Implementación API: Autorización

Vamos a explicar como vamos a authorizar 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 podriamos utilizar cualquier otra herramienta compatible en el estandard antes citado.

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

issue: mango

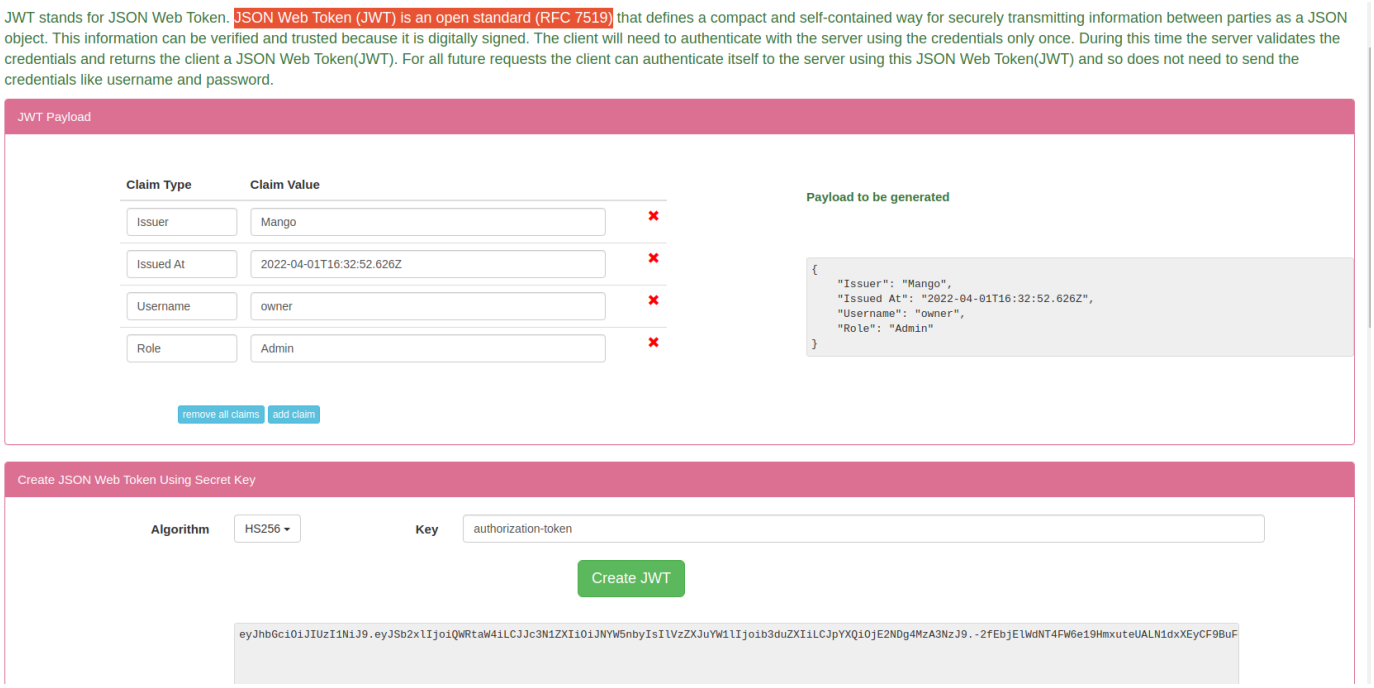
Issued At: 2022-04-01T16:32:52.626Z

Username: owner

Role: Admin

Y la clave:

key: authorization-token



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

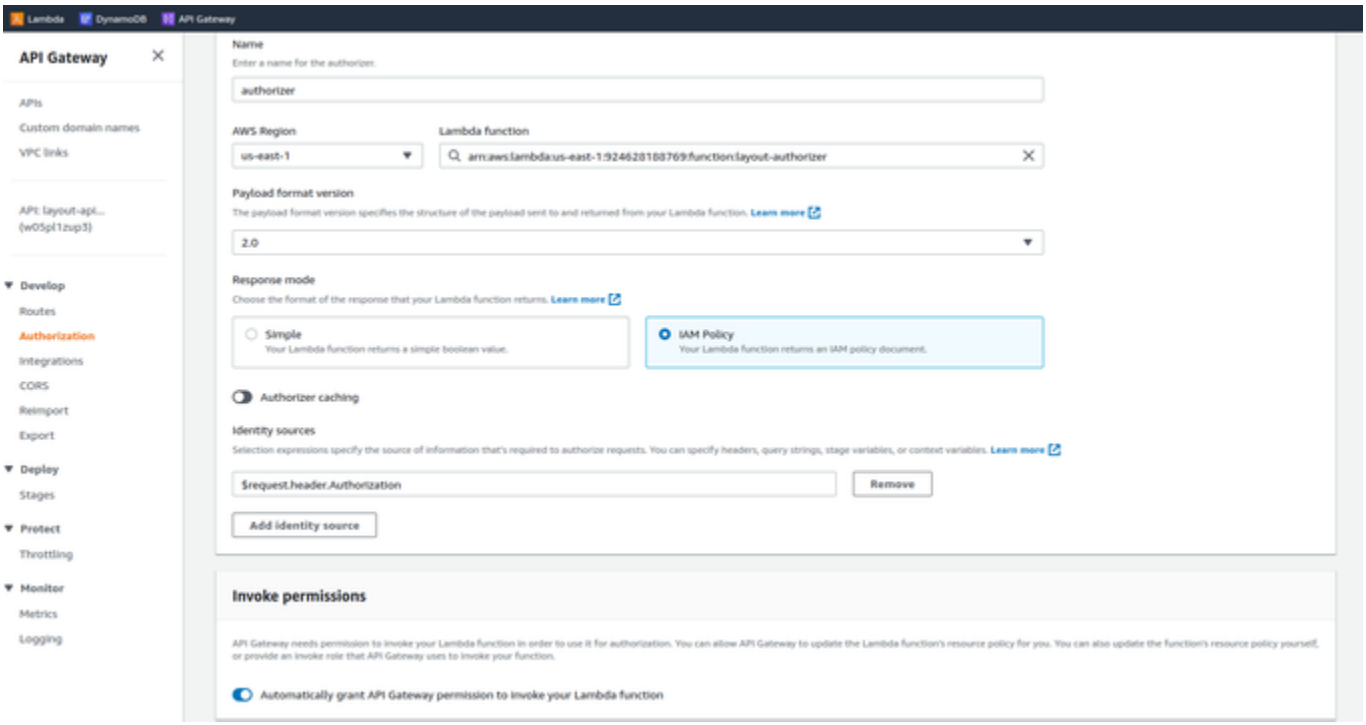
```
exports.handler = async (event) => {
// Define JSON Web Token from these data usinh HS256 hashing algorithm
/*
Token payload:
{
"Issuer": "Mango",
"Issued At": "2022-04-01T16:32:52.626Z",
"Username": "owner",
"Role": "Admin"
}
```

