

Tamarco

framework de microservicios

José Melero Fernández
jmelerofernandez@gmail.com
5 octubre 2019




Índice

- Introducción
 - System73 y PolyNet
 - Microservicios
- Primera aproximación
 - Librerías
 - Problemas
- Tamarco
 - Ciclo de vida
 - Configuración
 - Ejemplo
- Conclusiones



- **Introducción**
 - System73 y PolyNet
 - Microservicios
- Primera aproximación
 - Librerías
 - Problemas
- Tamarco
 - Ciclo de vida
 - Configuración
 - Ejemplo
- Conclusiones



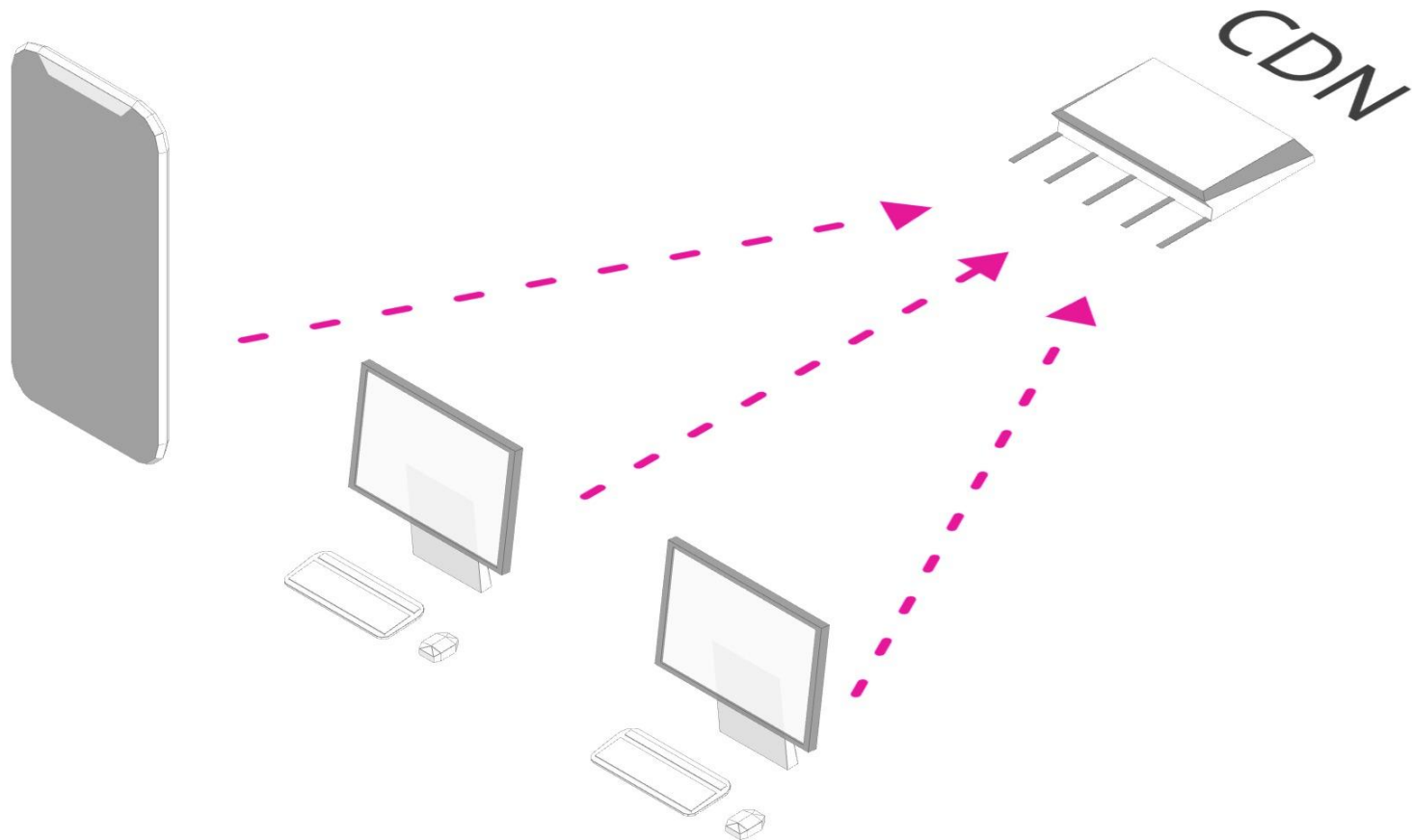
```
1 class TamarcoPresentation:
2
3     def __init__(self):
4         ...
```

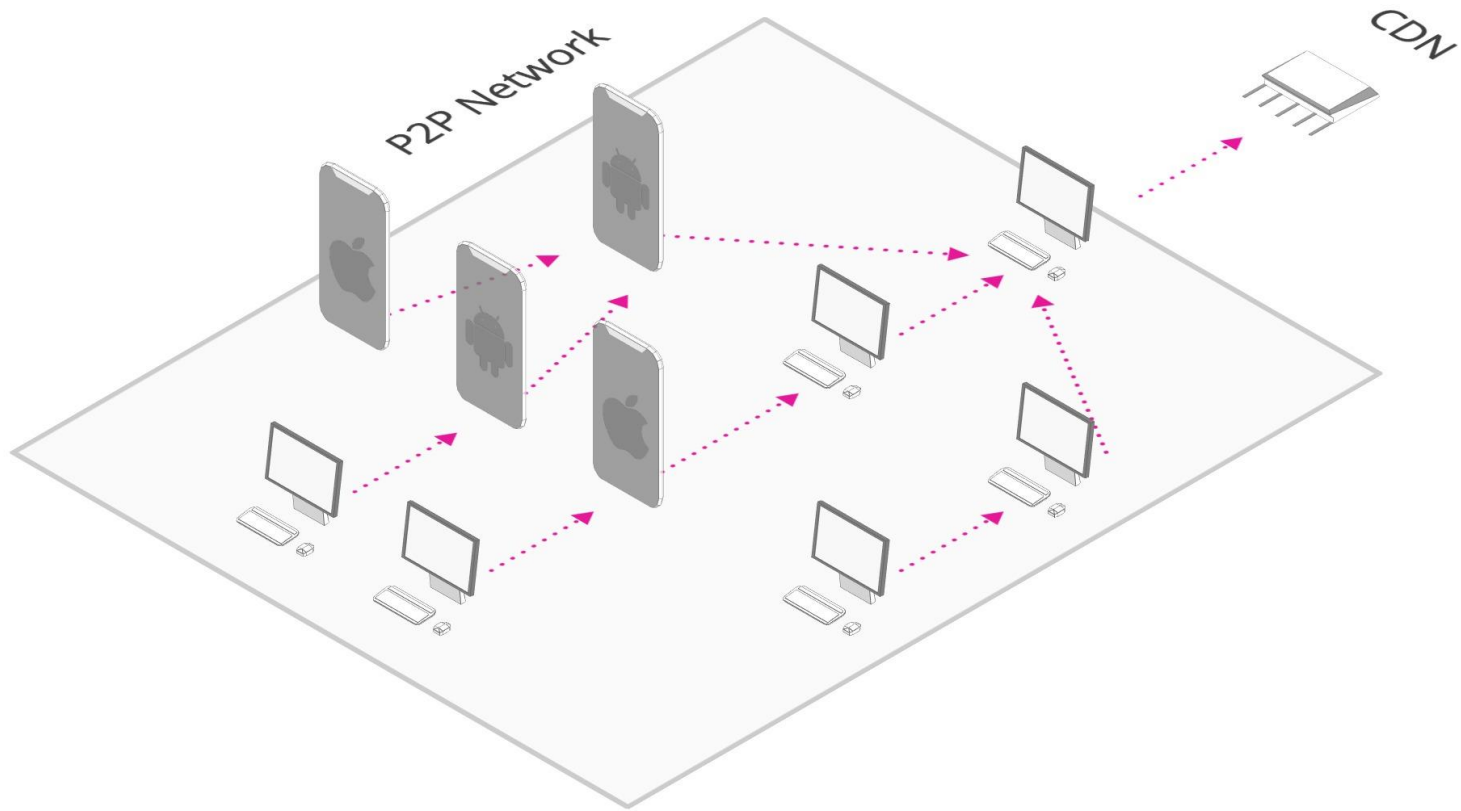


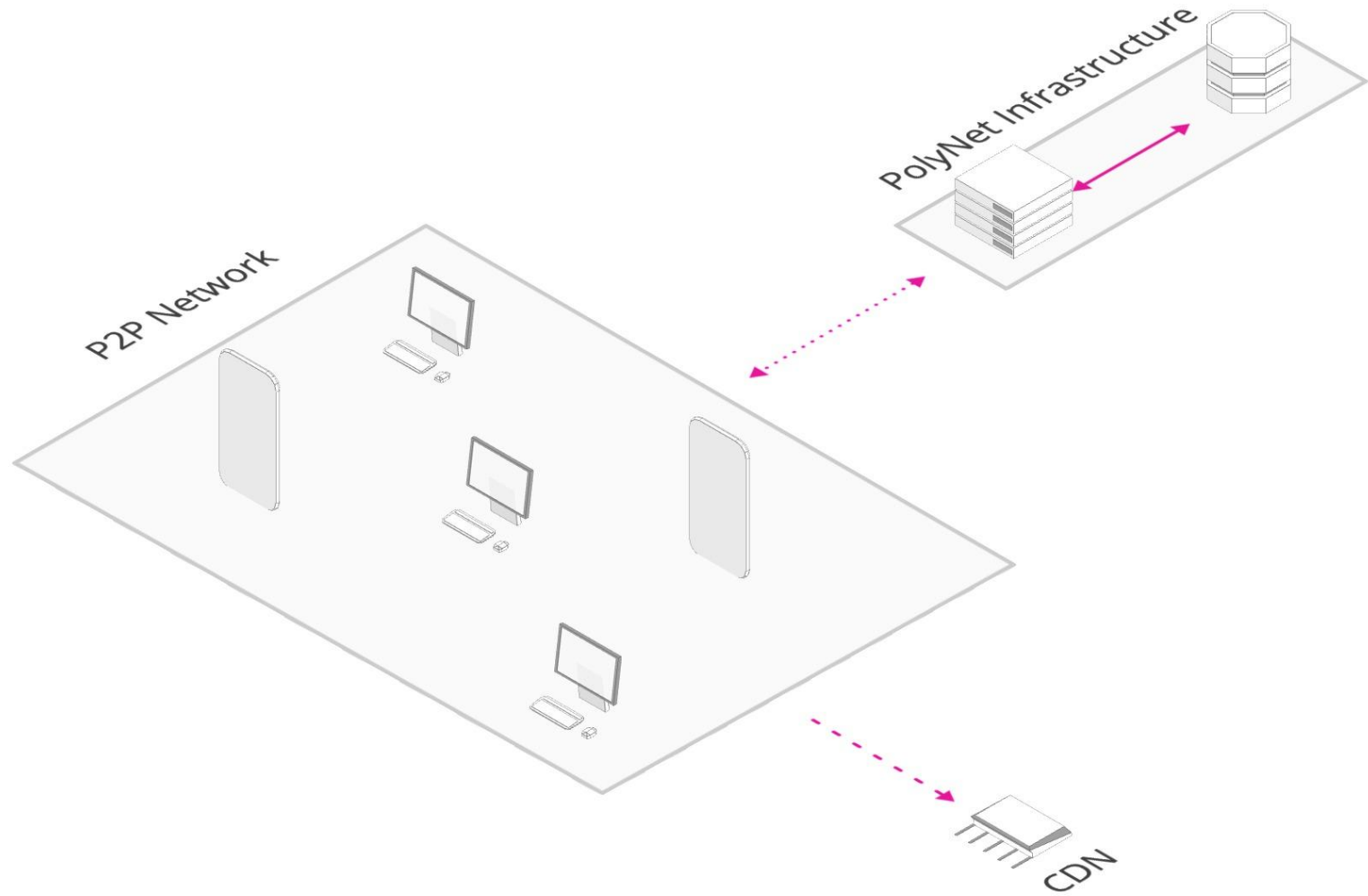
SYSTEM73
Vanquish Technology SLU

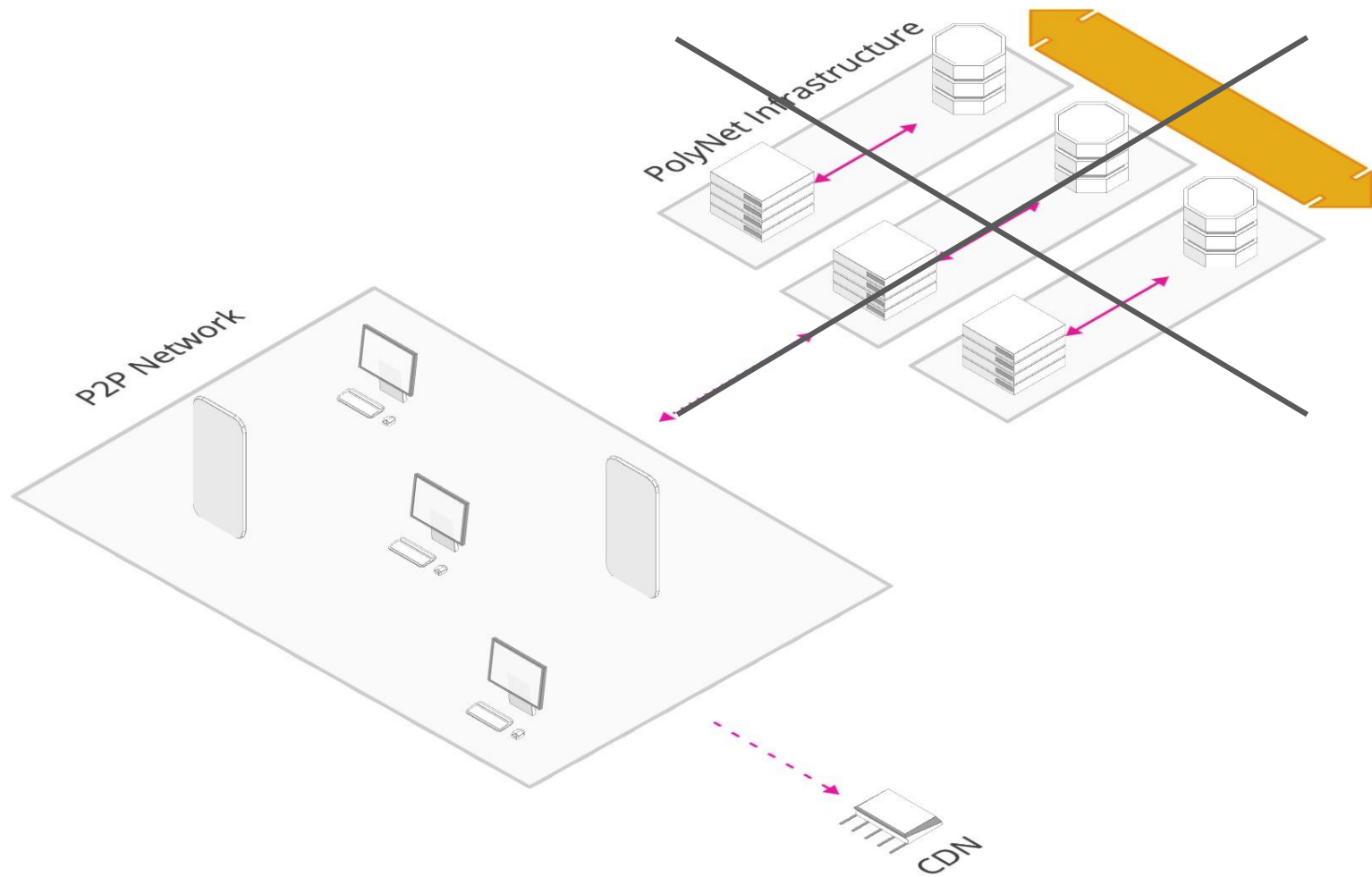
The background of the slide is a deep blue color. It features a complex, abstract network of white lines and dots. The dots, representing nodes, are of varying sizes and are scattered across the frame. The lines, representing connections, are thin and crisscross the background, creating a sense of depth and connectivity. The overall effect is reminiscent of a digital or neural network.

