

API Gateway y CORS

IIC2173 – 2025.2

Para hoy

01

API Gateway

La fachada a nuestro sistema

02

AWS API Gateway

Implementación directa de Amazon

03

CORS

Añadiendo otra capa de seguridad

04

Demo

Demo de todos los conceptos



01

API Gateway

One gateway to rule them all, Simplify your backend with an API Gateway (Chat GPT)



Un Proxy a Nuestro Sistema

Nos ofrece una puerta de entrada común a nuestro sistema y le quita responsabilidades comunes a nuestros servicios internos

https://www.tibco.com/es/reference-center/what-is-an-api-gateway

Propiedades

Ruteo y agregación

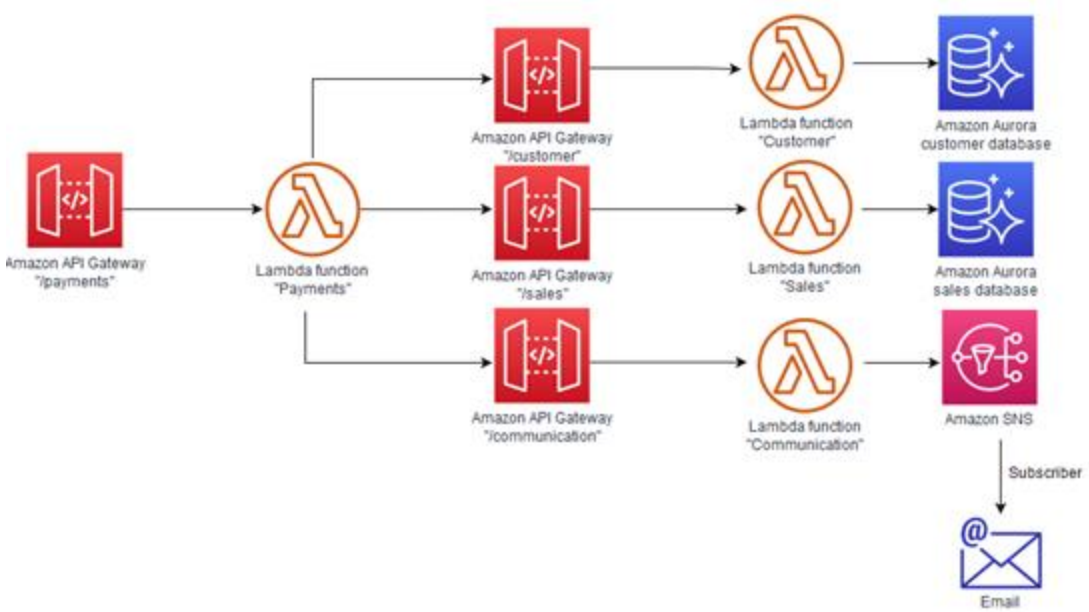
Permite ajustar las rutas con facilidad y redirigir requests

Integrabilidad

Integra diversos servicios y los ofrece de forma transparente. (dev y user)

SPoC (y SPoF)

Control centralizado de servicios a ofrecer

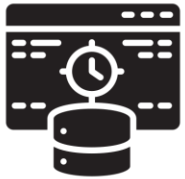


Responsabilidades



Recopilación de Métricas

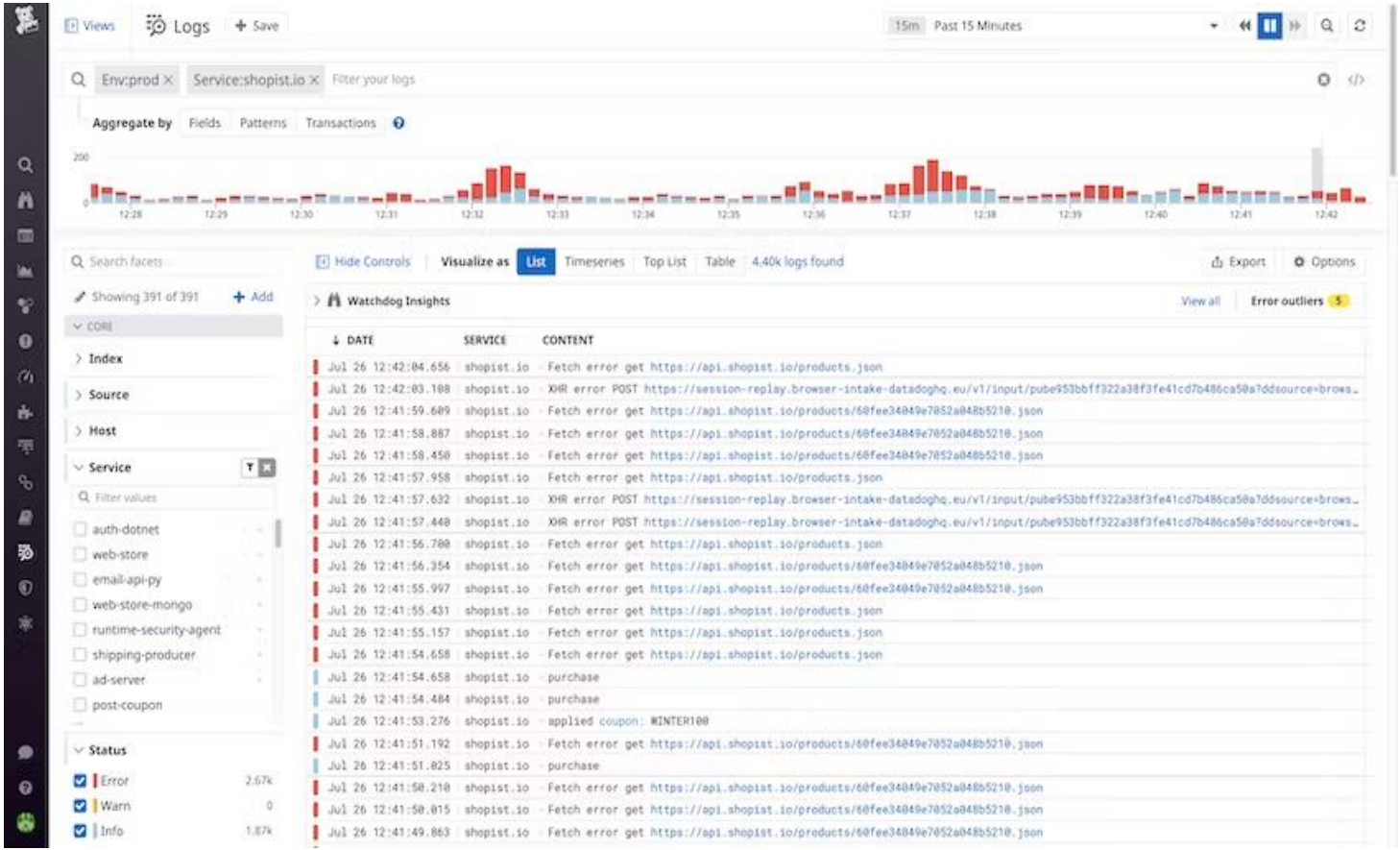
- Logging Distribuido
- Analytics
- Mejora en la Observabilidad



Caché

- Capacidad de retener requests ya hechas anteriormente
- Mejora la performance

https://en.wikipedia.org/wiki/Cross-origin_resource_sharing#/media/File:Flowchart_showing_Simple_and_Preflight_XHR.svg



https://www.datadoghq.com/knowledge-center/log-aggregation/

Responsabilidades



Seguridad

- Política CORS
- Throtling (limitación intencional de ancho de banda)



Autenticación

- Role Based Authentication
- Integraciones OAuth2
- Basic Auth
- HMAC
- Etc.

https://en.wikipedia.org/wiki/Cross-origin_resource_sharing#/media/File:Flowchart_showing_Simple_and_Preflight_XHR.svg

¿Como se asocia con RNF?

+ Seguridad (Entrada Unificada)
+ Integrabilidad (Entrada Unificada)
+ Mantenibilidad (Recopilación de métricas)
+ Performance (caching)
+ Escalabilidad (caching)
- Consistencia (En caso de caching)
- Disponibilidad (Single Point of Failure)*

* No es del todo cierto en la práctica dado el paradigma serverless

...

¡Serverless!



100% bajo demanda

Pago por uso
Atomizado en
ms/GB/CPUs

...



Autogestionado

Mínimo
mantenimiento de
infra

...



Accessible

Diseñados para una
excelente
experiencia de
desarrollador

...

Diversos proveedores

Kong



Azure

Amazon
API Gateway

02



Amazon
API Gateway


Connected Users and Streaming Dashboards
Web and Mobile Applications
IoT Devices
Private Applications: VPC and On-Premises
Amazon API Gateway
Create, publish, maintain, monitor, and secure APIs at any scale
API Gateway cache
Amazon CloudWatch
AWS Lambda
Amazon EC2
Amazon Kinesis
Amazon DynamoDB
Other AWS services
Publicly accessible endpoints
Private Applications
VPC and On-Premises
Data Center

Features interesantes

01

Integrabilidad con AWS

Muchísimos productos se integran suavemente, especialmente Lambda

02

Mantenibilidad

Completamente cloud, deployment de versiones y documentación extensiva

03

Monitoreo

Es posible exportar data a Cloudwatch u otros proveedores

04

Seguridad y Throttling

Permite mejorar el control usando features de AWS

Cuanto cuesta?

AWS

Llamadas a la API

Número de solicitudes (por mes)	Precio (por millón)
Primeros 333 millones	3,50 USD
Próximos 667 millones	2,80 USD
Próximos 19 mil millones	2,38 USD
Más de 20 000 millones	1,51 USD

GCP

Llamadas a la API por mes y cuenta de facturación	Coste por cada millón de llamadas a la API
Hasta dos millones	0,00 USD
De dos millones a mil millones	3,00 USD
Más de mil millones	1,50 USD

HTTP vs REST

Features	REST API	HTTP API
CORS configuration	✓	✓
Test invocations	✓	
Caching	✓	
User-controlled deployments	✓	✓
Automatic deployments		✓
Custom gateway responses	✓	
Canary release deployments	✓	
Request validation	✓	
Request parameter transformation	✓	✓
Request body transformation	✓	

Entre otras opciones de

- Monitoreo
- Integraciones
- Manejo
- Autorización

En general REST API es mucho más completo pero más caro y restrictivo

03

CORS

Estricto, necesario, molesto

¡Video 100 seg!



https://www.youtube.com/watch?v=4KHiSt0oLJ0



Fomentando una internet más segura

CORS nos permite proteger recursos de un servidor frente a clientes no deseados mediante la política **same-origin**

