Profit4Cloud

professionals in cloud engineering

TECH UPDATE

Serverless APIs with AWS

About Us

Company

- 40 employees we're hiring!
- **MISSION:** Helping clients with our Cloud expertise
- Resident of Entrada 100

Me

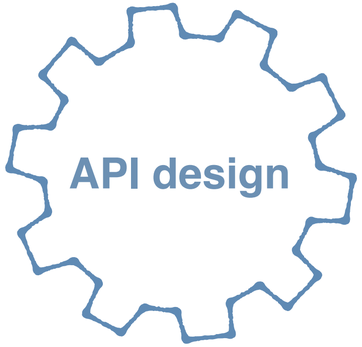
- Dick Eimers, Cloud Solution Architect
- 12+ years experience designing and creating software
- **Convinced the future of IT is in the Cloud** one way or another ;-)

Profit4Cloud

Plan

Part 1

API design



Share basic API design tips to create APIs that work learned from doing it wrong a few times

Part 2

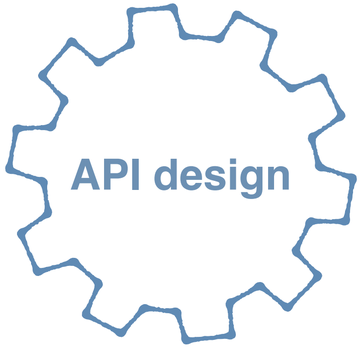
Serverless implementation

Introduce Amazon's serverless building blocks for creating such APIs

Show you example code!

Profit4Cloud

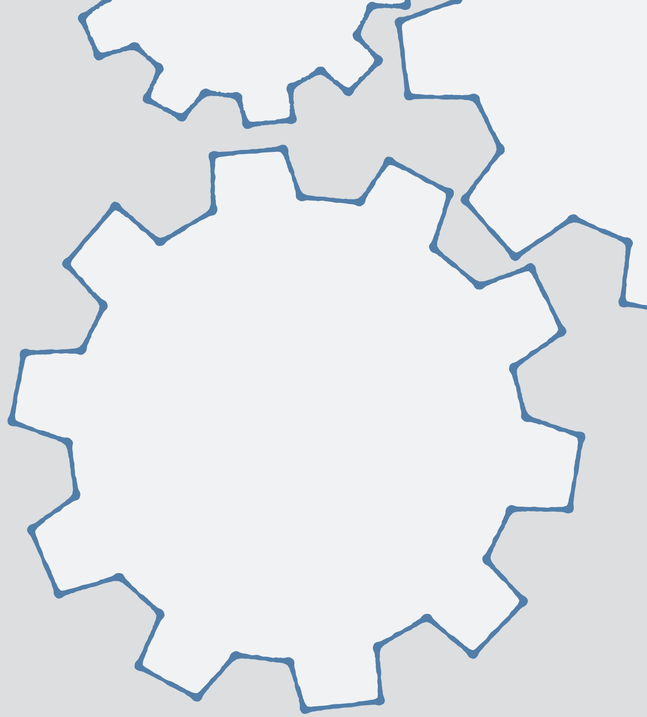
Part 1

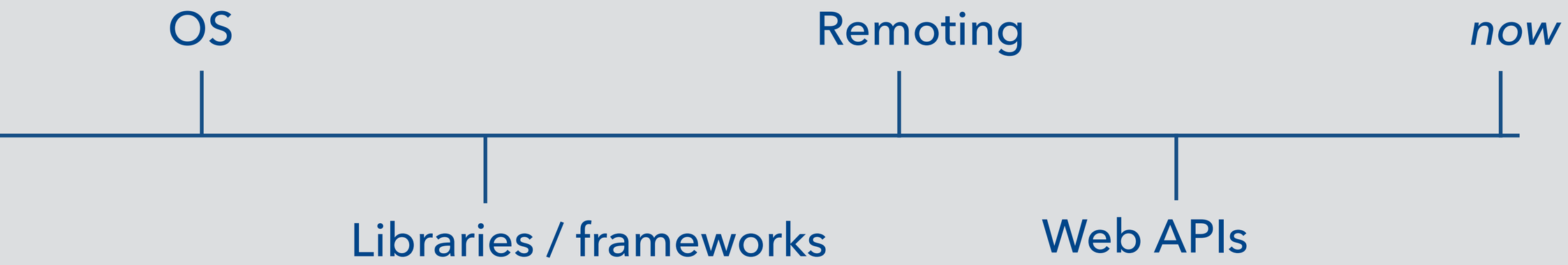


Profit4Cloud

History of APIs

In general terms, it is a set of clearly defined methods of communication between various software components.





