Building an enterprise-ready Lambda Experience

Héctor Rodes, CTO @ Adhara
Álex González, Head of Back @ BBVA Next

ADHARA

BBVA


True Story


Context

BBVA

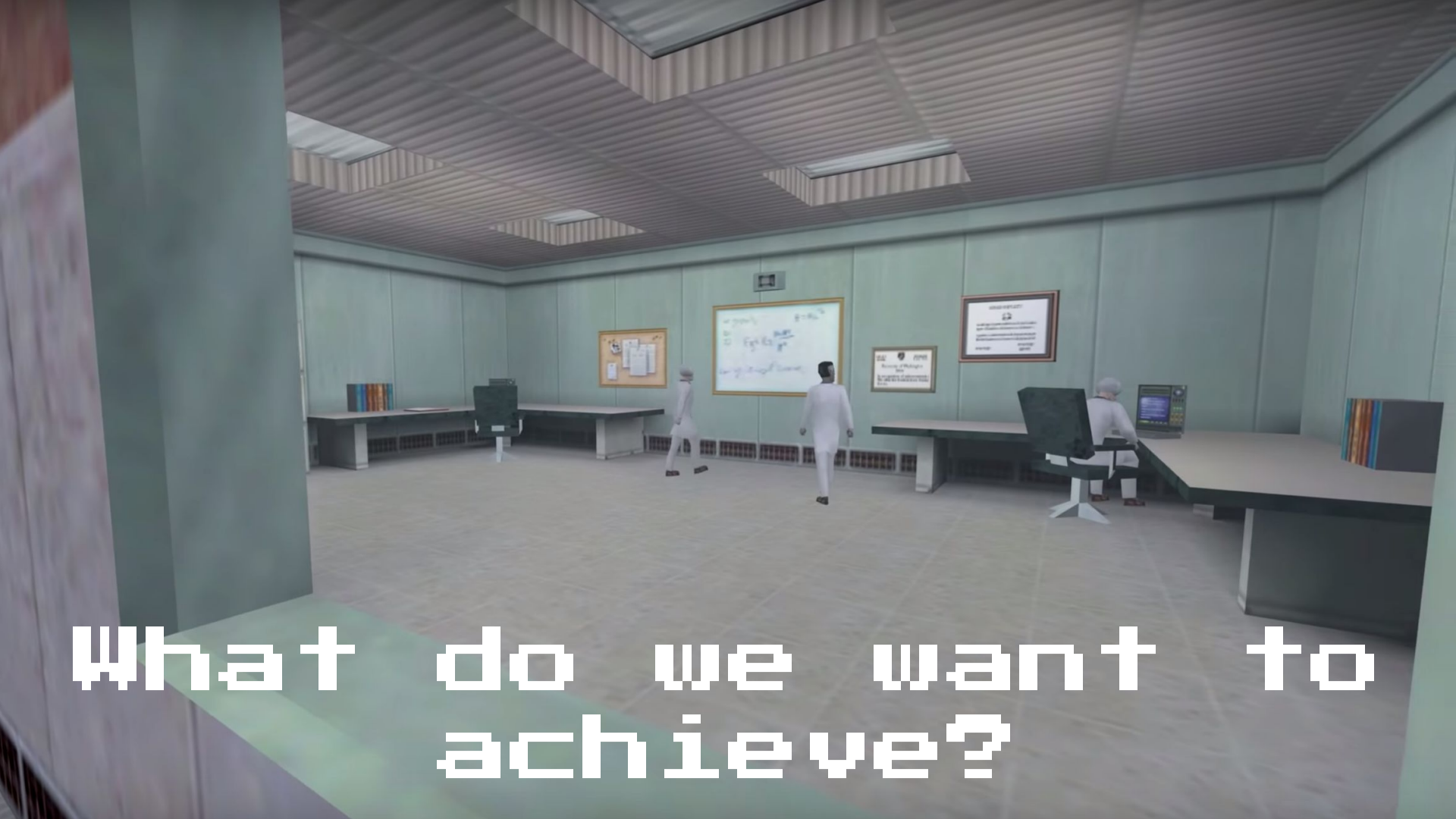
- Financial services
- Presence in +10 countries
- 2 private data centers (America, Europe)
- +10K IT professionals
- Building internal cloud services since 2014

