗	docs	GitHub 1.2.0 ↗
OpenTelemetry para Azure Monitor	PyPI 1.6.13 ↗	docs	GitHub 1.6.13 ↗
Azure Remote Rendering	PyPI 1.0.0b2 ↗	docs	GitHub 1.0.0b2 ↗
Automatización de llamadas de comunicación	PyPI 1.4.0 ↗	docs	GitHub 1.4.0 ↗
Chat de Comunicación	PyPI 1.3.0 ↗	docs	GitHub 1.3.0 ↗
Correo electrónico de comunicación	PyPI 1.0.0 ↗ PyPI 1.0.1b1 ↗	docs	GitHub 1.0.0 ↗ GitHub 1.0.1b1 ↗
Identidad comunicativa	PyPI 1.5.0 ↗	docs	GitHub 1.5.0 ↗
Comunicación JobRouter	PyPI 1.0.0 ↗ PyPI 1.1.0b1 ↗	docs	GitHub 1.0.0 ↗ GitHub 1.1.0b1 ↗
Mensajes de comunicación	PyPI 1.1.0 ↗ PyPI 1.2.0b1 ↗	docs	GitHub 1.1.0 ↗ GitHub 1.2.0 B1 ↗
Números de Teléfono de Comunicación	PyPI 1.3.0 ↗ PyPI 1.4.0b2 ↗	docs	GitHub 1.3.0 ↗ GitHub 1.4.0b2 ↗
Salas de comunicación	PyPI 1.2.0 ↗	docs	GitHub 1.2.0 ↗
Comunicación SMS	PyPI 1.1.0 ↗	docs	GitHub 1.1.0 ↗
Confidential Ledger	PyPI 1.1.1 ↗ PyPI 1.2.0b1 ↗	docs	GitHub 1.1.1 ↗ GitHub 1.2.0 B1 ↗
Container Registry	PyPI 1.2.0 ↗	docs	GitHub 1.2.0 ↗
Seguridad del contenido	PyPI 1.0.0 ↗	docs	GitHub 1.0.0 ↗
Comprensión del lenguaje conversacional	PyPI 1.1.0 ↗	docs	GitHub 1.1.0 ↗
Core - Cliente - Core	PyPI 1.35.0 ↗	docs	GitHub 1.35.0 ↗
Core - Cliente - Core HTTP	PyPI 1.0.0b6 ↗	docs	GitHub 1.0.0b6 ↗
Core - Cliente - Experimental	PyPI 1.0.0b4 ↗	docs	GitHub 1.0.0b4 ↗
Core - Cliente - Seguimiento de OpenTelemetry	PyPI 1.0.0b12 ↗	docs	GitHub 1.0.0b12 ↗

Name	Package	Docs	Source
Seguimiento básico de Opencensus	PyPI 1.0.0b10 ↗	docs	GitHub 1.0.0b10 ↗
Cosmos DB	PyPI 4.9.0 ↗ PyPI 4.14.0b2 ↗	docs	GitHub 4.9.0 ↗ GitHub 4.14.0b2 ↗
Defender EASM	PyPI 1.0.0b1 ↗	docs	GitHub 1.0.0b1 ↗
Centro de desarrollo	PyPI 1.0.0 ↗	docs	GitHub 1.0.0 ↗
Actualización de dispositivos	PyPI 1.0.0 ↗	docs	GitHub 1.0.0 ↗
Digital Twins	PyPI 1.3.0 ↗	docs	GitHub 1.3.0 ↗
Inteligencia Documental	PyPI 1.0.2 ↗	docs	GitHub 1.0.2 ↗
Traducción de documentos	PyPI 1.1.0 ↗	docs	GitHub 1.1.0 ↗
Event Grid	PyPI 4.22.0 ↗	docs	GitHub 4.22.0 ↗
Event Hubs	PyPI 5.15.0 ↗	docs	GitHub 5.15.0 ↗
Face	PyPI 1.0.0b2 ↗	docs	GitHub 1.0.0b2 ↗
FarmBeats	PyPI 1.0.0b2 ↗	docs	GitHub 1.0.0b2 ↗
Form Recognizer	PyPI 3.3.3 ↗	docs	GitHub 3.3.3 ↗
Anonimización de Salud	PyPI 1.0.0 ↗ PyPI 1.1.0b1 ↗	docs	GitHub 1.0.0 ↗ GitHub 1.1.0b1 ↗
Generación de perfiles de cáncer de Health Insights	PyPI 1.0.0b1 ↗	docs	GitHub 1.0.0b1 ↗
Coincidencia clínica de Health Insights	PyPI 1.0.0b1 ↗	docs	GitHub 1.0.0b1 ↗
Información de radiología de Health Insights	PyPI 1.1.0 ↗	docs	GitHub 1.1.0 ↗
Identity	PyPI 1.24.0 ↗	docs	GitHub 1.24.0 ↗
Agente de identidad	PyPI 1.3.0 ↗	docs	GitHub 1.3.0 ↗
Ánálisis de imágenes	PyPI 1.0.0 ↗	docs	GitHub 1.0.0 ↗
Key Vault: Administración	PyPI 4.6.0 ↗	docs	GitHub 4.6.0 ↗
Key Vault: certificados	PyPI 4.10.0 ↗	docs	GitHub 4.10.0 ↗
Key Vault: claves	PyPI 4.11.0 ↗	docs	GitHub 4.11.0 ↗
Key Vault - Secretos	PyPI 4.10.0 ↗	docs	GitHub 4.10.0 ↗
Key Vault: dominio de seguridad	PyPI 1.0.0b1 ↗	docs	GitHub 1.0.0b1 ↗

Name	Package	Docs	Source
Load Testing	PyPI 1.0.1 ↗ PyPI 1.1.0b1 ↗	docs	GitHub 1.0.1 ↗ GitHub 1.1.0b1 ↗
Machine Learning	PyPI 1.28.1 ↗	docs	GitHub 1.28.1 ↗
Machine Learning - Almacén de características	PyPI 1.0.1 ↗		GitHub 1.0.1 ↗
Puntos de conexión privados administrados	PyPI 0.4.0 ↗	docs	GitHub 0.4.0 ↗
Geolocalización en mapas	PyPI 1.0.0b3 ↗	docs	GitHub 1.0.0b3 ↗
Representación de mapas	PyPI 2.0.0b2 ↗	docs	GitHub 2.0.0b2 ↗
Ruta en mapas	PyPI 1.0.0b3 ↗	docs	GitHub 1.0.0b3 ↗
Búsqueda de mapas	PyPI 2.0.0b2 ↗	docs	GitHub 2.0.0b2 ↗
Análisis multimedia en el Edge	PyPI 1.0.0b2 ↗	docs	GitHub 1.0.0b2 ↗
Autenticación de Realidad Mixta	PyPI 1.0.0b1 ↗	docs	GitHub 1.0.0b1 ↗
Supervisión de la ingesta	PyPI 1.1.0 ↗	docs	GitHub 1.1.0 ↗
Supervisión de registros de consultas	PyPI 2.0.0 ↗	docs	GitHub 2.0.0 ↗
Supervisión de métricas de consulta	PyPI 1.0.0 ↗	docs	GitHub 1.0.0 ↗
Exportador de OpenTelemetry	PyPI 1.0.0b41 ↗	docs	GitHub 1.0.0b41 ↗
Personalizer	PyPI 1.0.0b1 ↗		GitHub 1.0.0b1 ↗
Cuenta de Purview	PyPI 1.0.0b1 ↗	docs	GitHub 1.0.0b1 ↗
Administración de Purview	PyPI 1.0.0b1 ↗	docs	GitHub 1.0.0b1 ↗
Mapa de datos de Purview	PyPI 1.0.0b2 ↗	docs	GitHub 1.0.0b2 ↗
Examen de Purview	PyPI 1.0.0b2 ↗	docs	GitHub 1.0.0b2 ↗
Uso compartido de Purview	PyPI 1.0.0b3 ↗	docs	GitHub 1.0.0b3 ↗
Flujo de trabajo de Purview	PyPI 1.0.0b2 ↗	docs	GitHub 1.0.0b2 ↗
Respuesta a preguntas	PyPI 1.1.0 ↗	docs	GitHub 1.1.0 ↗
Registro de esquemas	PyPI 1.3.0 ↗	docs	GitHub 1.3.0 ↗
Registro de esquema: Avro	PyPI 1.0.0 ↗	docs	GitHub 1.0.0 ↗
Service Bus	PyPI 7.14.2 ↗	docs	GitHub 7.14.2 ↗
Spark	PyPI 0.7.0 ↗	docs	GitHub 0.7.0 ↗

Name	Package	Docs	Source
Almacenamiento: Blobs	PyPI 12.26.0 ↗ PyPI 12.27.0b1 ↗	docs	GitHub 12.26.0 ↗ GitHub 12.27.0b1 ↗
Storage: Blobs Changefeed	PyPI 12.0.0b5 ↗	docs	GitHub 12.0.0b5 ↗
Storage: lago de datos de archivos	PyPI 12.21.0 ↗ PyPI 12.22.0b1 ↗	docs	GitHub 12.21.0 ↗ GitHub 12.22.0b1 ↗
Storage - Compartición de archivos	PyPI 12.22.0 ↗ PyPI 12.23.0b1 ↗	docs	GitHub 12.22.0 ↗ GitHub 12.23.0b1 ↗
- Colas de almacenamiento	PyPI 12.13.0 ↗ PyPI 12.14.0b1 ↗	docs	GitHub 12.13.0 ↗ GitHub 12.14.0b1 ↗
Synapse: AccessControl	PyPI 0.7.0 ↗	docs	GitHub 0.7.0 ↗
Synapse – Artifacts	PyPI 0.20.0 ↗	docs	GitHub 0.20.0 ↗
Synapse: supervisión	PyPI 0.2.0 ↗	docs	GitHub 0.2.0 ↗
Tables	PyPI 12.7.0 ↗	docs	GitHub 12.7.0 ↗
Text Analytics	PyPI 5.3.0 ↗	docs	GitHub 5.3.0 ↗
Traducción de texto	PyPI 1.0.1 ↗	docs	GitHub 1.0.1 ↗
TimeZones	PyPI 1.0.0b1 ↗	docs	GitHub 1.0.0b1 ↗
unknown	PyPI 1.0.0b1 ↗	docs	GitHub 1.0.0b1 ↗
Web PubSub	PyPI 1.3.0 ↗	docs	GitHub 1.3.0 ↗
Cliente de Web PubSub	PyPI 1.1.0 ↗	docs	GitHub 1.1.0 ↗
Core - Administración - Core	PyPI 1.6.0 ↗	docs	GitHub 1.6.0 ↗
Administración de recursos - Astro	PyPI 1.0.0b1 ↗	docs	GitHub 1.0.0b1 ↗
Administración de recursos: centro de desarrollo	PyPI 1.1.0 ↗	docs	GitHub 1.1.0 ↗
Administración de recursos - Elastic SAN	PyPI 1.1.0 ↗ PyPI 1.2.0b2 ↗	docs	GitHub 1.1.0 ↗ GitHub 1.2.0b2 ↗
Administración de recursos: centro de seguridad	PyPI 1.0.0b2 ↗	docs	GitHub 1.0.0b2 ↗
Administración de recursos: Advisor	PyPI 9.0.0 ↗ PyPI 10.0.0b1 ↗	docs	GitHub 9.0.0 ↗ GitHub 10.0.0b1 ↗
Administración de recursos: Agrifood	PyPI 1.0.0b3 ↗	docs	GitHub 1.0.0b3 ↗

Name	Package	Docs	Source
Administración de recursos - Centro para desarrolladores de AKS	PyPI 1.0.0b1 ↗ GitHub 1.0.0b1 ↗	docs	
Administración de recursos: administración de alertas	PyPI 1.0.0 ↗ PyPI 2.0.0b2 ↗	docs	GitHub 1.0.0 ↗ GitHub 2.0.0b2 ↗
Administración de recursos: Centro API	PyPI 1.0.0 ↗	docs	GitHub 1.0.0 ↗
Administración de recursos: API Management	PyPI 5.0.0 ↗	docs	GitHub 5.0.0 ↗
Administración de recursos: automatización del cumplimiento de aplicaciones	PyPI 1.0.0 ↗	docs	GitHub 1.0.0 ↗
Administración de recursos: Configuración de la aplicación	PyPI 5.0.0 ↗	docs	GitHub 5.0.0 ↗
Administración de recursos: App Service	PyPI 9.0.0 ↗	docs	GitHub 9.0.0 ↗
Administración de recursos: Application Insights	PyPI 4.1.0 ↗ PyPI 5.0.0b1 ↗	docs	GitHub 4.1.0 ↗ GitHub 5.0.0b1 ↗
Administración de recursos - Datos de Arc	PyPI 1.0.0 ↗ PyPI 2.0.0b1 ↗	docs	GitHub 1.0.0 ↗ GitHub 2.0.0b1 ↗
Administración de recursos: Arize AI Observability Eval	PyPI 1.0.0 ↗	docs	GitHub 1.0.0 ↗
Administración de recursos: atestación	PyPI 1.0.0 ↗ PyPI 2.0.0b1 ↗	docs	GitHub 1.0.0 ↗ GitHub 2.0.0b1 ↗
Administración de recursos: autorización	PyPI 4.0.0 ↗ PyPI 5.0.0b1 ↗	docs	GitHub 4.0.0 ↗ GitHub 5.0.0b1 ↗
Administración de recursos: Automanage	PyPI 1.0.0 ↗ PyPI 2.0.0b1 ↗	docs	GitHub 1.0.0 ↗ GitHub 2.0.0b1 ↗
Administración de recursos: Automatización	PyPI 1.0.0 ↗ PyPI 1.1.0b4 ↗	docs	GitHub 1.0.0 ↗ GitHub 1.1.0b4 ↗
Administración de recursos: Búsqueda de Azure AI	PyPI 9.2.0 ↗	docs	GitHub 9.2.0 ↗
Administración de recursos: Azure Stack	PyPI 1.0.0 ↗ PyPI 2.0.0b1 ↗	docs	GitHub 1.0.0 ↗ GitHub 2.0.0b1 ↗
Administración de recursos: Azure Stack HCI	PyPI 7.0.0 ↗ PyPI 8.0.0b4 ↗	docs	GitHub 7.0.0 ↗ GitHub 8.0.0b4 ↗
Administración de recursos: Azure VMware Solution	PyPI 9.1.0 ↗	docs	GitHub 9.1.0 ↗
Administración de recursos - Infraestructura BareMetal	PyPI 1.0.0 ↗ PyPI 1.1.0b2 ↗	docs	GitHub 1.0.0 ↗ GitHub 1.1.0b2 ↗

Name	Package	Docs	Source
Administración de recursos: Batch	PyPI 18.0.0 ↗	docs	GitHub 18.0.0 ↗
Administración de recursos: facturación	PyPI 7.0.0 ↗	docs	GitHub 7.0.0 ↗
Administración de recursos: ventajas de la facturación	PyPI 1.0.0b1 ↗	docs	GitHub 1.0.0b1 ↗
Administración de recursos: Bot Service	PyPI 2.0.0 ↗	docs	GitHub 2.0.0 ↗
Administración de recursos: optimización de carbono	PyPI 1.0.0 ↗	docs	GitHub 1.0.0 ↗
Administración de recursos: Análisis de cambios	PyPI 1.0.0 ↗	docs	GitHub 1.0.0 ↗
Administración de recursos: Chaos	PyPI 2.0.0 ↗	docs	GitHub 2.0.0 ↗
Administración de recursos: Cloudhealth	PyPI 1.0.0b1 ↗	docs	GitHub 1.0.0b1 ↗
Administración de recursos: Cognitive Services	PyPI 13.7.0 ↗	docs	GitHub 13.7.0 ↗
Administración de recursos: Commerce	PyPI 6.0.0 ↗ PyPI 6.1.0b1 ↗	docs	GitHub 6.0.0 ↗ GitHub 6.1.0b1 ↗
Administración de recursos: comunicación	PyPI 2.1.0 ↗	docs	GitHub 2.1.0 ↗
Administración de recursos: Compute	PyPI 35.0.0 ↗	docs	GitHub 35.0.0 ↗
Administración de recursos: flota de cómputo	PyPI 1.0.0 ↗	docs	GitHub 1.0.0 ↗
Administración de recursos: programación de cálculo	PyPI 1.1.0 ↗ PyPI 1.2.0b1 ↗	docs	GitHub 1.1.0 ↗ GitHub 1.2.0 B1 ↗
Administración de recursos: Confidential Ledger	PyPI 1.0.0 ↗ PyPI 2.0.0b5 ↗	docs	GitHub 1.0.0 ↗ GitHub 2.0.0b5 ↗
Administración de recursos - Confluent	PyPI 2.1.0 ↗	docs	GitHub 2.1.0 ↗
Administración de recursos: caché conectada	PyPI 1.0.0b1 ↗	docs	GitHub 1.0.0b1 ↗
Administración de recursos - VMWare conectado	PyPI 1.0.0 ↗	docs	GitHub 1.0.0 ↗
Administración de recursos: consumo	PyPI 10.0.0 ↗ PyPI 11.0.0b1 ↗	docs	GitHub 10.0.0 ↗ GitHub 11.0.0b1 ↗
Administración de recursos: Container Apps	PyPI 3.2.0 ↗	docs	GitHub 3.2.0 ↗
Administración de recursos: Container Instances	PyPI 10.1.0 ↗ PyPI 10.2.0b1 ↗	docs	GitHub 10.1.0 ↗ GitHub 10.2.0b1 ↗
Administración de recursos: entorno de ejecución de Container Orchestrator	PyPI 1.0.0b1 ↗	docs	GitHub 1.0.0b1 ↗

Name	Package	Docs	Source
Administración de recursos: Container Registry	PyPI 14.0.0 ↗ PyPI 14.1.0b1 ↗	docs	GitHub 14.0.0 ↗ GitHub 14.1.0b1 ↗
Administración de recursos: servicio de contenedor	PyPI 39.0.0 ↗	docs	GitHub 39.0.0 ↗
Administración de recursos: flota de servicio de contenedor	PyPI 3.1.0 ↗	docs	GitHub 3.1.0 ↗
Administración de recursos - Controles de seguridad de servicios de contenedores	PyPI 1.0.0b1 ↗	docs	GitHub 1.0.0b1 ↗
Administración de recursos: Content Delivery Network	PyPI 13.1.1 ↗	docs	GitHub 13.1.1 ↗
Administración de recursos: Cosmos DB	PyPI 9.8.0 ↗ PyPI 10.0.0b5 ↗	docs	GitHub 9.8.0 ↗ GitHub 10.0.0b5 ↗
Administración de recursos: Cosmos DB para PostgreSQL	PyPI 1.0.0 ↗ PyPI 1.1.0b1 ↗	docs	GitHub 1.0.0 ↗ GitHub 1.1.0b1 ↗
Administración de recursos: Gestión de costos	PyPI 4.0.1 ↗	docs	GitHub 4.0.1 ↗
Administración de recursos: proveedores personalizados	PyPI 1.0.0 ↗ PyPI 1.1.0b1 ↗	docs	GitHub 1.0.0 ↗ GitHub 1.1.0b1 ↗
Administración de recursos: Data Box	PyPI 3.1.0 ↗	docs	GitHub 3.1.0 ↗
Administración de recursos: Data Box Edge	PyPI 2.0.0 ↗ PyPI 3.0.0b1 ↗	docs	GitHub 2.0.0 ↗ GitHub 3.0.0b1 ↗
Administración de recursos: Data Factory	PyPI 9.2.0 ↗	docs	GitHub 9.2.0 ↗
Administración de recursos: Data Lake Analytics	PyPI 1.0.0b2 ↗	docs	GitHub 1.0.0b2 ↗
Administración de recursos: Data Lake Store	PyPI 1.0.0 ↗ PyPI 1.1.0b1 ↗	docs	GitHub 1.0.0 ↗
Administración de recursos: migración de datos	PyPI 10.0.0 ↗ PyPI 10.1.0b2 ↗	docs	GitHub 10.0.0 ↗ GitHub 10.1.0b2 ↗
Administración de recursos: protección de datos	PyPI 1.4.0 ↗	docs	GitHub 1.4.0 ↗
Administración de recursos: Data Share	PyPI 1.0.0 ↗ PyPI 1.1.0b1 ↗	docs	GitHub 1.0.0 ↗ GitHub 1.1.0b1 ↗
Administración de recursos: Monitor de base de datos	PyPI 1.0.0b1 ↗	docs	GitHub 1.0.0b1 ↗
Administración de recursos: Databricks	PyPI 2.0.0 ↗	docs	GitHub 2.0.0 ↗
Administración de recursos: Datadog	PyPI 2.1.0 ↗	docs	GitHub 2.1.0 ↗
Administración de recursos: Defender EASM	PyPI 1.0.0b1 ↗	docs	GitHub 1.0.0b1 ↗

Name	Package	Docs	Source
Administración de recursos: Dellstorage	PyPI 1.0.0b1	docs	GitHub 1.0.0b1
Administración de recursos: Mapa de dependencias	PyPI 1.0.0b1	docs	GitHub 1.0.0b1
Administración de recursos: Administrador de Despliegue	PyPI 1.0.0 PyPI 2.0.0b1	docs	GitHub 1.0.0 GitHub 2.0.0b1
Administración de recursos: virtualización de escritorio	PyPI 2.0.0	docs	GitHub 2.0.0
Administración de recursos: Dev Spaces	PyPI 1.0.0b3	docs	GitHub 1.0.0b3
Administración de recursos - Servicios de Aprovisionamiento de Dispositivos	PyPI 1.1.0 PyPI 1.2.0b2	docs	GitHub 1.1.0
Administración de recursos - Registro de dispositivos	PyPI 1.0.0	docs	GitHub 1.0.0
Administración de recursos: actualización de dispositivos	PyPI 1.1.0	docs	GitHub 1.1.0
Administración de recursos - Infraestructura de DevOps	PyPI 1.0.0	docs	GitHub 1.0.0
Administración de recursos: DevTest Labs	PyPI 9.0.0 PyPI 10.0.0b2	docs	GitHub 9.0.0 GitHub 10.0.0b2
Administración de recursos: Digital Twins	PyPI 7.0.0	docs	GitHub 7.0.0
Administración de recursos: DNS	PyPI 9.0.0	docs	GitHub 9.0.0
Administración de recursos: solucionador DNS	PyPI 1.1.0	docs	GitHub 1.1.0
Administración de recursos: Durable Task	PyPI 1.0.0b2	docs	GitHub 1.0.0b2
Administración de recursos: Dynatrace	PyPI 2.0.0	docs	GitHub 2.0.0
Administración de recursos: orden perimetral	PyPI 2.0.0	docs	GitHub 2.0.0
Administración de recursos - Edge Zones	PyPI 1.0.0b2	docs	GitHub 1.0.0b2
Administración de recursos - Educación	PyPI 1.0.0b2	docs	GitHub 1.0.0b2
Administración de recursos: elástico	PyPI 1.0.0 PyPI 1.1.0b4	docs	GitHub 1.0.0 GitHub 1.1.0b4
Administración de recursos: Event Grid	PyPI 10.4.0 PyPI 10.5.0b1	docs	GitHub 10.4.0 GitHub 10.5.0b1
Administración de recursos: Event Hubs	PyPI 11.2.0 PyPI 12.0.0b1	docs	GitHub 11.2.0 GitHub 12.0.0b1
Administración de recursos: ubicación ampliada	PyPI 2.0.0	docs	GitHub 2.0.0
Administración de recursos - Fabric	PyPI 1.0.0	docs	GitHub 1.0.0

Name	Package	Docs	Source
Administración de recursos: Fluid Relay	PyPI 1.0.0 ↗ PyPI 1.1.0b1 ↗	docs	GitHub 1.0.0 ↗ GitHub 1.1.0b1 ↗
Administración de recursos: Front Door	PyPI 1.2.0 ↗	docs	GitHub 1.2.0 ↗
Administración de recursos: Servicios de Graph	PyPI 1.0.0 ↗	docs	GitHub 1.0.0 ↗
Administración de recursos: configuración de invitados	PyPI 1.0.0b2 ↗	docs	GitHub 1.0.0b2 ↗
Administración de recursos: HANA en Azure	PyPI 1.0.0 ↗ PyPI 1.1.0b1 ↗	docs	GitHub 1.0.0 ↗ GitHub 1.1.0b1 ↗
Administración de recursos: módulos de seguridad de hardware	PyPI 1.0.0 ↗	docs	GitHub 1.0.0 ↗
Administración de recursos: HDInsight	PyPI 9.0.0 ↗ PyPI 9.1.0b1 ↗	docs	GitHub 9.0.0 ↗ GitHub 9.1.0b1 ↗
Gestión de recursos - Contenedores HDInsight	PyPI 1.0.0b3 ↗	docs	GitHub 1.0.0b3 ↗
Administración de recursos: Health Bot	PyPI 1.0.0b2 ↗	docs	GitHub 1.0.0b2 ↗
Administración de recursos: Health Data AI Services	PyPI 1.0.0 ↗	docs	GitHub 1.0.0 ↗
Administración de recursos: API de sanidad	PyPI 2.1.0 ↗	docs	GitHub 2.1.0 ↗
Administración de recursos: computación híbrida	PyPI 9.0.0 ↗ PyPI 9.1.0b2 ↗	docs	GitHub 9.0.0 ↗ GitHub 9.1.0b2 ↗
Administración de recursos: conectividad híbrida	PyPI 1.0.0 ↗ PyPI 2.0.0b1 ↗	docs	GitHub 1.0.0 ↗ GitHub 2.0.0b1 ↗
Administración de recursos: Servicio de contenedor híbrido	PyPI 1.0.0 ↗	docs	GitHub 1.0.0 ↗
Administración de recursos: Kubernetes híbrido	PyPI 1.1.0 ↗ PyPI 1.2.0b2 ↗	docs	GitHub 1.1.0 ↗ GitHub 1.2.0b2 ↗
Administración de recursos: red híbrida	PyPI 2.0.0 ↗	docs	GitHub 2.0.0 ↗
Administración de recursos: Image Builder	PyPI 1.4.0 ↗	docs	GitHub 1.4.0 ↗
Administración de recursos: informes de impacto	PyPI 1.0.0b1 ↗	docs	GitHub 1.0.0b1 ↗
Administración de recursos - Informatica Data Management	PyPI 1.0.0 ↗	docs	GitHub 1.0.0 ↗
Administración de recursos: IoT Central	PyPI 9.0.0 ↗ PyPI 10.0.0b2 ↗	docs	GitHub 9.0.0 ↗ GitHub 10.0.0b2 ↗

