

Enviar correo con lambda

Creamos una aplicación en google para poder mandar el correo para eso vamos a Gestionar tu cuenta de Google después a seguridad, en búsqueda ponemos Contraseñas de aplicación

The screenshot shows the left sidebar of the Google Account settings with 'Seguridad' selected. A search bar at the top right contains the text 'Contraseñas de aplicación'. Below it, a list titled 'Resultados de la cuenta de Google' includes 'Contraseñas de aplicación' under 'Seguridad', which is highlighted. Other items listed are 'Gestor de contraseñas', 'Notificaciones de inicio de sesión', 'Herramientas de introducción de texto', and 'Iniciar sesión con contraseñas de aplicación'. At the bottom, there's a link to 'Artículos del Centro de Ayuda'.

despues elige el nombre de la aplicación que vas a crear

← Contraseñas de aplicación

Las contraseñas de aplicación te ayudan a iniciar sesión en tu cuenta de Google en aplicaciones y servicios antiguos que no son compatibles con los estándares de seguridad modernos.

Las contraseñas de aplicación son menos seguras que usar aplicaciones y servicios actualizados que utilicen estándares de seguridad modernos. Antes de crear una contraseña de aplicación, debes comprobar si tu aplicación la necesita para iniciar sesión.

[Más información](#)

A large input field is shown with the placeholder text 'No tienes ninguna contraseña de aplicación.' Below it, instructions say 'Para crear una contraseña específica de la aplicación, escribe el nombre de la aplicación a continuación...'. A blue button labeled 'Nombre de la aplicación' is followed by the text 'EnviarCORreos'.

y cuando das a crear te va a salir una contraseña cual tienes que guardar en algún sitio, que nos va a hacer falta más adelante

Contraseña de aplicación generada

Tu contraseña de aplicación para el dispositivo

uqxz qwfk jwwv rlx

Cómo utilizarla

Accede a la sección de configuración de tu cuenta de Google en la aplicación o el dispositivo que estés intentando configurar. Sustituye tu contraseña por la contraseña de 16 caracteres que se muestra arriba.

Al igual que la contraseña normal, esta contraseña de aplicación ofrece acceso completo a tu cuenta de Google. No tendrás que recordarla, así que no la escribas ni la compartas con nadie.

[Hecho](#)

Y ahora vamos a por AWS y primero que tenemos que hacer es entrar en AWS academy cloud



AWS academy cloud

Más preguntas :

¿Cuánto cuesta el curso de AWS?

¿Qué es AWS Academy?

¿AWS Educate es gratis?

¿Cuánto vale la certificación AWS?



Instructure

<https://awsacademy.instructure.com> · Traducir esta página



Sin título

学生の方はこちらからログインしてください。 已注册课程的学生请在这里登录. Educators who have access to the **AWS Academy Portal**).

después entró en mi panel de control

Panel de control

:

[AWS Academy Cloud Architectin...](#)

ACAv3EN-US-LTI13-140228



Vamos a modules

The screenshot shows the AWS Academy dashboard. On the left is a sidebar with icons for Account, Dashboard, Courses, Calendar, and Inbox. The main area displays the course title "ACAv3EN-US-LTI13". Below the title are links for Home, Modules (which is selected and highlighted in blue), Discussions, Grades, and Lucid (Whiteboard). A vertical scrollbar is visible on the right side of the main content area.

en modules buscamos a

Guided lab: Implementing a Serverless Architecture on AWS
que están en modules 14

