

AWS API Gateway



**DE HOGESCHOOL
MET HET NETWERK**

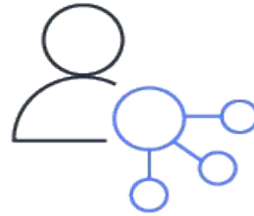
Elfde-Liniestraat 24, 3500 Hasselt, www.pxl.be

REST(ful) APIs

An API is a software mechanism that simplifies development by doing the following:



Abstracting
implementation details

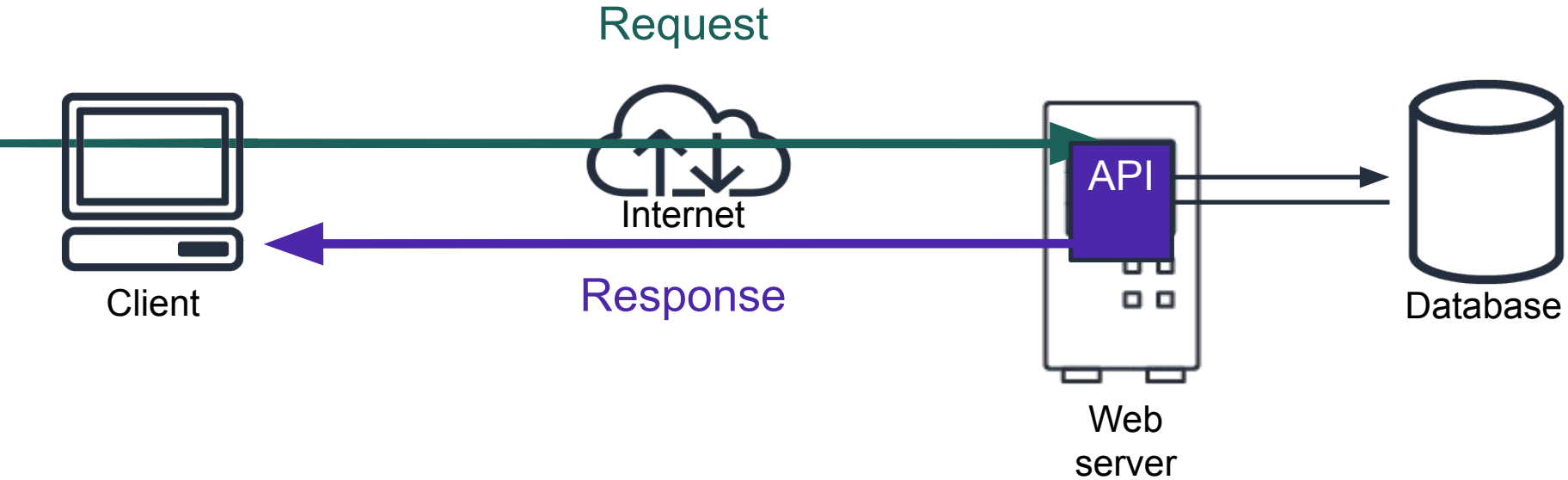


Exposing only objects or
actions that a developer
needs



Establishing how an
information provider and
an information user
communicate

REST(ful) APIs



REST(ful) APIs

Representational State Transfer (REST)

- Architectural style
- Standard way of structuring requests from a client to a server

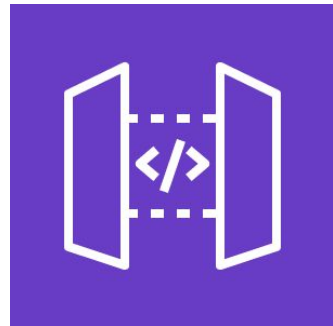
RESTful API

- An API that adheres to REST principles

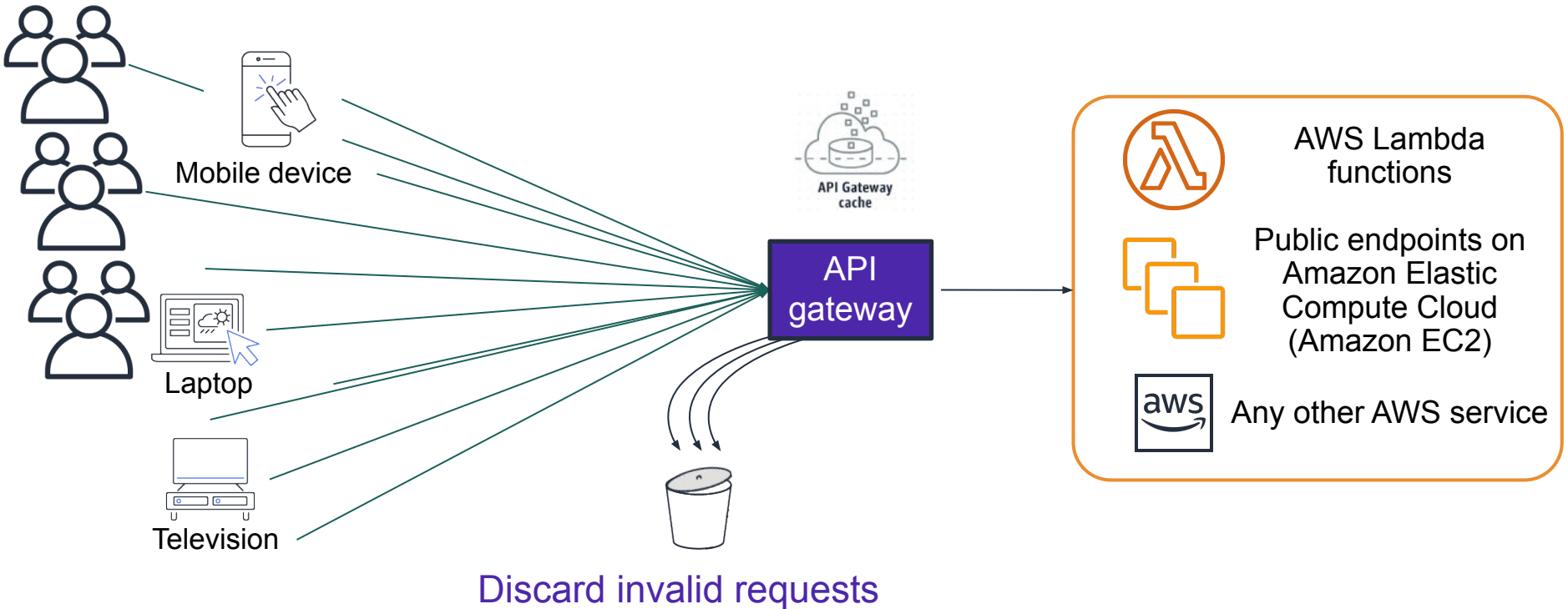


API Gateway

- Fully managed, serverless service
- Servers as a proxy that:
 - Abstract common functionality from the server
 - Reduce the number of requests that reach the server
 - Limit the request rate
 - Allow or disallow requests by source
 - Validate the developer key
 - ...

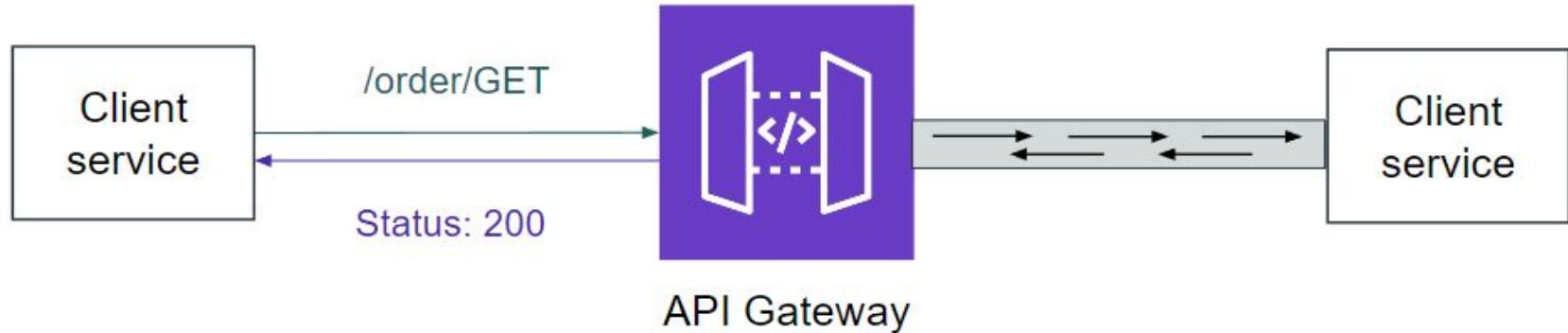


API Gateway



Supported protocols

HTTP vs Websockets



Types of HTTP Gateways

REST API

- Gives the developer full control over API requests and responses
- Supports features that are not yet available in HTTP APIs

HTTP API

- Simplifies the development of APIs that require only API proxy functionality
- Designed for the lowest cost and lowest latency

Creating an API Gateway

Using the CLI:

```
aws apigatewayv2 create-api --name my-api --protocol-type HTTP --target  
arn:aws:lambda:us-east-2:123456789012:function:function-name
```

General url structure of an API gateway:

`https://{restapi_id}.execute-api.{region}.amazonaws.com/{stage_name}/`

|
{restapi_id}
API identifier

|
{region}
The AWS Region

|
{stage_name}
Stage name of the API
deployment

Stages

- API configuration can be deployed to a stage
- Stages are different environments

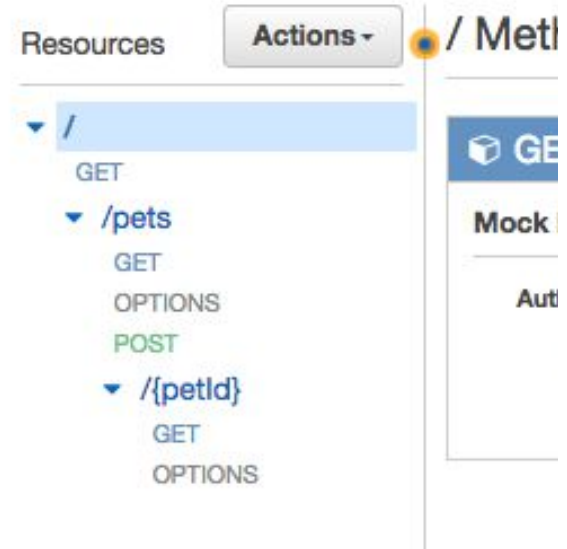
For example:

- apiV1
- dev*
- beta
- prod*
- As many stages as you need

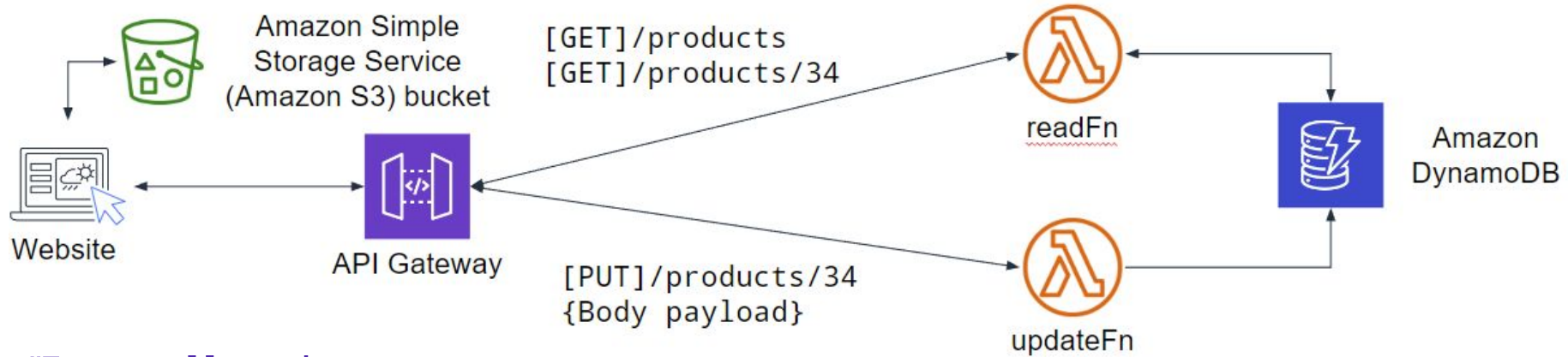
** Best practise for environments is to have multiple AWS accounts per environment*

Resources

- Structure of the API is shown as a tree of nodes
 - Each resource is a new endpoint in the API gateway
- API methods defined on each resource
- Each method can be linked to an endpoint
eg. Lambda, EC2 http endpoint, ...



Resources



#To get all products

[GET] <https://api-id.execute-api.us-east-2.amazonaws.com/products>

#To get a specific product by ID (34)

[GET] <https://api-id.execute-api.us-east-2.amazonaws.com/products/34>

#To update (replace) a specific product by ID (34)

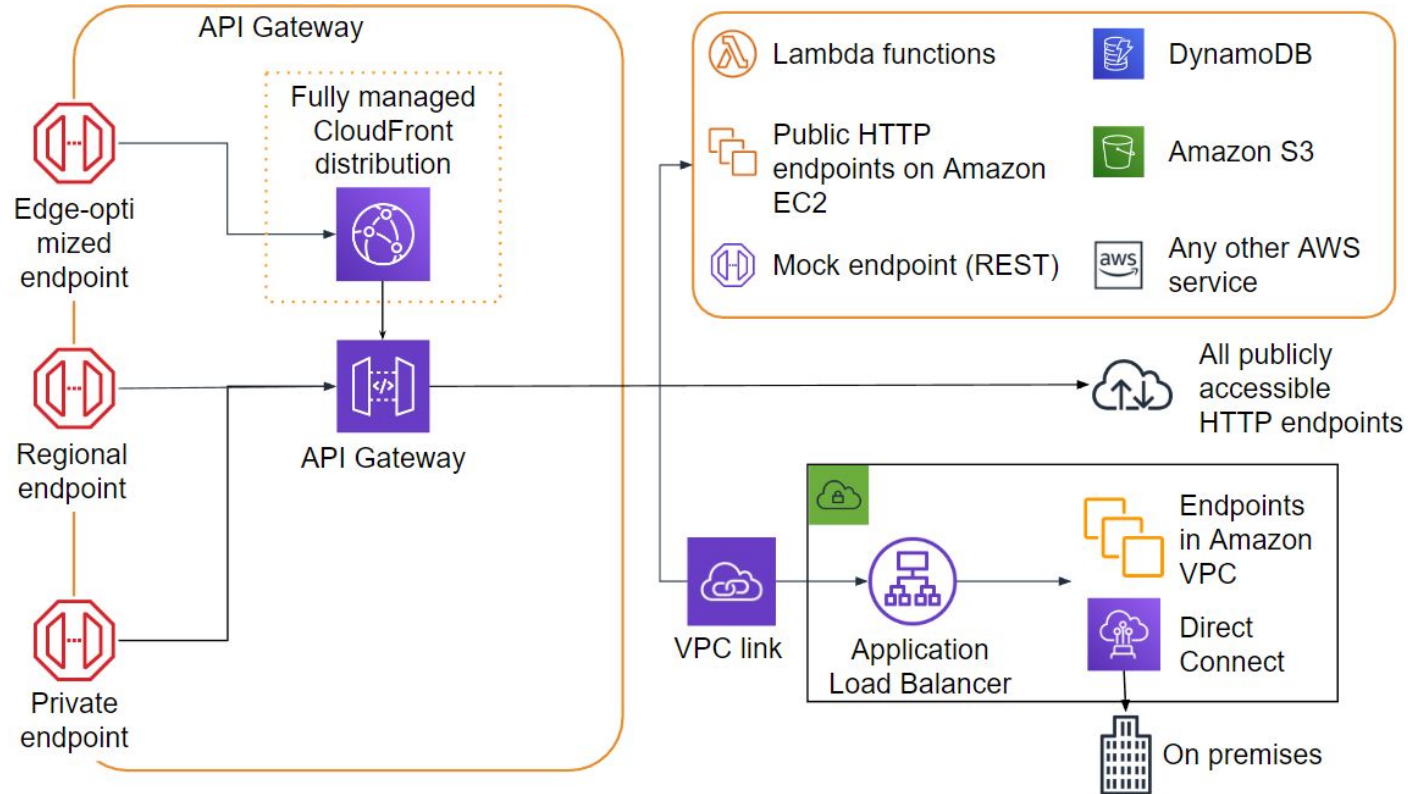
[PUT] <https://api-id.execute-api.us-east-2.amazonaws.com/products/34>

[BODY] #some keys and values

Endpoint types

- **Edge optimised:** when the clients accessing the API are distributed globally. Creates a CloudFront distribution that is configured to route requests to your API
- **Regional endpoint:** when majority of the clients accessing the API are located in the same region as the API itself
- **Private endpoint:** when the API needs to be accessible only from within your Amazon VPC or through a Direct Connect connection

Backend integrations



Creating an endpoint

Using the AWS console

- Choose the request type and URI
- Choose the linked resource (lambda, ec2, ...)
- Multiple request types per URI are possible

Routes

Routes for calculator-app

Create

Search

▼ /calc

▼ /add

GET

Route details

GET /calc/add (ID: q4zbbsa)

Authorization

Authorizers protect your API against

qxz2e0

Configure

Integration

The integration is the backend resou

ogwpl5b

Configure

Create a route

Route and method [Info](#)

Choose a method and enter a path to create a route. You can also sp

ANY

ANY

GET

POST

PUT

/

Transforming requests / responses

Transform data before going to your endpoint

eg. Lambda functions cannot handle headers/params
-> convert data to a json object

Note: Lambda's have integrated prebuild proxies

Provide information about the target backend that this method will call and whether

Integration type ☐ Lambda Function ⓘ

☒ HTTP ⓘ

☐ Mock ⓘ

☐ AWS Service ⓘ

☐ VPC Link ⓘ

Use HTTP Proxy integration ☐ ⓘ

HTTP method POST ✎

Endpoint URL <https://demo-url.com/api/customer> ✎

Content Handling Passthrough ✎ ⓘ

Use Default Timeout ☒ ⓘ

URL Path Parameters

Name	Mapped from ⓘ
------	---------------

No path parameters

[+ Add path](#)

URL Query String Parameters

Name	Mapped from ⓘ
------	---------------

No query strings

[+ Add query string](#)

HTTP Headers

Name	Mapped from ⓘ
------	---------------

No headers

[+ Add header](#)

Controlling access to REST APIs

- IAM Resource policies
 - Allow only certain AWS resources to access your API
 - More about this later in this course

Resource Policy

Configure access control to this private API using a Resource Policy. Access can be controlled by IAM condition and/or IP range. If the Principal in the policy is set to *, other authorization types can be used alongside the resource secured with AWS IAM auth, including unsecured resources. Changes to this policy require a deployment to take

```
1 {  
2   "Version": "2012-10-17",  
3   "Statement": [  
4     {  
5       "Effect": "Allow",  
6       "Principal": {  
7         "AWS": [  
8           "arn:aws:iam::{{otherAWSAccountID}}:root",  
9           "arn:aws:iam::{{otherAWSAccountID}}:user/{{otherAWSUserName}}",  
10          "arn:aws:iam::{{otherAWSAccountID}}:role/{{otherAWSRoleName}}"  
11        ]  
12      },  
13      "Action": "execute-api:Invoke",  
14      "Resource": [  
15        "execute-api:/{stageNameOrWildcard*}/{httpVerbOrWildcard*}/{res"   
16      ]  
17    }  
18  ]  
19 }
```

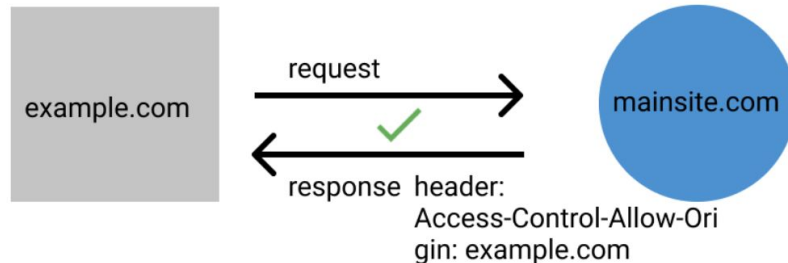
Controlling access to REST APIs

- AWS WAF (Web Application Firewall)
 - Protect APIs from common web exploits such as:
 - SQL injection
 - XSS
 - ...
 - Block request based on origination (geo, ips, user-agents, bots, ...)

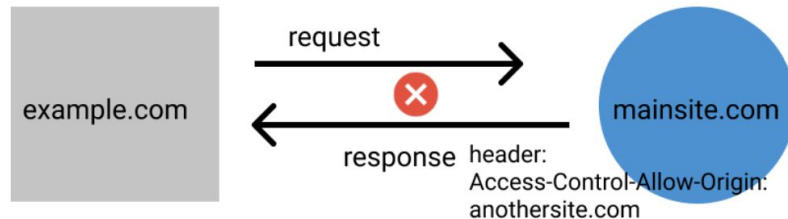
Controlling access to REST APIs

- CORS (Cross Origin Resource Sharing)

- Browser based security
- Blocks requests based on
 - origin
 - request type
 - headers



Good: Origin is in response header



Error: Origin not in response header

Controlling access to REST APIs

- Rate limiting
 - Throttle rates per stage or per method
 - Can be linked to API keys
 - eg. 10k requests / month for a free tier API key
 - 100K req / month for a premium type API key that costs 5\$ / month
 - example: Twitter API, Google Maps API

Authentication on REST APIs

- Open APIs, no authentication
- IAM authentication
- JWT-based authentication
 - integrates with identity providers such as Amazon Cognito
 - OpenId Connect / OAuth2
- Lambda authorizers
 - Seperate function that validates a bearer token

The API gateway validates the tokens

Simple Authentication example (1/3)

- Create an AWS cognito user pool
 - AWS provides a web page (Cognito Hosted UI)
 - This web app provides simple user registration / login / forgot password features
 - Alternative is to use the AWS sdk to inject users in to the pool from your app

The screenshot displays the AWS Cognito console interface. At the top, the 'User pool overview' section provides key details for the 'calculator-app' user pool, including its ARN, ID, and the estimated number of users (1). Below this, the 'Getting started' section offers navigation links for 'Users', 'Groups', 'Sign-in experience', 'Sign-up experience', 'Messaging', 'App integration', and 'User pools'. The 'Users' tab is currently selected, showing a list of users. The list includes columns for 'User name', 'Email address', and 'Email verified'. A single user named 'dries' is listed with a verified email address. The interface also includes a search bar and a 'Property' dropdown menu set to 'User name'.