Conceptos clave

Origen

Nombre de host que está haciendo la petición

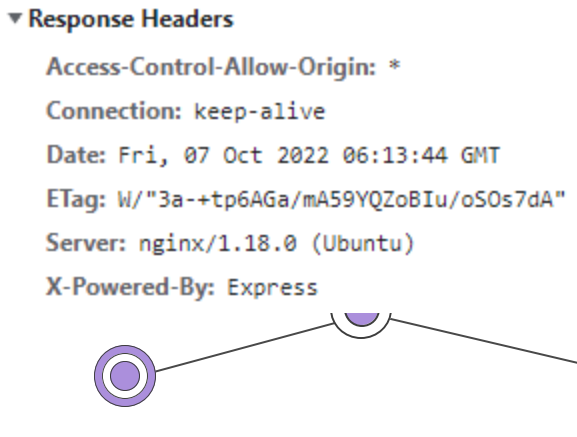
scheme://hostname:port

1 combinación = 1 origen

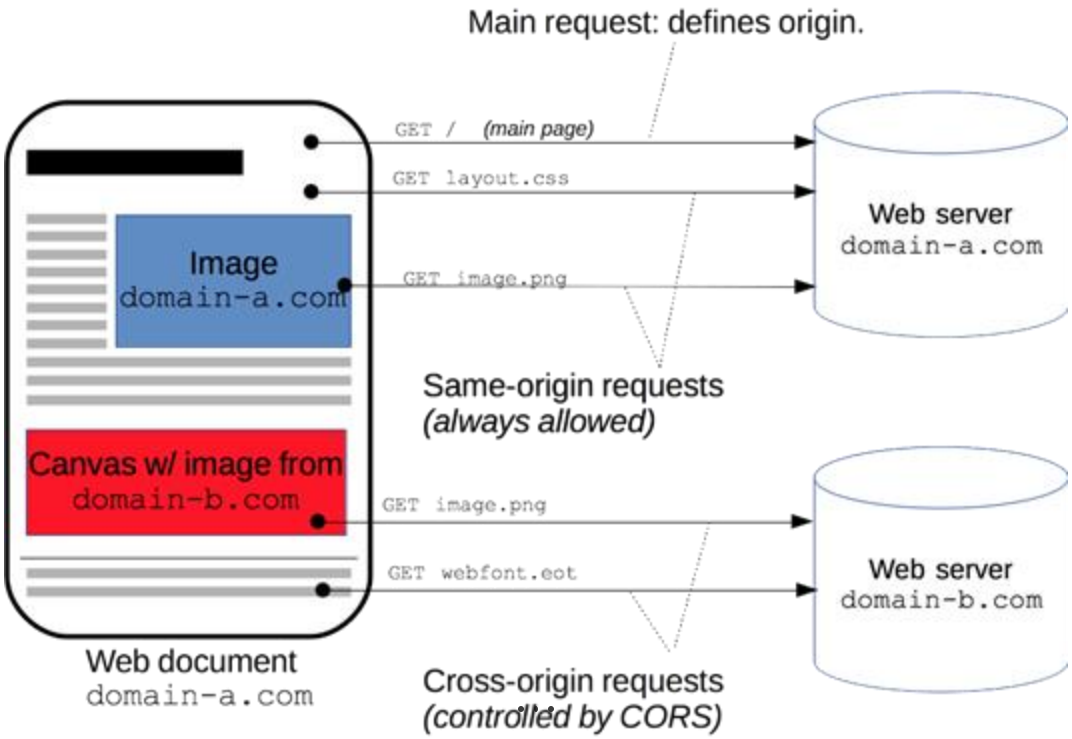
Origin: https://developer.mozilla.org

Headers

Encabezados que se envían con cada request HTTP



Funcionamiento de CORS



https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS/cors_principle.png

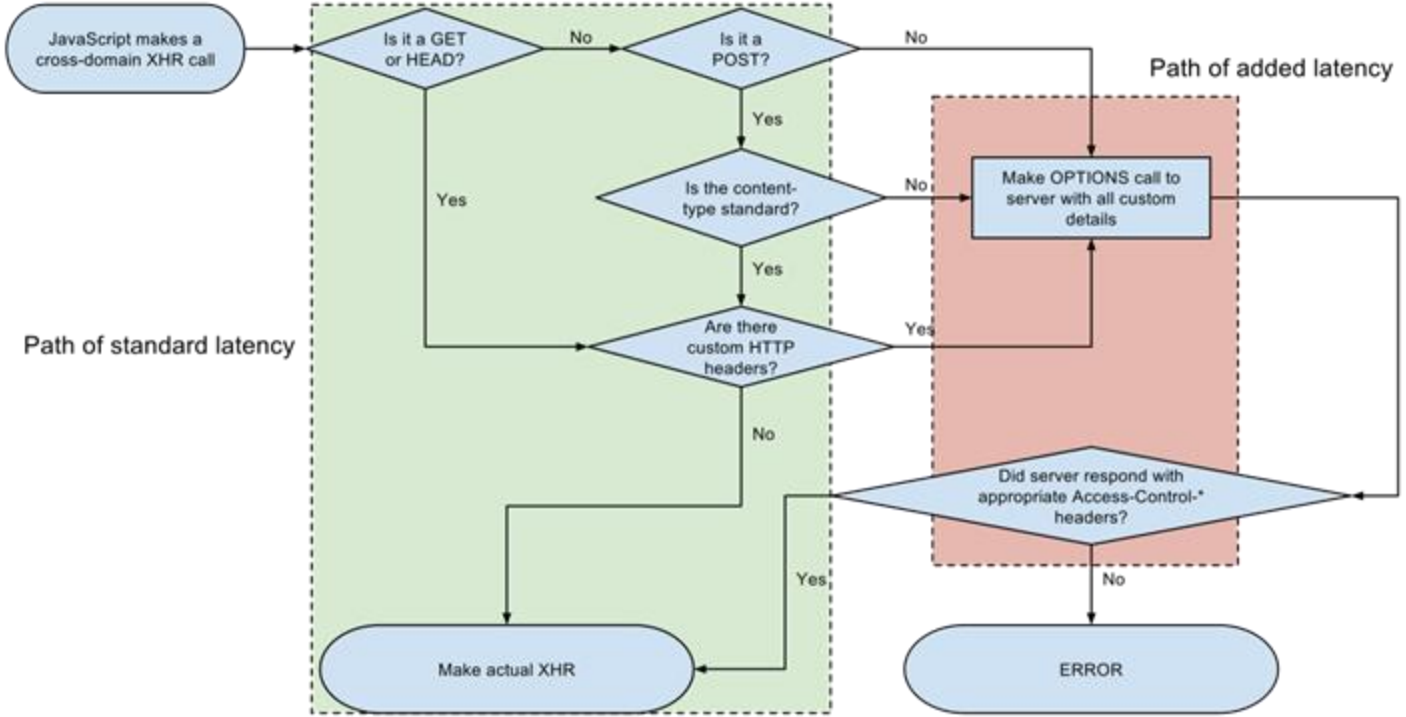
Headers de CORS

`Access-Control-Allow-Origin`	➔ ¿Qué orígenes están permitidos?
`Access-Control-Allow-Credentials`	➔ ¿Puedo enviar cookies?
`Access-Control-Expose-Headers`	➔ ¿Qué headers pueden leer los scripts?
`Access-Control-Max-Age`	➔ ¿Puedo cachear los permisos de CORS?
`Access-Control-Allow-Methods`	➔ ¿Qué métodos puedo llamar?
`Access-Control-Allow-Headers`	➔ ¿Qué headers puedo enviar?

Esto para requests que no vienen del mismo origen del servidor (Cross-origin)

https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS/cors_principle.png

Funcionamiento de Preflight



https://en.wikipedia.org/wiki/Cross-origin_resource_sharing#/media/File:Flowchart_showing_Simple_and_Preflight_XHR.svg

Para más información consultar MDN y especificación W3C

MDN: https://developer.mozilla.org/es/docs/Web/HTTP/CORS
https://developer.mozilla.org/es/docs/Glossary/CORS