Name	Package	Docs	Source
Administración de recursos: IoT Firmware Defense	PyPI 1.0.0 ↗ PyPI 2.0.0b1 ↗	docs	GitHub 1.0.0 ↗ GitHub 2.0.0b1 ↗
Administración de recursos: IoT Hub	PyPI 4.0.0 ↗ PyPI 5.0.0b1 ↗	docs	GitHub 4.0.0 ↗ GitHub 5.0.0b1 ↗
Administración de recursos: operaciones de IoT	PyPI 1.0.0 ↗	docs	GitHub 1.0.0 ↗
Administración de recursos: Key Vault	PyPI 12.0.0 ↗	docs	GitHub 12.0.0 ↗
Administración de recursos: configuración de Kubernetes	PyPI 3.1.0 ↗	docs	GitHub 3.1.0 ↗
Administración de recursos: Kubernetesconfiguration-Extensions	PyPI 1.0.0b1 ↗	docs	GitHub 1.0.0b1 ↗
Administración de recursos: Kubernetesconfiguration-Extensiontypes	PyPI 1.0.0b1 ↗	docs	GitHub 1.0.0b1 ↗
Administración de recursos: Kubernetesconfiguration-Fluxconfigurations	PyPI 1.0.0b1 ↗	docs	GitHub 1.0.0b1 ↗
Administración de recursos: Kubernetesconfiguration-Privatelinkscopes	PyPI 1.0.0b1 ↗	docs	GitHub 1.0.0b1 ↗
Administración de recursos: Kusto	PyPI 3.4.0 ↗	docs	GitHub 3.4.0 ↗
Administración de recursos: servicios de laboratorio	PyPI 2.0.0 ↗ PyPI 2.1.0b1 ↗	docs	GitHub 2.0.0 ↗ GitHub 2.1.0b1 ↗
Administración de recursos - Lambdatesthyperexecute	PyPI 1.0.0 ↗	docs	GitHub 1.0.0 ↗
Administración de recursos: instancia grande	PyPI 1.0.0b1 ↗	docs	GitHub 1.0.0b1 ↗
Administración de recursos: prueba de carga	PyPI 1.0.0 ↗	docs	GitHub 1.0.0 ↗
Administración de recursos: Log Analytics	PyPI 12.0.0 ↗ PyPI 13.0.0b7 ↗	docs	GitHub 12.0.0 ↗ GitHub 13.0.0b7 ↗
Administración de recursos: Logic Apps	PyPI 10.0.0 ↗ PyPI 10.1.0b1 ↗	docs	GitHub 10.0.0 ↗ GitHub 10.1.0b1 ↗
Administración de recursos - Computación para Aprendizaje Automático	PyPI 1.0.0b2 ↗	docs	GitHub 1.0.0b2 ↗
Administración de recursos: Machine Learning Services	PyPI 1.0.0 ↗ PyPI 2.0.0b2 ↗	docs	GitHub 1.0.0 ↗ GitHub 2.0.0b2 ↗
Administración de recursos: mantenimiento	PyPI 2.1.0 ↗ PyPI 2.2.0b2 ↗	docs	GitHub 2.1.0 ↗ GitHub 2.2.0b2 ↗

Name	Package	Docs	Source
Administración de recursos: aplicaciones administradas	PyPI 1.0.0b1 ↗	docs	GitHub 1.0.0b1 ↗
Administración de recursos: Grafana administrado	PyPI 1.1.0 ↗ PyPI 2.0.0b1 ↗	docs	GitHub 1.1.0 ↗ GitHub 2.0.0b1 ↗
Administración de recursos: tejido de red administrada	PyPI 1.0.0 ↗	docs	GitHub 1.0.0 ↗
Administración de recursos: identidad de servicio administrado	PyPI 7.1.0 ↗	docs	GitHub 7.1.0 ↗
Administración de recursos: servicios administrados	PyPI 6.0.0 ↗ PyPI 7.0.0b2 ↗	docs	GitHub 6.0.0 ↗ GitHub 7.0.0b2 ↗
Administración de recursos: Grupos de administración	PyPI 1.0.0 ↗ PyPI 1.1.0b2 ↗	docs	GitHub 1.0.0 ↗ GitHub 1.1.0b2 ↗
Administración de recursos: asociado de administración	PyPI 1.0.0 ↗ PyPI 1.1.0b1 ↗	docs	GitHub 1.0.0 ↗ GitHub 1.1.0b1 ↗
Administración de recursos: Mapas	PyPI 2.1.0 ↗	docs	GitHub 2.1.0 ↗
Administración de recursos: pedidos de Marketplace	PyPI 1.1.0 ↗ PyPI 1.2.0b2 ↗	docs	GitHub 1.1.0 ↗ GitHub 1.2.0b2 ↗
Administración de recursos: evaluación de la migración	PyPI 1.0.0b1 ↗	docs	GitHub 1.0.0b1 ↗
Administración de recursos - SAP de detección de migración	PyPI 1.0.0b1 ↗	docs	GitHub 1.0.0b1 ↗
Administración de recursos: Mixed Reality	PyPI 1.0.0 ↗ PyPI 1.1.0b1 ↗	docs	GitHub 1.0.0 ↗ GitHub 1.1.0b1 ↗
Administración de recursos: red móvil	PyPI 3.3.0 ↗	docs	GitHub 3.3.0 ↗
Administración de recursos - Clúster de Mongo	PyPI 1.0.0 ↗ PyPI 1.1.0b1 ↗	docs	GitHub 1.0.0 ↗ GitHub 1.1.0b1 ↗
Administración de recursos - Mongodbatlas	PyPI 1.0.0 ↗	docs	GitHub 1.0.0 ↗
Administración de recursos: Monitoreo	PyPI 7.0.0 ↗ PyPI 8.0.0b1 ↗	docs	GitHub 7.0.0 ↗ GitHub 8.0.0b1 ↗
Administración de recursos: servidores flexibles de MySQL	PyPI 1.0.0b3 ↗	docs	GitHub 1.0.0b3 ↗
Administración de recursos: Neon Postgres	PyPI 1.0.0 ↗	docs	GitHub 1.0.0 ↗
Administración de recursos: NetApp Files	PyPI 13.6.0 ↗ PyPI 14.0.0b1 ↗	docs	GitHub 13.6.0 ↗ GitHub 14.0.0b1 ↗
Administración de recursos: Red	PyPI 29.0.0 ↗	docs	GitHub 29.0.0 ↗

Name	Package	Docs	Source
Administración de recursos: función de red	PyPI 1.0.0b1	docs	GitHub 1.0.0b1
Administración de recursos: Networkcloud	PyPI 2.1.0	docs	GitHub 2.1.0
Administración de recursos - New Relic Observability	PyPI 1.1.0	docs	GitHub 1.1.0
Administración de recursos: Nginx	PyPI 3.0.0 PyPI 3.1.0b2	docs	GitHub 3.0.0 GitHub 3.1.0b2
Administración de recursos: Notification Hubs	PyPI 8.0.0 PyPI 8.1.0b2	docs	GitHub 8.0.0 GitHub 8.1.0b2
Administración de recursos - Oep	PyPI 1.0.0b2	docs	GitHub 1.0.0b2
Administración de recursos - Onlineexperimentation	PyPI 1.0.0b1	docs	GitHub 1.0.0b1
Administración de recursos: administración de operaciones	PyPI 1.0.0 PyPI 2.0.0b1	docs	GitHub 1.0.0 GitHub 2.0.0b1
Administración de recursos - Oracle Database	PyPI 2.0.0	docs	GitHub 2.0.0
Administración de recursos: Orbital	PyPI 2.0.0	docs	GitHub 2.0.0
Administración de recursos: Palo Alto Networks - Firewall de próxima generación	PyPI 1.0.0 PyPI 2.0.0b1	docs	GitHub 1.0.0 GitHub 2.0.0b1
Administración de recursos: emparejamiento	PyPI 1.0.0 PyPI 2.0.0b1	docs	GitHub 1.0.0 GitHub 2.0.0b1
Administración de recursos: base de datos vectorial de Pinecone	PyPI 1.0.0b2	docs	GitHub 1.0.0b2
Administración de recursos: Planetarycomputer	PyPI 1.0.0b1	docs	GitHub 1.0.0b1
Gestión de recursos - Playwright	PyPI 1.0.0b1	docs	GitHub 1.0.0b1
Administración de recursos - Playwright Testing	PyPI 1.0.0	docs	GitHub 1.0.0
Administración de recursos - Perspectivas de políticas	PyPI 1.0.0 PyPI 1.1.0b5	docs	GitHub 1.0.0 GitHub 1.1.0b5
Administración de recursos - Portal	PyPI 1.0.0 PyPI 1.1.0b1	docs	GitHub 1.0.0 GitHub 1.1.0b1
Administración de recursos - Portalservicescopilot	PyPI 1.0.0b1	docs	GitHub 1.0.0b1
Administración de recursos: PostgreSQL	PyPI 10.1.0 PyPI 10.2.0b18	docs	GitHub 10.1.0 GitHub 10.2.0b18
Administración de recursos: servidores flexibles de PostgreSQL	PyPI 1.1.0 PyPI 1.2.0b1	docs	GitHub 1.1.0 GitHub 1.2.0 B1

Name	Package	Docs	Source
Administración de recursos: Power BI dedicado	PyPI 1.0.0 ↗ PyPI 1.1.0b1 ↗	docs	GitHub 1.0.0 ↗ GitHub 1.1.0b1 ↗
Administración de recursos: DNS privado	PyPI 1.2.0 ↗	docs	GitHub 1.2.0 ↗
Administración de recursos - Purestorageblock	PyPI 1.0.0 ↗	docs	GitHub 1.0.0 ↗
Administración de recursos: Purview	PyPI 1.0.0 ↗ PyPI 1.1.0b1 ↗	docs	GitHub 1.0.0 ↗ GitHub 1.1.0b1 ↗
Administración de recursos: Quantum	PyPI 1.0.0b5 ↗	docs	GitHub 1.0.0b5 ↗
Administración de recursos: Qumulo	PyPI 2.0.0 ↗	docs	GitHub 2.0.0 ↗
Administración de recursos: cuota	PyPI 2.0.0 ↗	docs	GitHub 2.0.0 ↗
Administración de recursos: Recovery Services	PyPI 3.1.0 ↗	docs	GitHub 3.1.0 ↗
Administración de recursos: copia de seguridad de Recovery Services	PyPI 9.2.0 ↗	docs	GitHub 9.2.0 ↗
Administración de recursos: replicación de datos de Recovery Services	PyPI 1.0.0 ↗	docs	GitHub 1.0.0 ↗
Administración de recursos: Site Recovery de Recovery Services	PyPI 1.3.0 ↗	docs	GitHub 1.3.0 ↗
Administración de recursos: Red Hat OpenShift	PyPI 2.0.0 ↗	docs	GitHub 2.0.0 ↗
Administración de recursos: Redis	PyPI 14.5.0 ↗	docs	GitHub 14.5.0 ↗
Administración de recursos: Redis Enterprise	PyPI 3.0.0 ↗ PyPI 3.1.0b4 ↗	docs	GitHub 3.0.0 ↗ GitHub 3.1.0b4 ↗
Administración de recursos: Relay	PyPI 1.1.0 ↗ PyPI 2.0.0b1 ↗	docs	GitHub 1.1.0 ↗ GitHub 2.0.0b1 ↗
Administración de recursos: reservas	PyPI 2.3.0 ↗	docs	GitHub 2.3.0 ↗
Administración de recursos: Conector de recursos	PyPI 1.0.0 ↗	docs	GitHub 1.0.0 ↗
Administración de recursos: Resource Graph	PyPI 8.0.0 ↗ PyPI 8.1.0b3 ↗	docs	GitHub 8.0.0 ↗ GitHub 8.1.0b3 ↗
Administración de recursos - Salud de recursos	PyPI 1.0.0b6 ↗	docs	GitHub 1.0.0b6 ↗
Administración de recursos: Resource Mover	PyPI 1.1.0 ↗	docs	GitHub 1.1.0 ↗
Administración de recursos: Resource-Bicep	PyPI 1.0.0b1 ↗	docs	GitHub 1.0.0b1 ↗
Administración de recursos: Resource-Deployments	PyPI 1.0.0b1 ↗	docs	GitHub 1.0.0b1 ↗