User name	Email address	Email verified
dries	[REDACTED]	Yes

Simple Authentication example (2/3)

- Create an app client for that user pool
 - allows unauthenticated API calls to that user pool
 - Defines token settings
 - Provide a callback URL to our frontend application that receives the token

App client information		
App client name calculator_app	Authentication flow session duration 3 minutes	Created time April 10, 2023
Client ID 4r3ccn4h4palnap192v469l0kl	Refresh token expiration 30 day(s)	Last updated time April 10, 2023
Client secret -	Access token expiration 60 minutes	
Authentication flows ALLOW_CUSTOM_AUTH ALLOW_REFRESH_TOKEN_AUTH ALLOW_USER_SRP_AUTH	ID token expiration 60 minutes	
	Advanced authentication settings Enable token revocation Enable prevent user existence errors	

Hosted UI [Info](#)

Configure the Hosted UI for this app client.

Hosted UI status

Available

Allowed callback URLs

http://localhost:4200/callback

Allowed sign-out URLs

-

Identity providers

Cognito user pool directory

OAuth grant types

Implicit grant

OpenID Connect scopes

email

openid

Simple Authentication example (3/3)

- Link the user pool and app client to a route in the API gateway
 - The issuer URI has the following structure:
<https://cognito-idp.{region}.amazonaws.com/{userPoolId}>
 - The Audience contains the app client identifier

Authorizer for route GET /calc/add
Detach authorizer

Authorizer name	Authorizer type	Authorizer ID
calculator-app-user-pool	JWT	qxz2e0

Identity source

When this authorizer is invoked, API Gateway will use this selection expression to determine the source of the token

\$request.header.Authorization

Issuer

The issuer URI of the Identity Provider

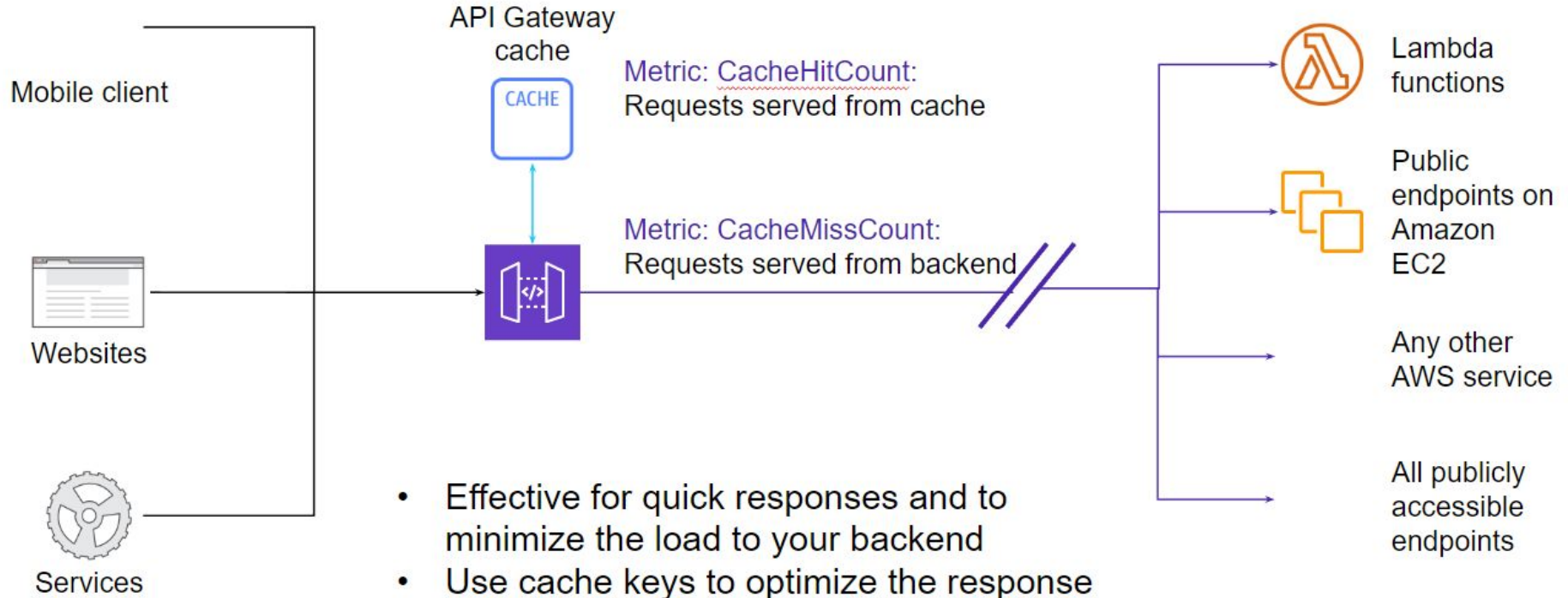
https://cognito-idp.us-east-1.amazonaws.com/us-east-1_XwVdOsUQC