Profit4Cloud

Web APIs

- Online networked service available via a **URL** communicating **HTTP**:

  http(s)://so.me.host:someport/some/path?some=param

  path query

- Input by interpreting the **path**, **query parameters**, **headers** and **request body**
- Output by **response codes** (e.g. 200 OK), **headers** and **response body**

Profit4Cloud

Body Content-Type

```
<?xml version="1.0" encoding="UTF-8" ?>
<product>
      <price>10</price>
      <product_id>12345</product_id>
      <name>some product</name>
      <description>none</description>
</product>
```

XML


```
{
  "product": {
    "price": 10,
    "product_id": "12345",
    "name": "some product",
    "description": “none"
  }
}
```

JSON

```
---
product:
  price: 10
  product_id: 12345
  name: some product
  description: none
```

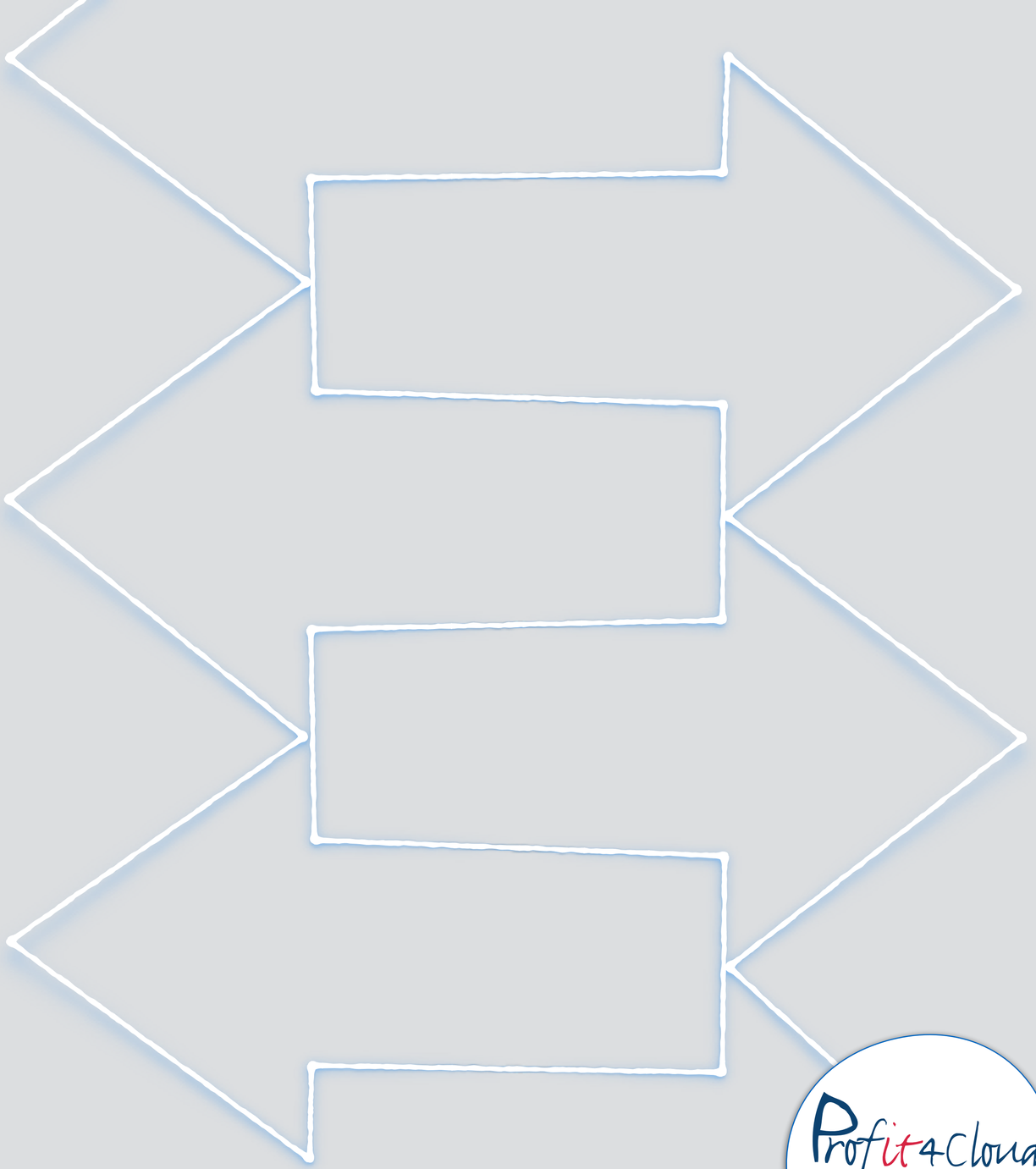
YAML

easier on the eyes →

Profit4Cloud

API Design

Who to think of when designing APIs?



Profit4Cloud

API Design

Roy T. Fielding
@fielding
Following

I'd really like to figure out how Microsoft manages to screw up every decision they make that involves deploying an HTTP header field.

1:48 AM - 1 Jun 2012

Roy T. Fielding
@fielding
Following

Replying to @kragen

Probably not. I have my own REST guidelines. Adobe has its own API guidelines. At some point, we need to publish a mix tape.