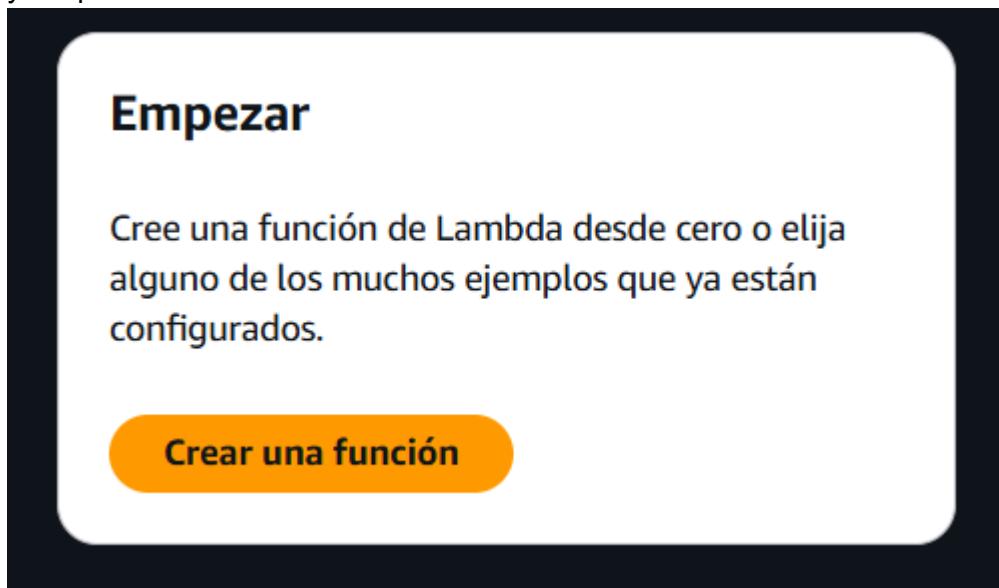
- ▼ Module 14: Building Serverless Architectures and Microservices
 - 🔗 Student guide
 - 🔗 Introduction
 - 🔗 Thinking serverless
 - 🔗 Architecting serverless microservices
 - 🔗 Building serverless architectures with AWS Lambda
 - 🔗 Demo: Using AWS Lambda with Amazon S3
 - 📝 Guided lab: Implementing a Serverless Architecture on AWS
56 pts

al entrar damos a start Lab y cuando AWS se pone en verde damos a el

The screenshot shows the AWS Lambda console. At the top, there is a status bar with the text "AWS" and a red dot. To the right of the status bar are several buttons: "Start Lab", "End Lab", "AWS Details", "Details", and a close button. Below the status bar, the text "nos manda a otra pantalla en cual en buscador ponemos Lambda" is displayed.

The screenshot shows the AWS search interface with the query "lambda". On the left, there's a sidebar with service categories: Servicios (8), Características (7), Recursos (1), Publicaciones de blog (31), Documentación (2639), Artículos de conocimiento (155), and Tutoriales (7). The main area displays three services: Lambda (orange icon, "Ejecute código sin tener que pensar en los servidores"), CodeBuild (blue icon, "Compile y pruebe código"), and AWS Signer (red icon, "Garantizar la confianza y la integridad del código"). A link "Ver los 8 resultados" is at the top right.

y después damos a crear función



elegimos Python 3.11

Información básica

Nombre de la función
Escriba un nombre para describir el propósito de la función.
ENviarCorreo2

El nombre de la función debe tener entre 1 y 64 caracteres, debe ser exclusivo de la región y no puede incluir espacios. Los caracteres válidos son a-z, A-Z, 0-9, guiones (-) y guiones bálgicos (–).

Tiempo de ejecución | Información
Elija el idioma para usar para escribir su función. Tenga en cuenta que el editor de código de la consola es compatible con solo Node.js, Python y Ruby.

Python 3.11 ↻

Arquitectura | Información
Elija la arquitectura del conjunto de instrucciones que desea para el código de la función.

arm64
 x86_64

Permisos | Información
De forma predeterminada, Lambda creará un rol de ejecución con permisos para cargar registros en Amazon CloudWatch Logs. Puede personalizar este rol predeterminado.

► Cambiar el rol de ejecución predeterminado

y en Cambiar el rol de ejecución predeterminado elegimos Uso de un rol existente y elegimos Lambda-Check-Stock-Role

▼ Cambiar el rol de ejecución predeterminado

Rol de ejecución

Seleccione un rol que defina los permisos de la función. Para crear un rol personalizado, vaya a la [consola de IAM](#).

- Creación de un nuevo rol con permisos básicos de Lambda
- Uso de un rol existente
- Creación de un nuevo rol desde la política de AWS templates

Rol existente

Seleccione un rol existente que haya creado para usarlo con esta función de Lambda. El rol debe tener permiso para cargar registros en Amazon CloudWatch Logs.

Lambda-Check-Stock-Role



Consulte el rol Lambda-Check-Stock-Role en la consola de IAM.

y damos a Crear una función.

Abajo en el código fuente pegamos este código

```
import json
import smtplib
from email.mime.text import MIMEText

SECRET_TOKEN = "mF8Xk9u2170aB5FZqvH2j1sW9cR8yTzP6aQvB0xE3fI"
GMAIL_USER = "smirnitskiinikolai@gmail.com"
GMAIL_PASS = "jksjvvmfgwgzmco"

def lambda_handler(event, context):
    # Verificar token en headers
    headers = event.get("headers") or {}
    token = headers.get("x-api-key")

    if not token or token != SECRET_TOKEN:
        return {
            "statusCode": 403,
            "body": json.dumps({"message": "Forbidden: Invalid token"})
        }

    # Leer datos del correo desde body
    try:
        body = json.loads(event.get("body") or "{}")
        to_email = body["to"]
        subject = body.get("subject", "Sin asunto")
        message = body.get("message", "")
    except Exception as e:
        return {
            "statusCode": 400,
            "body": json.dumps({"message": "Bad Request: Missing fields", "error": str(e)})
        }

    # Preparar correo
    msg = MIMEText(message)
```

```

msg["Subject"] = subject
msg["From"] = GMAIL_USER
msg["To"] = to_email

# Enviar correo
try:
    with smtplib.SMTP_SSL("smtp.gmail.com", 465) as server:
        server.login(GMAIL_USER, GMAIL_PASS)
        server.sendmail(GMAIL_USER, to_email, msg.as_string())
except Exception as e:
    return {
        "statusCode": 500,
        "body": json.dumps({"message": "Error sending email",
"error": str(e)})
    }

return {
    "statusCode": 200,
    "body": json.dumps({"message": "Correo enviado
correctamente!"})
}

```

jksjvvmfgwgzmcok

en el codigo teneis que meter la contraseña que habeis compiado al hacer la aplicacion de google en

GMAIL_PASS = "Aquí hay que pegar la contraseña"

y en GMAIL_USER = "El correo electrónico en cual habéis creado la aplicación"

y después da a deploy si todo sale bien te va a salir "Se ha actualizado correctamente la función ENviarCORreо2"

The screenshot shows the AWS Lambda function editor interface. At the top, a green banner says "Se ha actualizado correctamente la función EnviarCorreo2.". The left sidebar has sections for EXPLORER, ENVIARCORREO2 (with lambda_function.py selected), and TEST EVENTS [NONE SELECTED] (+ Create new test event). The main area shows the code for lambda_function.py:

```

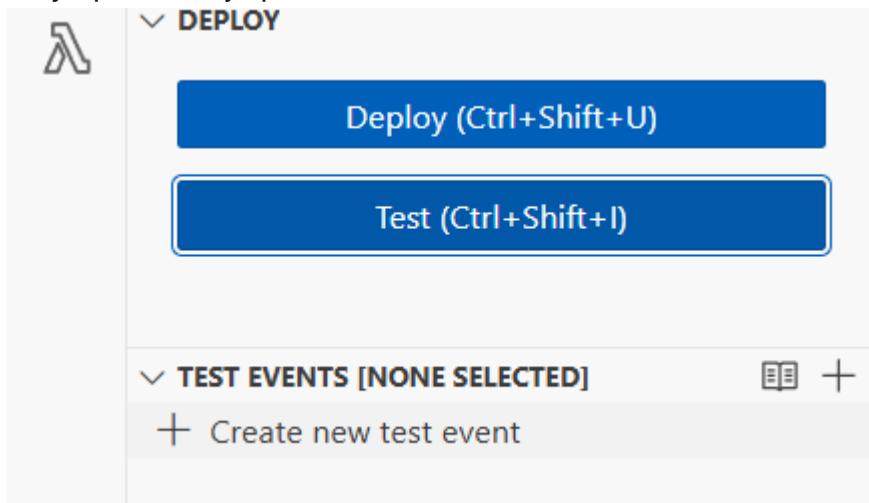
EXPLORER
ENVIARCORREO2
lambda_function.py

λ lambda_function.py
...
def lambda_handler(event, context):
    except Exception as e:
        ...
        # Preparar correo
        msg = MIMEText(message)
        msg["Subject"] = subject
        msg["From"] = GMAIL_USER
        msg["To"] = to_email

    # Enviar correo
    try:
        with smtplib.SMTP_SSL("smtp.gmail.com", 465) as server:
            server.login(GMAIL_USER, GMAIL_PASS)
            server.sendmail(GMAIL_USER, to_email, msg.as_string())
    except Exception as e:
        return {
            "statusCode": 500,
            "body": "Error al enviar el correo electrónico: " + str(e)
}

```

y después a Test, pero para hacer test tienes que añadir un test event, para eso un poco mas abajo que test hay opción de create new event test