Name	Package	Docs	Source
Administración de recursos: Resource-Deploymentscripts	PyPI 1.0.0b1	docs	GitHub 1.0.0b1
Administración de recursos: Resource-Deploymentstacks	PyPI 1.0.0b1	docs	GitHub 1.0.0b1
Administración de recursos: Resource-Templatespecs	PyPI 1.0.0b1	docs	GitHub 1.0.0b1
Administración de recursos: Recursos	PyPI 24.0.0 PyPI 25.0.0b1	docs	GitHub 24.0.0 GitHub 25.0.0b1
Administración de recursos: Recursos	PyPI 24.0.0 PyPI 25.0.0b1	docs	GitHub 24.0.0 GitHub 25.0.0b1
Administración de recursos - Scvmm	PyPI 1.0.0	docs	GitHub 1.0.0
Administración de recursos - Secretsstoreextension	PyPI 1.0.0b1	docs	GitHub 1.0.0b1
Administración de recursos: seguridad	PyPI 7.0.0	docs	GitHub 7.0.0
Administración de recursos: Security Insights	PyPI 1.0.0 PyPI 2.0.0b2	docs	GitHub 1.0.0 GitHub 2.0.0b2
Administración de recursos: autoayuda	PyPI 1.0.0 PyPI 2.0.0b4	docs	GitHub 1.0.0 GitHub 2.0.0b4
Administración de recursos: consola serie	PyPI 1.0.0 PyPI 1.1.0b1	docs	GitHub 1.0.0 GitHub 1.1.0b1
Administración de recursos: Service Bus	PyPI 9.0.0 PyPI 10.0.0b1	docs	GitHub 9.0.0 GitHub 10.0.0b1
Administración de recursos: Service Fabric	PyPI 2.1.0 PyPI 2.2.0b1	docs	GitHub 2.1.0 GitHub 2.2.0b1
Administración de recursos: clústeres administrados de Service Fabric	PyPI 2.0.0 PyPI 2.1.0b3	docs	GitHub 2.0.0 GitHub 2.1.0b3
Administración de recursos: Service Linker	PyPI 1.1.0 PyPI 1.2.0b3	docs	GitHub 1.1.0 GitHub 1.2.0b3
Administración de recursos: Redes de servicio	PyPI 2.0.0 PyPI 2.1.0b1	docs	GitHub 2.0.0 GitHub 2.1.0b1
Administración de recursos: SignalR	PyPI 1.2.0 PyPI 2.0.0b2	docs	GitHub 1.2.0 GitHub 2.0.0b2
Administración de recursos: Sitemanager	PyPI 1.0.0b1	docs	GitHub 1.0.0b1
Administración de recursos: Sphere	PyPI 1.0.0	docs	GitHub 1.0.0
Administración de recursos - Detección de aplicaciones de	PyPI 1.0.0b1	docs	GitHub 1.0.0b1

Name	Package	Docs	Source
Spring			
Administración de recursos: SQL	PyPI 3.0.1 ↗ PyPI 4.0.0b22 ↗	docs	GitHub 3.0.1 ↗ GitHub 4.0.0b22 ↗
Administración de recursos: máquina virtual de SQL	PyPI 1.0.0b6 ↗	docs	GitHub 1.0.0b6 ↗
Administración de recursos - Grupo en espera	PyPI 2.0.0 ↗	docs	GitHub 2.0.0 ↗
Administración de recursos: Storage	PyPI 23.0.1 ↗	docs	GitHub 23.0.1 ↗
Administración de recursos - Acciones de almacenamiento	PyPI 1.0.0 ↗	docs	GitHub 1.0.0 ↗
Administración de recursos: caché de almacenamiento	PyPI 2.0.0 ↗	docs	GitHub 2.0.0 ↗
Administración de recursos: Storage Mover	PyPI 2.1.0 ↗	docs	GitHub 2.1.0 ↗
Administración de recursos: grupo de almacenamiento	PyPI 1.0.0 ↗ PyPI 1.1.0b1 ↗	docs	GitHub 1.0.0 ↗ GitHub 1.1.0b1 ↗
Administración de recursos: sincronización de almacenamiento	PyPI 1.0.0 ↗ PyPI 2.0.0b1 ↗	docs	GitHub 1.0.0 ↗ GitHub 2.0.0b1 ↗
Administración de recursos: Storagediscovery	PyPI 1.0.0b1 ↗	docs	GitHub 1.0.0b1 ↗
Administración de recursos: Stream Analytics	PyPI 1.0.0 ↗ PyPI 2.0.0b2 ↗	docs	GitHub 1.0.0 ↗ GitHub 2.0.0b2 ↗
Administración de recursos: suscripciones	PyPI 3.1.1 ↗ PyPI 3.2.0b1 ↗	docs	GitHub 3.1.1 ↗ GitHub 3.2.0b1 ↗
Administración de recursos: soporte técnico	PyPI 7.0.0 ↗	docs	GitHub 7.0.0 ↗
Administración de recursos: Synapse	PyPI 2.0.0 ↗ PyPI 2.1.0b7 ↗	docs	GitHub 2.0.0 ↗ GitHub 2.1.0b7 ↗
Administración de recursos: Terraform	PyPI 1.0.0b1 ↗	docs	GitHub 1.0.0b1 ↗
Administración de recursos: Time Series Insights	PyPI 1.0.0 ↗ PyPI 2.0.0b1 ↗	docs	GitHub 1.0.0 ↗ GitHub 2.0.0b1 ↗
Administración de recursos: Administrador de Tráfico	PyPI 1.1.0 ↗	docs	GitHub 1.1.0 ↗
Administración de recursos: Trusted Signing	PyPI 1.0.0b1 ↗	docs	GitHub 1.0.0b1 ↗
Administración de recursos: VMware Solution by CloudSimple	PyPI 1.0.0b2 ↗	docs	GitHub 1.0.0b2 ↗
Administración de recursos: Servicios de Voz	PyPI 1.0.0 ↗	docs	GitHub 1.0.0 ↗
Administración de recursos: Web PubSub	PyPI 2.0.0 ↗	docs	GitHub 2.0.0 ↗

Name	Package	Docs	Source
Administración de recursos: ponderaciones y sesgos	PyPI 1.0.0b1 ↗	docs	GitHub 1.0.0b1 ↗
Administración de recursos: cargas de trabajo	PyPI 1.0.0 ↗	docs	GitHub 1.0.0 ↗
Administración de recursos - Instancia virtual SAP de gestión de cargas de trabajo	PyPI 1.0.0 ↗	docs	GitHub 1.0.0 ↗
azure-communication-administration	PyPI 1.0.0b4 ↗		
azureml-fsspec	PyPI 1.0.0 ↗		
Batch	PyPI 14.2.0 ↗ PyPI 15.0.0b2 ↗	docs	GitHub 14.2.0 ↗ GitHub 15.0.0b2 ↗
Core - Cliente - Seguimiento de OpenCensus	PyPI 1.0.0b10 ↗	docs	GitHub 1.0.0b10 ↗
Servicios de Aprovisionamiento de Dispositivos	PyPI 1.0.0b1 ↗		
Servicios de Aprovisionamiento de Dispositivos	PyPI 1.2.0 ↗		
Ink Recognizer	PyPI 1.0.0b1 ↗		GitHub 1.0.0b1 ↗
Dispositivo de IoT	PyPI 2.12.0 ↗ PyPI 3.0.0b2 ↗		
IoT Hub	PyPI 2.7.0 ↗		
iotedgedev	PyPI 3.3.7 ↗		
iotedgehubdev	PyPI 0.14.18 ↗		
Key Vault	PyPI 4.2.0 ↗		GitHub 4.2.0 ↗
Datos de Kusto	PyPI 2.0.0 ↗		
Machine Learning	PyPI 1.2.0 ↗		
Aprendizaje Automático - Tabla	PyPI 1.3.0 ↗		
Supervisión de Machine Learning	PyPI 0.1.0a3 ↗		
Personalizer	PyPI 0.1.0 ↗		GitHub 0.1.0 ↗
Administración de Purview	PyPI 1.0.0b1 ↗		GitHub 1.0.0b1 ↗
Service Fabric	PyPI 8.2.0.0 ↗	docs	GitHub 8.2.0.0 ↗
Speech	PyPI 1.14.0 ↗		
Storage	PyPI 0.37.0 ↗		GitHub 0.37.0 ↗

Name	Package	Docs	Source
Storage: lago de datos de archivos	PyPI 0.0.51 ↗		
Text Analytics	PyPI 1.0.2 ↗		
Uamqp	PyPI 1.6.11 ↗		GitHub 1.6.11 ↗
azure-agrifood-nspkg	PyPI 1.0.0 ↗		
azure-ai-language-nspkg	PyPI 1.0.0 ↗		
azure-ai-translation-nspkg	PyPI 1.0.0 ↗		
azure-iot-nspkg	PyPI 1.0.1 ↗		
azure-media-nspkg	PyPI 1.0.0 ↗		
azure-messaging-nspkg	PyPI 1.0.0 ↗		
azure-mixedreality-nspkg	PyPI 1.0.0 ↗		
azure-monitor-nspkg	PyPI 1.0.0 ↗		
azure-purview-nspkg	PyPI 2.0.0 ↗		
azure-security-nspkg	PyPI 1.0.0 ↗		
Paquete de espacio de nombres de conocimiento de Cognitive Services	PyPI 3.0.0 ↗		GitHub 3.0.0 ↗
Paquete de espacio de nombres de idiomas de Cognitive Services	PyPI 3.0.1 ↗		GitHub 3.0.1 ↗
Paquete de espacio de nombres de Cognitive Services	PyPI 3.0.1 ↗		GitHub 3.0.1 ↗
Paquete de espacio de nombres de búsqueda de Cognitive Services	PyPI 3.0.1 ↗		GitHub 3.0.1 ↗
Paquete de espacio de nombres de visión de Cognitive Services	PyPI 3.0.1 ↗		GitHub 3.0.1 ↗
Paquete de espacio de nombres de comunicación	PyPI 0.0.0b1 ↗	docs	
Paquete de espacio de nombres principal	PyPI 3.0.2 ↗		GitHub 3.0.2 ↗
Paquete de espacio de nombres de datos	PyPI 1.0.0 ↗	docs	
Paquete de espacio de nombres de Digital Twins	PyPI 1.0.0 ↗		
Paquete de espacio de nombres de Key Vault	PyPI 1.0.0 ↗		GitHub 1.0.0 ↗
Paquete de espacio de nombres de búsqueda	PyPI 1.0.0 ↗		GitHub 1.0.0 ↗

Name	Package	Docs	Source
Paquete de espacio de nombres de Storage	PyPI 3.1.0 ↗		GitHub 3.1.0 ↗
Paquete de espacio de nombres de Synapse	PyPI 1.0.0 ↗		GitHub 1.0.0 ↗
Paquete de espacio de nombres de Text Analytics	PyPI 1.0.0 ↗		GitHub 1.0.0 ↗
Administración de recursos	PyPI 5.0.0 ↗		GitHub 5.0.0 ↗
Administración de recursos: común	PyPI 0.20.0 ↗		
apiview-stub-generator	PyPI 0.3.7 ↗		
azure-pylint-guidelines-checker	PyPI 0.0.8 ↗		
Herramientas de desarrollo	PyPI 1.2.0 ↗		GitHub 1.2.0 ↗
Doc Warden	PyPI 0.7.2 ↗		GitHub 0.7.2 ↗
Tox Monorepo	PyPI 0.1.2 ↗		GitHub 0.1.2 ↗

Reference

Services

[Active Directory](#)

[Advisor](#)

[Authorization](#)

[Batch AI](#)

[Batch](#)

[Billing](#)

[CDN](#)

[Cognitive Services](#)

[Commerce](#)

[Consumption](#)

[Container Instance](#)

[Container Registry](#)

[Container Service](#)

[Cosmos DB](#)

[Data Box Edge / Data Box Gateway](#)

[Data Factory](#)

[Data Lake Analytics](#)

[Data Lake Store](#)

[Deployment Manager](#)

[DevTest Labs](#)

[DNS](#)

[Event Hub](#)

[Event Grid](#)

[Functions](#)

[Graph RBAC](#)

[Management Groups](#)

[Hana on Azure](#)

[HDInsight](#)

[IoT](#)

[Key Vault](#)

[Lab Services](#)

[Log Analytics](#)

[Logic Apps](#)

[Managed Services](#)

[Media Services](#)

[Mixed Reality](#)

[Monitoring](#)

[PostgreSQL/MySQL](#)

[Network](#)

[Notification Hubs](#)

[Policy](#)

[Power BI](#)

[Recovery Services](#)

[Recovery Services Backup](#)

[Redis](#)

[Relay](#)

[Resources](#)

[Resource Graph](#)

[Scheduler](#)

[Search](#)

[Service Bus](#)

[Service Fabric](#)

[SQL](#)

[Storage](#)

[Virtual Machines](#)

[Traffic Manager](#)

[Web Apps](#)

[Other](#)

Hospedaje de aplicaciones de Python en Azure

21/05/2025

Azure ofrece varias opciones para hospedar la aplicación, cada una adecuada para distintos niveles de control y responsabilidad. Para obtener información general sobre estas opciones, consulte [Hospedaje de aplicaciones en Azure](#).

En general, seleccionar una opción de hospedaje implica equilibrar el control con la responsabilidad de la administración. Cuanto más control requiera sobre la infraestructura, más responsabilidad tendrá para la administración de uno o varios recursos.

Se recomienda empezar con Azure App Service, que proporciona un entorno altamente administrado con una sobrecarga administrativa mínima. A medida que evolucionan sus necesidades, puede explorar otras opciones que ofrecen mayor flexibilidad y control, como Azure Container Apps, Azure Kubernetes Service (AKS) o, en última instancia, Azure Virtual Machines, que proporcionan el mayor control, pero también requieren la mayor cantidad de mantenimiento.

Las opciones de hospedaje de este artículo se presentan en orden desde más gestionadas (menos responsabilidad de su parte) a menos gestionadas (más control y responsabilidad).

- **Hospedaje de aplicaciones web con Azure App Service:**
 - [Inicio rápido: implementación de una aplicación web de Python \(Django o Flask\) en Azure App Service](#)
 - [Implementación de una aplicación web Python \(Django o Flask\) con PostgreSQL en Azure](#)
 - [Creación e implementación de una aplicación web de Flask en Azure con una identidad administrada asignada automáticamente por el sistema](#)
 - [Configuración de una aplicación de Python para Azure App Service](#)
- **Red de entrega de contenido con aplicaciones web estáticas de Azure**
 - [Hospedaje de sitios web estáticos en Azure Storage](#)
 - [Inicio rápido: creación del primer sitio estático con Azure Static Web Apps](#)
- **Hospedaje sin servidor con Azure Functions:**
 - [Inicio rápido: Creación de una función de Python en Azure desde la línea de comandos](#)
 - [Inicio rápido: Creación de una función en Azure con Python mediante Visual Studio Code](#)
 - [Conexión de Azure Functions a Azure Storage mediante herramientas de línea de comandos](#)

- Conexión de Azure Functions a Azure Storage mediante Visual Studio Code
- **Hospedaje de contenedores con Azure:**
 - Introducción a python Container Apps en Azure
 - Implementación de un contenedor en App Service
 - Implementación de un contenedor en Azure Container Apps
 - Inicio rápido: Implementación de un clúster de Azure Kubernetes Service mediante la CLI de Azure
 - Implementación de un contenedor en Azure Container Instances mediante la CLI de Azure
 - Cree la primera aplicación de contenedor de Service Fabric en Linux
- **Operaciones intensivas en cómputo y de larga duración con Azure Batch**
 - Uso de Python para crear y ejecutar un trabajo de Azure Batch
 - Tutorial: Ejecución de una carga de trabajo de procesamiento de archivos paralelos con Azure Batch mediante Python
 - Tutorial: Ejecución de scripts de Python mediante Azure Data Factory mediante Azure Batch
- **Recursos informáticos escalables a petición con Azure Virtual Machines:**
 - Inicio rápido: Uso de la CLI de Azure para implementar una máquina virtual Linux en Azure

Soluciones de datos para aplicaciones de Python en Azure

05/06/2025

Azure ofrece una amplia gama de soluciones de almacenamiento y base de datos totalmente administradas, incluidas las bases de datos relacionales, NoSQL y en memoria, con compatibilidad con tecnologías propietarias y de código abierto. También puede elegir entre los servicios de objetos, bloques y almacenamiento de archivos. Los artículos siguientes pueden ayudarle a empezar a usar estas opciones con Python en Azure.

Bases de datos

- **PostgreSQL:** compile aplicaciones empresariales escalables, seguras y totalmente administradas mediante PostgreSQL de código abierto. Puede escalar PostgreSQL de un solo nodo para un alto rendimiento o migrar cargas de trabajo de PostgreSQL y Oracle existentes a la nube.
 - [Inicio rápido: Uso de Python para conectarse y consultar datos en Azure Database for PostgreSQL: servidor flexible](#)
 - [Inicio rápido: Uso de Python para conectarse y consultar datos en Azure Database for PostgreSQL: servidor único](#)
 - [Desplegar una aplicación web de Python \(Django o Flask\) con PostgreSQL en Azure App Service](#)
- **MySQL:** cree aplicaciones escalables mediante una base de datos MySQL totalmente administrada e inteligente en la nube.
 - [Inicio rápido: Use Python para conectarse a datos y consultarlos en Azure Database for MySQL con la opción Servidor flexible](#)
 - [Inicio rápido: Uso de Python para conectarse y consultar datos en Azure Database for MySQL](#)
- **Azure SQL:** cree aplicaciones escalables con una plataforma de base de datos SQL totalmente administrada e inteligente en la nube.
 - [Inicio rápido: Uso de Python para consultar una base de datos de Azure SQL o Azure SQL Managed Instance](#)

NoSQL, blobs, tablas, archivos, grafos y memorias caché

- **Cosmos DB:** cree aplicaciones de baja latencia, alta disponibilidad a escala global o migre Cassandra, MongoDB y otras cargas de trabajo noSQL a la nube.
 - [Inicio rápido: Biblioteca cliente de Azure Cosmos DB for NoSQL para Python](#)
 - [Inicio rápido: Azure Cosmos DB para MongoDB para Python con el controlador de MongoDB](#)
 - [Inicio rápido: Compilación de una aplicación de Cassandra con el SDK de Python y Azure Cosmos DB](#)
 - [Inicio rápido: Creación de una aplicación de API para Table con el SDK de Python y Azure Cosmos DB](#)
 - [Inicio rápido: Biblioteca de Azure Cosmos DB for Apache Gremlin para Python](#)
- **Blob Storage:** almacenamiento seguro y escalable de objetos para aplicaciones nativas de nube, lagos de datos, archivos, informática de alto rendimiento (HPC) y aprendizaje automático.
 - [Inicio rápido: Biblioteca cliente de Azure Blob Storage para Python](#)
 - [Ejemplos de Azure Storage con bibliotecas cliente de Python v12](#)
- **Azure Data Lake Storage Gen2:** lago de datos escalable y seguro optimizado para análisis de alto rendimiento.
 - [Uso de Python para administrar directorios y archivos en Azure Data Lake Storage Gen2](#)
 - [Uso de Python para administrar ACL en Azure Data Lake Storage Gen2](#)
- **Almacenamiento de archivos:** compartición de archivos en la nube de nivel empresarial, sencilla, segura y sin servidor.
 - [Desarrollo para Azure Files con Python](#)
- **Redis Cache:** acelere el rendimiento de las aplicaciones con un almacén de datos escalable en memoria compatible con código abierto.
 - [Inicio rápido: Uso de Azure Cache for Redis con Python](#)

Macrodatos y análisis

- **Análisis de Azure Data Lake:** servicio de análisis de pago por trabajo totalmente administrado que ofrece un procesamiento de datos paralelo eficaz con seguridad, auditoría y soporte técnico integrados de nivel empresarial.
 - [Administración de Azure Data Lake Analytics mediante Python](#)
 - [Desarrollo de U-SQL con Python para Azure Data Lake Analytics en Visual Studio Code](#)
- **Azure Data Factory:** un servicio de integración de datos totalmente administrado que le permite crear, orquestar y automatizar visualmente el movimiento y la transformación de datos en varios orígenes de datos.

- Inicio rápido: Creación de una factoría de datos y una canalización con Python
 - Transformación de datos mediante la ejecución de una actividad de Python en Azure Databricks
- **Azure Event Hubs:** un servicio de ingesta de telemetría totalmente administrado y a gran escala diseñado para recopilar, transformar y almacenar millones de eventos por segundo desde aplicaciones y dispositivos conectados.
 - Envío o recepción de eventos desde Event Hubs mediante Python
 - Capturar datos de Event Hubs en Azure Storage y leerlos mediante Python (azure-eventhub)
- **HDInsight:** un servicio en la nube totalmente administrado que ejecuta marcos populares de código abierto, como Hadoop y Spark, respaldados por un Acuerdo de Nivel de Servicio de 99.9% para el análisis de macrodatos de nivel empresarial.
 - Usar las herramientas Spark y Hive para Visual Studio Code
- **Azure Databricks:** una plataforma de análisis basada en Apache® Spark™ totalmente administrada, rápida y colaborativa optimizada para cargas de trabajo de macrodatos e inteligencia artificial en Azure.
 - Conexión a Azure Databricks desde Excel, Python o R
 - Introducción a Azure Databricks
 - Tutorial: Azure Data Lake Storage Gen2, Azure Databricks & Spark
- **Azure Synapse Analytics:** un servicio de análisis totalmente administrado que unifica la integración de datos, el almacenamiento de datos empresariales y el análisis de macrodatos en una sola plataforma.
 - Inicio rápido: Uso de Python para consultar una base de datos en Azure SQL Database o Azure SQL Managed Instance (incluye Azure Synapse Analytics)

Administración de identidades y acceso para aplicaciones de Python en Azure

05/06/2025

En Azure, la administración de identidades y acceso (IAM) para aplicaciones de Python implica dos conceptos clave:

- **Autenticación:** comprobación de la identidad de un usuario, un grupo, un servicio o una aplicación
- **Autorización:** determinación de las acciones que la identidad puede realizar en los recursos de Azure

Azure proporciona varias opciones de IAM para ajustarse a los requisitos de seguridad de la aplicación. En este artículo se incluyen vínculos a recursos esenciales para ayudarle a empezar.

Para más información, consulte [Recomendaciones para la administración de identidades y acceso](#).

Conexiones sin contraseña

Siempre que sea posible, se recomienda usar identidades administradas para simplificar la administración de identidades y mejorar la seguridad. Las identidades administradas admiten la autenticación sin contraseña, lo que elimina la necesidad de insertar credenciales confidenciales (como contraseñas o secretos de cliente) en variables de código o entorno. Las identidades administradas están disponibles para servicios de Azure, como App Service, Azure Functions y Azure Container Apps. Permiten que las aplicaciones se autentiquen en los servicios de Azure sin necesidad de administrar las credenciales.

Los siguientes recursos muestran cómo usar el SDK de Azure para Python con autenticación sin contraseña a través de [DefaultAzureCredential](#). `DefaultAzureCredential` es ideal para la mayoría de las aplicaciones que se ejecutan en Azure, ya que admite sin problemas entornos de desarrollo y producción locales mediante el encadenamiento de varios tipos de credenciales en un orden seguro e inteligente.

- [Información general: Conexión sin contraseña para los servicios de Azure](#)
- [Autenticación de aplicaciones de Python en servicios de Azure mediante el SDK de Azure para Python](#)
- [Uso de DefaultAzureCredential en una aplicación](#)

- [Inicio rápido: Biblioteca cliente de Azure Blob Storage para Python con conexiones sin contraseña](#)
- [Guía de inicio rápido: Enviar y recibir mensajes en las colas de Azure Service Bus con conexiones sin contraseña](#)
- Creación e implementación de una aplicación web de Flask en Azure con una identidad administrada asignada automáticamente por el sistema
- Creación e implementación de una aplicación web de Django en Azure con una identidad administrada de asignación de usuario

Conector de servicio

Muchos recursos de Azure que se usan habitualmente en las aplicaciones de Python admiten [Service Connector](#). Service Connector simplifica el proceso de configuración de conexiones seguras entre servicios de Azure. Automatiza la configuración de la autenticación, el acceso a la red y las cadenas de conexión entre servicios de proceso (como App Service o Container Apps) y servicios dependientes (como Azure Storage, Azure SQL o Cosmos DB). Esto reduce los pasos manuales, ayuda a aplicar procedimientos recomendados (como el uso de identidades administradas y puntos de conexión privados) y mejora la coherencia y la seguridad de la implementación.

- [Inicio rápido: Creación de una conexión de servicio en App Service desde Azure Portal](#)
- [Tutorial: Uso de Service Connector para compilar una aplicación de Django con Postgres en Azure App Service](#)

Bóveda de claves

El uso de una solución de administración de claves, como [Azure Key Vault](#), ofrece un mayor control sobre los secretos y las credenciales, aunque incluye una complejidad de administración adicional.

- [Inicio rápido: Biblioteca cliente de certificados de Azure Key Vault para Python](#)
- [Inicio rápido: Biblioteca cliente de claves de Azure Key Vault para Python](#)
- [Inicio rápido: Biblioteca cliente secreta de Azure Key Vault para Python](#)

Autenticación e identidad para iniciar sesión de usuarios en aplicaciones

Puede desarrollar aplicaciones de Python que permitan a los usuarios iniciar sesión con identidades de Microsoft (como cuentas de Azure AD) o cuentas sociales externas (como Google o Facebook). Una vez autenticada, la aplicación puede autorizar a los usuarios a acceder a sus propias API o API de Microsoft, como Microsoft Graph, para interactuar con recursos como perfiles de usuario, calendarios y correos electrónicos.

- [Inicio rápido: Inicio de sesión de usuarios y llamada a Microsoft Graph API desde una aplicación web de Python](#)
- [Temas de autenticación de aplicaciones web](#)
- [Inicio rápido: Adquisición de un token y llamada a Microsoft Graph desde una aplicación de demonio de Python](#)
- [Temas sobre el servicio backend, el demonio y la autenticación de scripts](#)

Aprendizaje automático para aplicaciones de Python en Azure

12/06/2025

Los artículos siguientes le ayudarán a empezar a trabajar con Azure Machine Learning. Las API REST de Azure Machine Learning v2, la extensión de la CLI de Azure y el SDK de Python están diseñadas para simplificar todo el ciclo de vida de aprendizaje automático y acelerar los flujos de trabajo de producción. Los vínculos de este artículo tienen como destino la versión 2, que se recomienda si va a iniciar un nuevo proyecto de aprendizaje automático.