1:47 AM - 20 Jul 2016

Roy T. Fielding
@fielding
Following

Replying to @mnot

@mnot Oy, yes, PATCH was something I created for the initial HTTP/1.1 proposal because partial PUT is never RESTful. ;-)

6:27 AM - 3 Dec 2012

Roy T. Fielding
@fielding
Following

The reason to make a real REST API is to get evolvability … a "v1" is a middle finger to your API customers, indicating RPC/HTTP (not REST)

12:33 AM - 9 Sep 2013

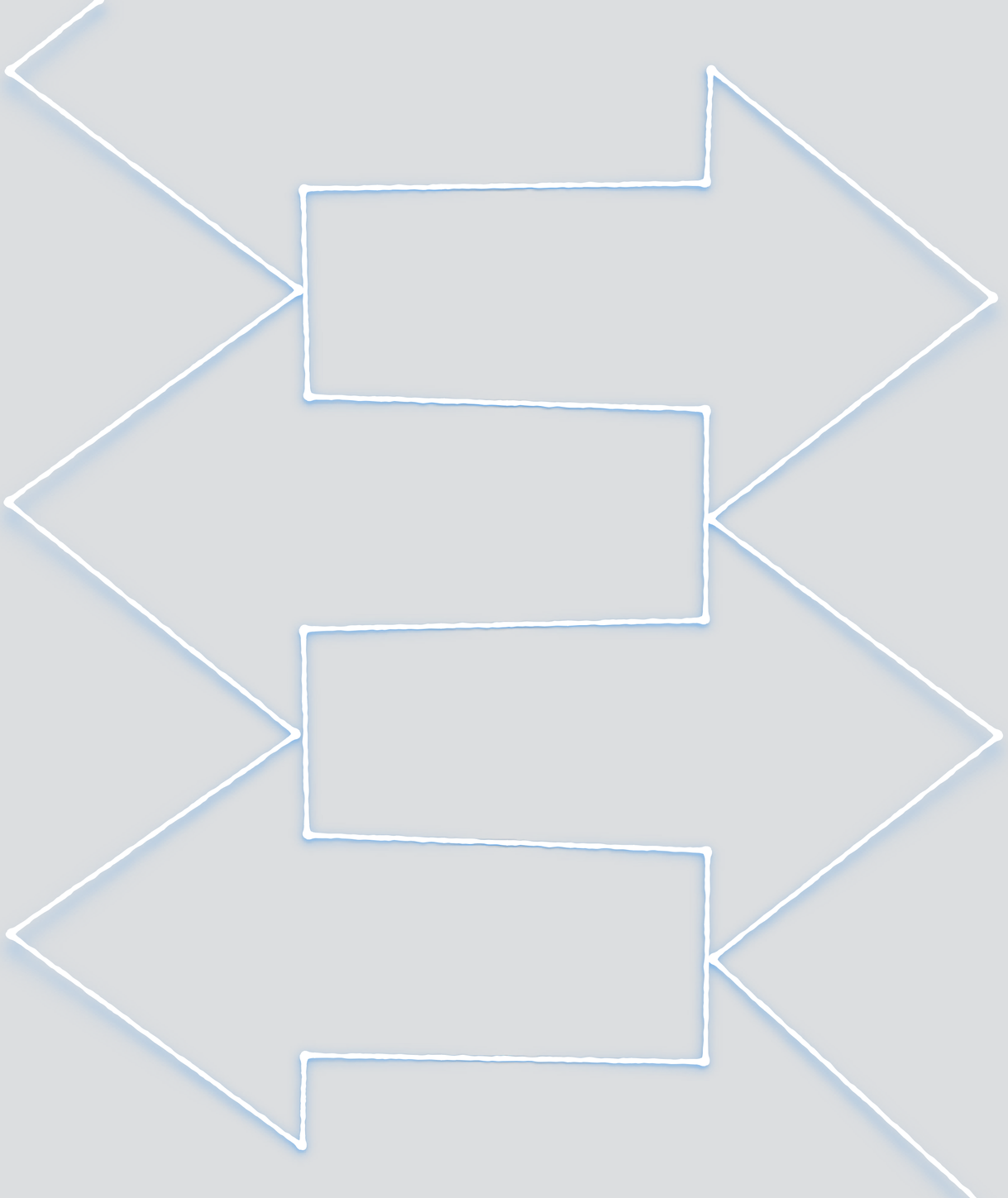
Roy T. Fielding
@fielding
Following

WTF is a "REST endpoint"?

Glass Explorer Edition | Google Developers
Develop Glassware
developers.google.com

11:04 AM - 17 Apr 2013

Profit4Cloud



Design for

<**developers**>

<you>

<me>

Profit4Cloud

<**developers**>

<you>

An API that works for <me> is

intuitive

and

well documented

Profit4Cloud

Being **intuitive** is not easy..

Profit4Cloud

General Principles

Try to build what the user expects

(don't try to be smart)

Profit4Cloud

Care about the documentation;

include examples where ever you can

Profit4Cloud

Be conservative in what you send
and liberal in what you accept

a.k.a. Postel's law, see https://en.wikipedia.org/wiki/Robustness_principle

Profit4Cloud

Specifics

URIs are preferably resource-oriented

	https://ap.i/products	https://ap.i/products/{id}
GET	List products	Get a product
POST	Create a new product	
PUT		Update or create a product
DELETE		Delete a product

Profit4Cloud

	https://ap.i/products	https://ap.i/products/{id}
GET	List products	Get a product
POST	Create a new car	
PUT		Update or create a product
PATCH		Partial update
DELETE		Delete a product

Profit4Cloud

Resources preferred, actions allowed..

