INTRODUCCIÓN A SERVERLESS

César Silgo Ortiz
Luis González Abundes

(BREVE) HISTORIA

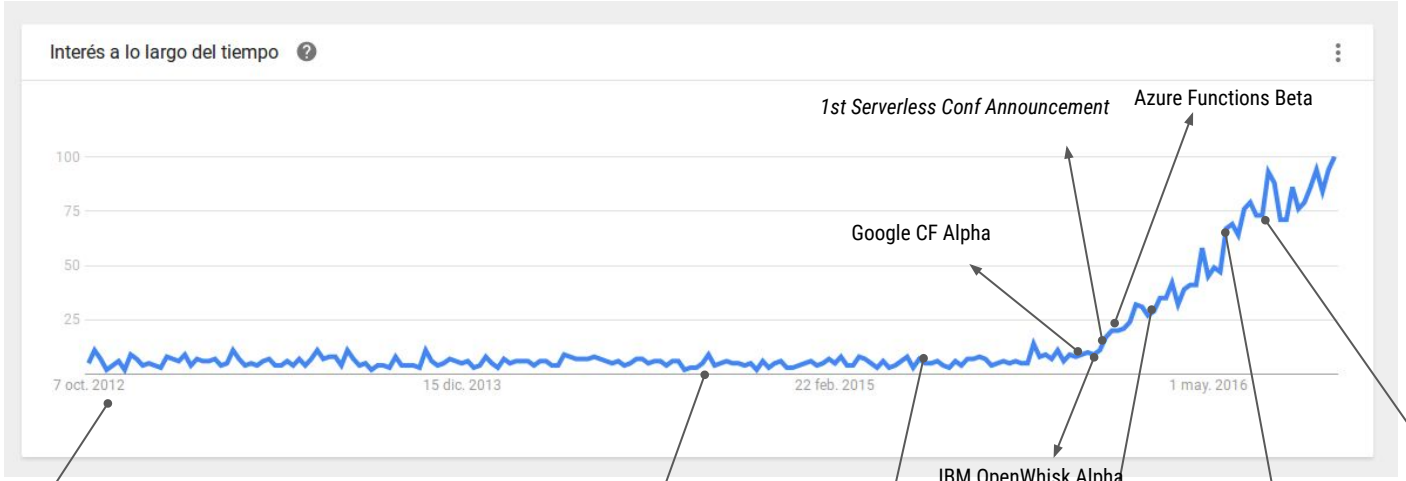
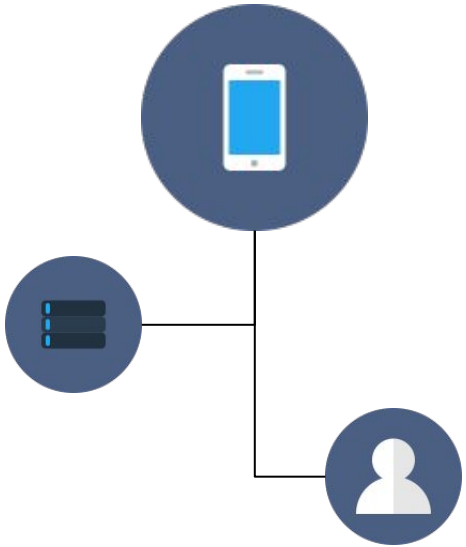


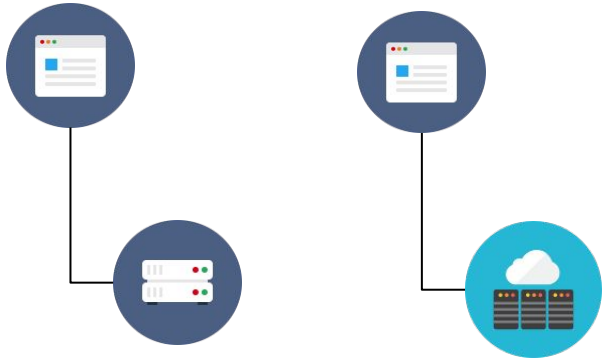
Image by Google Trends

¿QUÉ ES SERVERLESS?



General

(PaaS, SaaS, BaaS…)



Cloud Functions

(Código Backend en la nube, FaaS)

Images by David Cross

¿QUÉ ES UNA CLOUD FUNCTION?

```python
from __future__ import print_function
import boto3
import uuid
from PIL import Image

s3_client = boto3.client('s3')


def resize_image(image_path, resized_path):
    with Image.open(image_path) as image:
        image.thumbnail(tuple(x / 2 for x in image.size))
        image.save(resized_path)


def handler(event, context):
    for record in event['Records']:
        bucket = record['s3']['bucket']['name']
        key = record['s3']['object']['key']
        download_path = '/tmp/{}{}'.format(uuid.uuid4(), key)
        upload_path = '/tmp/resized-{}'.format(key)

        s3_client.download_file(bucket, key, download_path)
        resize_image(download_path, upload_path)
        s3_client.upload_file(upload_path, '{}resized'.format(bucket), key)
```

Una función es una unidad de trabajo que realiza una tarea atómica muy concreta y se ejecuta en “la nube”

EVENT-DRIVEN

Eventos predefinidos

Se recibe un fichero en S3

Llega una notificación de SNS

Se inserta un item en DynamoDB

Caso de uso: “Pegamento”

Evento planificado

Cada día, haz un backup de un disco

Cada hora, termina una instancia del entorno de producción

Cada segundo, lee 100 registros de un Stream de Kinesis

Caso de uso: “Operaciones”

Custom REST

Backend APP

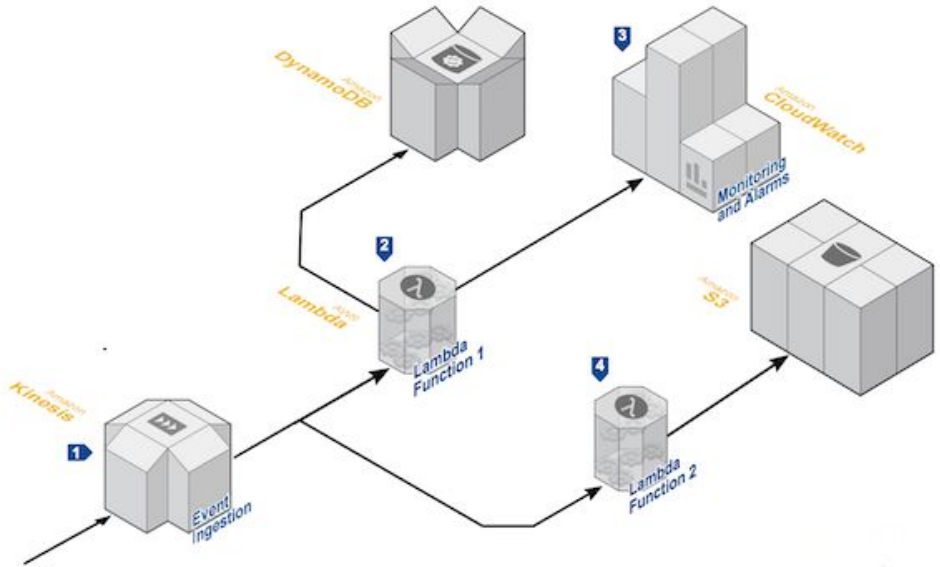
Mobile

IoT

Caso de uso: “REST API”

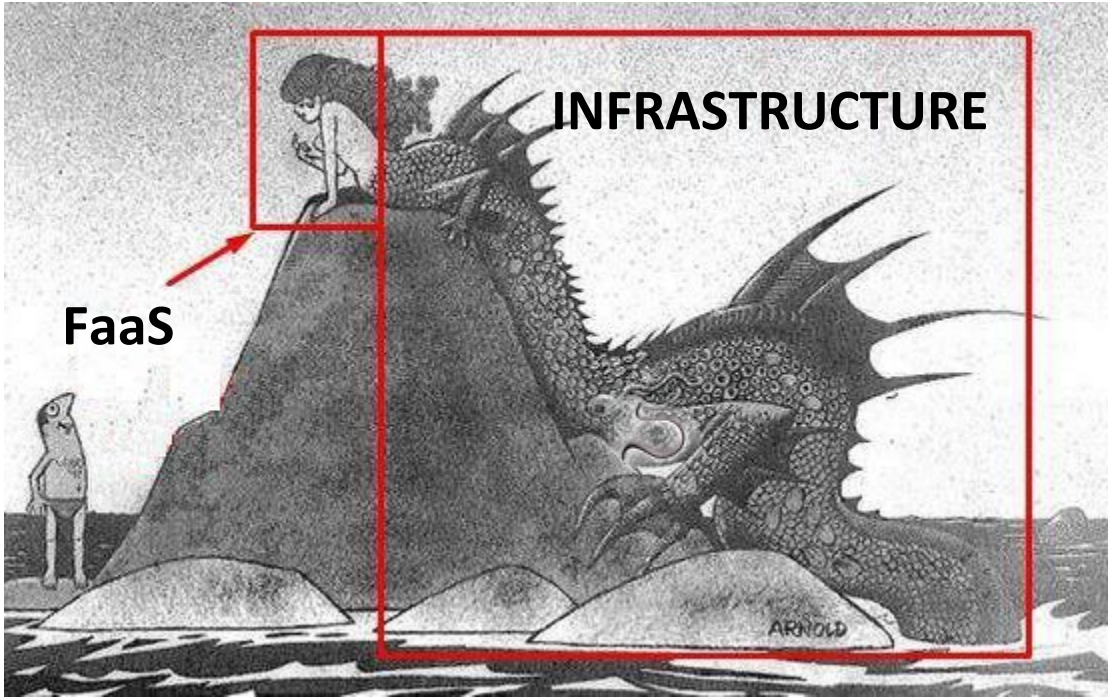
TECNOLOGÍAS SERVERLESS

EJEMPLO ARQUITECTURA SERVERLESS



https://github.com/awslabs/lambda-refarch-streamprocessing

DENTRO DE UNA CLOUD FUNCTION

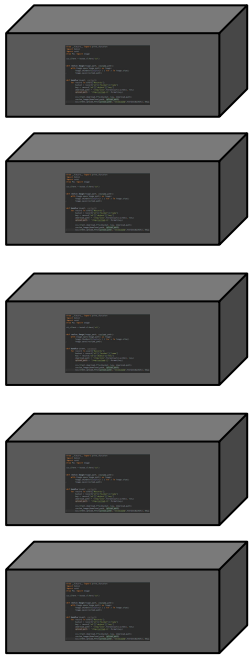


LINUX CONTAINERS

MARATHON

kubernetes

CICLO DE VIDA DE UNA FUNCIÓN



1. Un evento dispara la función
2. Se levanta un contenedor con el código de la función
3. La función se ejecuta
4. El contenedor, eventualmente, muere

RESTRICCIONES: STATELESS

- La vida de un contenedor es **efímera**
- Invocaciones sucesivas no tienen porqué ser ejecutadas por el mismo contenedor → “**Think Parallel**”
- Se pueden reutilizar recursos (ej: pool de conexiones), pero la lógica debe ser **atómica**
- **Externalizar** cualquier tipo de almacenamiento

RESTRICCIONES: RECURSOS LIMITADOS

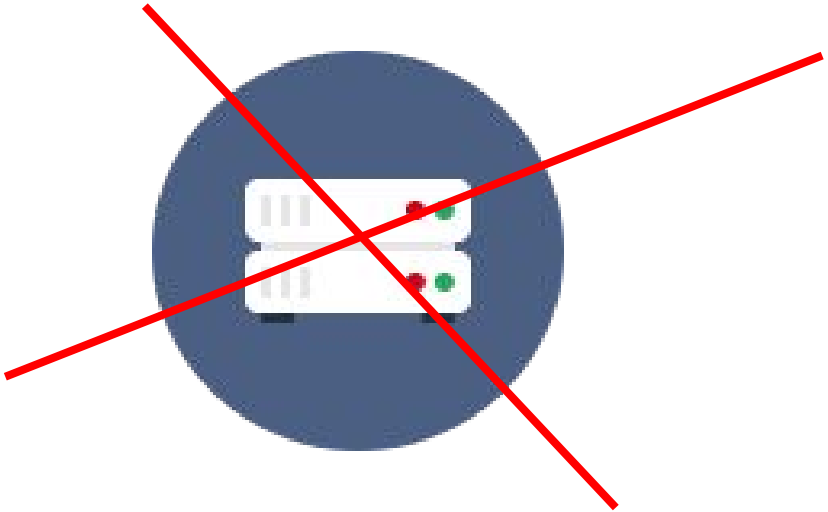
- Memoria: ~1 GB
- Disco: ~512 MB
- Tiempo: ~300 segundos

RESTRICCIONES: ARRANQUE

- El tiempo de arranque habitual está en decenas de milisegundos
- Podría llegar a varios segundos dependiendo de lo pesado que sea el código
- Especialmente determinante para backends REST

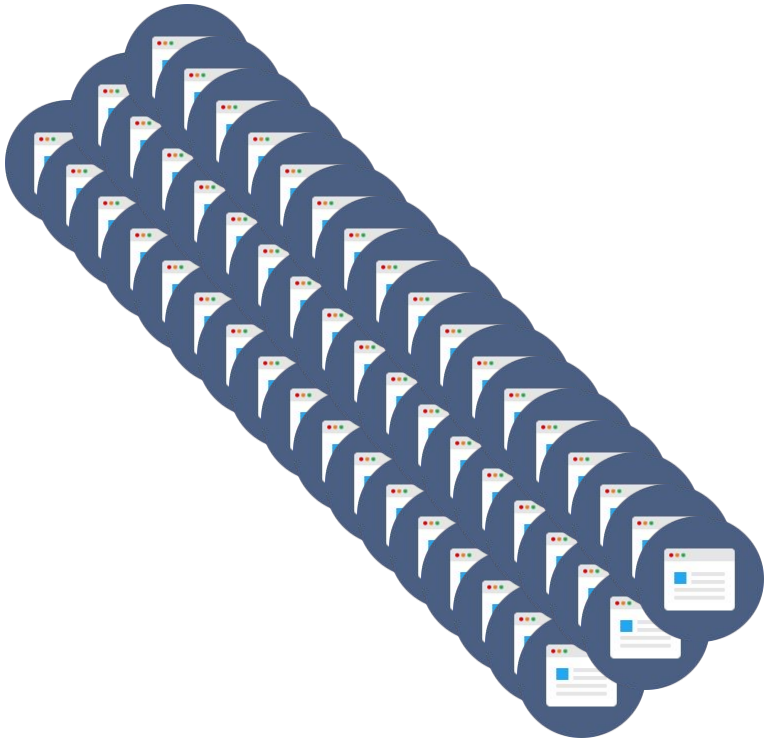
PROS & CONS

VENTAJAS: MENOS OPERACIONES



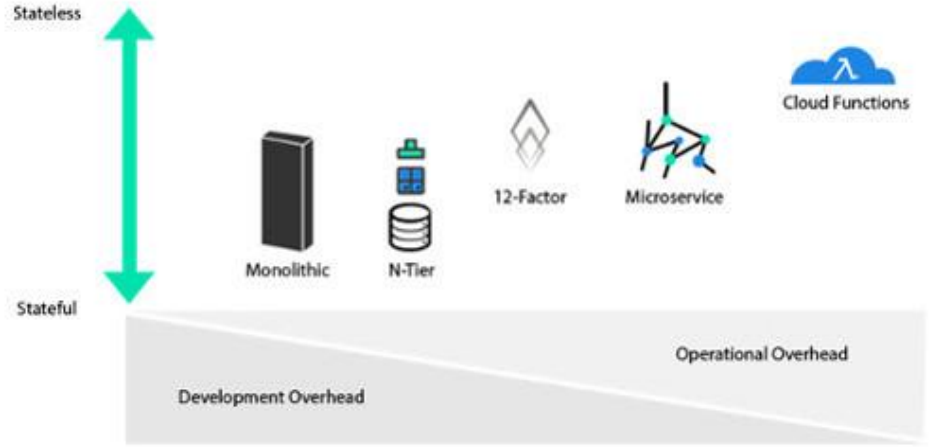
- No Servers
- No SysAdmin

NoOPS?



- Deployment
- Versioning
- Packing
- Testing & Debugging
- Auditing

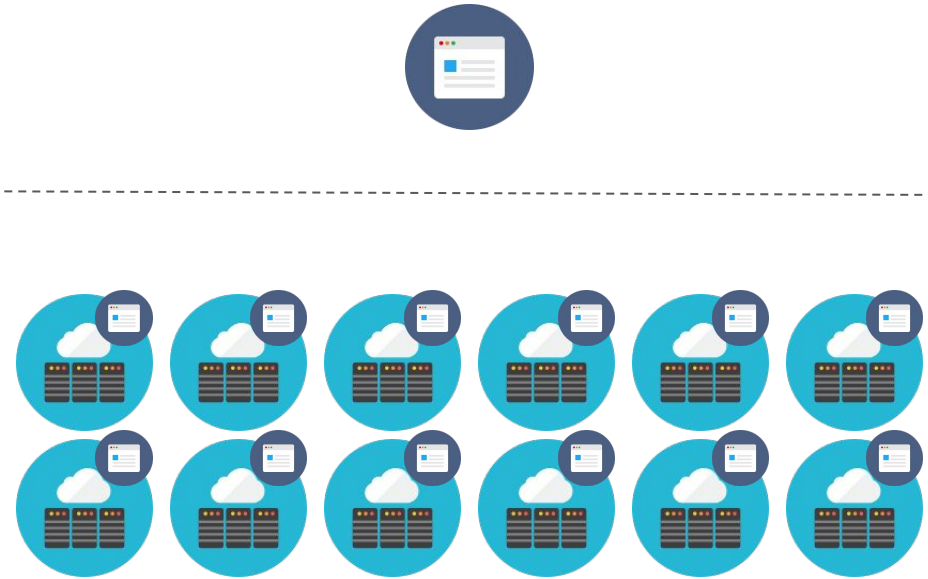
NoOPS?



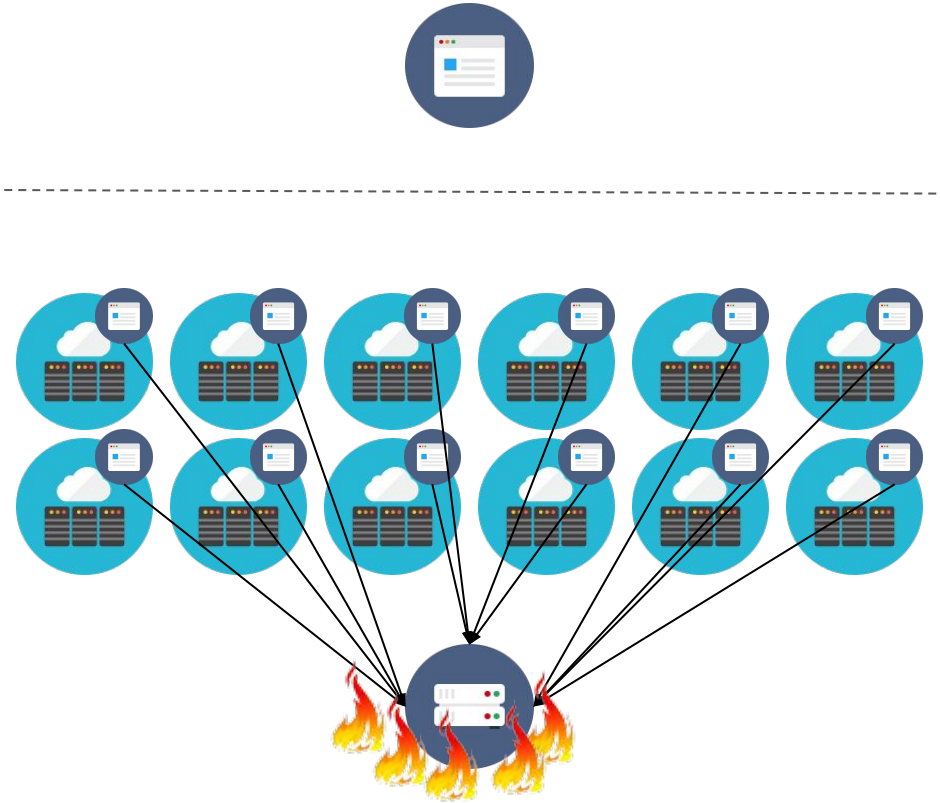
"Lambda Functions versus Infrastructure - Are we Trading Apples for Oranges?"

Dean Sheehan, Apr 2016

VENTAJAS: ESCALADO AUTOMÁTICO



ESCALADO AUTOMÁTICO: CUIDADO



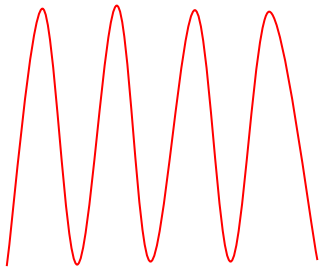
VENTAJAS: COSTES

- Menor esfuerzo manual
- Infraestructura compartida → Economía de escala
- Pago por ejecución (~0.20 USD por millón de peticiones)
- Cuidado con servicios adicionales (API Gateway)

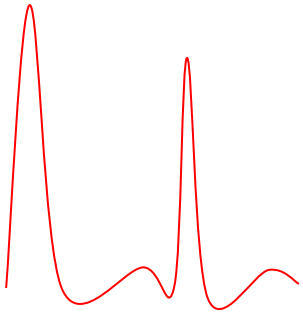
¿SIEMPRE ES MÁS BARATO?

ESCENARIOS

Picos predecibles



Picos no predecibles



Baja carga constante

Alta carga constante

VENTAJAS: TIME-TO-MARKET

- Menos carga manual -> Más agilidad
- Se reduce el coste de experimentación
- La inversión inicial es mucho más baja

CONSIDERACIONES

- Vendor lock-in
- Multi-tenancy
- No fine-tuning
- Stateful Apps
- Límites (ej: 1000 peticiones concurrentes)
- Diferentes modelos de seguridad y criterios específicos

COMPARATIVA (AGOSTO 2016)

GENERAL

	AWS Lambda	Azure Functions	Iron.io	Google CF
State	**Full production**	Preview	**Full production**	Alpha
Community acceptance	**Very high**	Low	**High**	Medium
Scalability	**Unlimited**	Low	Good	(Not tested)
Response time	**Very low**	Very high	**Low**	Low
Supported languages	Node.js, Java and Python	Batch, Bash, C#, Javascript, Powershell, F#, PHP and Python	**All* (PHP, Python, Ruby, Node.js, Java, .Net, Go, Scala, binary executables)**	Javascript
Max execution time	300 sec	**Unlimited**	**1 hour**	**Unlimited**
Local test	Not possible	Not possible	**Yes (Dockerized)**	Not possible

STRESS TEST

	AWS	Azure	Iron.io
Average request per second	500	47	250
Maximum request per second	500 (theoretical API Gateway limit)	200	400 (Free version)
Average response time	0.03 s	112 s	8 s

CONCLUSIONES

	AWS-lambda	Azure-functions	Iron.io	Google - Functions
Strengths	**Performance** Scalability Reliability	**Access by url** Grouping Automatic export/import	**Dockerized** open-source SDK's to most platforms	**Response Time (no stress)** Versioning with git No limits
Weaknesses	**Max execution time** Only accessible by events	**Performance** Monitoring Not versioning	**Few services** Startup time	**Only one language (JS)** Scalability

ESTADO DEL ARTE...

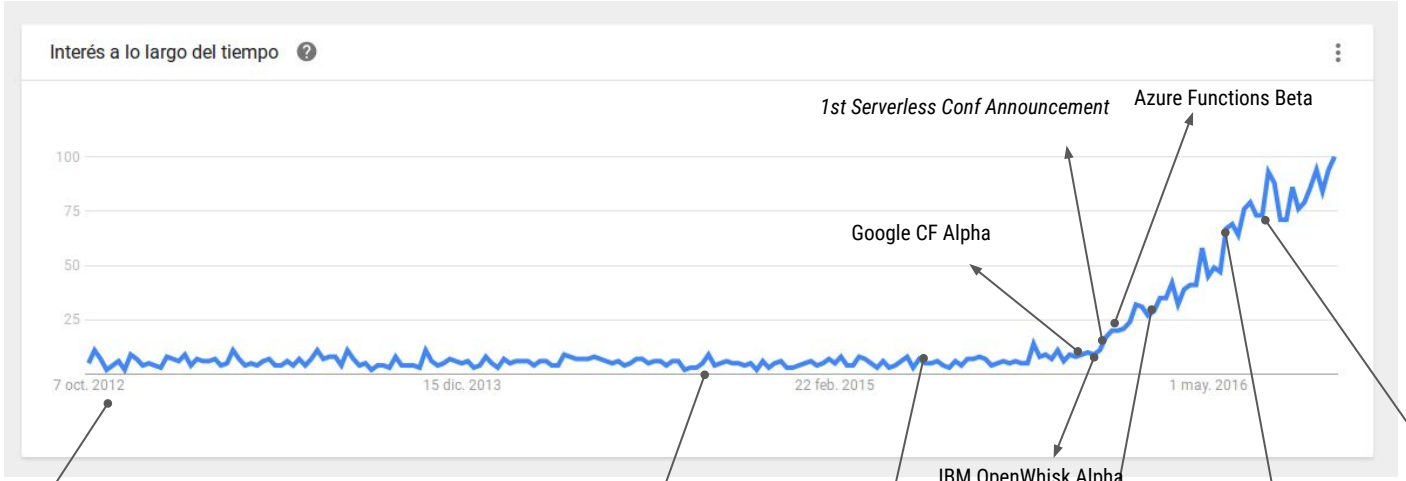
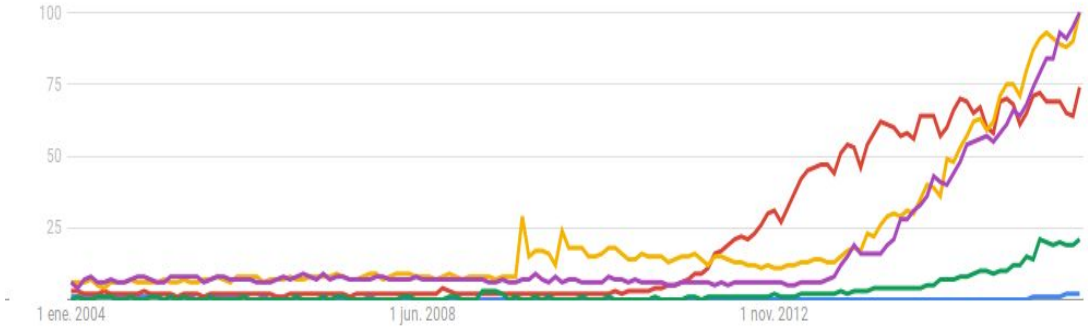


Image by Google Trends

… EN PERSPECTIVA



Docker
IoT
BigData
Deep Learning
Serverless

Image by Google Trends

THE PIONEER TAX

Alphas / Betas
Tooling & Resources
Best Practices
Know-how
Community
Bugs


Q&A

Serverless for REST Services (>1000 ⭐)

⭐ 11k **Serverless Framework** - Node.js, Python & Java functions on AWS Lambda, Azure Functions, Google CloudFunctions. https://github.com/serverless/serverless

⭐ 1.5k **Chalice** - Python microframework on AWS Lambda and API Gateway. https://github.com/awslabs/chalice

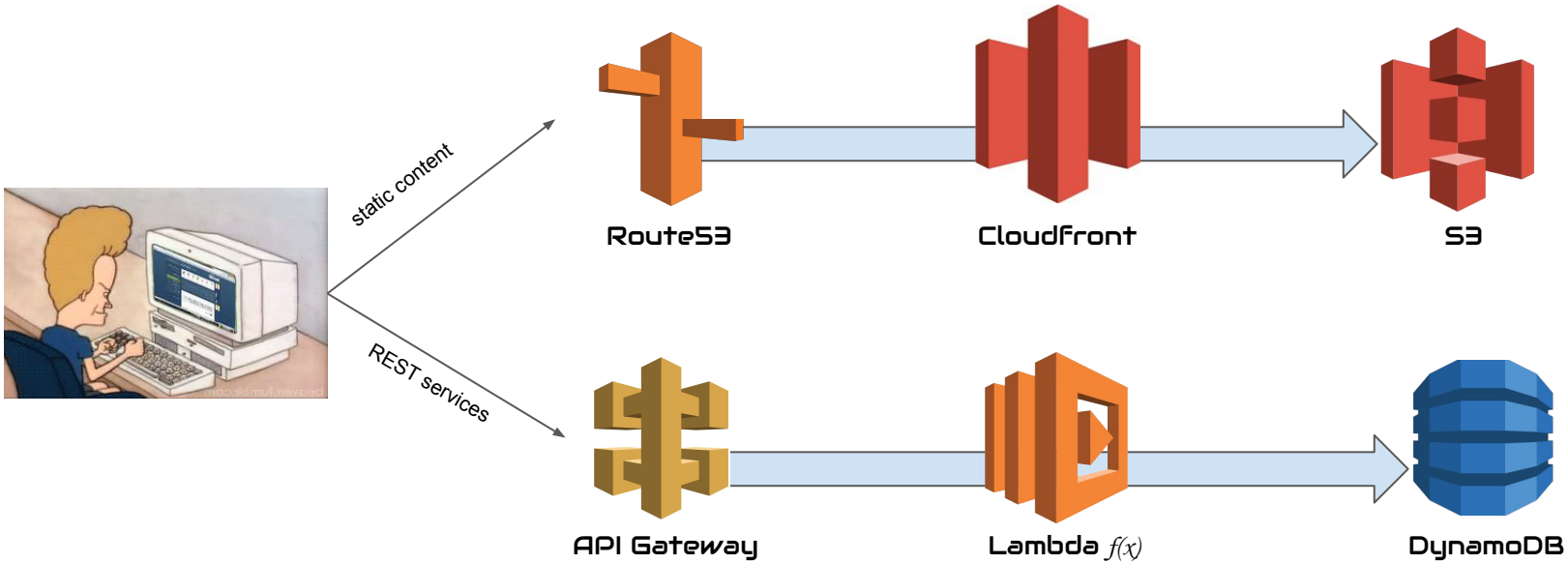
⭐ 1.3k **Zappa** - python WSGI on AWS Lambda and API Gateway. https://github.com/Miserlou/Zappa

⭐ 1.3k **Claudia** - node.js on AWS Lambda and API Gateway. https://github.com/claudiajs/claudia

⭐ 3.7k **APEX** - build, deploy, and manage AWS lambdas in Node.js, Golang, Python, Java. https://github.com/apex/apex

⭐ 1.5k **λ Gordon** - create, wire and deploy AWS Lambdas using CloudFormation. https://github.com/jorgebastida/gordon

Serverless REST Architecture on AWS

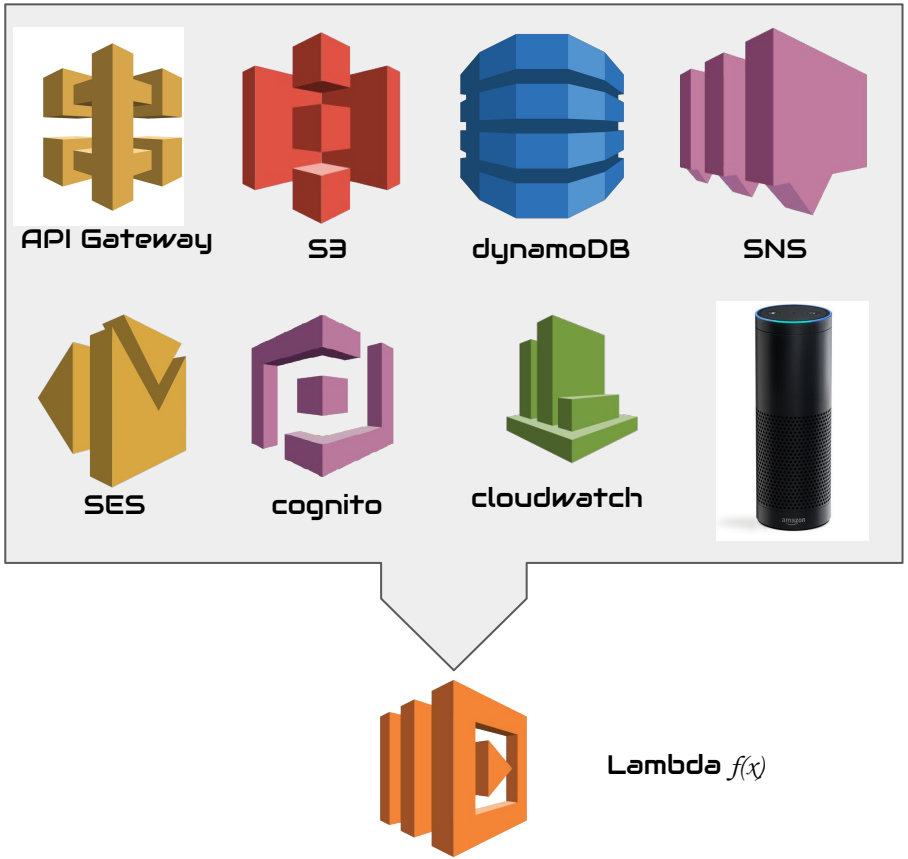


Chalice DEMO!

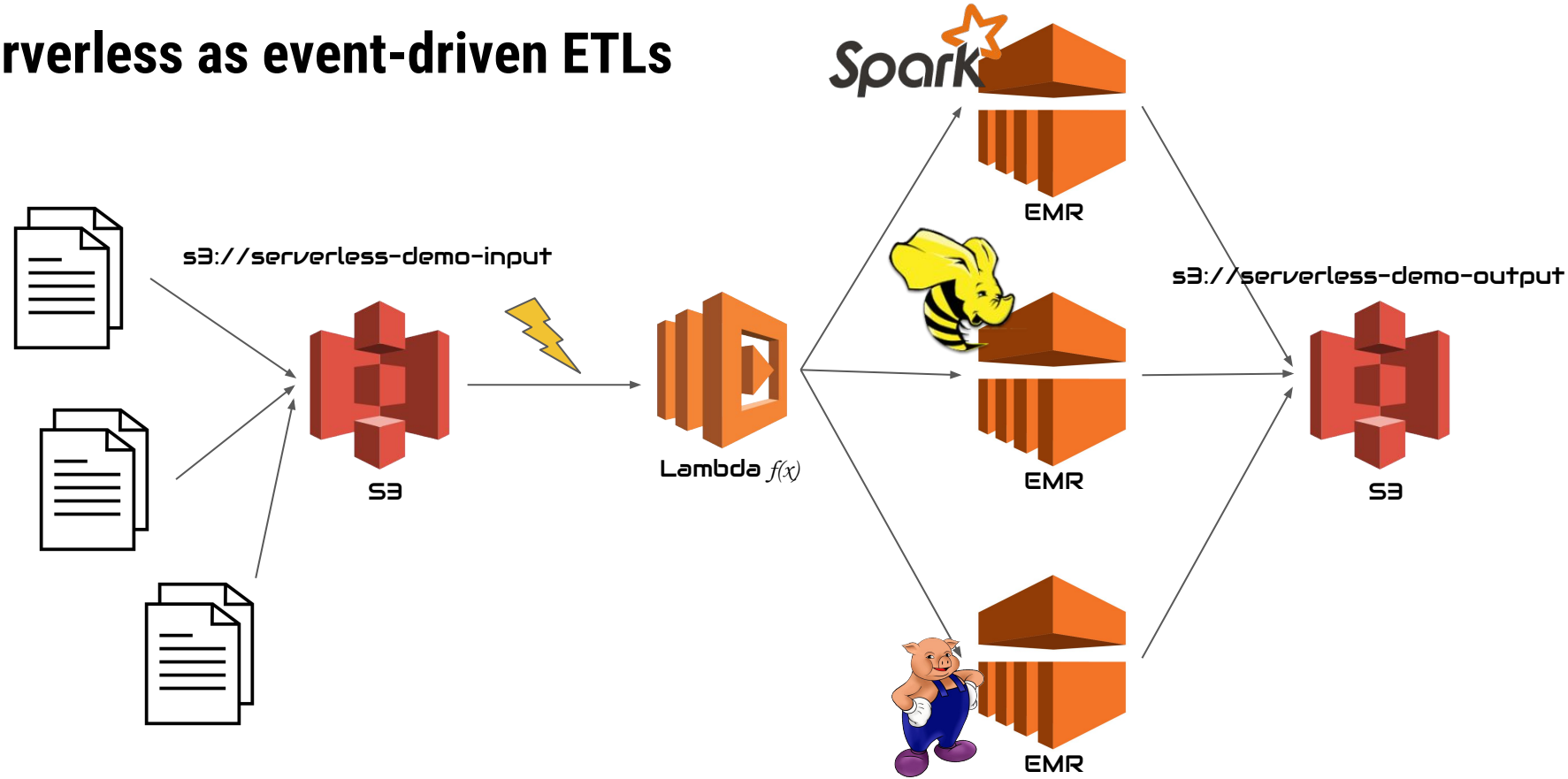
Serverless as Glue for Apps/AWS - Event sources

S3 - object-created or object-deleted
DynamoDB - stream-based
Kinesis Streams - kinesis-record
SNS - notification
SES - email-receiving
Cognito - sync-trigger
CloudFormation - create-request
CloudWatch Logs - event-log
CloudWatch Events - cron-expressions
Echo - Alexa Kits, Alexa Home
API Gateway - api-requests

& **invoke-async** with CLI or SDKs



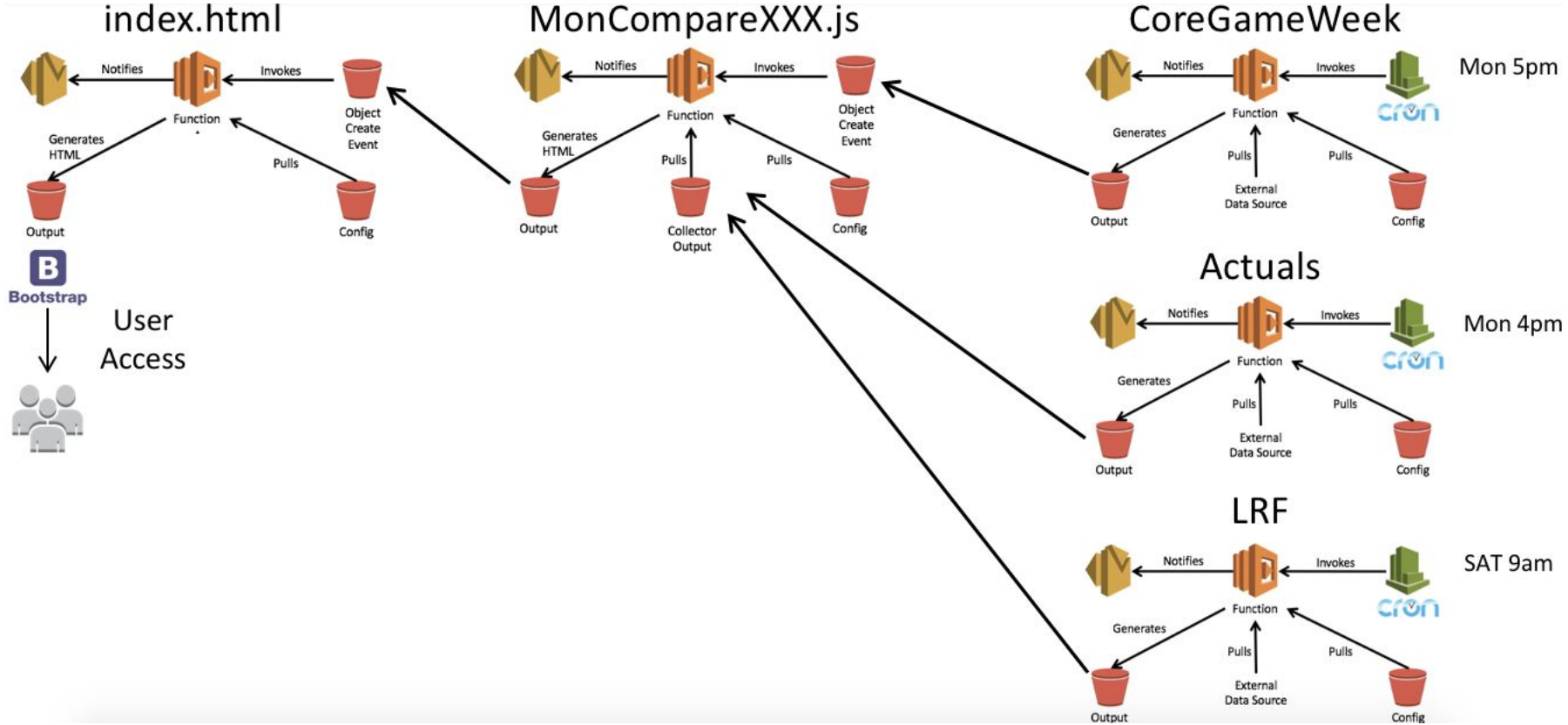
Serverless as event-driven ETLs



https://blogs.aws.amazon.com/bigdata/post/Tx1R28PXR3NAO1I/How-Expedia-Implemented-Near-Real-time-Analysis-of-Interdependent-Datasets
https://blogs.aws.amazon.com/bigdata/post/Tx3KH6BEUL2SGVA/Building-Scalable-and-Responsive-Big-Data-Interfaces-with-AWS-Lambda


dev
ops
λ
ETL DEMO!

Another example - **fmlnerd.com**



https://fmlnerd.com/2016/08/16/30k-page-views-for-0-21-a-serverless-story/


Q&A

Gracias!