PolyNet

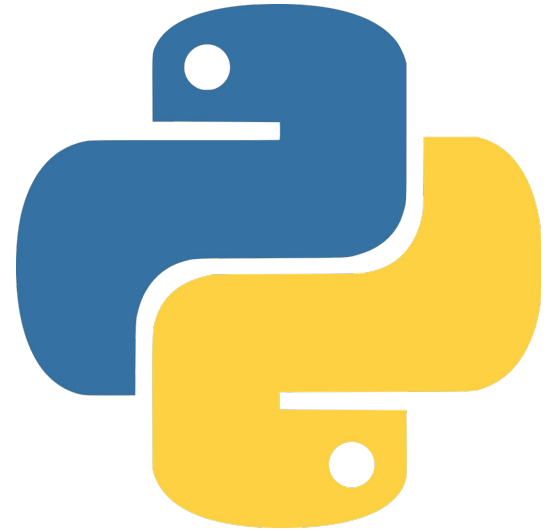




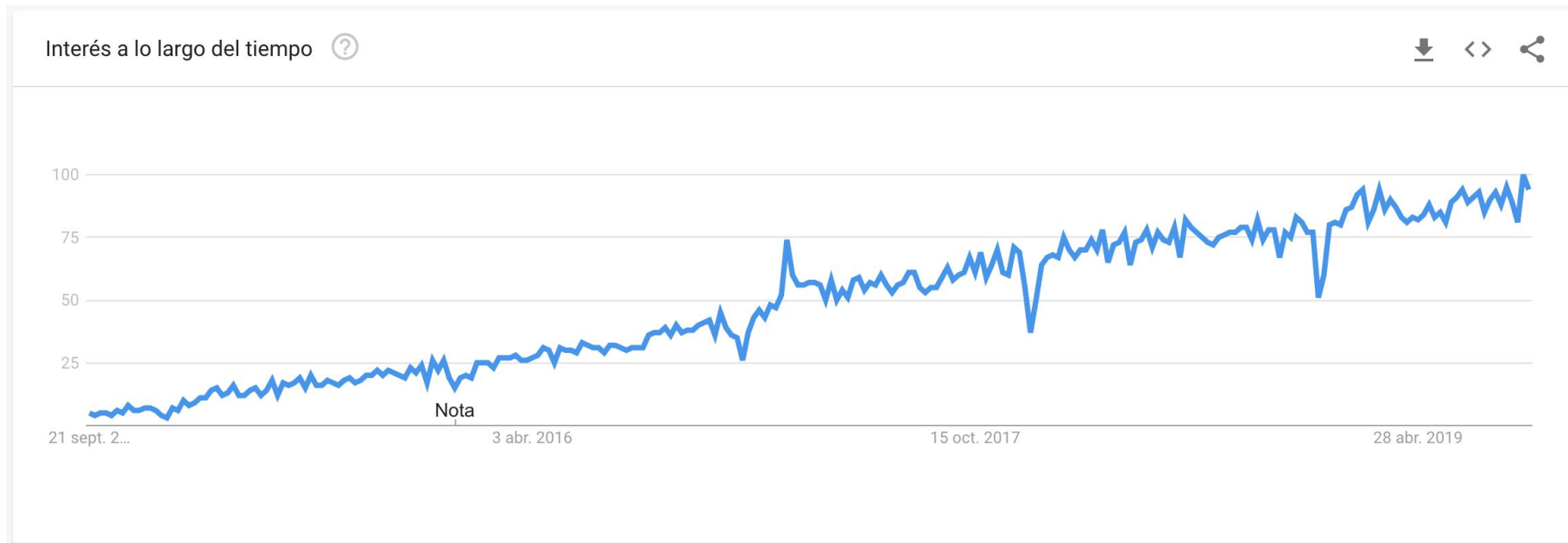




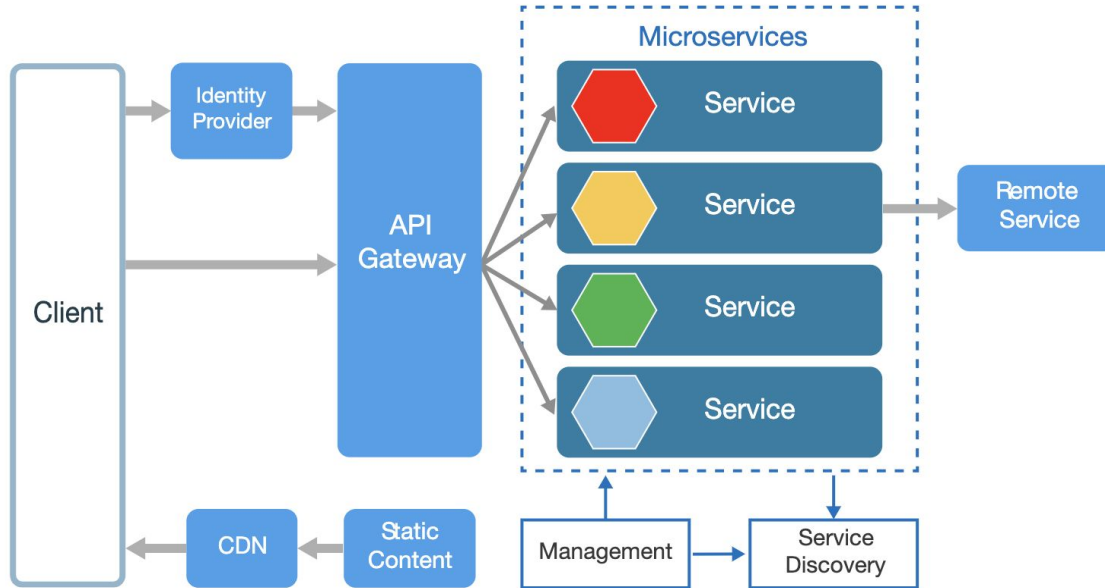
Microservicios + Python



Microservicios



Microservicios

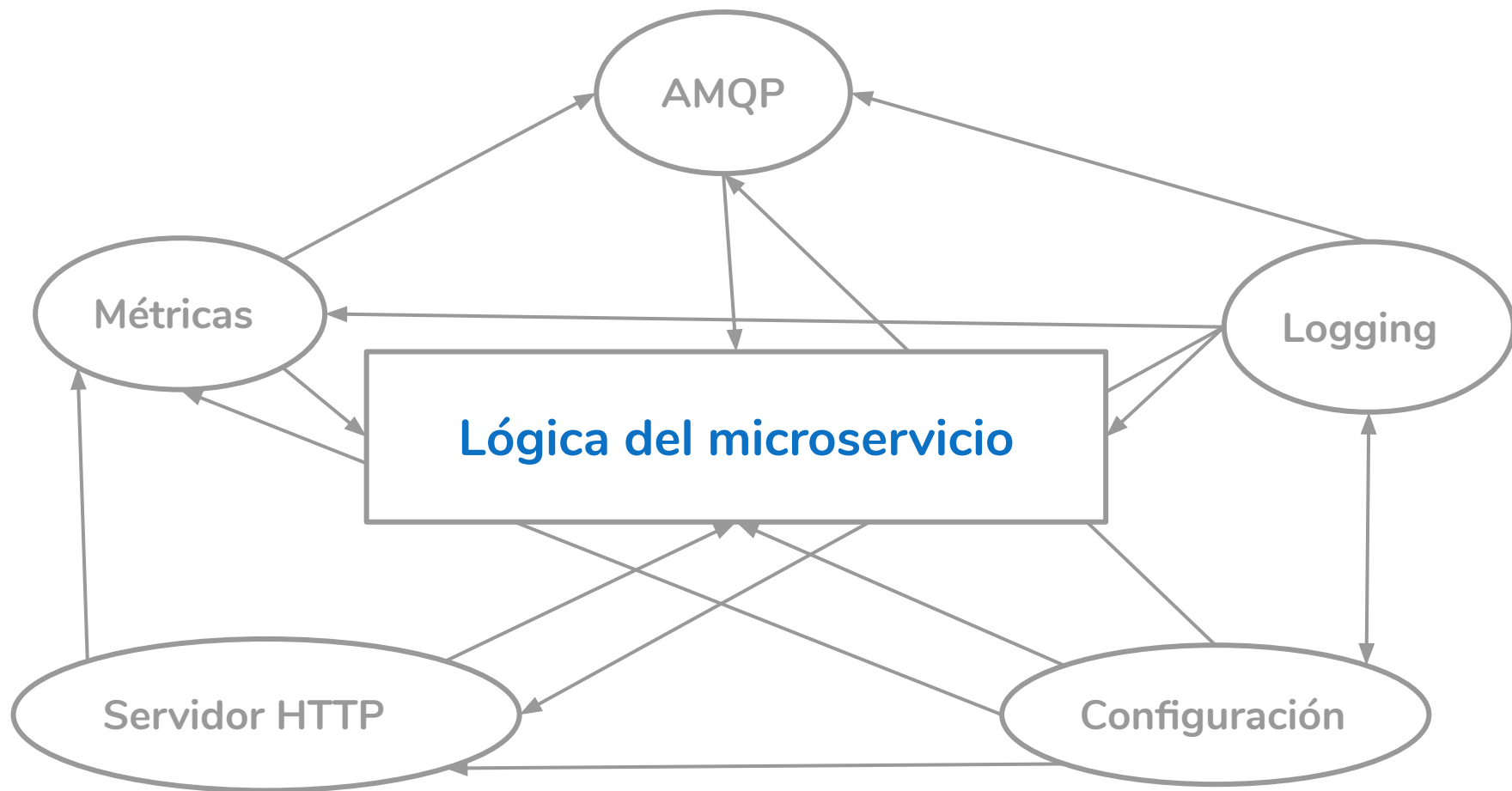


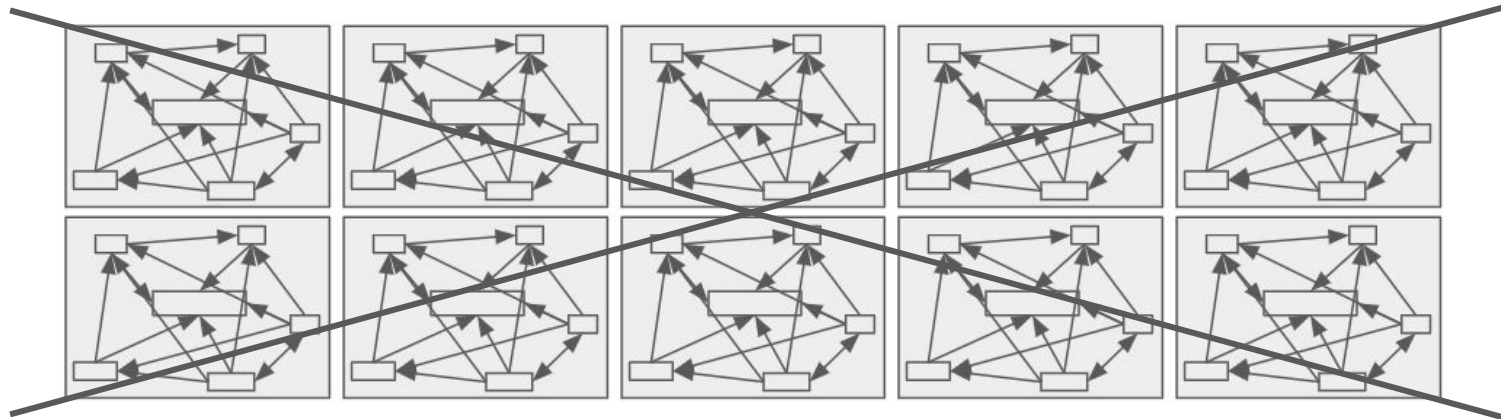
- Son pequeños e independientes
- Son responsables de conservar sus propios datos.
- Se comunican entre sí mediante API bien definidas.
- Pueden usar tecnologías y lenguajes diferentes.

Requieren de muchas tecnologías y herramientas para usarlos de forma eficiente.

