Taller Multi-Cloud: Azure

Despliegue Serverless: Azure Functions (Container & Code)

4. Configuración de Azure CLI y Azure Functions

> **¿Qué es una Azure Function?**
>
> Es una unidad de código (similar a nuestra función `read_root`) que se ejecuta en la nube en respuesta a un **evento** (un request HTTP, un mensaje en una cola, un timer, etc.). Se agrupan en un contenedor llamado **Function App**.

Paso 4.1: Instalación y Login

Asumimos que el Azure CLI está instalado. Iniciamos sesión para gestionar los recursos.

```
# 1. Login con Azure CLI
az login

# 2. Establecer la ubicación y un nombre base (evitar conflictos)
# Reemplace [SU_NOMBRE]
AZURE_LOCATION="eastus"
RESOURCE_GROUP="rg-serverless-[SU_NOMBRE]"
APP_NAME="func-hello-api-[SU_NOMBRE]"
ACR_NAME="acrhelloapi[SU_NOMBRE]"

# 3. Crear el Grupo de Recursos (donde vivirán los recursos)
az group create --name $RESOURCE_GROUP --location $AZURE_LOCATION
```

5. Despliegue Rápido (Método ZIP / Código Directo)

Aunque nos enfocaremos en Docker, es vital conocer el camino rápido: subir el código Python directamente a Azure Functions.

Paso 5.1: Crear la Function App

Creamos el contenedor lógico (Function App) en un Plan de Consumo Serverless (Paga-por-ejecución).

```
# Crear la Function App en Plan de Consumo (Serverless puro)
az functionapp create \
    --resource-group $RESOURCE_GROUP \
    --consumption-plan-location $AZURE_LOCATION \
    --runtime python \
    --runtime-version 3.11 \
    --functions-version 4 \
    --name $APP_NAME
```

Paso 5.2: Despliegue del Código

Normalmente, se usa `func azure functionapp publish`, pero como nuestra aplicación FastAPI no es un proyecto tradicional de Azure Function, en este paso, la consola de Azure es más práctica o usaría herramientas como **Azure Function Core Tools** (que adaptan el código). **Para el ejercicio, este paso sirve para entender la diferencia.**

Paso 5.3: Prueba y Verificación

Una vez desplegada una función, Azure le asigna una URL con una clave de acceso (Key). Revisaríamos el monitoreo en el Portal de Azure para verificar el escalado a cero y el uso de la memoria.

6. Despliegue Avanzado: Azure Functions con Docker

Aquí usamos la imagen Docker que construimos en la Hora 1. Esto nos da control total sobre el ambiente (Python 3.11, FastAPI, etc.).

Paso 6.1: Crear Azure Container Registry (ACR)

ACR es el repositorio privado de Docker en Azure. Necesitamos un lugar donde almacenar nuestra imagen.

```
# Crear el ACR
az acr create \
    --resource-group $RESOURCE_GROUP \
    --name $ACR_NAME \
    --sku Basic \
    --admin-enabled true

# Login con Docker para poder hacer 'push'
az acr login --name $ACR_NAME
```

Paso 6.2: Etiquetar y Subir la Imagen (Push)

Ahora etiquetamos nuestra imagen local para que Docker sepa a qué ACR debe enviarla y la subimos.

```
# 1. Etiquetar la imagen local con el nombre de ACR
docker tag simplehelloapi:latest $ACR_NAME.azurecr.io/helloapi:v1

# 2. Subir la imagen al ACR
docker push $ACR_NAME.azurecr.io/helloapi:v1
```

Resultado: ¡Nuestra imagen Docker está ahora en la nube de Azure!

Paso 6.3: Crear la Function App Serverless desde el Contenedor

Creamos una Function App especial que usará el **plan elástico** de Azure para manejar contenedores, manteniendo el modelo Serverless.

```
# Crear un Plan de App Service (Necesario para contenedores personaliza
# Usamos un plan ELÁSTICO para mantener la escalabilidad Serverless
az appservice plan create \
    --resource-group $RESOURCE_GROUP \
    --name appservice-serverless-plan \
    --location $AZURE_LOCATION \
    --is-linux \
    --sku ElasticPremium \
    --number-of-workers 1

# Crear la Function App usando la imagen de ACR
az functionapp create \
    --resource-group $RESOURCE_GROUP \
    --name $APP_NAME-docker \
    --plan appservice-serverless-plan \
    --deployment-container-image-name $ACR_NAME.azurecr.io/helloapi:v1
    --docker-registry-server-user $(az acr show --name $ACR_NAME --quer
    --docker-registry-server-password $(az acr credential show --name $
```

Paso 6.4: Prueba Final en Azure

Azure tarda unos minutos en arrancar el primer contenedor. Una vez listo, puede acceder a la URL proporcionada por Azure (desde el portal o con `az functionapp show`).

```
# URL de Prueba (Ejemplo - la ruta exacta puede variar según el plan)
```

```
https://[APP_NAME]-docker.azurewebsites.net/api/hello?name=AzureUser
```