- The **resource-oriented alternative feels far-stretched**:

  POST **/messages/123/resend**

  POST **/products/456/rate**

- Search for multiple resource

  GET /**search?q=term**

Profit4Cloud

Use of..

path	query params
Use for what is mandatory	Use for what is optional
GET **/products**	GET **/products?sort=asc.name**
GET **/products/456**	
GET **/products/456/parts/4**	

Use the path to select a specific service and params to modify its behavior

Profit4Cloud

Use of..

headers

Use for what is stable during the conversation

Content negotiation	Accept / Content-Type
Authentication	Authorization

body

Use for communication (partial) resource representation

Profit4Cloud

Use of..

status codes	status codes
Minimal	Better
200 - OK	200 - OK
	201 - Created
	204 - No Content
400 - Bad Request	400 - Bad Request
	401 - Unauthorized
	403 - Forbidden
404 - Not Found	404 - Not Found
500 - Internal Server Error	500 - Internal Server Error

Profit4Cloud

Server-side conversation state is evil

A stateful service is harder to scale and less resilient (but that is your problem)

For <developers>, an API that requires **special flows of service calls are more complicated**

GETs with side-effects are evil

It is not allowed per HTTP specification;
<developers> (and hence libraries)
expect GET calls to be safe

Add hypermedia for optional use

- A hypermedia-driven API is one that **informs the client what we can do next**
- Allows for a class of changes to the API such that these **changes will not break the client**
- HAL, JSON-LD, Collection+JSON, SIREN, JSON-API, .. too many standards

```
{
 "price":10,
 "product_id":"123",
 "name":"some product",
 "description":"none",
 "_links":{
   "self":{
    "href":"https://ap.i/products/123"
   },
   "rate":{
    "href":"https://ap.i/products/123/rate",
    "method":"POST
   }
   …
 }
}
```

JSON
with links

Profit4Cloud

Important API topics we don't have time for today..

Profit4Cloud

Part 2

Serverless API implementation

aws

Profit4Cloud

Serverless

- No servers (you need to care about)
- Autoscaled with true pay as you go
- Always on

If you don't have to care about infrastructure you can focus on creating ~~business value~~.

cool stuff

Profit4Cloud

Serverless building blocks on AWS

AWS Lambda

- Lets you **just run code** without provisioning or managing servers
  - Node.js, Java, .Net Core, Python
- **Event-driven** with many supported event sources
  - S3, SNS, RDS, API Gateway, ..
- Billed compute time *x* memory usage

logo

Profit4Cloud

Serverless building blocks on AWS

logo

Amazon API Gateway

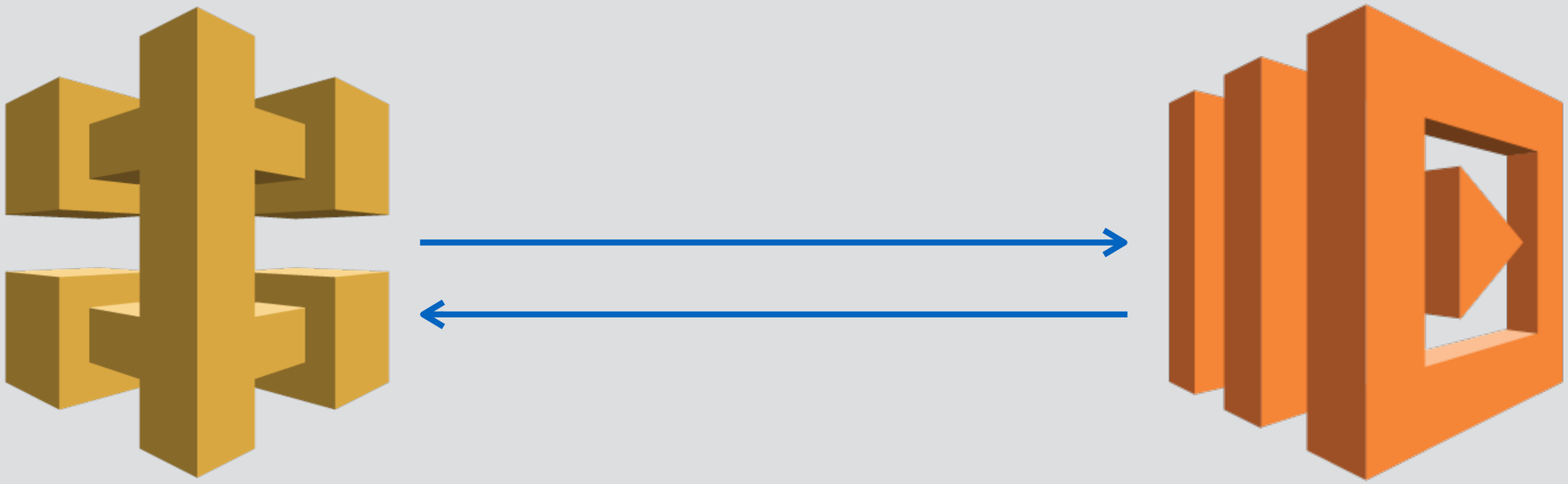
- Managed service that makes it easy for developers to create, publish, monitor, and secure APIs at any scale
  - SDK generation
  - OpenAPI 2.0 (Swagger) support
  - Authentication integrations
- Billed per call + data out

Profit4Cloud

Serverless APIs on AWS

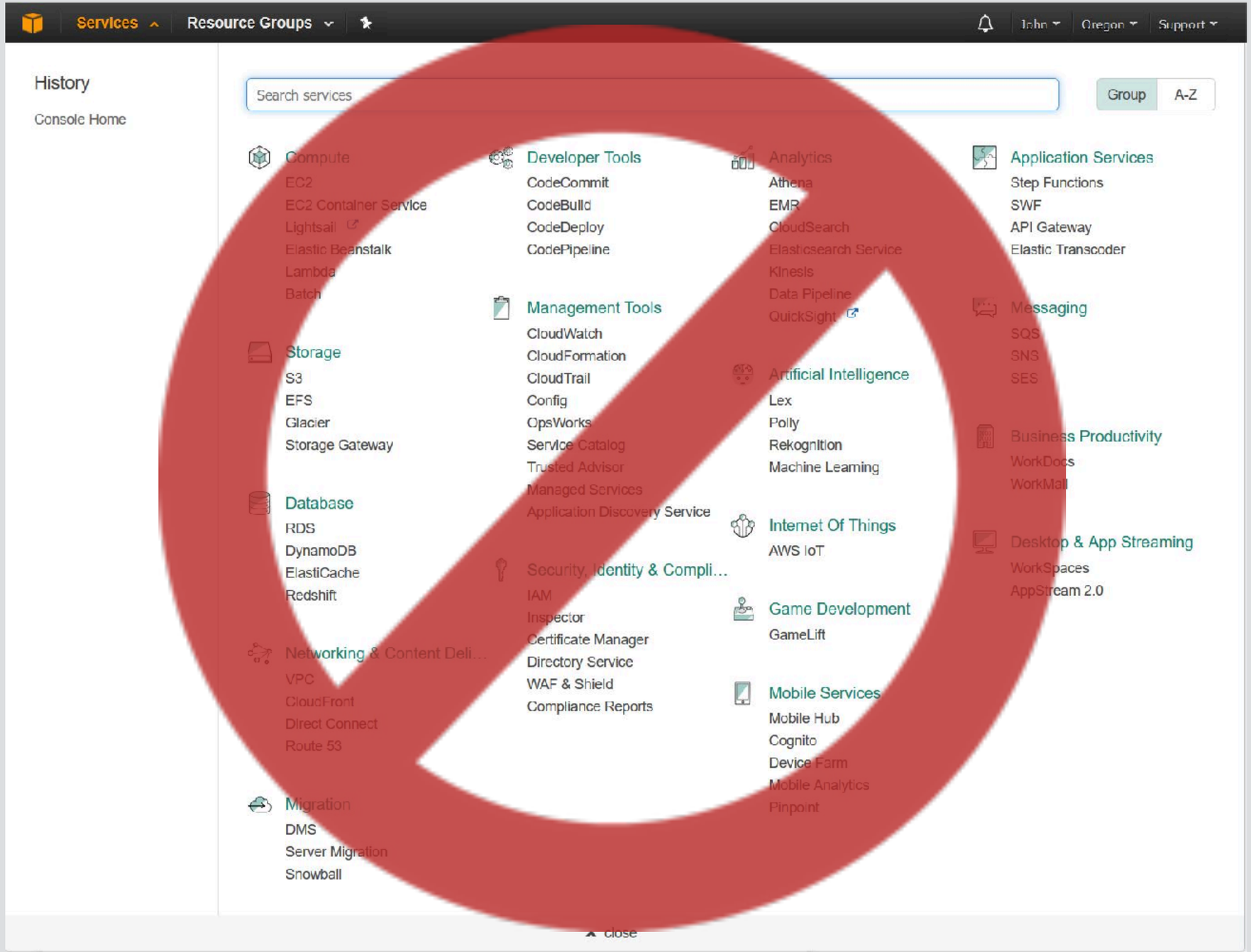
API Gateway

Lambda



The combination of these services is an interesting alternative to traditional backends

Profit4Cloud



Profit4Cloud

Serverless APIs on AWS

AWS CloudFormation

Infrastructure-as-code

- Use *template* files to declaratively define infrastructure resources
- Use CLI tools to deploy and update *stacks* based on the template

logo

Programmable infrastructure allows you to manage your infrastructure like you manage your code.