Cómo empezar

En Azure Machine Learning, el área de trabajo es el recurso principal que organiza y administra todo lo que se crea, como conjuntos de datos, modelos y experimentos.

- [Inicio rápido: Introducción a Azure Machine Learning](#)
- [Administración de áreas de trabajo de Azure Machine Learning en el portal o con el SDK de Python \(v2\)](#)
- [Ejecuta cuadernos de Jupyter en tu área de trabajo](#)
- [Tutorial: Desarrollo de modelos en una estación de trabajo en la nube](#)

Implementación de modelos

Implemente modelos para predicciones de aprendizaje automático en tiempo real y de baja latencia.

- [Tutorial: Diseñador: Implementación de un modelo de Machine Learning](#)
- [Implementación y puntuación de un modelo de aprendizaje automático mediante un punto de conexión en línea](#)

Aprendizaje Automático Automatizado

MI automatizado (AutoML) hace referencia al proceso de optimización del desarrollo de modelos de aprendizaje automático mediante la automatización de sus tareas repetitivas y lentas.

- [Entrenamiento de un modelo de regresión con AutoML y Python \(SDK v1\)](#)
- [Configuración del entrenamiento de AutoML para datos tabulares con la CLI de Azure Machine Learning y el SDK de Python \(v2\)](#)

Acceso a datos

Con Azure Machine Learning, puede importar datos desde el equipo local o conectarse a los servicios de almacenamiento en la nube existentes.

- [Creación y administración de recursos de datos](#)
- [Tutorial: Carga, acceso y exploración de los datos en Azure Machine Learning](#)
- [Acceso a datos en un trabajo](#)

Flujos de aprendizaje automático

Use canalizaciones de aprendizaje automático para crear flujos de trabajo que conecten distintas fases del proceso de aprendizaje automático.

- [Uso de Azure Pipelines con Azure Machine Learning](#)
- [Creación y ejecución de canalizaciones de aprendizaje automático mediante componentes con el SDK de Azure Machine Learning v2](#)
- [Tutorial: Creación de canalizaciones de ML de producción con el SDK v2 de Python en un cuaderno de Jupyter Notebook](#)

Servicios de inteligencia artificial para aplicaciones de Python en Azure

17/06/2025

Servicios de Azure AI es un servicio de inteligencia artificial (IA) basado en la nube que ayuda a los desarrolladores a crear inteligencia cognitiva en las aplicaciones sin tener aptitudes o conocimientos directos de IA o ciencia de datos. Hay servicios de inteligencia artificial listos para visión por computadora y procesamiento de imágenes, análisis de lenguaje y traducción, voz, toma de decisiones, búsqueda y Azure OpenAI que puede usar en sus aplicaciones de Python.

Debido a la naturaleza dinámica de los servicios de Azure AI, la mejor manera de encontrar material de introducción para Python es comenzar en la página del centro de [servicios de Azure AI](#) y, a continuación, encontrar el servicio específico que busca.

1. En la página central, seleccione un servicio para acceder a su página de inicio de documentación. Por ejemplo, para [Azure AI Vision](#).
2. En la página de aterrizaje del servicio, seleccione una categoría del servicio. Por ejemplo, en Computer Vision, seleccione [Análisis de imágenes](#).
3. En la documentación, busque **Inicios rápidos** en la tabla de contenido. Por ejemplo, en la documentación de Análisis de imágenes, en Inicios rápidos, hay una [guía de inicio rápido de la versión 4.0 \(versión preliminar\)](#).
4. En los artículos de inicio rápido, elija el lenguaje de programación Python si existe o la API REST.

Si no ve un inicio rápido, en el cuadro de búsqueda de la tabla de contenido, escriba *Python* para buscar artículos relacionados con Python.

Además, puede ir a la [descripción general de los módulos de Azure Cognitive Services para Python](#) para aprender sobre los módulos de SDK de Python disponibles. (Azure Cognitive Services es el nombre anterior de los servicios de Azure AI. La documentación se está actualizando actualmente para reflejar el cambio).

[Vaya a la página del centro de servicios de Azure AI. >>>](#)

La documentación de Azure AI Search se encuentra en una parte independiente de la documentación:

- [Inicio rápido: Buscar texto completo con los SDK de Azure](#)

- Uso de Python e IA para generar contenido que se puede buscar a partir de blobs de Azure

Mensajería, eventos e IoT para aplicaciones de Python en Azure

05/06/2025

Los siguientes artículos le ayudarán a empezar a trabajar con los servicios de mensajería, ingesta y procesamiento de eventos e Internet de las cosas (IoT) en Azure.

Mensajería

Los servicios de mensajería de Azure permiten que diferentes componentes y aplicaciones se comuniquen fácilmente, independientemente del idioma que usen o de dónde se hospeden, ya sea en la misma nube, en varias nubes o en el entorno local.

- Notificaciones
 - [Uso de Notification Hubs desde Python](#)
- Colas
 - [Inicio rápido: Biblioteca cliente de Azure Queue Storage para Python](#)
 - [Inicio rápido: Envío y recepción de mensajes desde colas de Azure Service Bus \(Python\)](#)
 - [Envío de mensajes a un tema de Azure Service Bus y recepción de mensajes de suscripciones al tema \(Python\)](#)
- Funcionalidad web en tiempo real (SignalR)
 - [Inicio rápido: Creación de una aplicación sin servidor con Azure Functions y Azure SignalR Service en Python](#)
- Azure Web PubSub
 - [Cómo crear un WebPubSubServiceClient con Python e Identidad de Azure](#)

Eventos

Azure Event Hubs y Azure Event Grid son dos servicios clave para controlar eventos en Azure. Proporcionan funcionalidades para la ingesta, el procesamiento y el enrutamiento de eventos en varias aplicaciones y servicios.

Estos servicios permiten crear arquitecturas controladas por eventos y procesar eventos en tiempo real.

- Event Hubs
 - [Inicio rápido: Envío o recepción de eventos desde Event Hubs mediante Python](#)

- Inicio rápido: Captura de datos de Event Hubs en Azure Storage y léelo mediante Python ([azure-eventhub](#))
- Event Grid
 - Inicio rápido: Enrutamiento de eventos personalizados al punto de conexión de web con la CLI de Azure y Event Grid
 - Ejemplos de Python de la biblioteca cliente de Azure Event Grid

Internet de las cosas (IoT)

Internet de las cosas (IoT) hace referencia a un conjunto de servicios administrados y de plataforma en todo el perímetro y la nube que conectan, supervisan y controlan los recursos de IoT. IoT también incluye herramientas de seguridad de dispositivos, sistemas operativos y análisis de datos para ayudarle a crear, implementar y administrar aplicaciones de IoT de forma eficaz.

- IoT Hub
 - Inicio rápido: Envío de telemetría desde un dispositivo IoT Plug and Play a Azure IoT Hub
 - Envío de mensajes de nube a dispositivo con IoT Hub
 - Carga de archivos desde el dispositivo a la nube con IoT Hub
 - Programación y difusión de trabajos
 - Inicio rápido: Control de un dispositivo conectado a IoT Hub
- Aprovisionamiento de dispositivos
 - Inicio rápido: Aprovisionamiento de un dispositivo simulado de certificado X.509
 - Tutorial: Aprovisionamiento de dispositivos mediante grupos de inscripción de clave simétrica
 - Tutorial: Aprovisionamiento de varios dispositivos X.509 mediante grupos de inscripción
- IoT Central/IoT Edge
 - Tutorial: Creación y conexión de una aplicación cliente a la aplicación de Azure IoT Central
 - Tutorial: Desarrollo de módulos de IoT Edge mediante Visual Studio Code

Otros servicios para aplicaciones de Python en Azure

12/06/2025

Los servicios a los que se hace referencia en este artículo para Python son especializados, cada uno diseñado para abordar un conjunto específico de problemas. El término *otros servicios* incluye ofertas de Azure más allá de las categorías fundamentales de proceso, redes, almacenamiento y bases de datos. En este artículo se muestran ejemplos de áreas como administración y gobernanza, medios, genómicos e Internet de las cosas (IoT). Para obtener una lista completa de los [productos](#) disponibles de Azure.

- **Transmisión multimedia:**
 - [Conectarse a la API de Azure Media Services v3](#)
- **Automatización:**
 - [Tutorial: Creación de un runbook de Python](#)
- **DevOps:**
 - [Uso de CI/CD con Acciones de GitHub para implementar una aplicación web de Python en Azure App Service en Linux](#)
 - [Compilación e implementación de una aplicación en la nube de Python con la herramienta de código abierto de la CLI para desarrolladores de Azure \(azd\)](#)
 - [Compilación, prueba e implementación de aplicaciones de Python con Azure Pipelines](#)
- **Internet de las cosas y la asignación geográfica:**
 - [Tutorial: Enrutamiento de vehículos eléctricos mediante Azure Notebooks](#)
 - [Tutorial: Unión de datos de sensor con datos de previsión meteorológica mediante Azure Notebooks](#)
- **Burrows-Wheeler Aligner (BWA) y el kit de herramientas de análisis de genoma (GATK):**
 - [Inicio rápido: Ejecución de un flujo de trabajo a través del servicio Microsoft Genomics](#)
- **Administración de recursos**
 - [Inicio rápido: Ejecución de la primera consulta de Resource Graph mediante Python](#)
- **Gestión de máquinas virtuales:**
 - [ejemplo: uso de las bibliotecas de Azure para crear una máquina virtual](#)