W3C Spec: https://www.w3.org/TR/2020/SPSD-cors-20200602/

04

Auth0

Como usamos API Gateway con nuestra autentificación

Hablemos de la autenticación



Queremos que sea de terceros

No queremos preocuparnos de su mantenibilidad



Altamente escalable

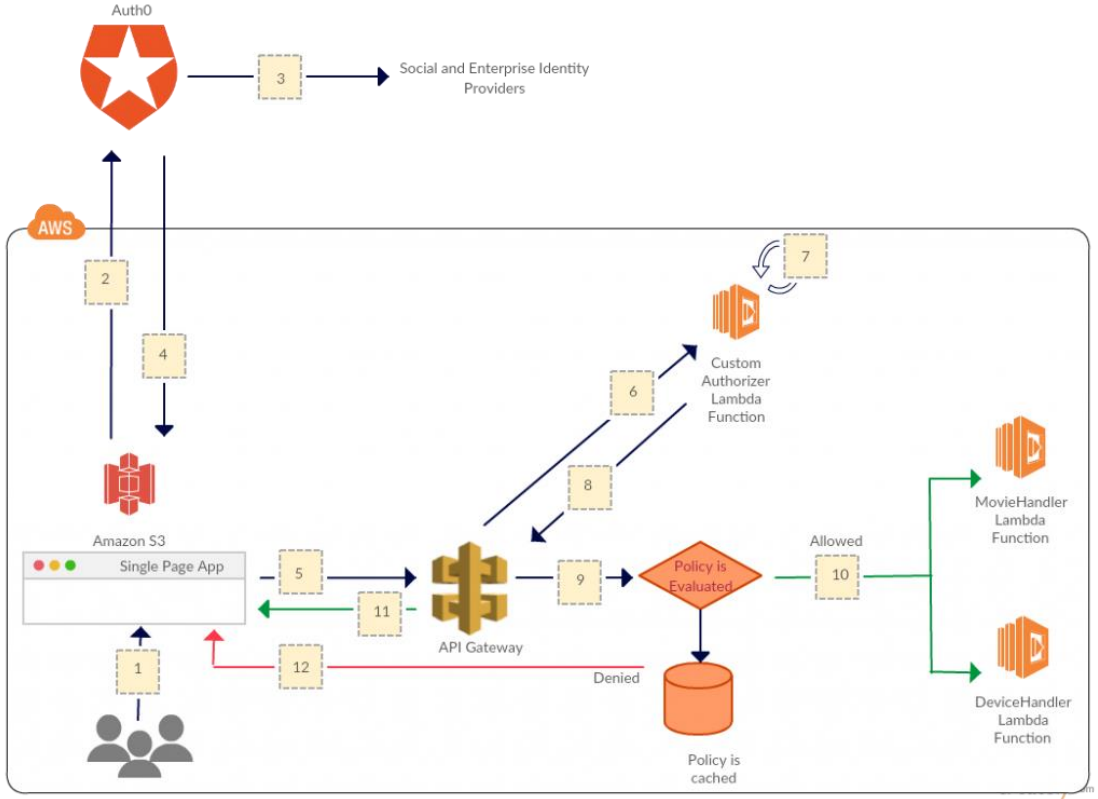
Debe ser un servidor aparte para poder mejorarlo



Queremos proteger nuestros endpoints

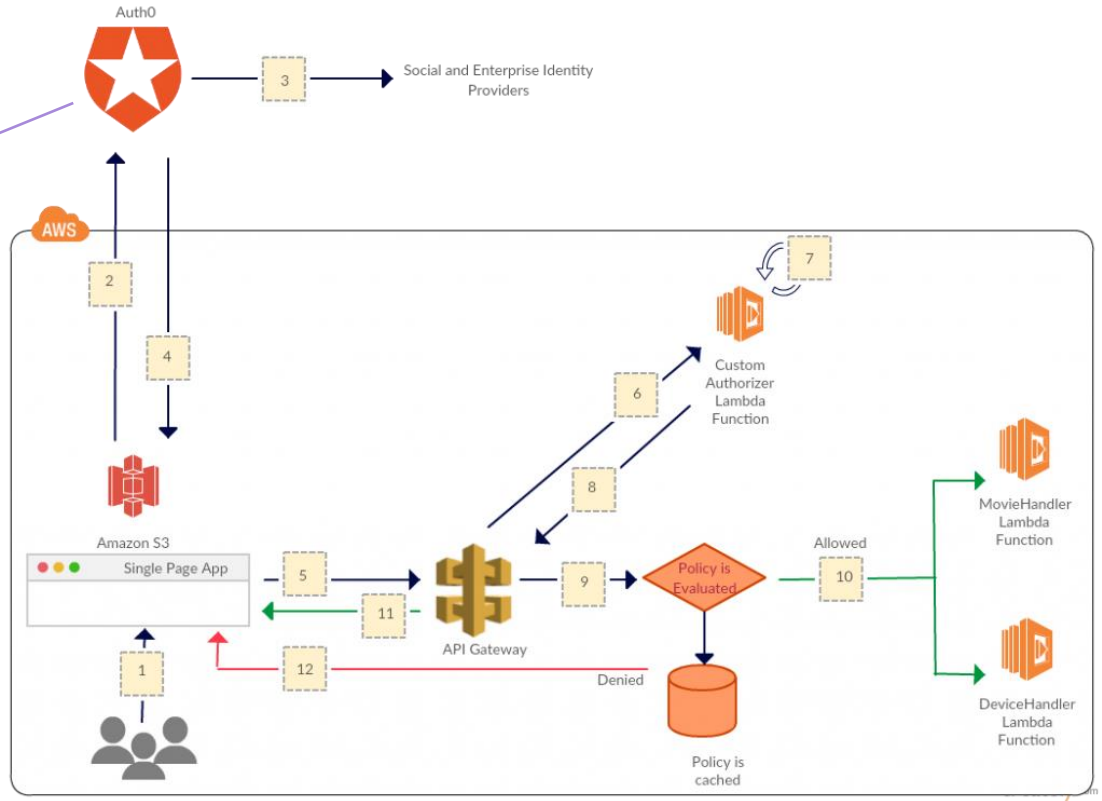
Que nuestros endpoints sean accedidos por un usuario autorizado