Profit4Cloud

Serverless APIs on AWS

AWS Serverless Application Model

- CloudFormation extension optimised for serverless applications
- Support anything CloudFormation supports
- Open specification (Apache 2.0)

Convention-over-configuration

by transformation

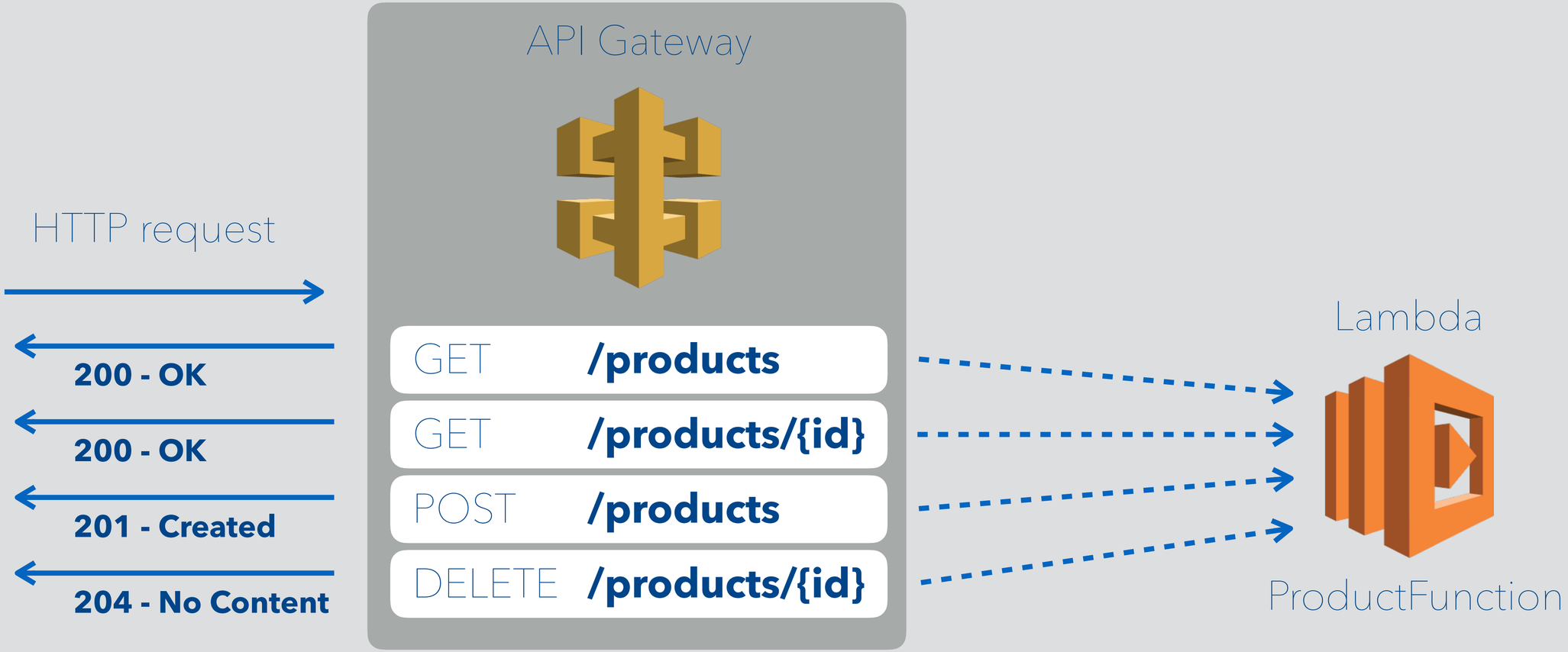
Profit4Cloud

Demo1

Minimalistic API

