## **📌 Título del Proyecto**

**Gestor de Tareas en la Nube con Azure y Flask**

## **🧑‍💻 Integrantes del Equipo**

Kenneth J. Cosme Ortiz – R00598658 – [kenjoel2002@hotmail.com](mailto:kenjoel2002@hotmail.com)

## **🎯 Descripción General**

Esta aplicación permite gestionar tareas básicas a través de una interfaz web desarrollada con Flask y alojada en Azure.  
 **¿Qué hace?** Permite añadir, marcar como completadas y eliminar tareas.  
 **¿A quién va dirigida?** A estudiantes y personas que necesiten una herramienta básica para organizar sus tareas.  
 **¿Qué problema resuelve?** Brinda una solución práctica y simple para llevar el control de pendientes en la nube, accesible desde cualquier lugar.

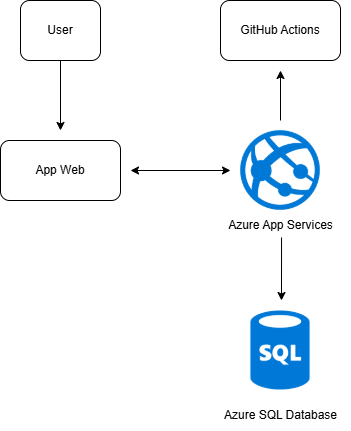
## **☁️ Servicios de Azure Utilizados**

|  |  |  |
| --- | --- | --- |
| **Servicio** | **Propósito dentro del proyecto** | **Gratuito en Azure for Students** |
| Azure App Service | Alojamiento de la app web Flask | ✅ Sí |
| Azure SQL Database | Almacenar las tareas creadas por los usuarios | ✅ Sí |
| GitHub Actions | Automatizar despliegue continuo desde GitHub | ✅ Sí |

## **🧱 Diagrama de Arquitectura**

**Descripción:**

* Usuario accede a la App Web
* Azure App Service ejecuta la app Flask
* Flask se conecta a Azure SQL Database
* GitHub Actions automatiza el despliegue desde GitHub



## **⚙️ Despliegue y Configuración**

### **1. Preparación Local**

* Clonar repositorio:  
   **git clone** [**https://github.com/KennethCosme/azure\_students\_project**](https://github.com/KennethCosme/azure_students_project)
* Crear entorno virtual:  
   **python -m venv venv**  
   **Activar (Windows): venv\Scripts\activate**
* Instalar dependencias:  
  **pip install -r requirements.txt**
* Ejecutar la aplicación localmente:  
   **python app.py**

### **2. Configuración en Azure**

* Azure SQL Database creada desde el portal.
* Se configuraron variables de entorno en App Service:
  + **SQL\_SERVER**
  + **SQL\_DATABASE**
  + **SQL\_USERNAME**
  + **SQL\_PASSWORD**
* App Service con stack: Python 3.8, Linux, Plan F1 (gratuito)
* Se permitió la IP del cliente en el firewall del servidor SQL.

### **3. Automatización con GitHub Actions**

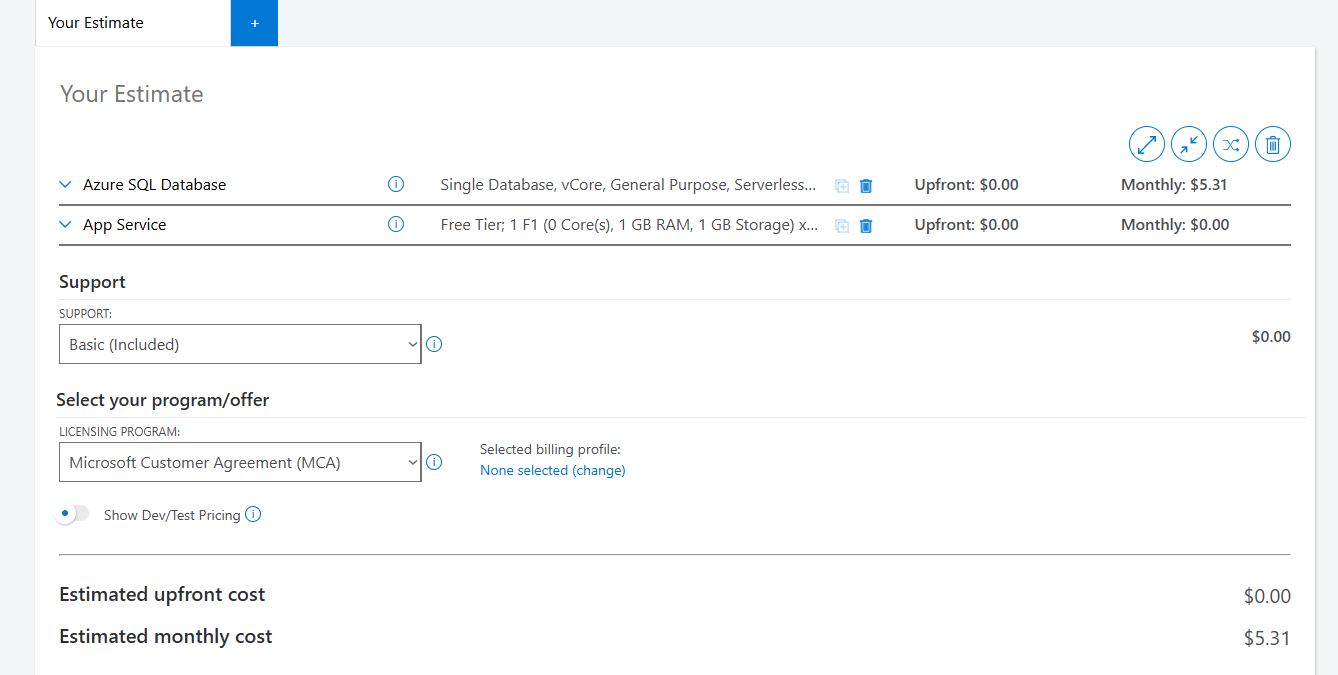
* GitHub conectado al Deployment Center.
* Azure detectó automáticamente el proyecto Flask.
* **requirements.txt** permitió la instalación automática de dependencias.
* Despliegue automático tras cada cambio en main.

## **💻 Enlace a la Aplicación Desplegada**

[**https://app-tasks-kc-fvc7dkfdgufmaeeg.eastus2-01.azurewebsites.net**](https://app-tasks-kc-fvc7dkfdgufmaeeg.eastus2-01.azurewebsites.net/)

## **💸 Estimación del Costo (Azure Pricing Calculator)**

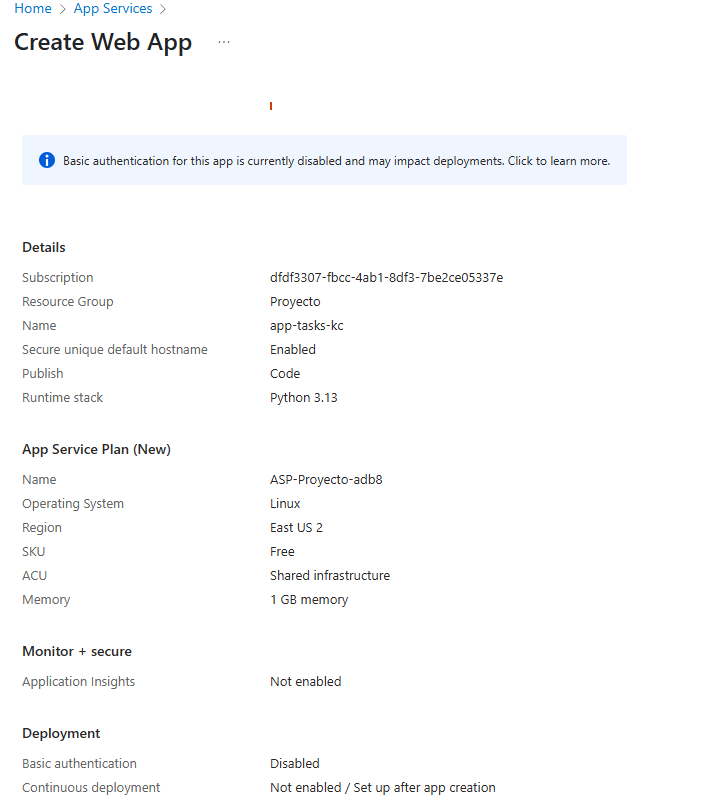
* Azure App Service (F1 - Gratis)
* Azure SQL Database (S0 - nivel más básico)



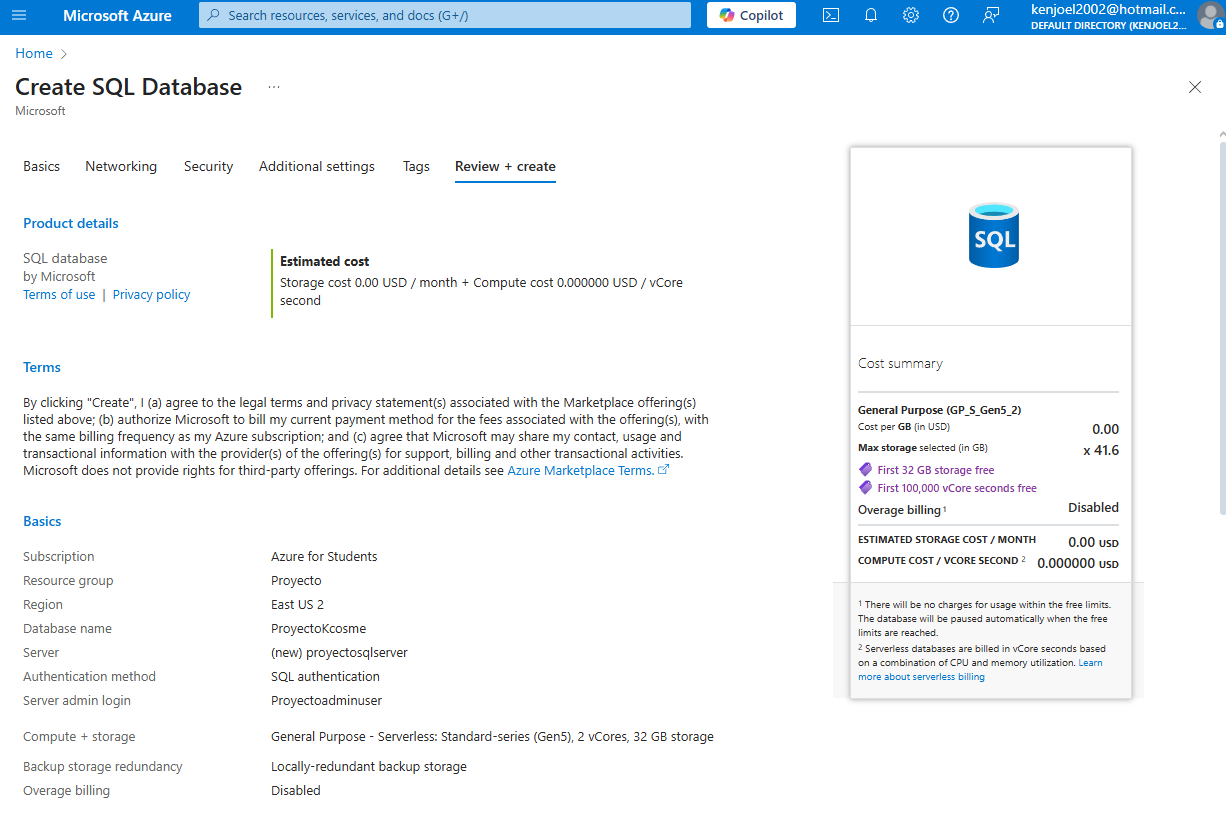
**Costo estimado sin beneficios gratuitos:** $5 USD mensuales