¿Cómo se ve este flujo?

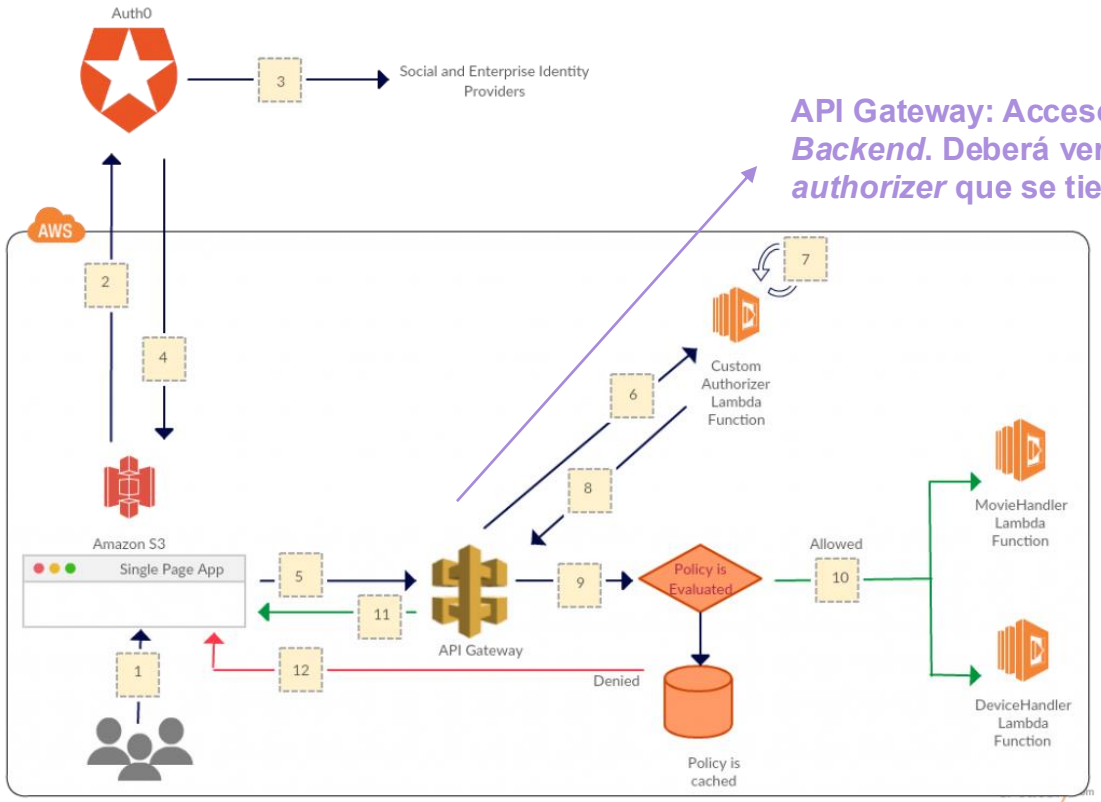


¿Cómo se ve este flujo?

Auth0: Servicio de autenticación de terceros



¿Cómo se ve este flujo?



API Gateway: Acceso a nuestro *Backend*. Deberá verificar con el *custom authorizer* que se tienen los accesos

¡Demo!