Profit4Cloud

Demo1

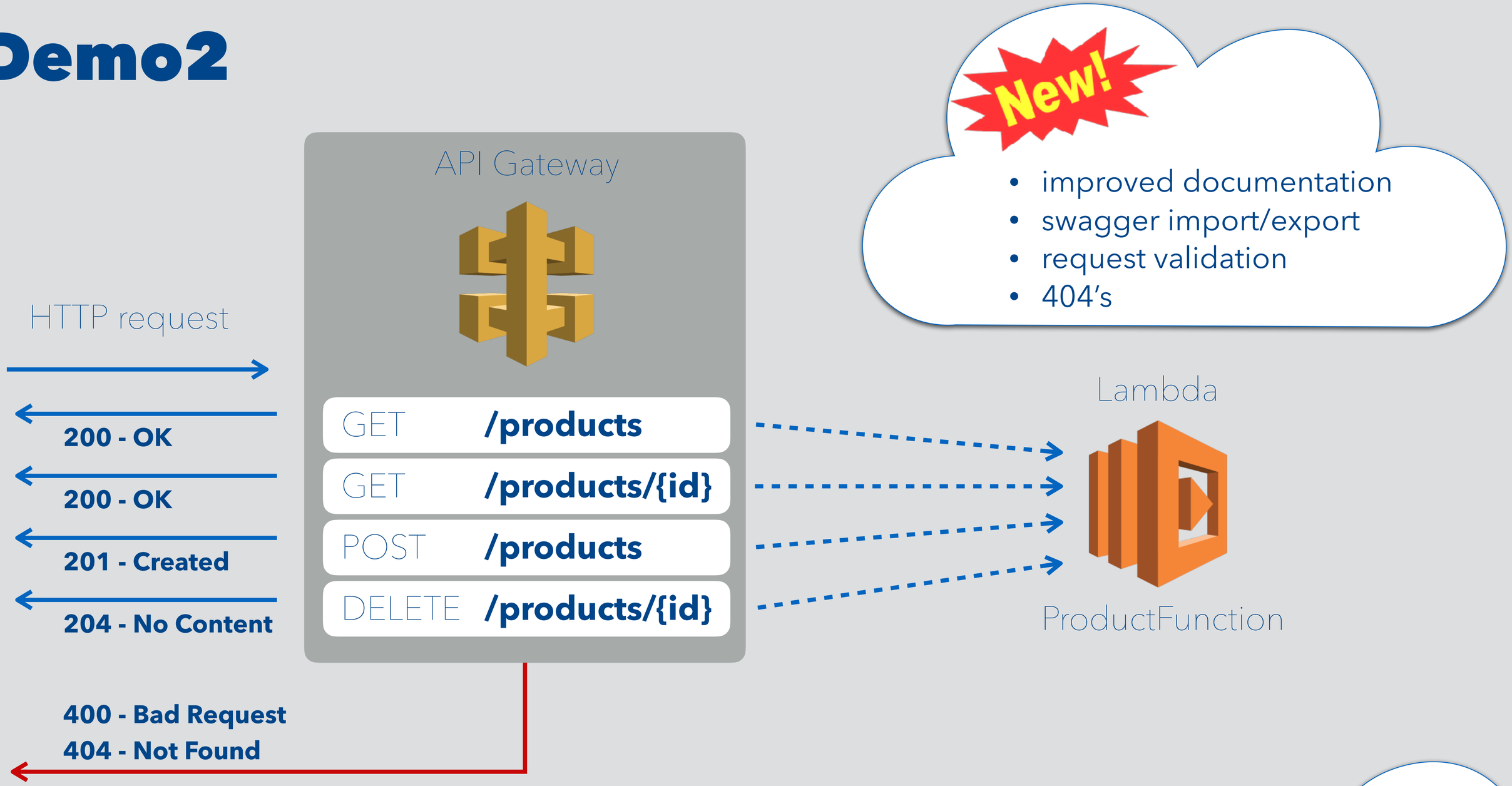


Demo2

Swagger-ish API

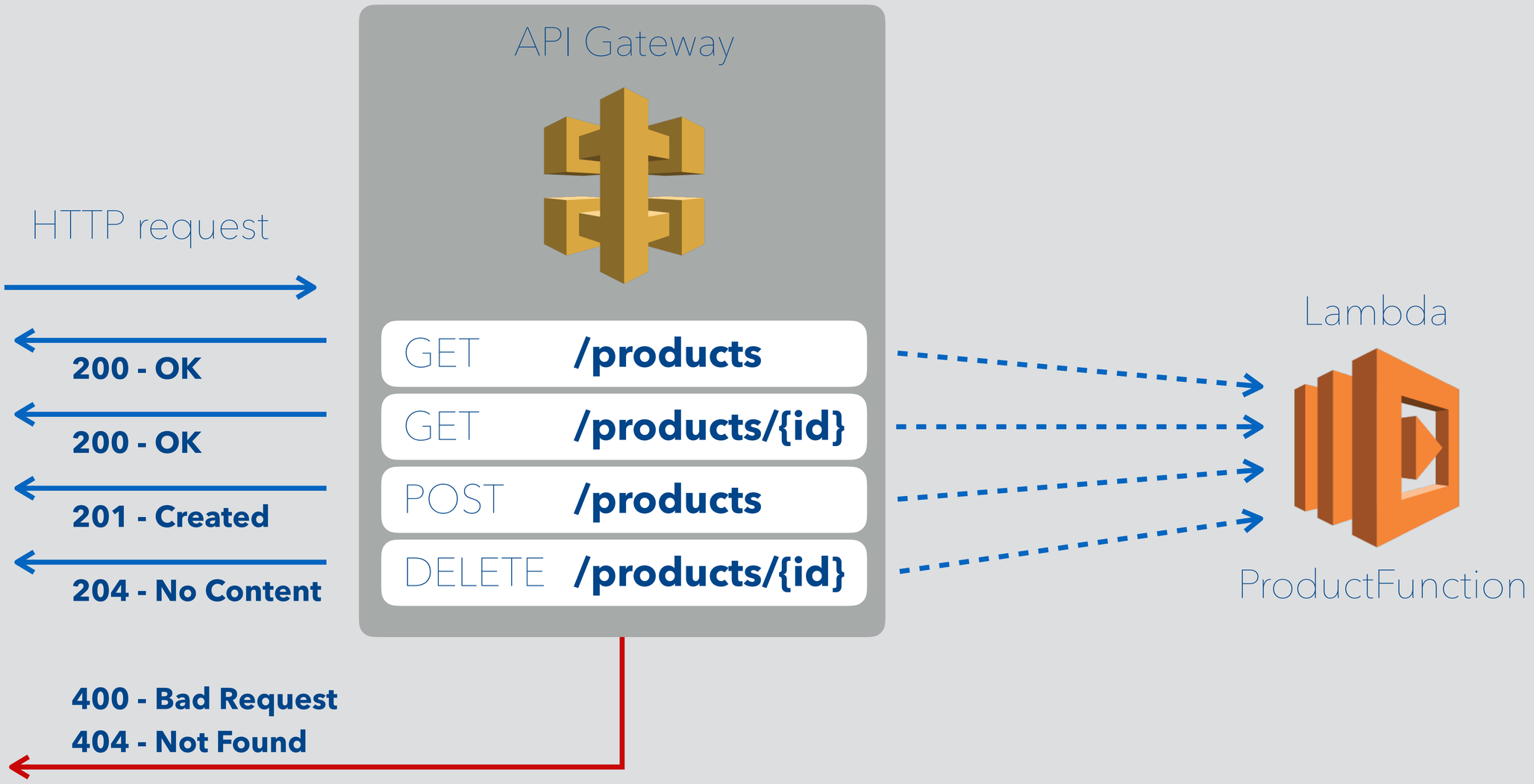
Profit4Cloud

Demo2



Profit4Cloud

Demo2

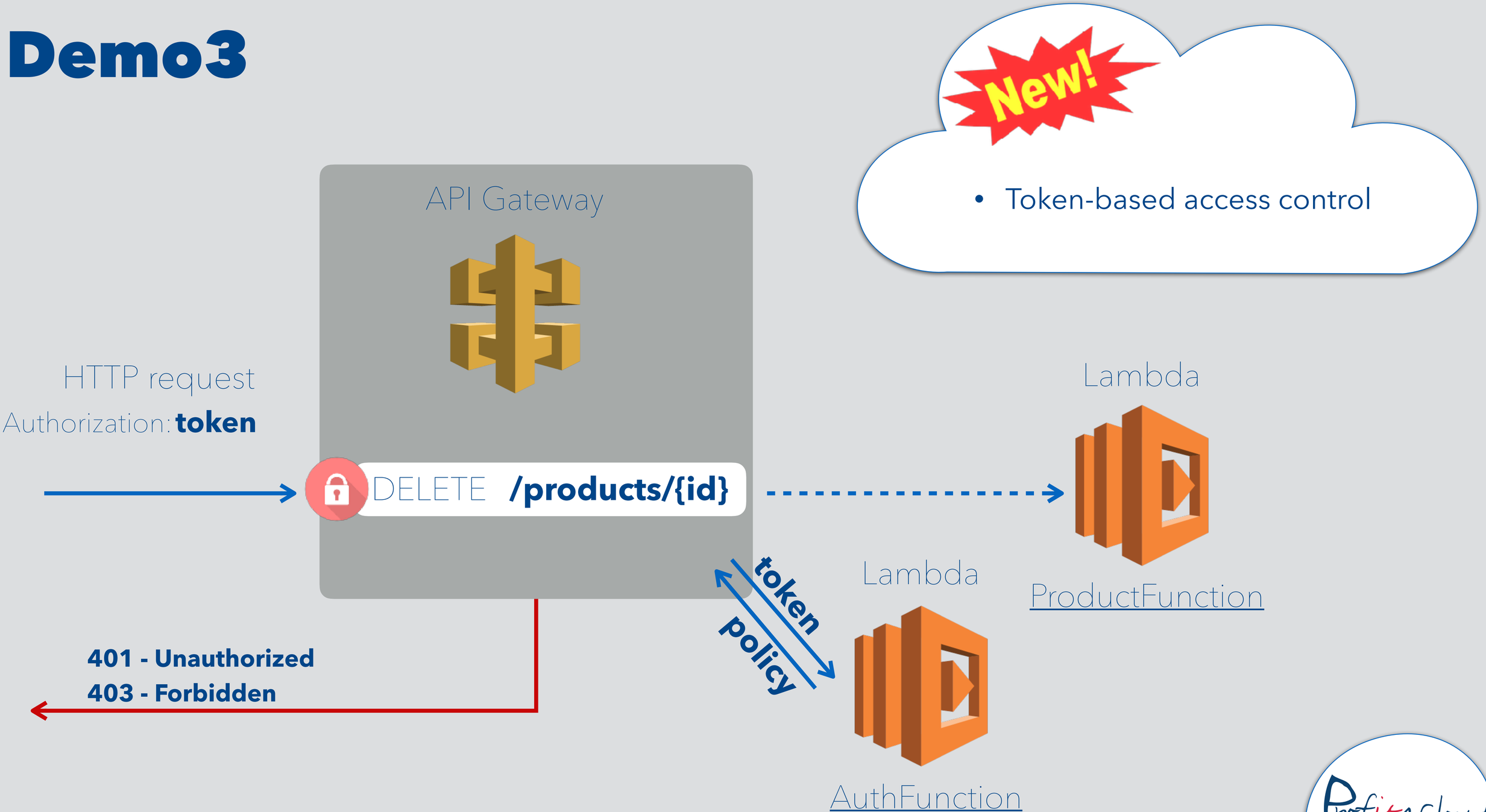


Demo3

API with Authorization

Profit4Cloud

Demo3



Profit4Cloud

To conclude

Recap

- Shared API design tips yeah, we only scratched the surface
- Introduced Amazon's serverless building blocks for creating such APIs

Reproducible (infra-as-)code to production APIs that scale

- Using SAM/CloudFormation to provision API Gateway and Lambda's
- With Swagger-ish configuration and documentation
- Awaiting (micro)frameworks to increase productivity even further..

Profit4Cloud