Metemos este el código

```
{
  "headers": {
    "x-api-key": "Tu token"
  },
  "body": "{\"to\": \"el correo a donde queres que llega el mensaje\", \"subject\": \"Prueba\", \"message\": \"Hola desde Lambda\"}"
}
```

y damos a Save

The screenshot shows the AWS Lambda function editor. On the left, the code file 'lambda_function.py' is displayed:

```

13     lambda_handler(event, context):
14         try:
15             subject = event['subject']
16             to_email = event['to']
17             GMAIL_USER = os.environ['GMAIL_USER']
18             GMAIL_PASS = os.environ['GMAIL_PASS']
19             msg = MIMEText(subject=subject, to=to_email)
20             server = smtplib.SMTP('smtp.gmail.com', 465)
21             server.login(GMAIL_USER, GMAIL_PASS)
22             server.sendmail(GMAIL_USER, to_email, msg.as_string())
23         except Exception as e:
24             return {
25                 "statusCode": 500,
26                 "body": json.dumps({"message": "Error sending email"})
27             }

```

On the right, the 'Create new test event' dialog is open, showing the event name 'EwentCorreo' and sharing settings (Private selected). The event JSON field contains:

```

1 {
2   "headers": {
3     "x-api-key": "mF8Xk9u2l7oab5FZqvH2j1sW9cR8yTzP6aQvB0xE3t"
4   },
5   "body": "{\"to\": \"n.smirnitskii@gmail.com\", \"subject\": \"Test Email\"}"
6 }

```

y después a invoke

y abajo me sale esto

Status: Succeeded
Test Event Name: EwentCorreo

Response:

```

{
  "statusCode": 200,
  "body": "{\"message\": \"Correo enviado correctamente!\"}"
}

```

Function Logs:

```

START RequestId: d64ee90b-3e98-4447-9b8b-c312695860ac Version: $LATEST
END RequestId: d64ee90b-3e98-4447-9b8b-c312695860ac
REPORT RequestId: d64ee90b-3e98-4447-9b8b-c312695860ac Duration: 977.50 ms Billed Duration: 978 ms Memory Size: 128 MB Max Memory Used: 48 MB
Request ID: d64ee90b-3e98-4447-9b8b-c312695860ac

```

y al entrar al correo puedo ver que si que se ha enviado el mensaje

smirnitskiinikolai 2 Prueba - Hola desde Lambda

Ahora vamos a crear un enlace para que si entras en enlace encia el correo automaticamente
vamos a configuración/URL de la función
damos a Crear una URL de función

[Crear una URL de función](#)

No hay ninguna URL de la función

No se ha configurado ninguna URL de la función.

[Crear una URL de función](#)

en Configurar la URL de la función elegimos none y guardamos

Configurar la URL de la función

URL de la función [Información](#)

Utilice URL de función para asignar puntos de enlace HTTP(S) a la función de Lambda.

Tipo de autorización

Elija el tipo de autorización para la URL de la función. [Más información](#)

AWS_IAM

Solo los usuarios y roles de IAM autenticados pueden realizar solicitudes a la URL de la función.

NONE

Lambda no realizará la autenticación de IAM en las solicitudes a la URL de la función. El punto de conexión de la URL será público a menos que implemente su propia lógica de autorización en la función.

Permisos de URL de función

ⓘ Cuando elige el tipo de autenticación **NONE**, Lambda crea automáticamente la siguiente política basada en recursos y la asocia a la función. Esta política hace que la función sea pública para cualquier persona que tenga la URL de la función. Puede editar la política más adelante. Para limitar el acceso a los usuarios y roles de IAM autenticados, elija el tipo de autenticación **AWS_IAM**.

▼ Instrucción de política

```
1 · [ ] "Version": "2012-10-17",
2 ·   "Statement": [
3 ·     {
4 ·       "StatementId": "FunctionURLAllowPublicAccess",
5 ·       "Effect": "Allow",
6 ·       "Principal": "*",
7 ·       "Action": "lambda:InvokeFunctionUrl",
8 ·       "Resource": "arn:aws:lambda:us-east-1:003421586898:function:ENviarCorreo2",
9 ·       "Condition": {}
10 · }
```

y así tenemos url para nuestra función

Descripción

-

Última modificación

hace 26 minutos

ARN de la función

 [arn:aws:lambda:us-east-1:003421586898:function:ENviarCorreo2](#)

URL de la función [Información](#)

 <https://znxiv6fx7dg77xripowifytuea0hvknslambda-url.us-east-1.on.aws/> 

Ahora vamos a hacer parte del código en kotlin es decir vamos a android studio mete este código

Primero vamos a meter este implementación en el build.gradle

```
dependencies {
    implementation("com.squareup.okhttp3:okhttp:4.11.0")
}
```

y luego mete este código pegamos en nuevo archivo, por ejemplo yo he llamado EnviarCoreo

```
package com.example.reto1_dam_2025_26.userInt.screens
```

```
import okhttp3.MediaType.Companion.toMediaTypeOrNull
import okhttp3.OkHttpClient
```

```

import okhttp3.Request
import okhttp3.RequestBody.Companion.toRequestBody
import org.json.JSONObject

fun enviarCorreoLambda(toEmail: String = "alumno@example.com") {
    val url = "https://cu4bxatvf5x3fdbwbayfw2h6pi0ivour.lambda-url.us-east-1.on.aws/"
    val token = "mF8Xk9u2l7OaB5FZqvH2j1sW9cR8yTzP6aQvB0xE3fl"

    val json = JSONObject().apply {
        put("to", toEmail)
        put("subject", "¡Registro exitoso!")
        put("message", "Bienvenido/a, tu cuenta ha sido creada correctamente 🎉")
    }

    val client = OkHttpClient()

    val requestBody = json.toString()
        .toRequestBody("application/json; charset=utf-8".toMediaTypeOrNull())

    val request = Request.Builder()
        .url(url)
        .post(requestBody)
        .addHeader("x-api-key", token) // ✅ correcto
        .build()

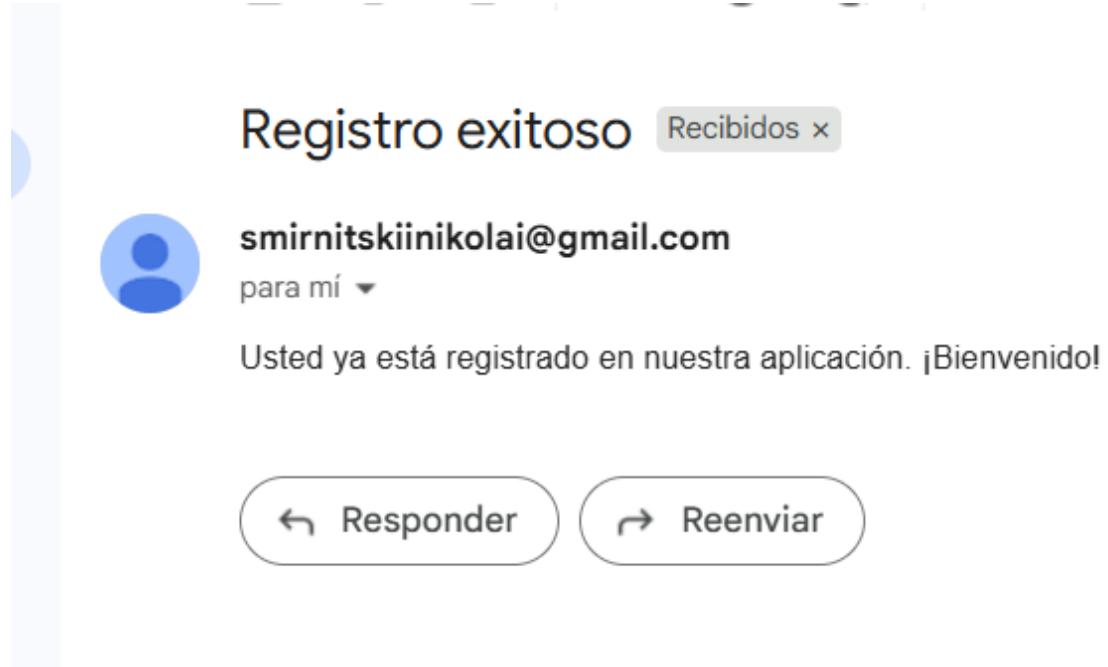
    Thread {
        try {
            val response = client.newCall(request).execute()
            if (response.isSuccessful) {
                println("✅ Correo enviado correctamente: ${response.body?.string()}")
            } else {
                println("❌ Error al enviar correo: ${response.code} ->
${response.body?.string()}")
            }
        } catch (e: Exception) {
            e.printStackTrace()
        }
    }.start()
}

```

Por último hay que implementar la función de enviar correos en login en mi caso tenía que añadir estas líneas

```
// Cuando el registro es exitoso, enviamos el correo  
com.example.reto1_dam_2025_26.utils.enviarCorreoRegistro(state.email)  
"Registro exitoso! Se ha enviado un correo de confirmación."
```

al registrar nuevo usuario al correo que ha puesto dicho usuario llega un correo



y si el usuario pone el correo que no existe a mi correo llega un mensaje diciendo que
No se ha encontrado la dirección



Mail Delivery Subsystem <mailer-daemon@googlemail.com>
para mí ▾



No se ha encontrado la dirección

Tu mensaje no se ha entregado a **naia@gmail.com** porque no se ha encontrado la dirección o esta no puede recibir correo.

[MÁS INFORMACIÓN](#)