Our team WAS offering

- Simplified compute experiences fully integrated with Bank tools including (but not limited to)...
- **Containers as a service** (based on k8s/openshift)
- **“Google App Engine/Heroku” like** service (based on k8s/openshift)

Our team IS offering

- ❑ Simplified compute experiences fully integrated with Bank tools including (but not limited to)...
- ❑ Containers as a service (based on k8s/openshift)
- ❑ “Google App Engine/Heroku” like service (based on k8s/openshift)
- ❑ **Lambdas**


What do we want to achieve?

Enterprise

READY

Alex Go{,5z} @
@agonzalezro

When I say: "Enterprise Ready", what comes to your mind? I need an image.

6:42 PM - 17 May 2019

4 Likes

16 4

@ferdef, @abelgvidal, @javierprovecho & @robermorales

Industry constraints

- Financial regulated industry:
  Security, confidentiality, auditable, data location…

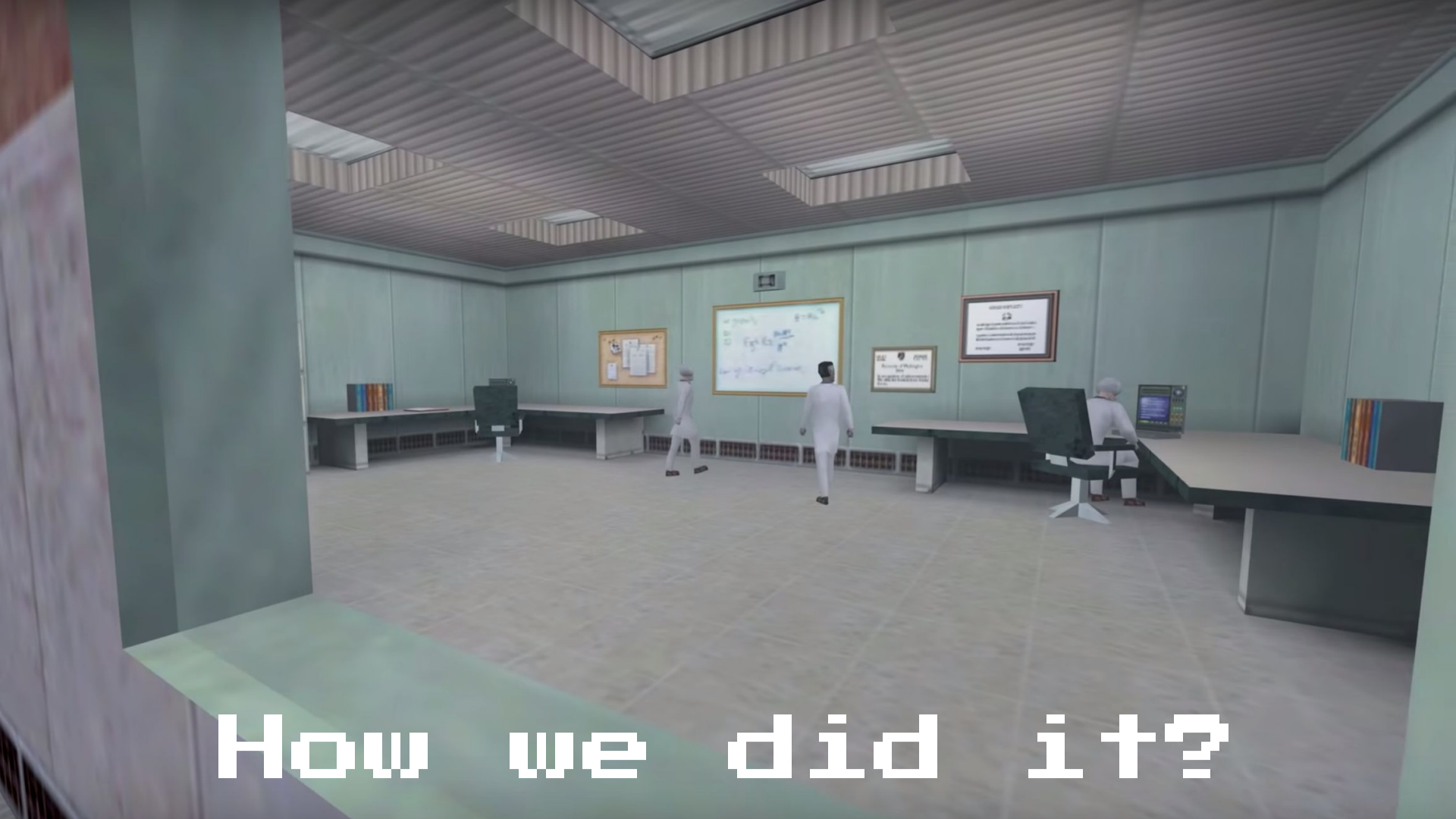
Company constraints

- BBVA internal rules and tools
  (ex: distributed tracing collector, security and compliance checks, ...)

experience

We wanted a lambda experience similar to public cloud offering

JUST THAT SIMPLE!


How we did it?


1.UX

Our API 1/2

Inspired by AWS Lambda on how to implement a function

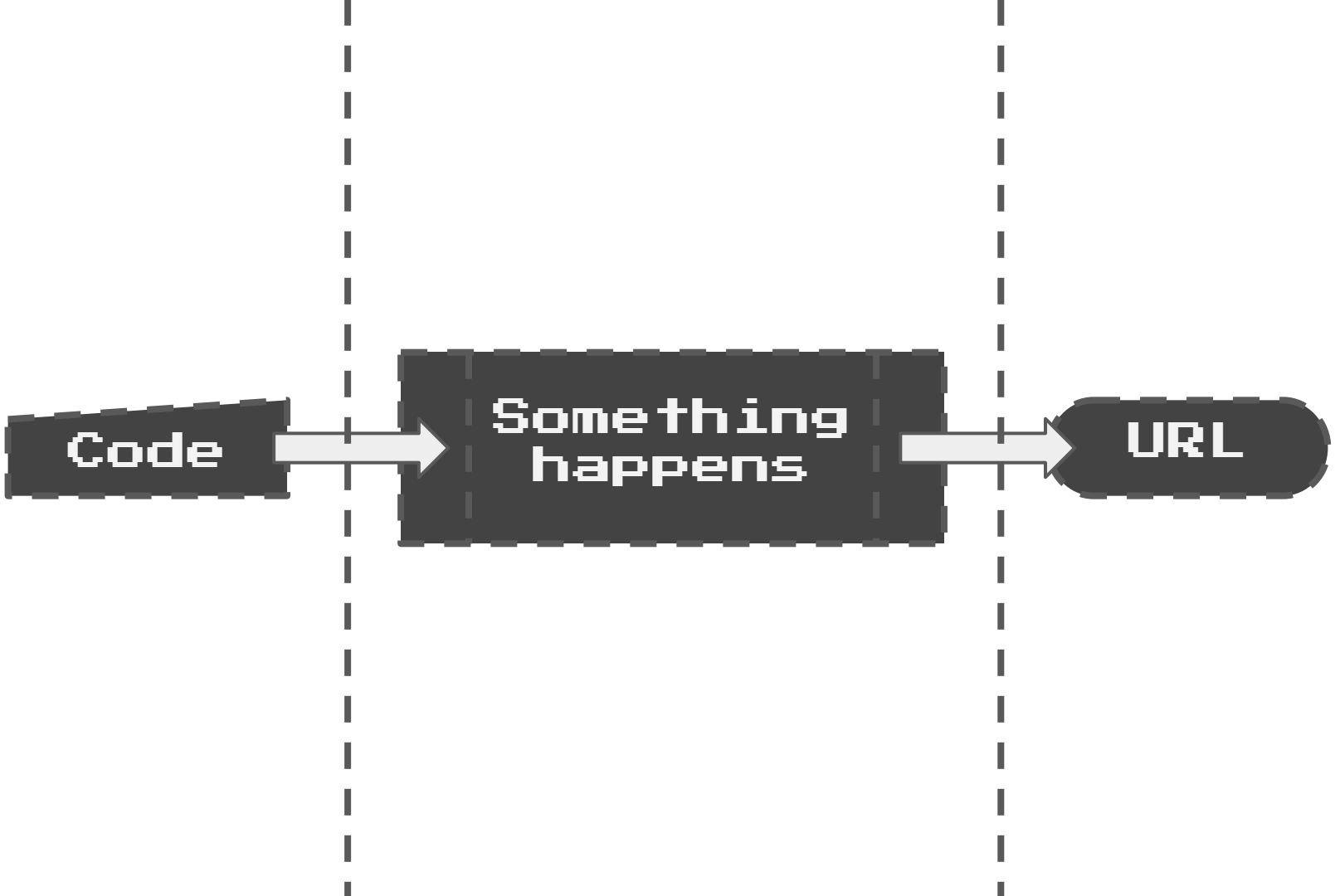
```
import io.e3r.lambda.context.Context;

public class IdentityCardLetter {
    public String getIdentityCardLetter(
                        String identityCardNumber,
                        final Context context) {

        return “your code goes here”;
    }
}
```

Our API 2/2

- Inspired by Google Cloud Functions on how to manage the functions
  - RESTful API
  - Function resource to create, get, update, delete a function
  - Execute function:
    `.../namespace/{id}/function/{id}:[call|async-call]`
- The big difference
  - Code is pushed to git repositories (only allowed option)
  - After code is pushed internal pipelines do their magic (mainly security and compliance)


Code
Something happens
URL

Our API 2/2

Example: Deploy your function

```
curl -X POST https://lambda.domain -d
{
  “code”:”[codeReference]”,
  “entryPoint”:”mypackage.MyClass.theFunction”
}
```

2. Homemade vs Product

State of the art 1/2

- First option was to use an existing solution. Some evaluated: Openwhisk, Openfaas, Knative, Kubeless…

- Problems not solved yet (or at least when we started)
  - Easy extension to be integrated with BBVA tools (security, logs, tracing, monitoring, …)
  - Multi region
  - Multitenancy (BBVA-way)
  - Security compliance
  - GRPC

State of the art 2/2

- We had an internal implementation of a compute service similar to Google App Engine / Heroku
- We had internal certified execution stacks
- Evolution of that service using the certified stacks was evaluated

Our decisions

- ❑ Offer the right UX “wrapping” the real implementation
- ❑ Evolve internal “App Engine” compute service to execute functions
- ❑ Use certified stacks as functions execution environment
- ❑ Keep evaluating products (future replacement of custom development without breaking the UX)

We want to build

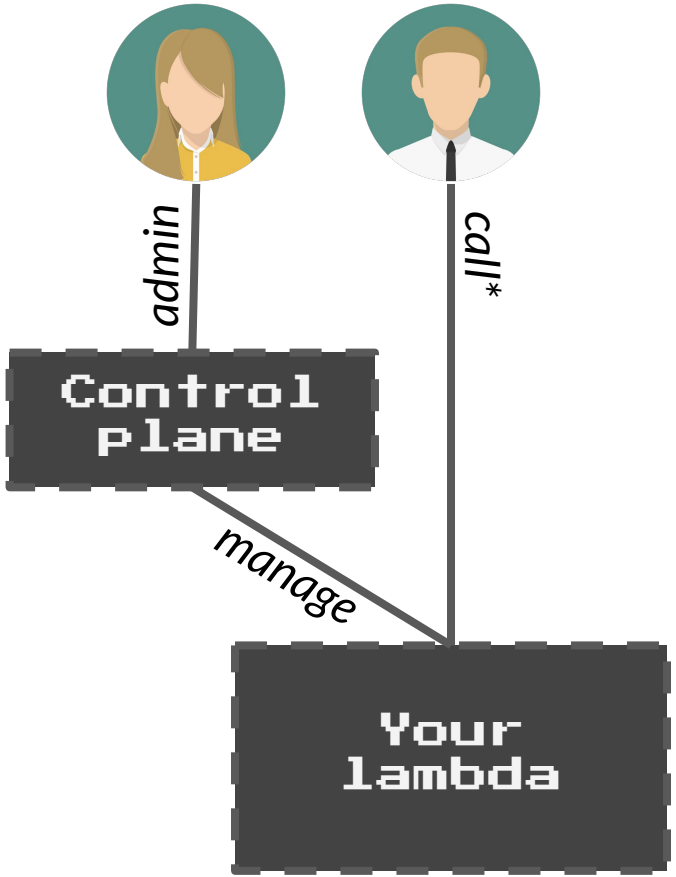
The context

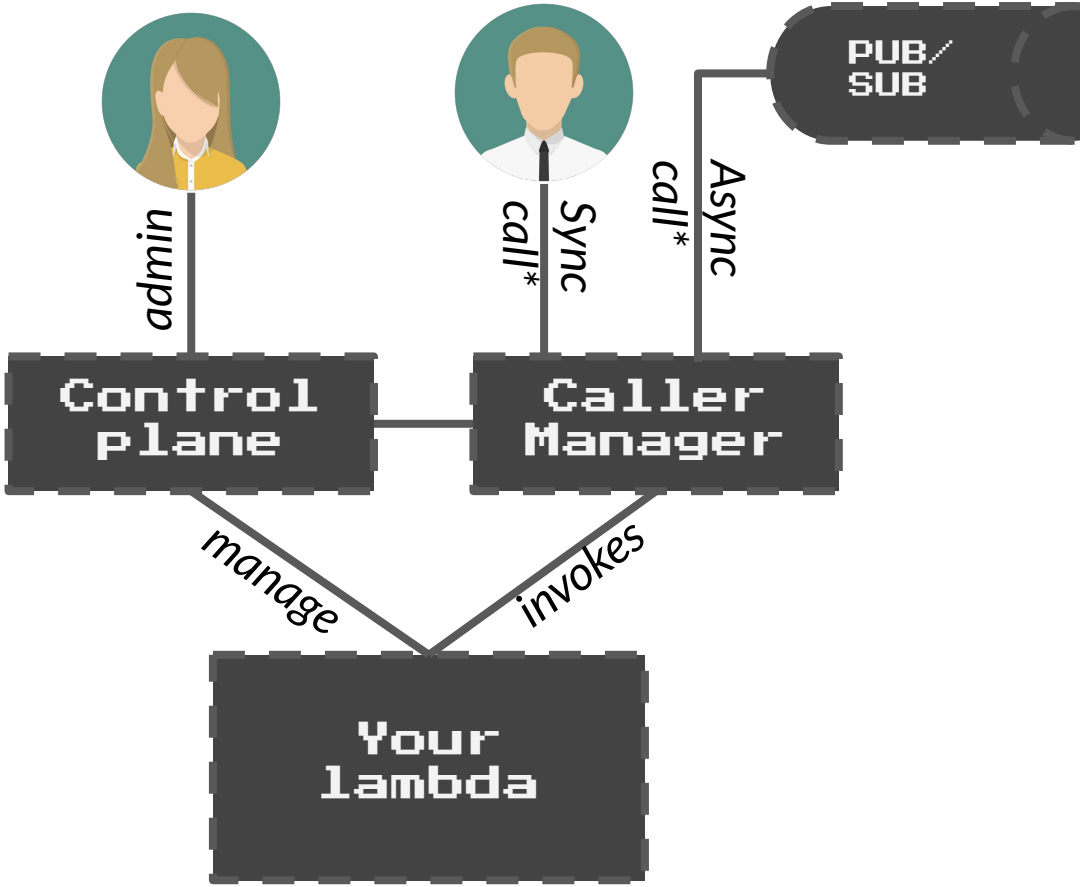
May be it's better

3. Control plane

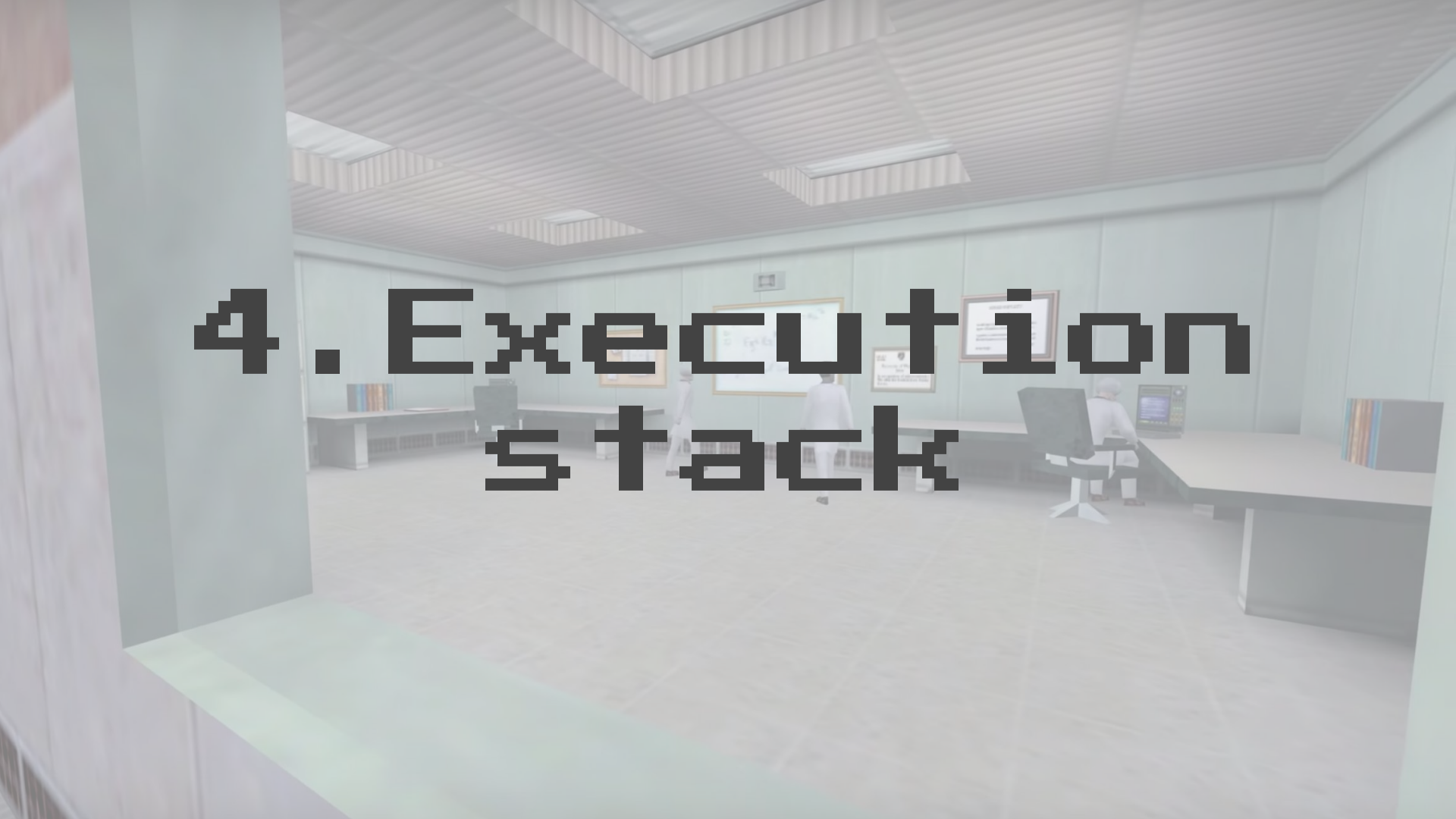
Control Plane

- Main control plane to manage lambda lifecycle
- Caller Manager providing access to deployed lambdas

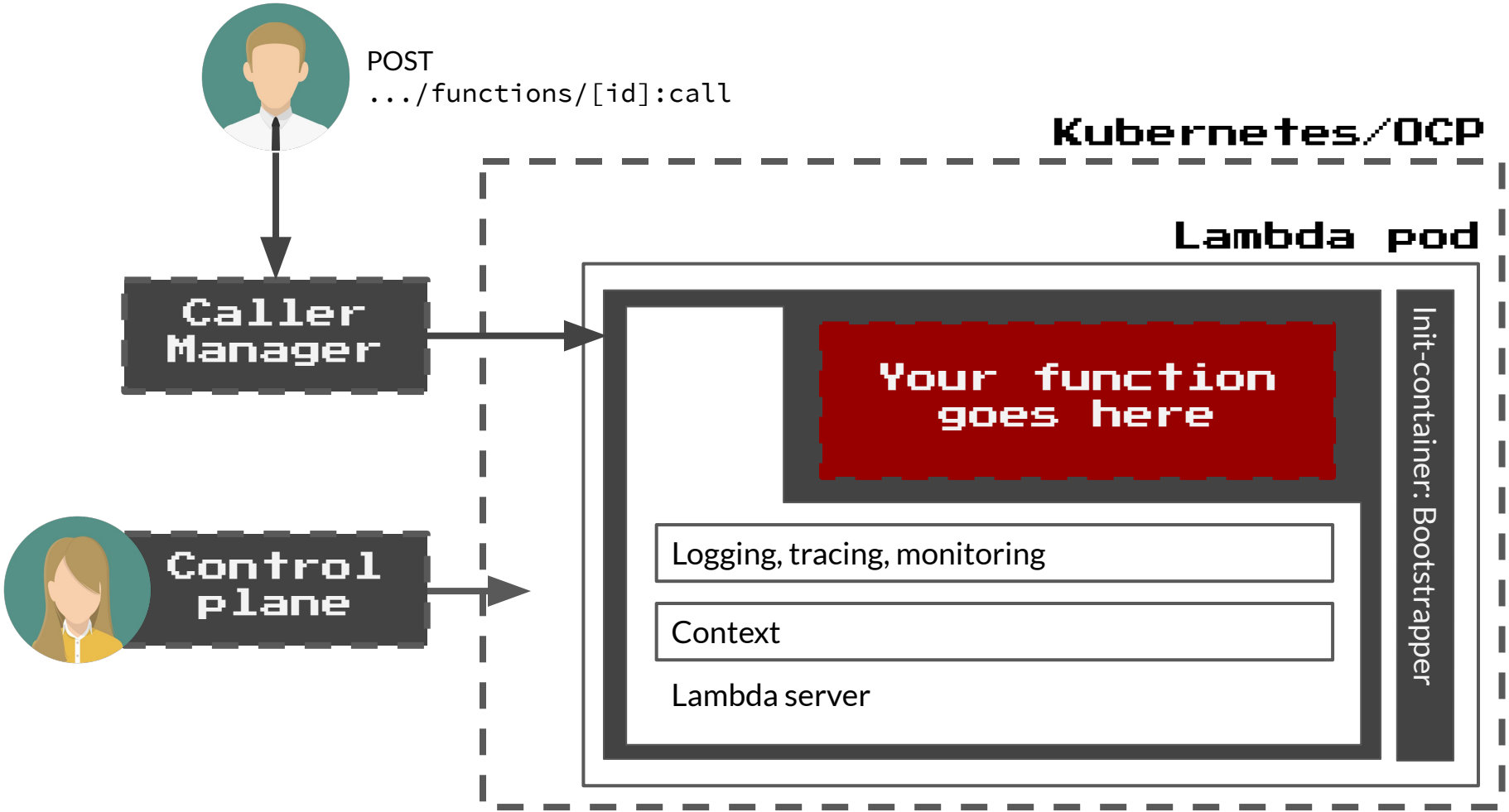




** Gateways, load balancers, firewalls… not represented here for the sake of simplicity*



4. Execution stack


POST
.../functions/[id]:call
Kubernetes/OCP
Lambda pod
Caller Manager
Control plane
Your function goes here
Logging, tracing, monitoring
Context
Lambda server
Init-container: Bootstrapper

Execution stack

- Add a new language implies to build a new lambda server implementing the internal json rpc protocol for that language and the setup process

- Add the internal pipelines needed to ensure software quality and vulnerabilities checking


DEMO
TIME

Windows

A fatal exception 0E has occurred at 0028:C562F1B7 in VXD ctpci9x(05) + 00001853. The current application will be terminated.

* Press any key to terminate the current application.
* Press CTRL+ALT+DEL again to restart your computer. You will lose any unsaved information in all applications.

Press any key to continue _

Thanks!

github.com/landistas

@hector_rodes
@agonzalezro