Audience

The audience associated with this authorizer

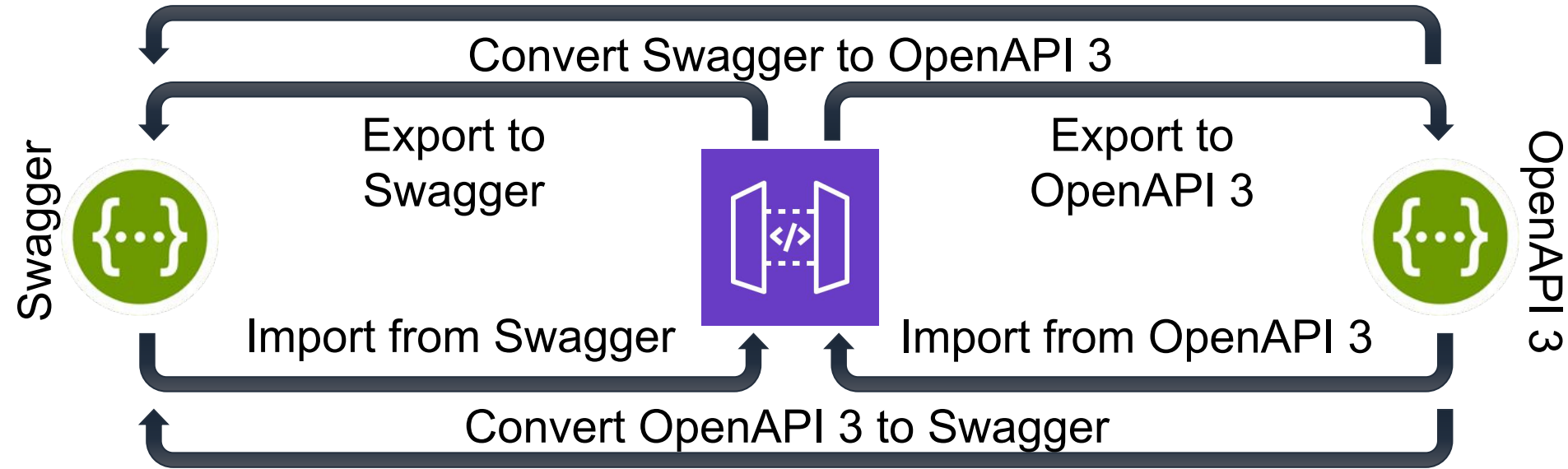
- 4r3ccn4h4palnap192v469l0kl

Caching



- Effective for quick responses and to minimize the load to your backend
- Use cache keys to optimize the response
 - Path, headers, query strings
- Can be set up per stage or method

Import / Export APIs



API gateway labs



- [Online lab platform](#)
 - AWSGen (Lambda)
 - Deploy a lambda function an in AWS VPC

