

Implementación API: Autorización

Vamos a explicar como vamos a autorizar los endpoints del CRUD de la gestión de layouts

Como se se ha explicado en la arquitectura anterior, vamos a crear un authorizer implementado através de un Lambda que va a devolver un Policy Document v2.0 de AWS indicando si puede o no puede acceder el usuario al endpoint en funcion de un access web token previamente generado por nosotros.

STEP 01: generación de un access web token

Vamos a generar un JSON Web Token (JWT) basado en el estandar abierto (RFC 7519). Hemos utilizado lo hemos generado de forma [online](#) através de esta web, aunque podríamos utilizar cualquier otra herramienta compatible en el estandar antes citado.

Los datos de partida para generar nuestro token son estos claims:

issue: mango

Issued At: 2022-04-01T16:32:52.626Z

Username: owner

Role: Admin

Y la clave:

key: authorization-token

JWT stands for JSON Web Token. **JSON Web Token (JWT) is an open standard (RFC 7519)** that defines a compact and self-contained way for securely transmitting information between parties as a JSON object. This information can be verified and trusted because it is digitally signed. The client will need to authenticate with the server using the credentials only once. During this time the server validates the credentials and returns the client a JSON Web Token (JWT). For all future requests the client can authenticate itself to the server using this JSON Web Token (JWT) and so does not need to send the credentials like username and password.

JWT Payload

Claim Type

Claim Value

Issuer

Mango

✖

Issued At

2022-04-01T16:32:52.626Z

✖

Username

owner

✖

Role

Admin

✖

remove all claims

add claim

Payload to be generated

{
 "Issuer": "Mango",
 "Issued At": "2022-04-01T16:32:52.626Z",
 "Username": "owner",
 "Role": "Admin"
}

Create JSON Web Token Using Secret Key

Algorithm

HS256 ▾

Key

authorization-token

Create JWT

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXLTQwMTAwNDg0MzA3NzJ9LWZlbiBldWNT4FW6e19HmxuteUALN1dxXEyCF9BuF

STEP 02: Implementación del lambda authorizer

Para implementar el authorizer vamos a crear un lambda que llamaremos layout-authorizer basado nuevamente en nodeJs con la siguiente implementación

```
exports.handler = async (event) => {
// Define JSON Web Token from these data usinh HS256 hashing algorithm
/*
Token payload:
{
"Issuer": "Mango",
"Issued At": "2022-04-01T16:32:52.626Z",
"Username": "owner",
"Role": "Admin"
}
```

```

exports.handler = async (event) => {
    // Define JSON Web Token from these data usinh HS256 hashing
    algorithm
    /*
        Token payload:
        {
            "Issuer": "Mango",
            "Issued At": "2022-04-01T16:32:52.626Z",
            "Username": "owner",
            "Role": "Admin"
        }

        Key: authorization-token
    */

    const ACCESS_TOKEN = "eyJhbGciOiJIUzI1NiJ9.eyJJSb2xlIjoib3duZXIiLCJpYXQiOiE2NDg4MzA3NzJ9.-2fEbjElWdNT4FW6e19HmxuteUALNldxXEyCF9BuFQg";

    // check if the user's token is valid
    let auth = "Deny";

    if (event.headers.authorization == ACCESS_TOKEN)
        auth = "Allow";
    else
        auth = "Deny";

    // response with a policy document v2.0
    let authResponse = {
        "principalId": ACCESS_TOKEN,
        "policyDocument": {
            "Version": "2012-10-17",
            "Statement": [{
                "Action": "execute-api:Invoke",
                "Effect": auth,
                "Resource": event.routeArn
            }]
        }
    };

    return authResponse;
};

```

Como podemos ver la estructura del policy document devuelto tiene una parte configurable que es el estado del mismo que podrá ser: Allow o Deny en función de que el authorization token enviado en la cabecera de la petición corresponda con el esperado.

STEP 03: Crear un authorizer

Creamos un authorizer asociado a esta función lambda y lo adjuntamos a cada uno de las rutas definidas: GET, GET {code}, PUT y DELETE {code} desde la opción del API Gateway llamada Authorization. Dentro de la misma escogemos una ruta y desde la pestaña de Manage authorizations creamos una asociando el lamda de autorización antes creado

The screenshot shows the 'Create authorizer' form in the AWS API Gateway console. The form is titled 'API Gateway' and has a sidebar with navigation options: APIs, Custom domain names, VPC links, Develop (Routes, Authorization, Integrations, CORS, Reimport, Export), Deploy (Stages), Protect (Throttling), and Monitor (Metrics, Logging). The main form area is titled 'Name' and contains the following fields:

- Name:** Enter a name for the authorizer. The value 'authorizer' is entered.
- AWS Region:** A dropdown menu showing 'us-east-1'.
- Lambda function:** A search box with the value 'arn:aws:lambda:us-east-1:924628188769:function:layout-authorizer'.
- Payload format version:** A dropdown menu showing '2.0'.
- Response mode:** Two radio buttons: 'Simple' (selected) and 'IAM Policy'. The 'Simple' option has a description: 'Your Lambda function returns a simple boolean value.' The 'IAM Policy' option has a description: 'Your Lambda function returns an IAM policy document.'
- Authorizer caching:** A checkbox labeled 'Authorizer caching' is checked.
- Identity sources:** A section titled 'Identity sources' with a description: 'Selection expressions specify the source of information that's required to authorize requests. You can specify headers, query strings, stage variables, or context variables. Learn more'. It contains a text box with the value '\$request.header.Authorization' and a 'Remove' button.
- Invoke permissions:** A section titled 'Invoke permissions' with a description: 'API Gateway needs permission to invoke your Lambda function in order to use it for authorization. You can allow API Gateway to update the Lambda function's resource policy for you. You can also update the function's resource policy yourself, or provide an invoke role that API Gateway uses to invoke your function.' It contains a checkbox labeled 'Automatically grant API Gateway permission to invoke your Lambda function' which is checked.