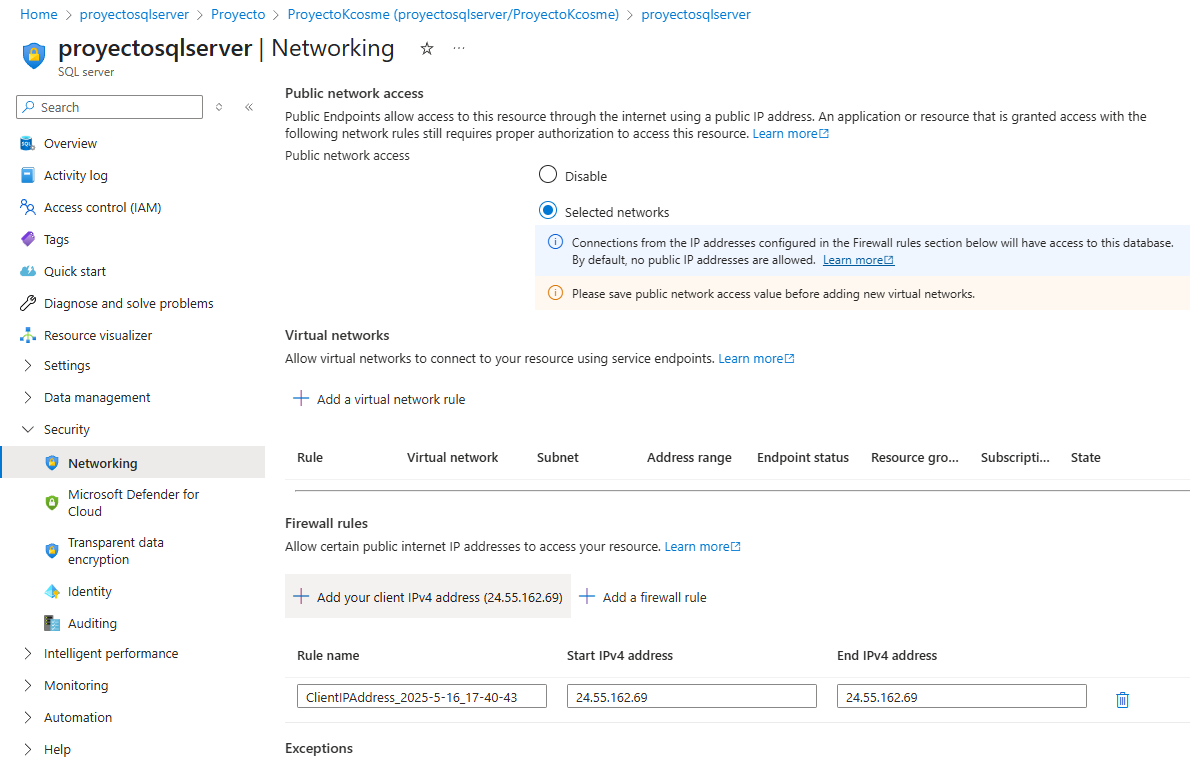
## **📁 Capturas del Portal de Azure**

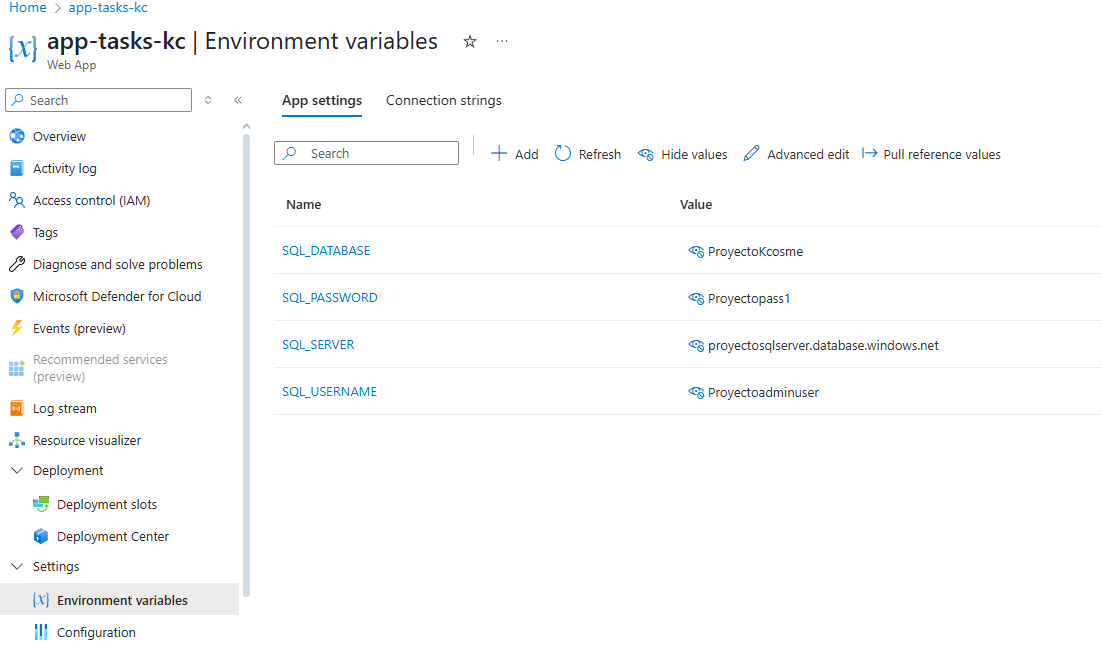
* App Service creado y configurado



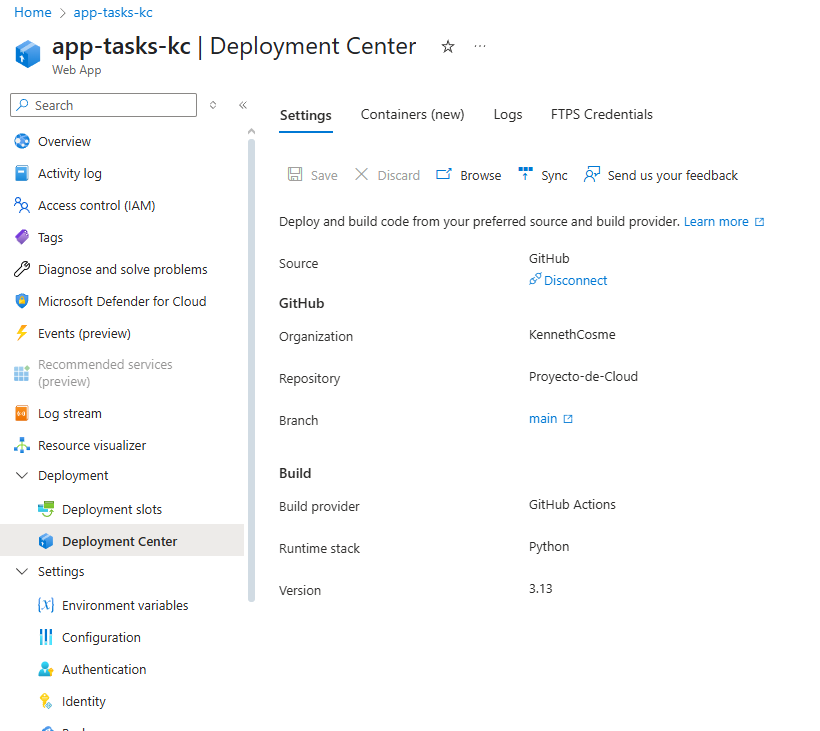
* SQL Database



* Firewall/IP configurado
* Variables de entorno



* GitHub Deployment Center



## **📘 Lecciones Aprendidas**

Aprendí cómo implementar aplicaciones en la nube utilizando los servicios de Azure.  
 Enfrenté dos errores importantes:

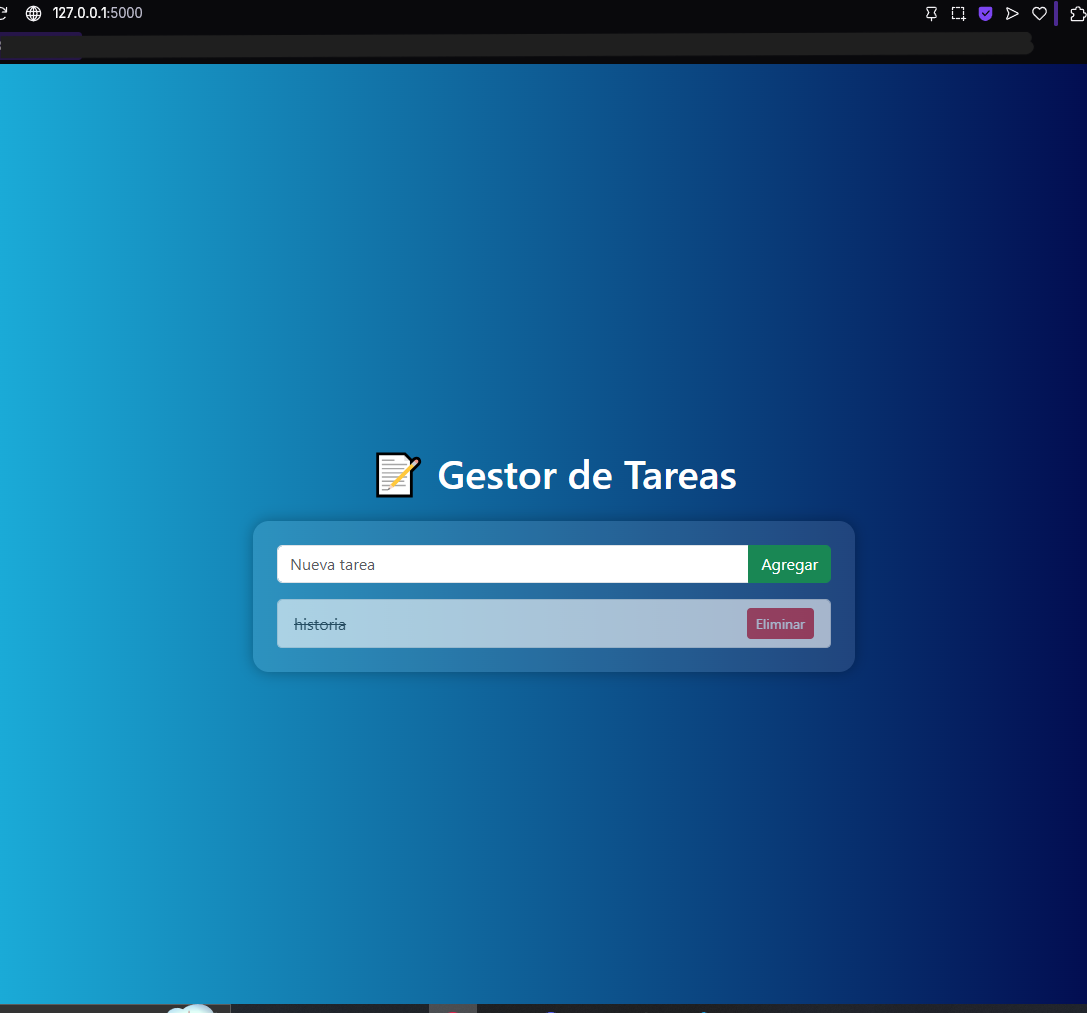
* Uno por datos faltantes en el archivo **app.py**
* Otro porque algunos archivos necesarios no estaban correctamente subidos a GitHub, lo que impedía que Azure App Service reconociera el proyecto.  
   Pude solucionarlos revisando la documentación del tutorial y validando los archivos requeridos.  
   Además, fue valioso aprender cómo GitHub Actions automatiza el despliegue sin necesidad de máquinas virtuales.

Esta experiencia me dio confianza en trabajar con herramientas de despliegue en la nube y me ayudó a entender cómo automatizar procesos reales en un entorno profesional.

## **📚 Repositorio del Código**

<https://github.com/KennethCosme>

## **📄 Instrucciones para Reproducir el Proyecto Localmente**

1. Clonar el repositorio:  
    **git clone** [**https://github.com/KennethCosme/azure\_students\_project**](https://github.com/KennethCosme/azure_students_project)
2. Instalar dependencias:  
    **pip install -r requirements.txt**
3. Crear base de datos (si aplica) o usar la ya conectada en Azure.
4. Configurar variables de entorno necesarias:
   1. **SQL\_SERVER**
   2. **SQL\_DATABASE**
   3. **SQL\_USERNAME**
   4. **SQL\_PASSWORD**
5. Ejecutar la app:  
    **python app.py**
6. Acceder desde **localhost** o desde Azure App Service.

## **✅ Revisión Final del Proyecto**

* La aplicación funciona correctamente en Azure App Service.
* Solo se utilizaron servicios gratuitos permitidos por Azure for Students.
* Se incluyó el diagrama de arquitectura y las capturas del portal.
* Toda la documentación fue completada con instrucciones claras.
* Se estimaron los costos del uso en Azure sin beneficios gratuitos.
* El código está disponible en GitHub.
* Se incluyó una reflexión sobre lo aprendido y los retos enfrentados.