- Introducción
 - System73 y PolyNet
 - Microservicios
- **Primera aproximación**
 - Librerías
 - Problemas
- Tamarco
 - Ciclo de vida
 - Configuración
 - Ejemplo
- Conclusiones





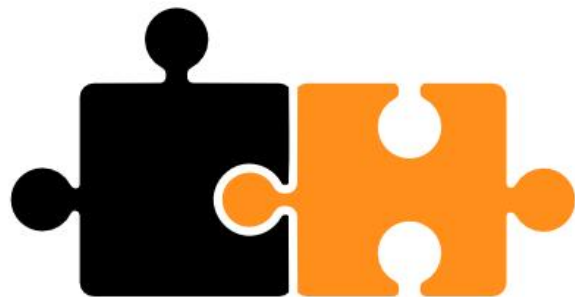


Problemas

- Código repetido entre todos los servicios
 - Muchos pull requests iguales
 - Mucho tiempo en tareas de mantenimiento
 - Código repetido también implica test repetidos
 - ...
- Comportamiento incoherente entre los microservicios
- El proceso de crear un microservicio es muy repetitivo
- Diferentes patrones de concurrencia



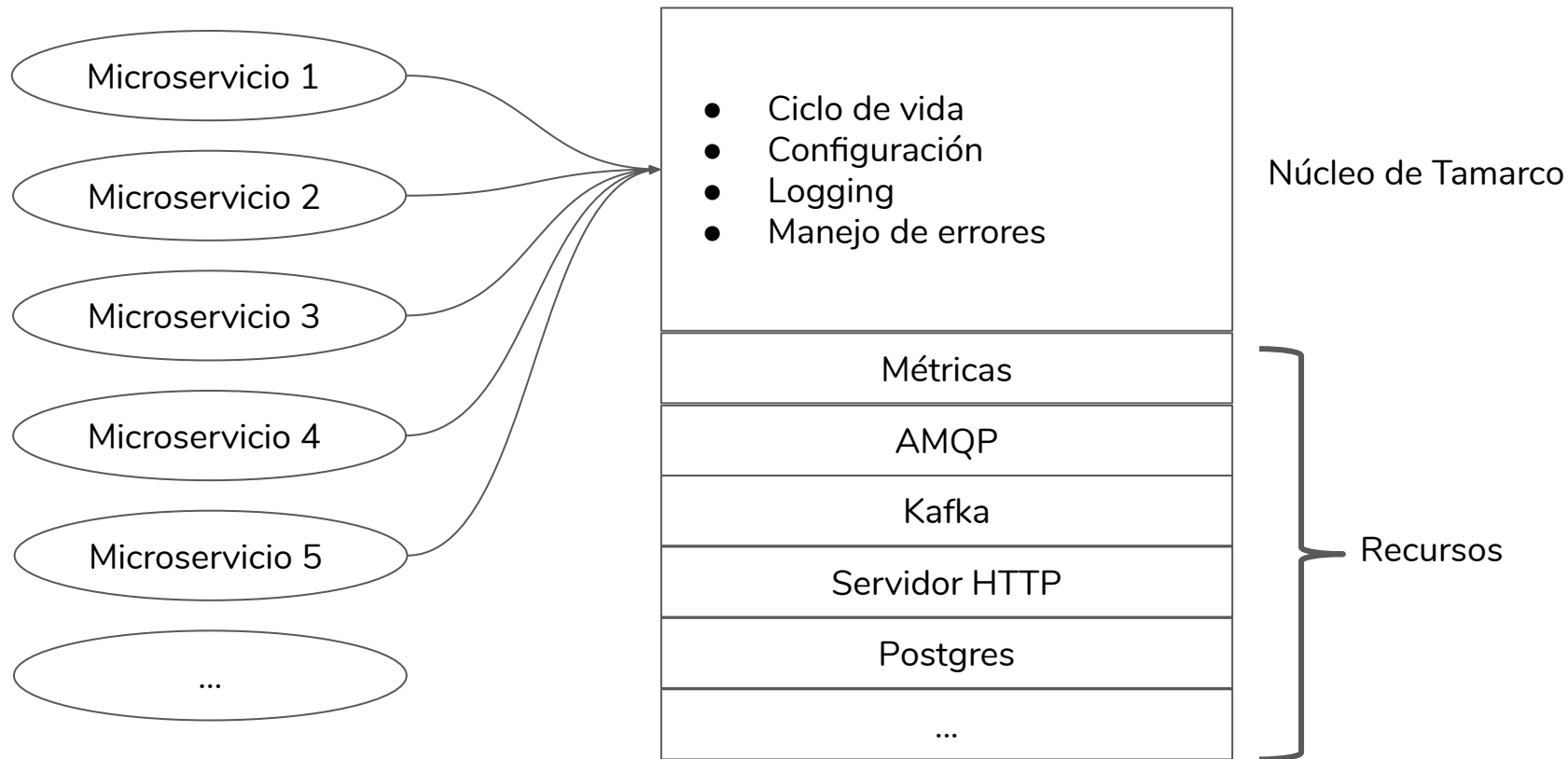
- Introducción
 - System73 y PolyNet
 - Microservicios
- Primera aproximación
 - Librerías
 - Problemas
- **Tamarco**
 - Ciclo de vida
 - Configuración
 - Ejemplo
- Conclusiones



TAMARCO

Características de Tamarco

- Gestión del ciclo de vida
- Configuración centralizada en etcd
- Configuración del *logging*
- Arquitectura orientada a recursos
- Basado en *asyncio*
- Nuevos servicios a partir de plantillas



Ciclo de vida



Inicio

Fases del inicio:

1. Logging provisional
2. Configuración
3. Logging
4. Pre start
5. Start
6. Post start

```
1 from tamarco import Microservice
2
3
4 class LifecycleMicroservice(Microservice):
5
6     async def pre_start(self):
7         print("Before pre_start of the service")
8         await super().pre_start()
9         print("After pre_start of the service")
10
11     async def start(self):
12         print("Before start of the service")
13         await super().start()
14         print("After start of the service")
15
16     async def post_start(self):
17         print("Before post_start of the service")
18         await super().post_start()
19         print("After post_start of the service")
```

Parada

Fases de la parada:

1. stop
2. post stop

```
1 from tamarco import Microservice
2
3
4 class LifecycleMicroservice(Microservice):
5
6     async def stop(self):
7         print("Before stop of the service")
8         await super().stop()
9         print("After stop of the service")
10
11     async def post_stop(self):
12         print("Before post stop of the service")
13         await super().post_stop()
14         print("After post stop of the service")
```

Concurrencia

```
8 import asyncio
9 import time
10
11 from tamarco.core.microservice import (Microservice,
12     MicroserviceContext, task, task_timer, thread)
13
14
15 class ConcurrencyExampleMicroservice(Microservice):
16
17     @task
18     async def task_example(self):
19         while True:
20             self.logger.info("task example, beep")
21             await asyncio.sleep(1)
22
23     @task_timer(interval=1000, autostart=True)
24     async def task_timer_example(self):
25         self.logger.info("task timer example, beeeep")
26
27     @thread
28     def thread_example(self):
29         while True:
30             self.logger.info("thread example, beeeep")
31             time.sleep(1)
```

Configuración

Configuración

Opciones de configuración:

- etcd
- Archivo YAML (desarrollo)
- Diccionario (desarrollo)



```
1 system:
2   deploy_name: PyconES2019
3   logging:
4     profile: DEVELOP
5     file_path: false
6     stdout: true
7   microservices:
8     my_admin:
9       default_user: tamarco
10      default_password: my_password
11      session_expiration_days: 3
12  resources:
13    amqp:
14      host: 127.0.0.1
15      port: 5672
16      vhost: /
17      user: microservice
18      password: 1234
19      connection_timeout: 10
20      queues_prefix: "prefix"
```



```
1 from tamarco.core.microservice import Microservice
2
3
4 class MyAdmin(Microservice):
5
6     async def pre_start(self):
7         await super().pre_start()
8         self.default_user = await self.settings.get(
9             '/system/microservices/my_admin/default_user')
10        self.default_password = await self.settings.get(
11            '/system/microservices/my_admin/default_password')
12        self.session_expiration_days = await self.settings.get(
13            '/system/microservices/my_admin/session_expiration_days')
14
```



- Base de datos clave valor
- Configuración centralizada
- Reporta cambios en los datos
- Fiable

Variables de entorno:

- TAMARCO_ETCD_HOST
- TAMARCO_ETCD_PORT

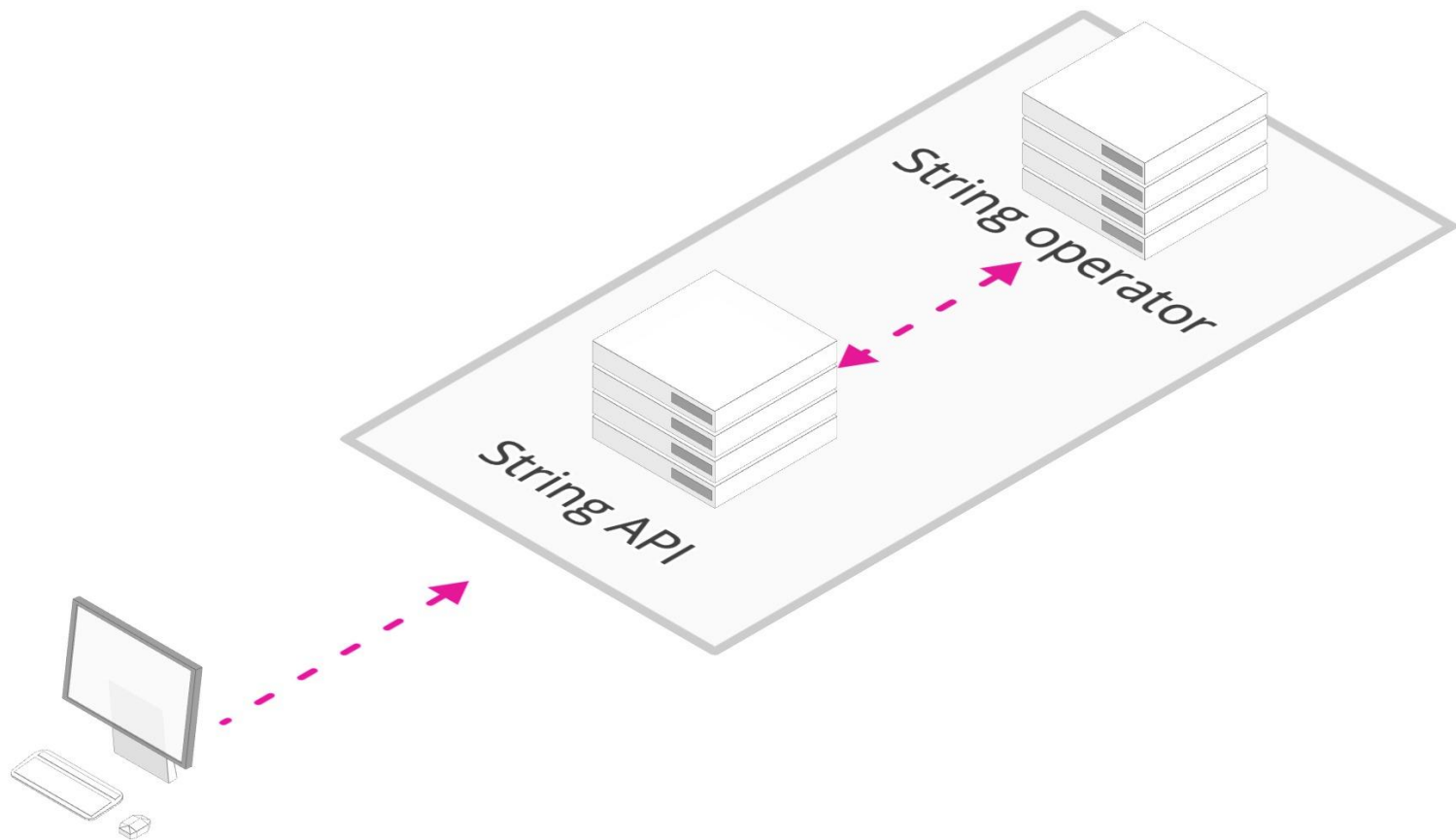
```
tamarco etcd write_yaml settings.yml
```

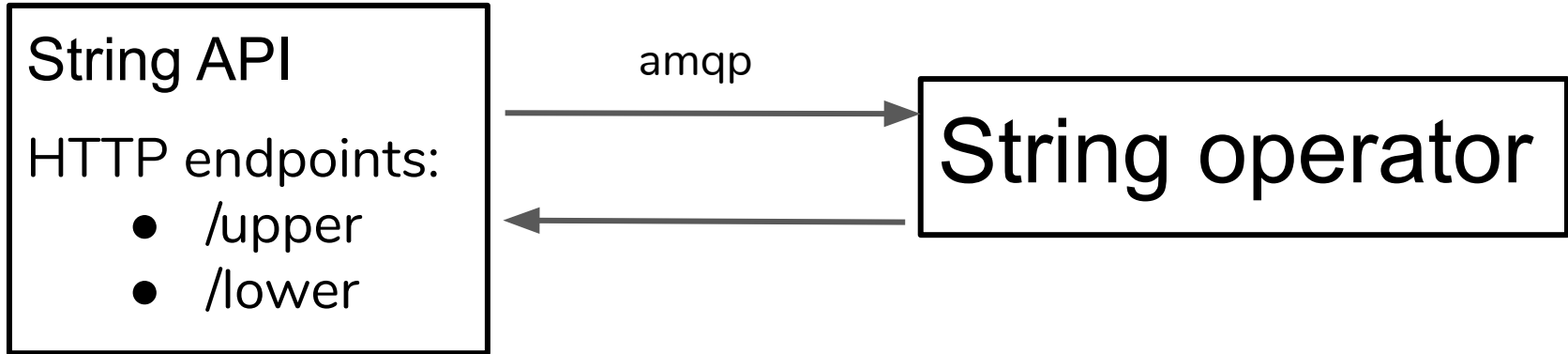
Logging

- Envío a Logstash
 - UDP
 - Redis
 - HTTP
- Etiquetas
 - Deploy
 - Nombre de servicio
 - Identificador de servicio
- Campos extra definidos por el usuario
- Traza de la excepción

```
1  system:
2    logging:
3      profile: DEVELOP
4      file_path: false
5      stdout: true
6      redis:
7        enabled: true
8        host: 127.0.0.1
9        port: 6379
10       password: my_password
11       ssl: false
12     http:
13       enabled: true
14       url: http://127.0.0.1
15       user:
16       password:
17       max_time_seconds: 15
18       max_records: 100
```

Ejemplo





- Introducción
 - System73 y PolyNet
 - Microservicios
- Primera aproximación
 - Librerías
 - Problemas
- Tamarco
 - Ciclo de vida
 - Configuración
 - Ejemplo
- Conclusiones

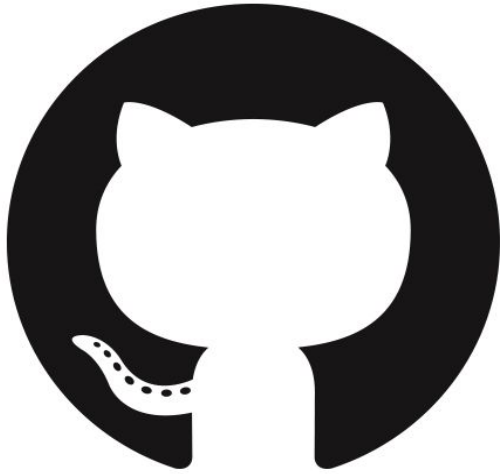


Conclusiones

- Un mes en desarrollar el framework
- Dos meses en tenerlo completamente integrado
- Sin grandes cambios desde la primera implementación
- Nos permitió escalar el número de servicios rápidamente
- Comportamiento coherente y predecible de los diferentes servicios

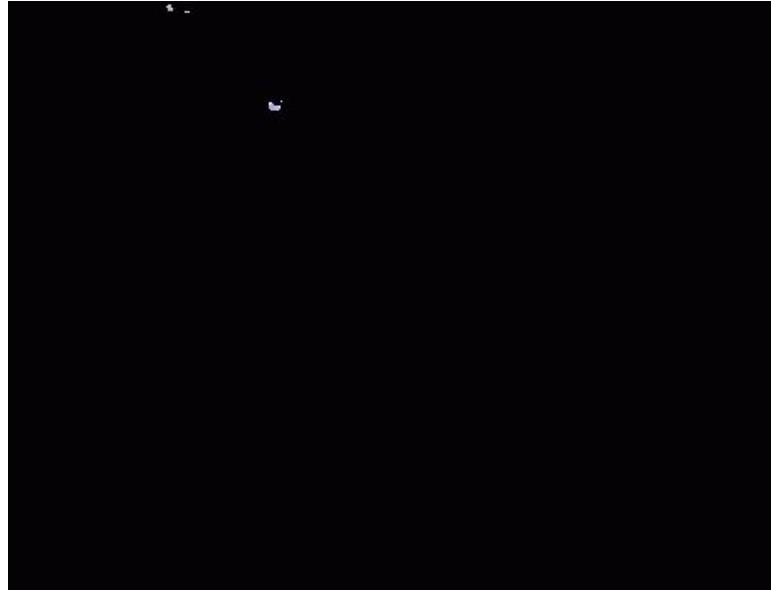
Problema principal:

- Rendimiento de Python insuficiente para ciertas tareas



System73/tamarco

Gracias por su atención



Tamarco

framework de microservicios

José Melero Fernández
jmelerofernandez@gmail.com
5 octubre 2019