The screenshot shows the 'Manage authorizers' page in the AWS API Gateway console. The page is titled 'API Gateway' and has a sidebar with navigation options: APIs, Custom domain names, VPC links, Develop (Routes, Authorization, Integrations, CORS, Reimport, Export), Deploy (Stages), Protect (Throttling), and Monitor (Metrics, Logging). The main form area is titled 'Authorization' and has a breadcrumb trail: 'API Gateway > Authorization > Manage'. The page contains the following sections:

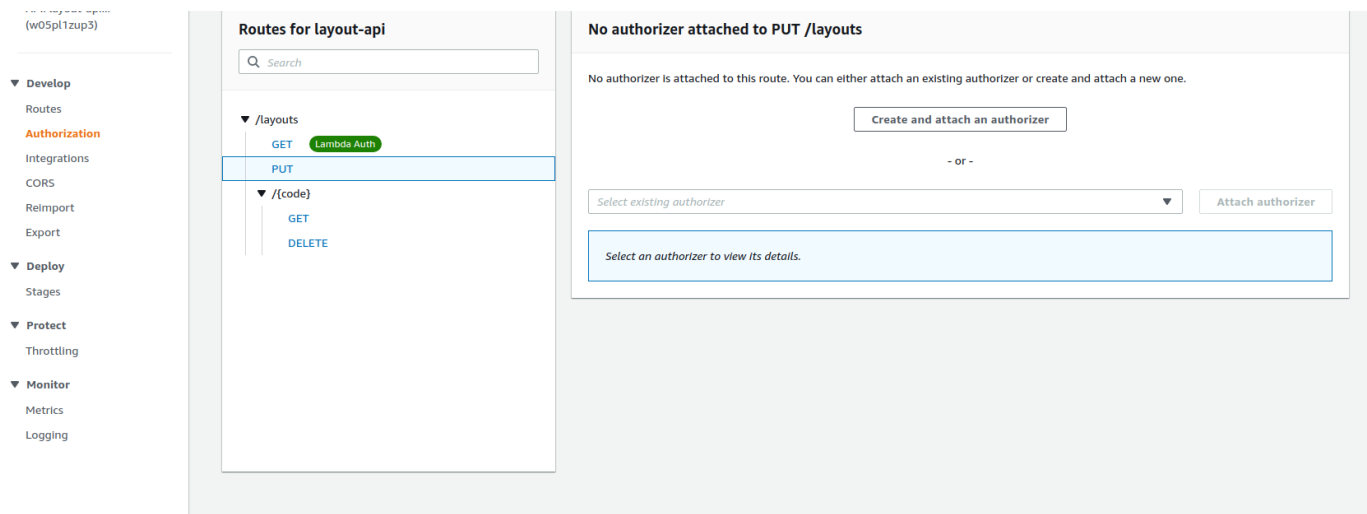
- Attach authorizers to routes:** A section with a 'Create' button.
- Manage authorizers:** A section with a search box and a list of authorizers. The list shows one authorizer named 'authorizer' with a radio button next to it.
- Authorizer details for authorizer:** A section with a 'Delete' button and an 'Edit' button. It contains the following details:
 - Authorizer type:** Lambda
 - Authorizer ID:** vv6qt3
 - Lambda function:** When this authorizer is invoked, API Gateway will invoke this Lambda function to determine whether to allow a request. The value is 'layout-authorizer (us-east-1)'.
 - Payload format version:** The payload format version of the structure of the payload sent to your Lambda function when invoking this authorizer. The value is '2.0'.
 - Response mode:** The type of structure of the response expected from your Lambda function when invoking this authorizer.
 - IAM Policy:**
 - Identity sources:** When this authorizer is invoked, API Gateway will use these selection expressions to determine the source of the token. The value is '\$request.header.Authorization'.
 - Authorizer cache duration:** The number of seconds that cached authorization results are valid. The value is '0'.

STEP 04: Adjuntar authorizer a cada ruta

Ahora tras crear la autorización la podemos adjuntar a esta ruta:

The screenshot shows the 'Manage authorizers' page in the AWS API Gateway console. The page is titled 'API Gateway' and has a sidebar with navigation options: APIs, Custom domain names, VPC links, Develop (Routes, Authorization, Integrations, CORS, Reimport, Export), Deploy (Stages), Protect (Throttling), and Monitor (Metrics, Logging). The main form area is titled 'Authorization' and has a breadcrumb trail: 'API Gateway > Authorization'. The page contains the following sections:

- Attach authorizers to routes:** A section with a 'Create' button.
- Manage authorizers:** A section with a search box and a list of authorizers. The list shows one authorizer named 'authorizer' with a radio button next to it.



STEP 05: Testing endpoints

Ahora los endpoints están securizados por el token antes generado por lo que para que nos den respuestas debemos de pasar el mismo en la cabecera de la petición la **clave authorization con el valor del token** antes generado como se puede ver en la imagen siguiente:

