Construir microservicios en Python. v3.0

Motivación

Build Microservices with Python (Microservice chassis Pattern)

Asked today Viewed 1 times

0

TL;DR: Is there any library or framework in Python that solves the most common problems of microservice architectures? I mean, a Microservice chassis Pattern

Java developers create microservices with Spring Boot they have everything done: there is a solution to apply the 12 factors and the key to other problems such as:

- Externalized configuration
- Logging
- Health checks
- Metrics
- circuit braker
- Distributed tracing
- etc.

I have found many Python libraries, boilerplates, blogs, but I couldn't find anything about creating "great" microservices with Python. Most of the documentation it's about: "Install flask, create routes, and... you have a microservice!". Of course, in Python we have many libraries like SQLAlchemy, Connexion, requests, Flask... but not a library or framework that covers the 90% of the usual problems that microservices create.

I've started a project with some colleagues to try to solve these issues:

https://github.com/python-microservices/microservices-scaffold

https://github.com/python-microservices/pyms

But everyday I think "it's impossible, it may exists a solution that is already done". So, does a microservice chassis pattern exist? (obviously, in Python)

python microservices

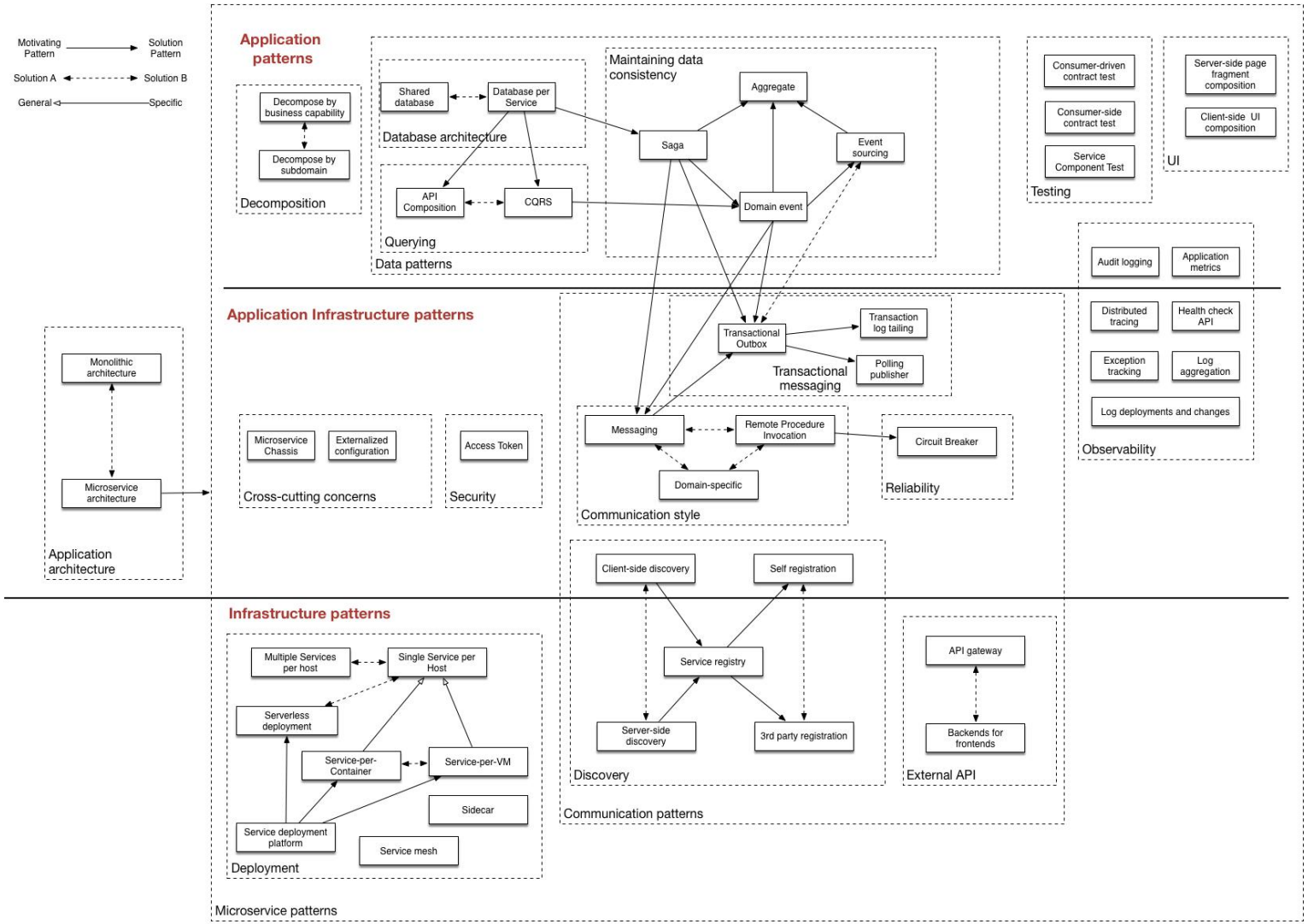
share edit close delete flag

asked just now
Avara
604 1 11 20

add a comment

Introducción

Patrones de microservicios



Patrón: Microservice chassis

- Configuración externalizada
- Trazabilidad de peticiones
- logging
- Health checks
- Metrics

https://microservices.io/patterns/microservice-chassis.html

Lo que ya tienen otros: Microservice chassis pattern

Java
- Spring Boot y Spring Cloud
- Dropwizard

Go
- Gizmo
- Micro
- Go kit

Lo que tiene Python:

ARQUETIPO

github.com/python-microservices

PyMS, la librería construida y basada en Flask que unifica todas las librerías necesarias para construir un microservicio. Entre ellas:

- Flask (obvio).
- Connexion.
- Prometheus.
- Opentracing y Jaegger.
- Anyconfig.
- swagger-ui.
- Cryptography.

Más literatura de cómo llegué a esto (DRY):
paradigmadigital.com/dev/como-construir-microservicios-en-python-1-2/
paradigmadigital.com/techbiz/microservice-chassis-pattern-python-2-2/

Librería y arquetipos: cómo contribuir

Librería Patrón Chasis para Microservicios:
github.com/python-microservices/pyms

Arquetipo en el que nos hemos basado:
github.com/python-microservices/microservices-scaffold

Template con Cookiecutter:
github.com/python-microservices/cookiecutter-pyms

Ejemplos

Ejemplos sencillos:
github.com/python-microservices/pyms/tree/master/examples

Ejemplo con Kubernetes:
github.com/python-microservices/microservices-chat

Ejemplo con Docker Compose:
github.com/avara1986/pivoandcode-2019-11-15

E[illegible]

REALIDAD

Microservicios en el mundo real

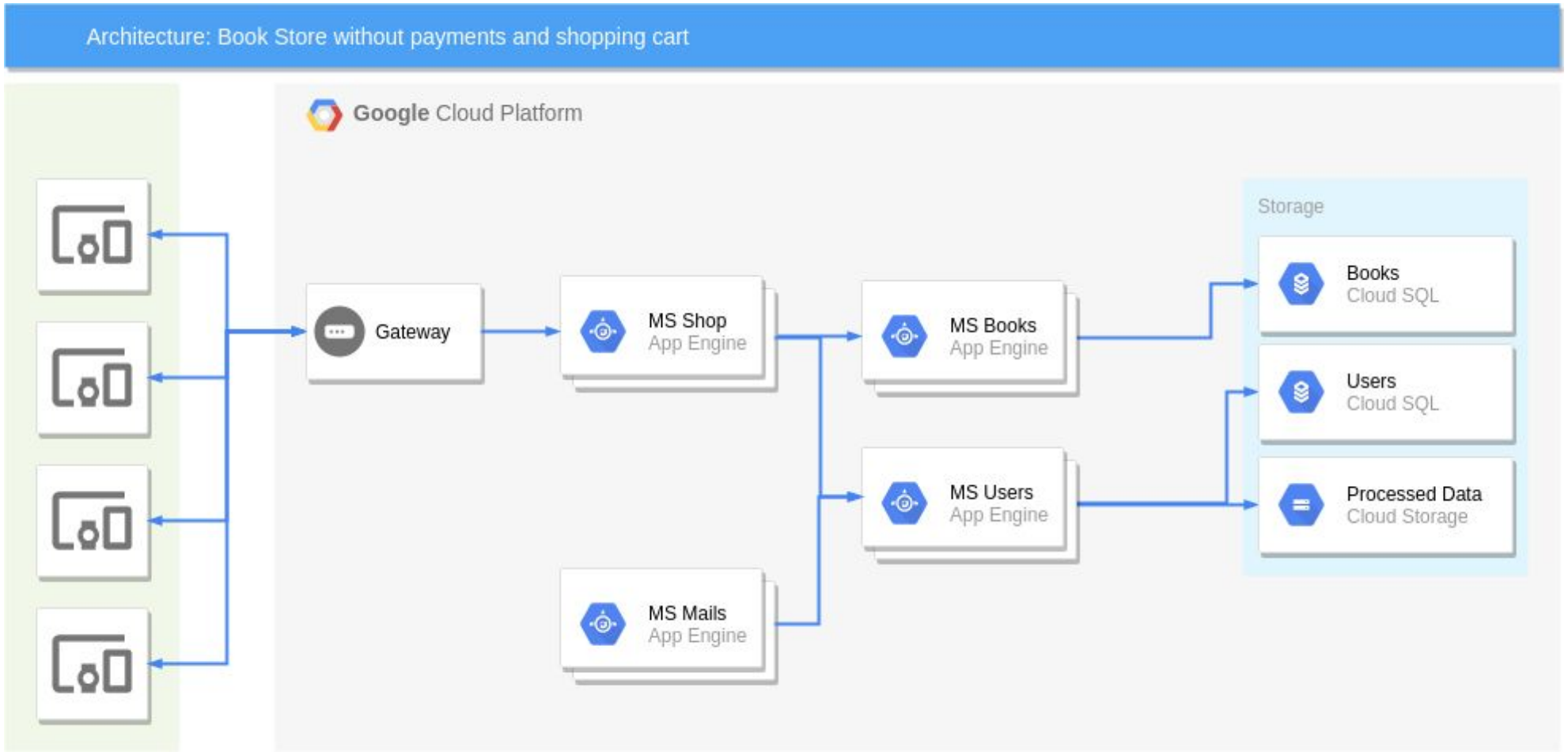
Blogs y tutoriales de internet al buscar “Build microservice in XXX”

- **Añade una ruta GET: /users/**
- **Añade otra ruta PUT: /users/**
- **devuelve un JSON:**

```
[{
        “id”: 1,
        “name”: “ImTheBoss”
}]
```

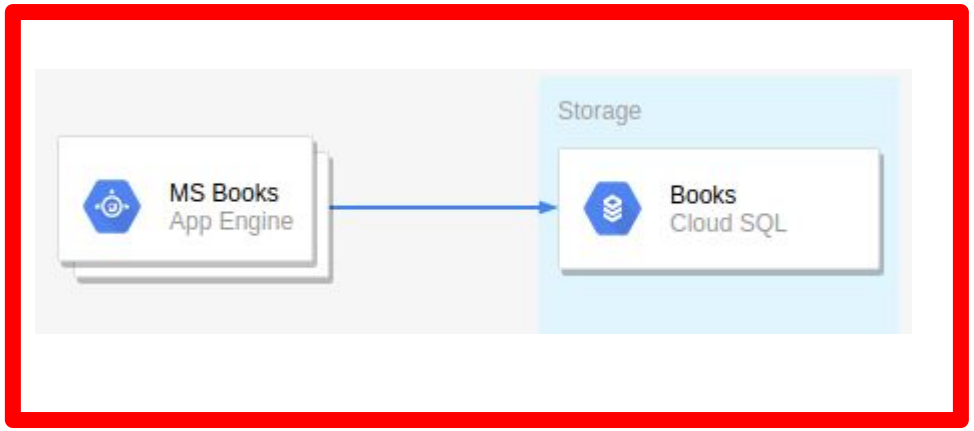
- **Actualiza Linkedin con “Microservice Architect”**
- **Wait...**

El cliente que te pide hacer esto:



El cliente que te pide hacer esto:

Problema: Configuración y desarrollo en local



Problema: Configuración y desarrollo en local

Antes: config.py

```
class Config:
    DEBUG = False
    TESTING = False
    APP_NAME = "Template"
    SQLALCHEMY_DATABASE_URI = "sqlite:/..."

class DevConfig(Config):

class TestConfig(Config):

class PreConfig(Config):

class ProConfig(Config):
```

Ahora: config.yaml

```
ms:

 DEBUG: false

 TESTING: false

 APP_NAME: Template

 SQLALCHEMY_DATABASE_URI : sqlite:/…
```

Problema: Configuración y desarrollo en local

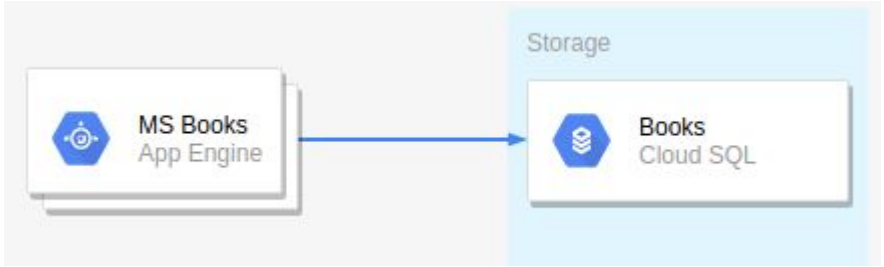
Docker compose

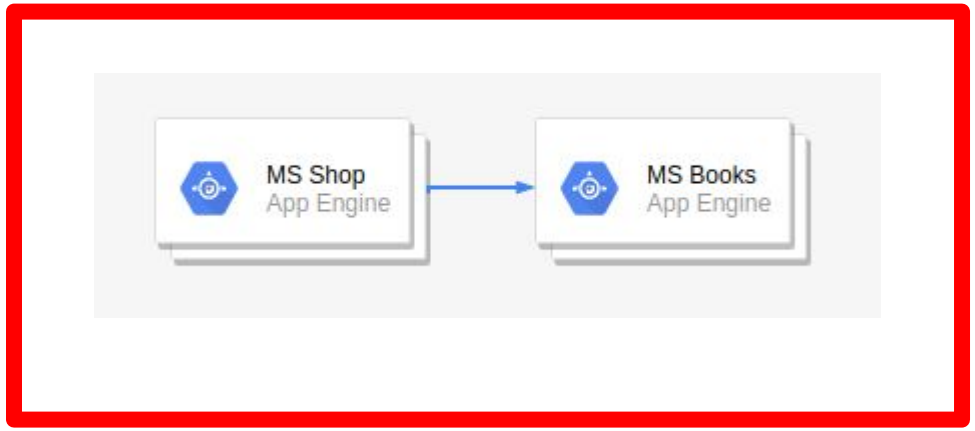
```
services:
  postgresql:
    image: postgres
  books:
    image:
    depends_on:
      - mysql
  shop:
    image:
    depends_on:
      - books
```

Kubernetes (Minikube) y Helm

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: {{ include "chat_db.fullname" . }}
 labels:
spec:
 replicas: {{ .Values.replicaCount }}
 template:
   metadata:
     labels:
       app.kubernetes.io/name: {{ include "chat_db.name" . }}
   spec:
     containers:
       - name: {{ .Chart.Name }}
         image: "{{ .Values.image.repository }}:{{
.Values.image.tag }}"
         ports:
           - name: http
             containerPort: 8080
             protocol: TCP
```

Problema: Trazabilidad y Logging





Problema: Logging

Mal:

```
DEBUG:pyms-requests: Response <Response [200]>
```

Guay:

```
{"timestamp": "2019-11-02T19:58:51.957358Z", "name": "pyms", "module": "requests", "funcName": "post",
"lineno": 205, "message": "Response <Response [200]>", "severity": "DEBUG", "service": "chat daas", "trace":
"678d1bb91836c636", "span": "40c17e8f5067a1dd", "parent": ""}
```

Opentracing

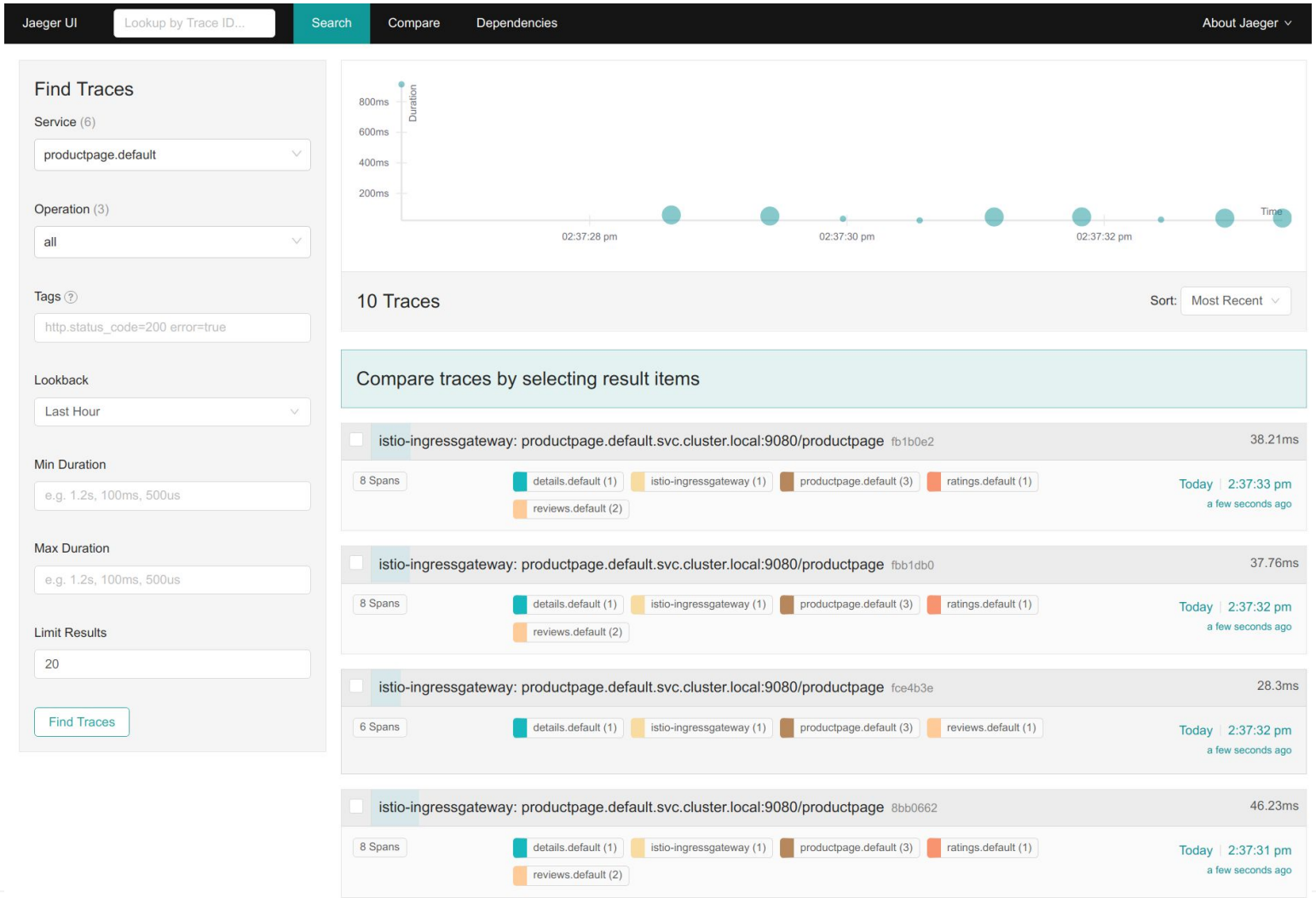
Iniciativa para estandarizar la trazabilidad de peticiones

Librerías para los lenguajes más populares: Python, C++, Go, Java, PHP, C#, Objective-C… y también Javascript

Variedad de sistemas de trazabilidad como **Jaegger** (creado por Uber y compatible con Zipkin), Apache SkyWalking...

Jaeger client + Jaeger UI

OPENTRACING



Otros métodos

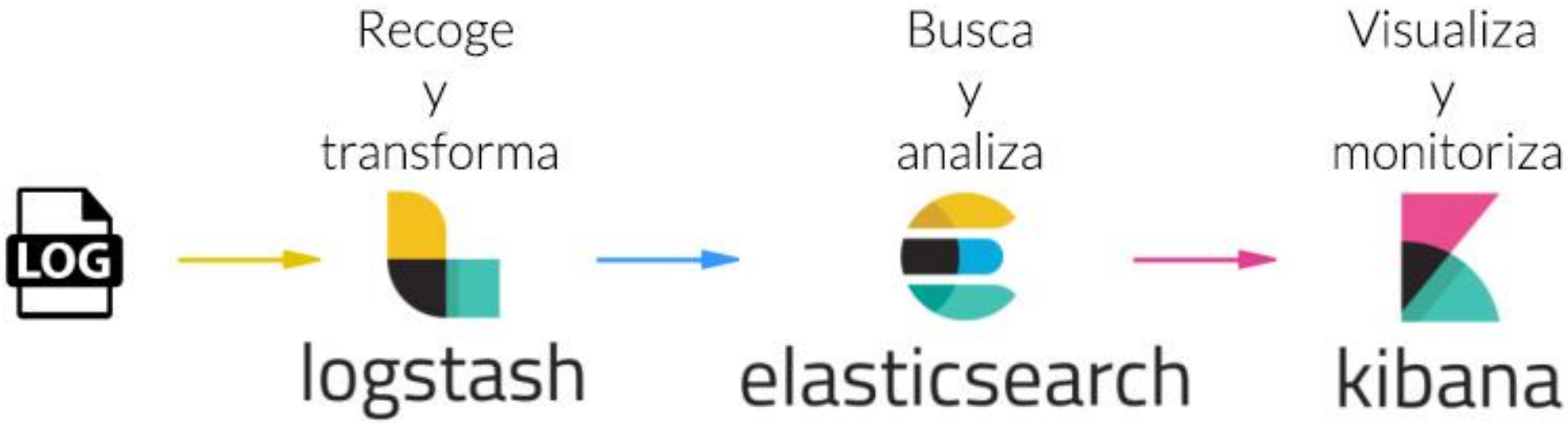
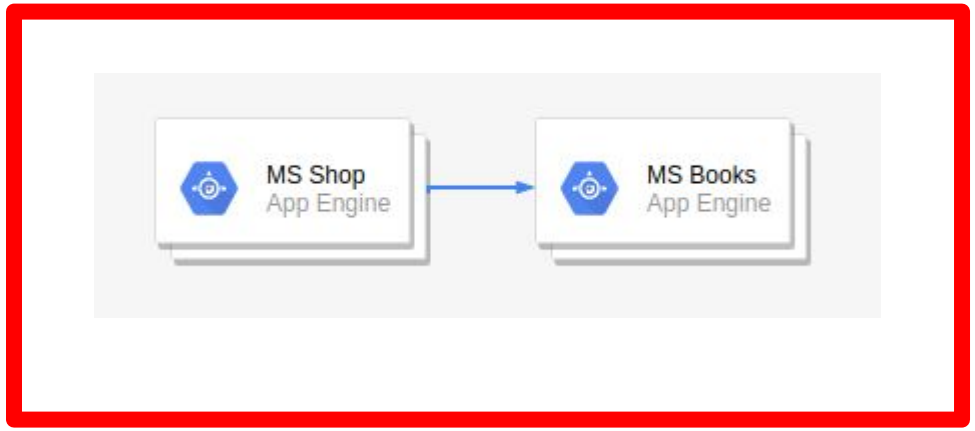


Imagen de Openwebinars

Problema: Healthcheck, metrics y Circuit Breaker



Problema: Healthcheck

A veces, una instancia puede ser incapaz de manejar solicitudes y aún estar en ejecución. Por ejemplo, es posible que se hayan quedado sin conexiones a la base de datos. Cuando esto ocurre, el sistema de monitorización debe generar una alerta. Además, el balanceador de carga o el registro de servicios no deben enrutar las solicitudes a la instancia de servicio fallida. Ejemplo:

URL:

http://localhost:8000/healtcheck

Respuesta:

```
curl -IX GET "http://localhost:8000/healtcheck"
HTTP/1.1 200
Content-Type: text/html; charset=utf-8
Date: Sun, 15 Sep 2019 12:32:22 GMT
```

Problema: Metrics

http://localhost:8000/metrics

```
# HELP go gc duration seconds A summary of the GC invocation durations.
# TYPE go gc duration seconds summary
go gc duration seconds{quantile="0"} 3.291e-05
go gc duration seconds{quantile="0.25"} 4.3849e-05
go gc duration seconds{quantile="0.5"} 6.2452e-05
go gc duration seconds{quantile="0.75"} 9.8154e-05
go gc duration seconds{quantile="1"} 0.011689149
go gc duration seconds sum 3.451780079
go_gc_duration_seconds_count 13118
```

¡GRACIAS POR VUESTRO TIEMPO!

github.com/avara1986

twitter.com/a_vara_n

linkedin.com/in/albertovara/

a.vara.1986@gmail.com