05 - Despliegue Serverless DevOps

Tipo	Area Temática	Duración
Taller	Automatización de despliegues en la nube	2 horas

Tema:

Despliegue de Aplicaciones Serverless mediante Pipelines

Objetivo

El objetivo de esta práctica es que los estudiantes configuren y gestionen un pipeline de despliegue en Azure DevOps para una aplicación serverless utilizando AWS Lambda y API Gateway. El pipeline automatizará la compilación, empaquetado, y despliegue de la función Lambda, permitiendo que cualquier cambio en el código se despliegue instantáneamente.

Requisitos Previos

- Cuenta en Azure DevOps.
- Acceso a una cuenta de AWS con permisos para crear y gestionar recursos como Lambda, API Gateway, IAM, etc.
- Conocimiento básico de YAML y de la CLI de AWS.

Paso 1: Configuración Inicial

1.1. Crear un Proyecto en Azure DevOps

1. Inicia sesión en Azure DevOps.
2. Crea un nuevo proyecto seleccionando la opción **New Project**.
3. Asigna un nombre al proyecto, por ejemplo, `ServerlessDeployment`.
4. Selecciona la visibilidad del proyecto (privado o público) y haz clic en **Create**.

1.2. Crear un Repositorio Git en Azure DevOps

1. Dentro del proyecto recién creado, ve a la sección **Repos**.
2. Crea un nuevo repositorio y nómbralo, por ejemplo, `serverless-app`.
3. Clona el repositorio en tu máquina local utilizando el siguiente comando:

```
git clone <https://dev.azure.com/yourorganization/Serverle
ssDeployment/_git/serverless-app>
```

4. Navega al directorio del repositorio clonado:

```
cd serverless-app
```

1.3. Crear una Función Lambda en AWS

1. Accede a la consola de AWS y navega a **Lambda**.
2. Haz clic en **Create function**.
3. Selecciona la opción **Author from scratch**.
4. Asigna un nombre a la función, por ejemplo, `MyServerlessFunction`.
5. Selecciona **Runtime** como `Python 3.9` (o el lenguaje de tu preferencia).
6. En la sección **Permissions**, selecciona **Create a new role with basic Lambda permissions**.
7. Haz clic en **Create function**.

1.4. Configurar el Código de la Función Lambda

1. Navega a la sección **Code** de tu función Lambda.
2. Reemplaza el código predeterminado con el siguiente ejemplo:

```
import json

def lambda_handler(event, context):
    return {
        'statusCode': 200,
        'body': json.dumps('Hello from Lambda v1!')
    }
```

3. Haz clic en **Deploy** para guardar los cambios.

1.5. Crear un API Gateway

1. En la consola de AWS, navega a **API Gateway**.
2. Selecciona **Create API** y elige **REST API**.
3. Asigna un nombre a la API, por ejemplo, `MyServerlessAPI`.
4. Haz clic en **Create Resource** y añade un nuevo recurso con el nombre `hello`.
5. Crea un nuevo método `GET` para el recurso `hello` y selecciona **Lambda Function** como integración.
6. Asocia la función Lambda `MyServerlessFunction` creada anteriormente y despliega la API en un nuevo stage, por ejemplo, `dev`.

Paso 2: Configuración del Pipeline en Azure DevOps

2.1. Configurar el YAML del Pipeline

1. En Azure DevOps, navega a **Pipelines** y selecciona **Create Pipeline**.
2. Selecciona **Use the classic editor** y luego elige **Empty Job**.
3. Dentro del pipeline, selecciona **YAML** y añade el siguiente contenido al archivo:

```
trigger:
- master

pool:
  vmImage: 'ubuntu-latest'

steps:
- script: |
    echo "Empaquetando la función Lambda..."
    zip -r function.zip .
  displayName: 'Empaquetar la función Lambda'

- script: |
    aws lambda update-function-code --function-name MyServ
erlessFunction --zip-file fileb://function.zip
    echo "Cambio en el mensaje de respuesta para la actual
ización automática"
  displayName: 'Actualizar la función Lambda'

- script: |
    aws apigateway create-deployment --rest-api-id <API_ID
> --stage-name dev
  displayName: 'Desplegar API Gateway'
```

4. Asegúrate de reemplazar `<API_ID>` con el ID real de tu API Gateway y `<YOUR_S3_BUCKET>` con el nombre de tu bucket S3 donde se subirá el archivo ZIP.

2.2. Configurar Service Connections en Azure DevOps

1. Navega a **Project settings** > **Service connections**.
2. Configura una nueva conexión de servicio para AWS utilizando tus credenciales de AWS.
3. Asegúrate de que el pipeline tenga acceso a esta conexión de servicio.

2.3. Realizar un Cambio en el Código para la Actualización Automática

1. Edita el código de la función Lambda en tu repositorio local y cambia el mensaje de respuesta, por ejemplo:

```
import json

def lambda_handler(event, context):
    return {
        'statusCode': 200,
        'body': json.dumps('Hello from Lambda v2!')
    }
```

2. Haz un commit y push de los cambios al repositorio:

```
git add .
git commit -m "Actualizar mensaje de respuesta"
git push origin master
```

Paso 3: Despliegue y Validación

3.1. Ejecutar el Pipeline

1. Navega a **Pipelines** en Azure DevOps y selecciona el pipeline creado.
2. Haz clic en **Run pipeline** para iniciar el despliegue.
3. Verifica que la ejecución sea exitosa y que los logs no contengan errores.

3.2. Validar el Despliegue

1. Accede a la URL de la API Gateway generada y navega al recurso `/hello`.
2. Deberías ver la respuesta `Hello from Lambda v2!` en formato JSON.
3. Si todo funciona correctamente, has completado la práctica.

Paso 4: Resolución de Problemas Comunes

4.1. Errores de Permisos

- Verifica que la función Lambda tenga los permisos adecuados para ser invocada por API Gateway.
- Revisa las políticas IAM asociadas a la función Lambda y API Gateway.

4.2. Fallas en el Pipeline

- Revisa los logs de ejecución del pipeline para identificar problemas específicos.
- Asegúrate de que el archivo ZIP subido contiene todos los archivos necesarios y que no esté corrupto.

Conclusión

En esta práctica, has aprendido a configurar y desplegar una aplicación serverless utilizando un pipeline en Azure DevOps. Esta es una habilidad crítica para automatizar y gestionar el despliegue continuo de aplicaciones en entornos de producción. Además, has aprendido a implementar actualizaciones automáticas, lo que permite desplegar nuevas versiones de la aplicación sin intervención manual.

¡Felicitaciones por completar la práctica!