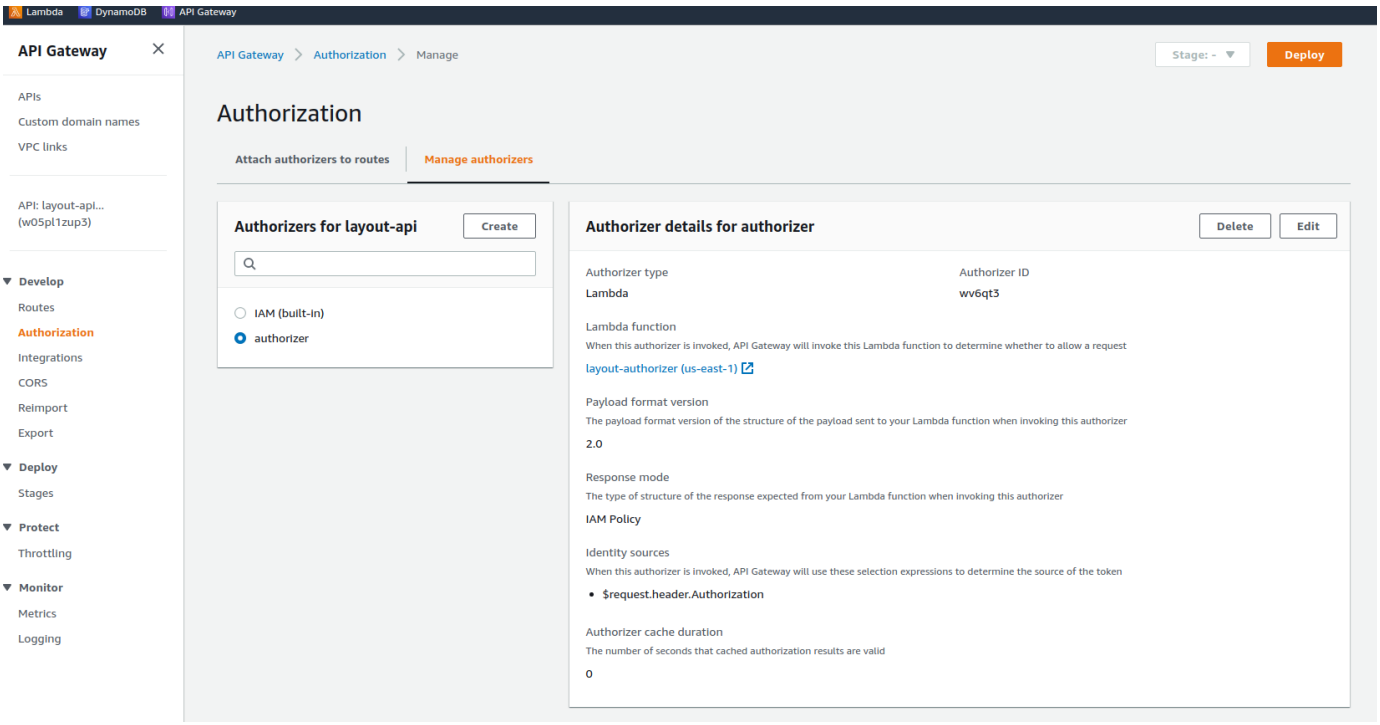
```
exports.handler = async (event) => {
    // Define JSON Web Token from these data usinh HS256 hashing
algorithm
    /*
        Token payload:
        {
            "Issuer": "Mango",
            "Issued At": "2022-04-01T16:32:52.626Z",
            "Username": "owner",
            "Role": "Admin"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

        Key: authorization-token
    */

    const ACCESS_TOKEN = "eyJhbGciOiJIUzI1NiJ9.
eyJSb2xlIjoiQWRtaW4iLCJJc3N1ZXIiOiJNYW5nbyIsIlVzZXJuYW1lIjoib3duZXIiLCJp
YXQiOjE2NDg4MzA3NzJ9.-2fEbjElWdNT4FW6e19HmxuteUALN1dxXEyCF9BuFQg";

    // 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

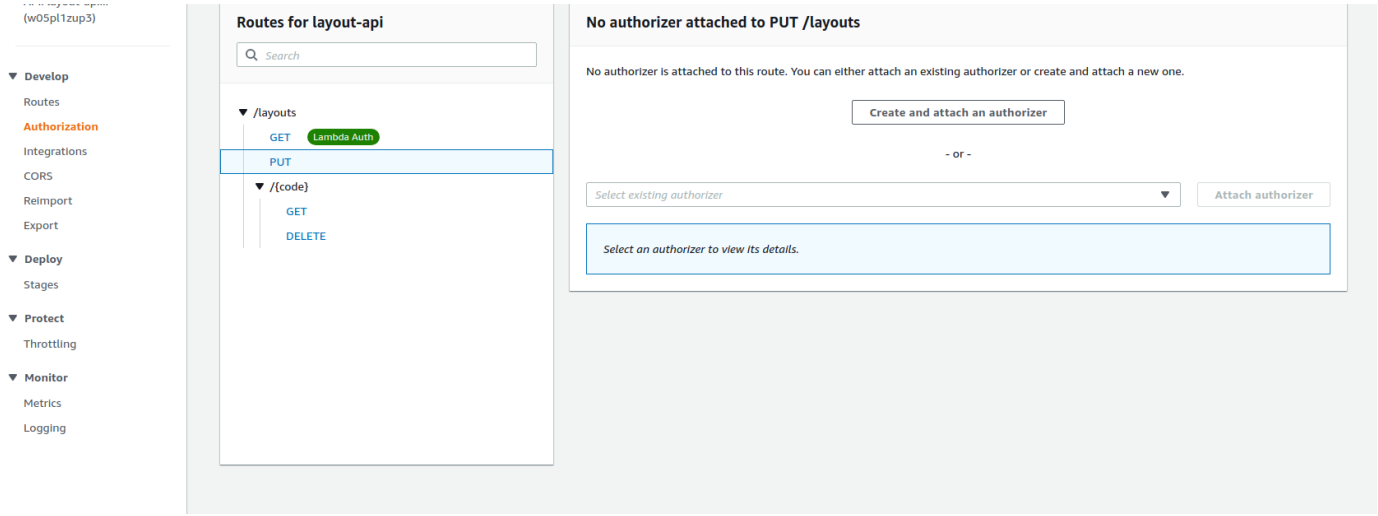
Creamos un authorizer asociado a esta función lambda y lo adjuntamos a cada uno de las rutas definidas: GET, GET {code}, PUT y DELETE {code} desde la opción del API Gateway llamada Authorization. Dentro de la misma escogemos una ruta y desde la pestaña de Manage authorizations creamos una asociando el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





STEP 04: Adjuntar authorizer a cada ruta

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



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: