OLIST ARCHITECTURE

OSVALDO

- Programador amador desde 1986, profissional desde 1989 e pythonista desde 2000
- Criador e ex-moderador da da python-brasil@yahoo (atual python-brasil@googlegroups)
- Criador da primeira versão do www.python.org.br
- Sócio fundador e ex-presidente da Associação Python Brasil
- Autor do (antigo) livro Python e Django
- Atualmente na Olist
- osvaldo@olist.com e @osantana (ou @osantanabr)

O QUE NÓS FAZEMOS?

Merchants

Online | Offline

olist

Submarino

AMERICANAS.com

mercado livre

extra

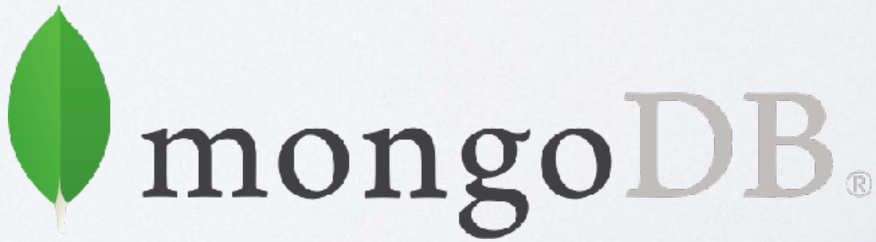
CASAS BAHIA

Walmart

BEFORE…

OLIST VERSION ONE (AKA V1)

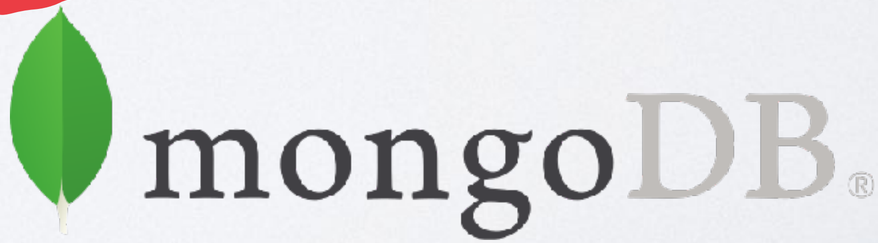
AngularJS

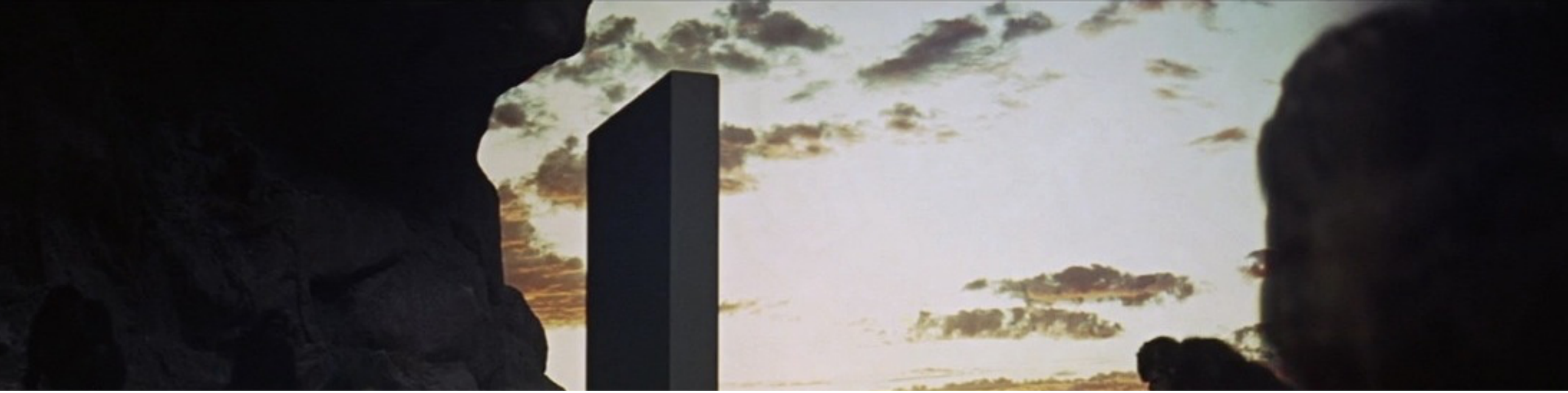


OLIST VERSION ONE (AKA V1)

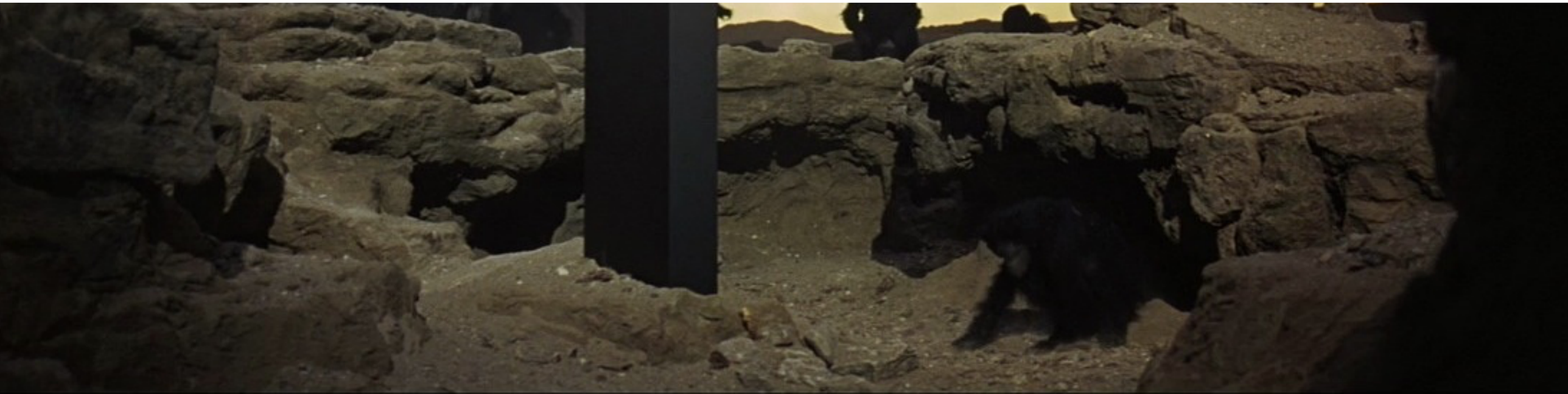
COMPLEX!

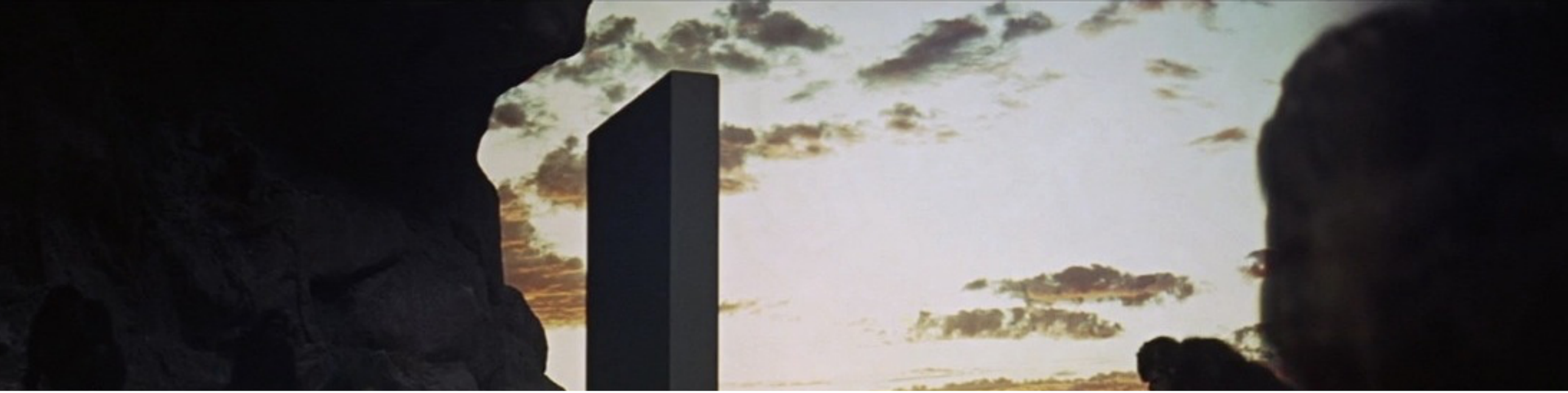
ularJS



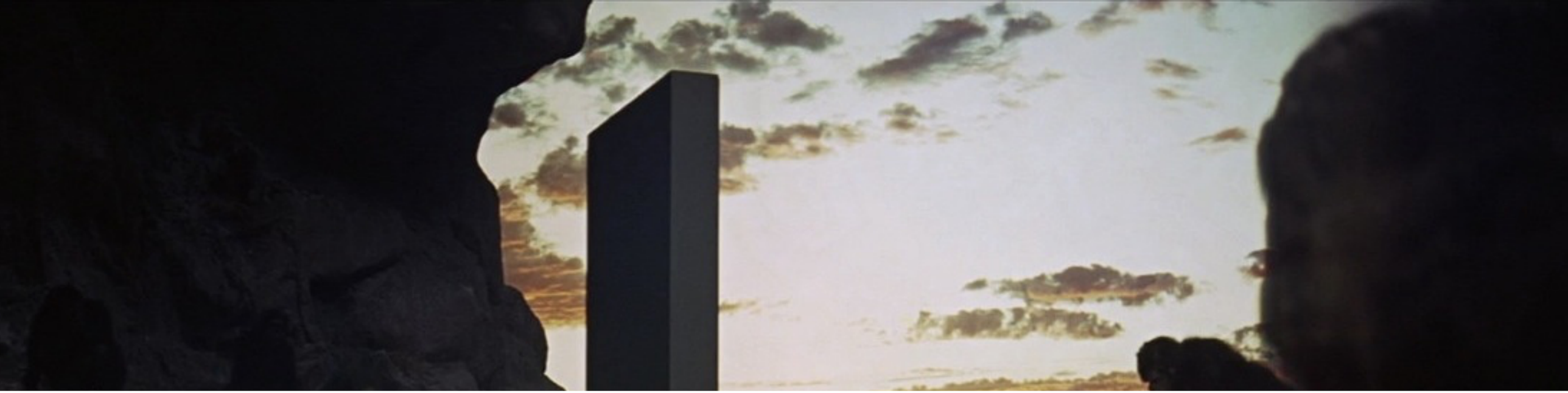


MONOLITHIC

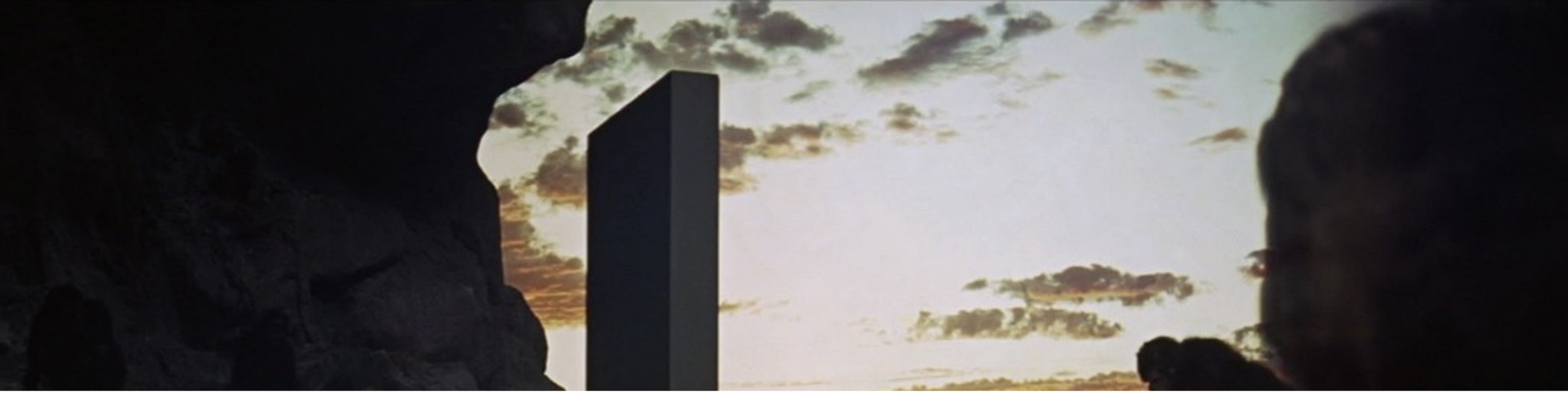




MONOLITHIC



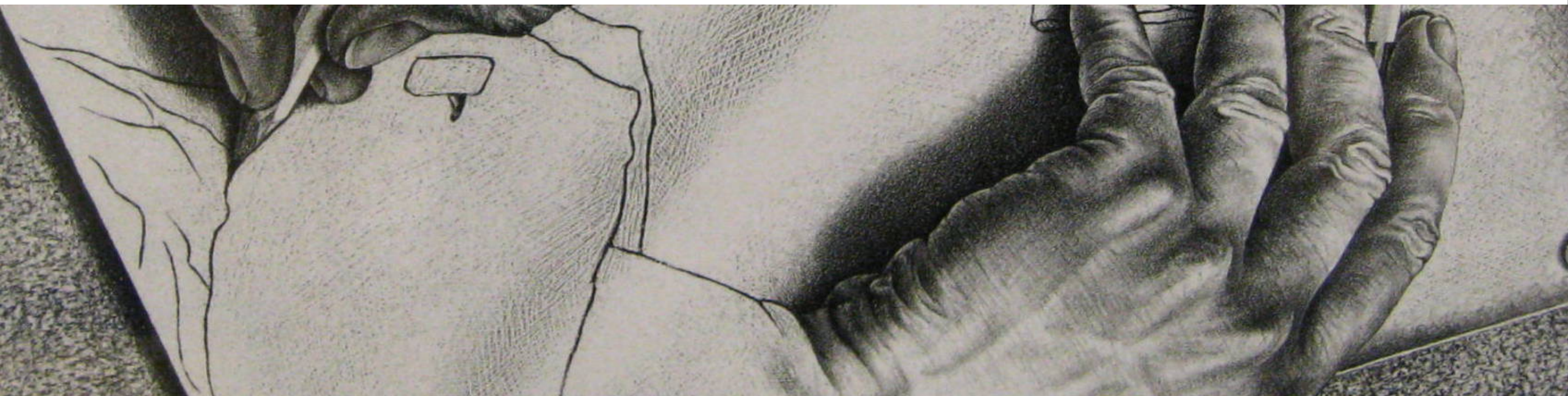
MONOLITHIC



MONOLITHIC

THEN…

LET'S WRITE A NEW VERSION



REQUIREMENTS

SIMPLICITY

SCALABILITY

RESILIENCE


MODULARITY

SAFETY

PREMISSE

- No matter if a **system** is internal or external, it eventually…
  - … goes **offline**…
  - … **crashes**…
  - … or **change their behaviour without notice**.

A NEW ARCHITECTURE

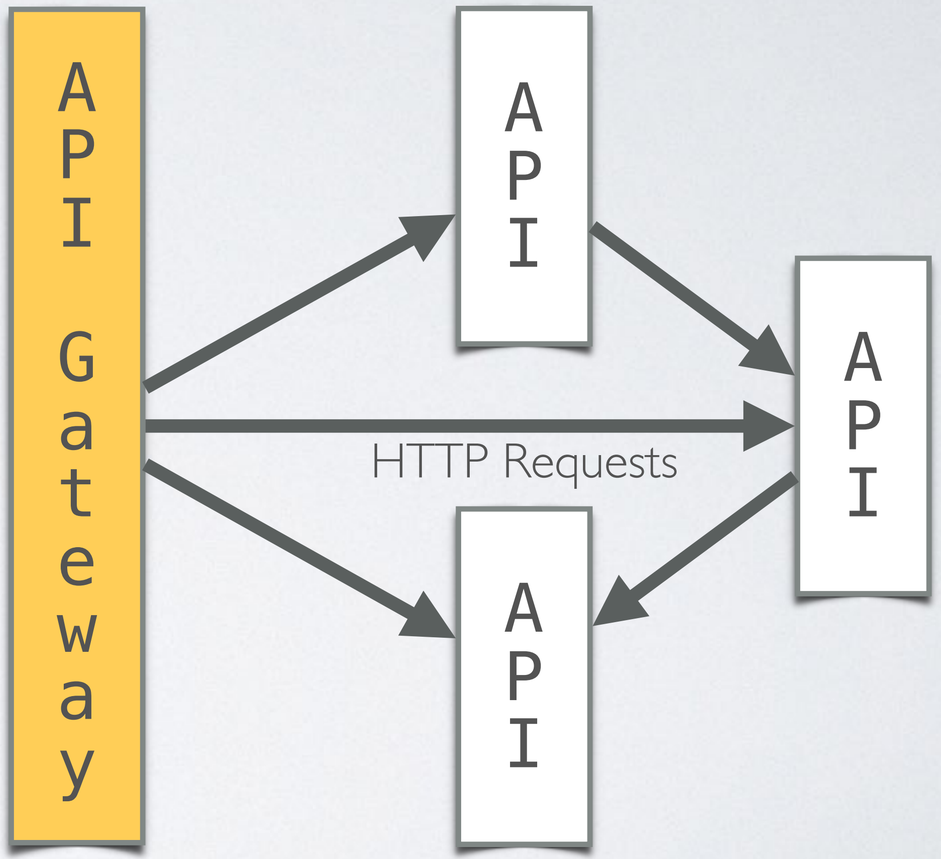
MICROSERVICES

MICROSERVICES or SOA?


I DON'T
CARE

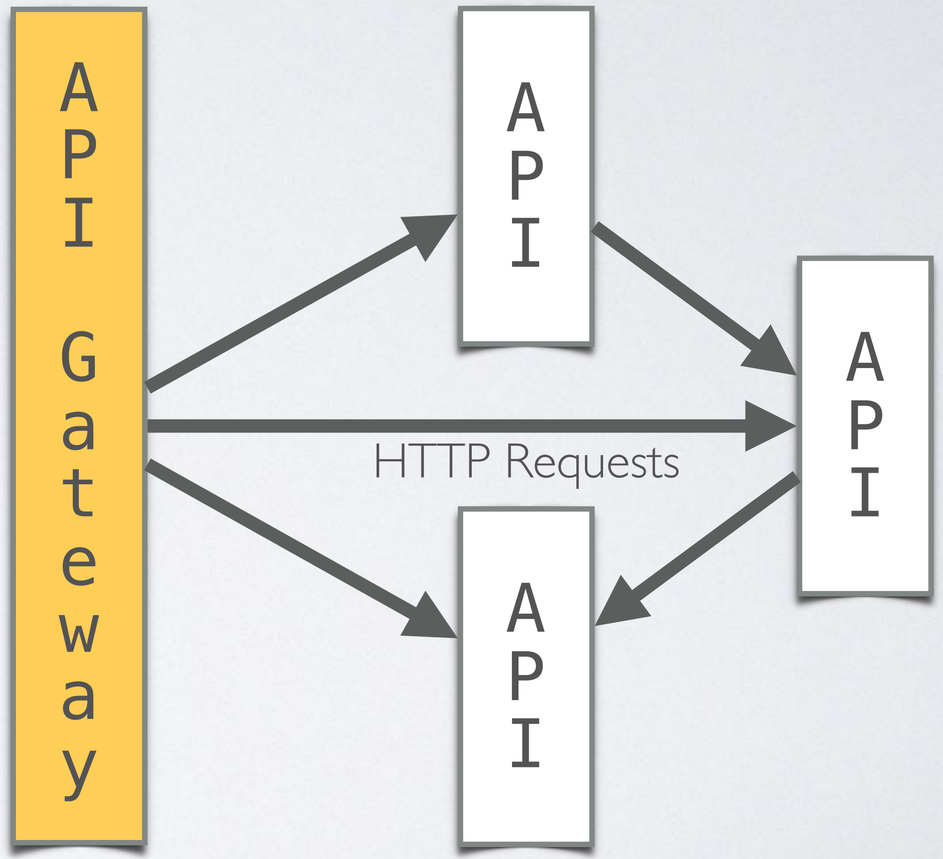
IMPLEMENTATION MODELS

COMMUNICATION VIA API



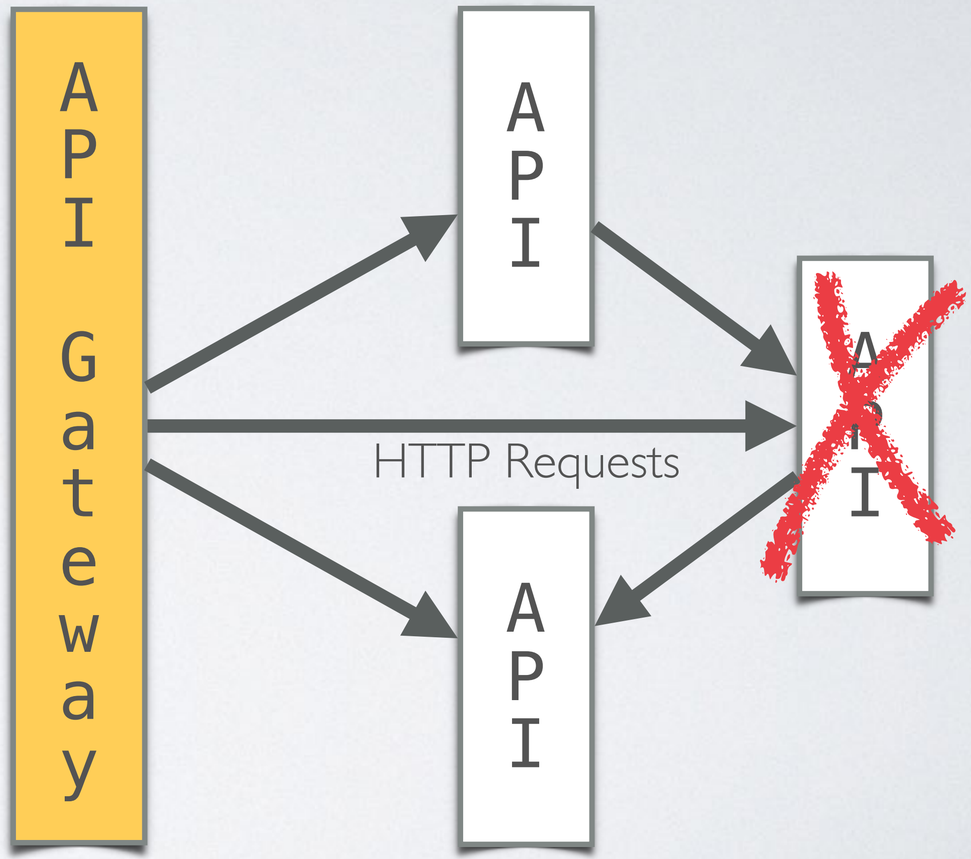
COMMUNICATION VIA API

- RESTful communication between services
- Synchronous Service Calls
- Strict dependency between services



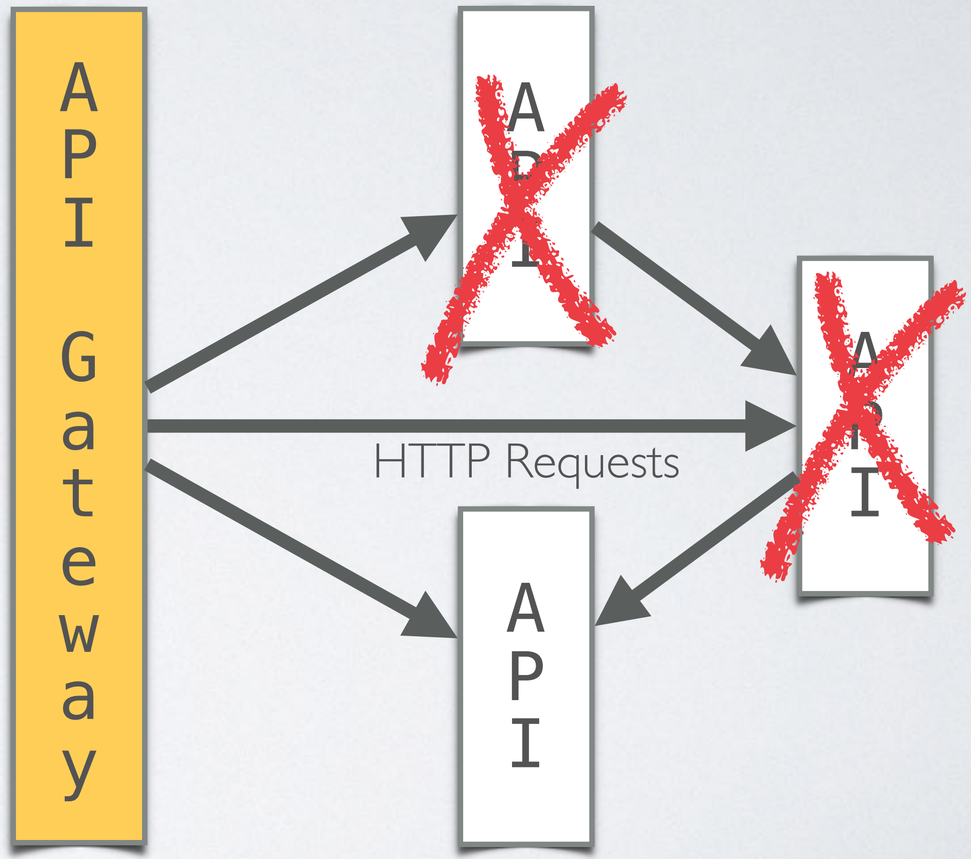
COMMUNICATION VIA API

- RESTful communication between services
- Synchronous Service Calls
- Strict dependency between services



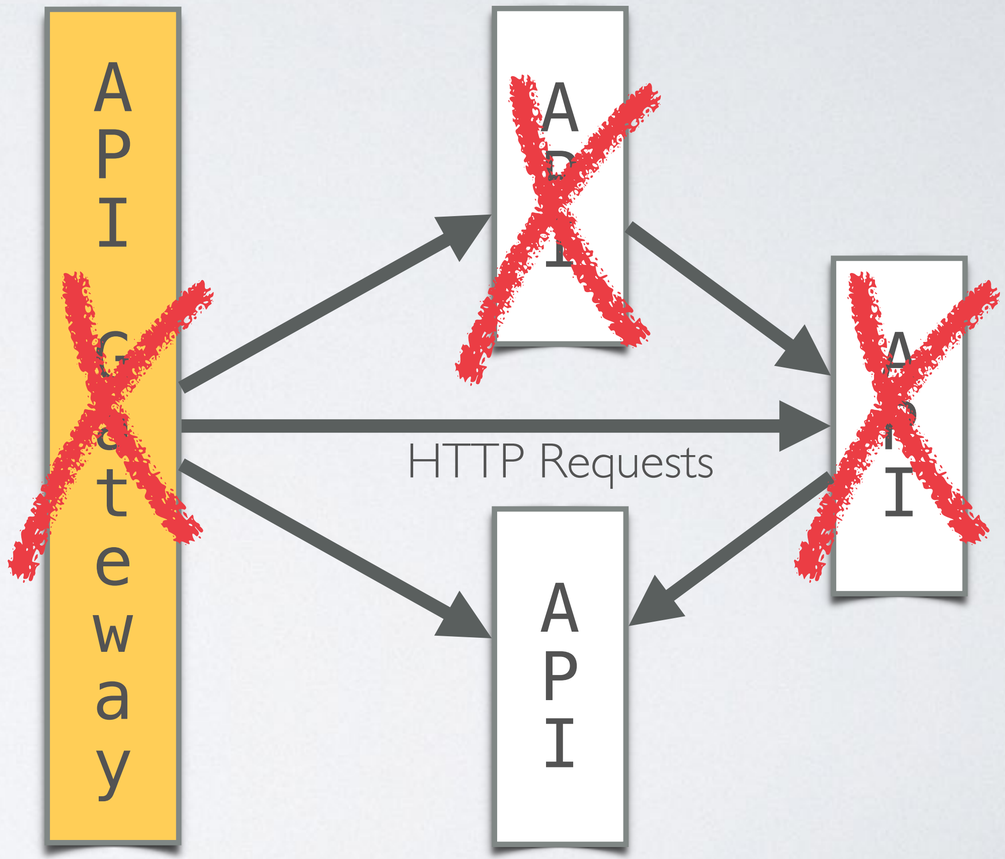
COMMUNICATION VIA API

- RESTful communication between services
- Synchronous Service Calls
- Strict dependency between services



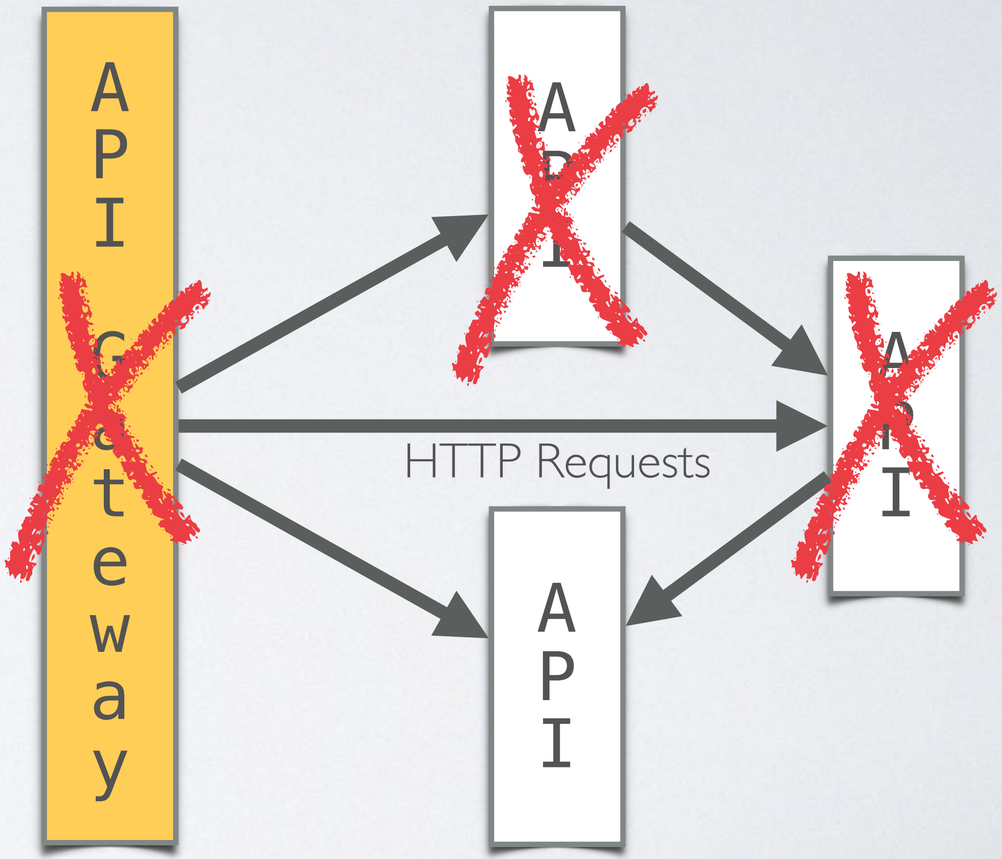
COMMUNICATION VIA API

- RESTful communication between services
- Synchronous Service Calls
- Strict dependency between services

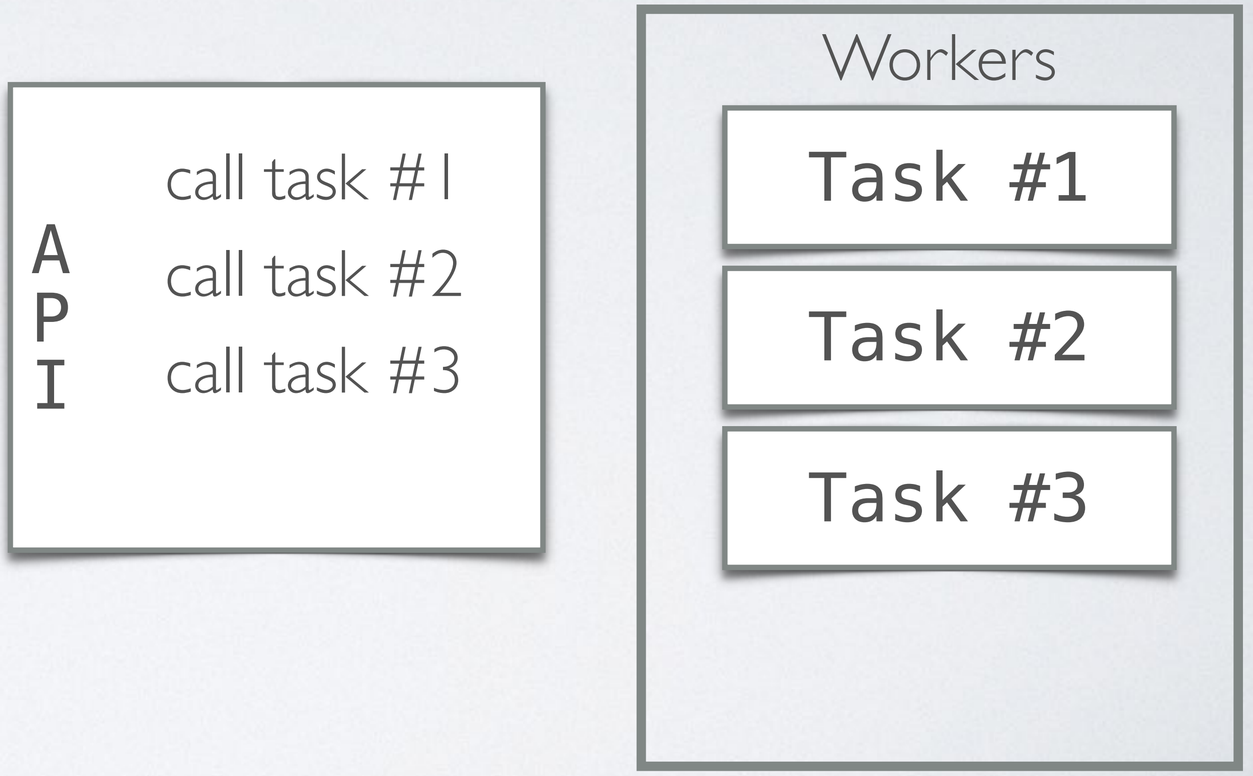


COMMUNICATION VIA API

- RESTful communication between services
- Synchronous Service Calls
- Strict dependency between services
- **No resilience**
- **No safety* (eg. data loss on request failures)**

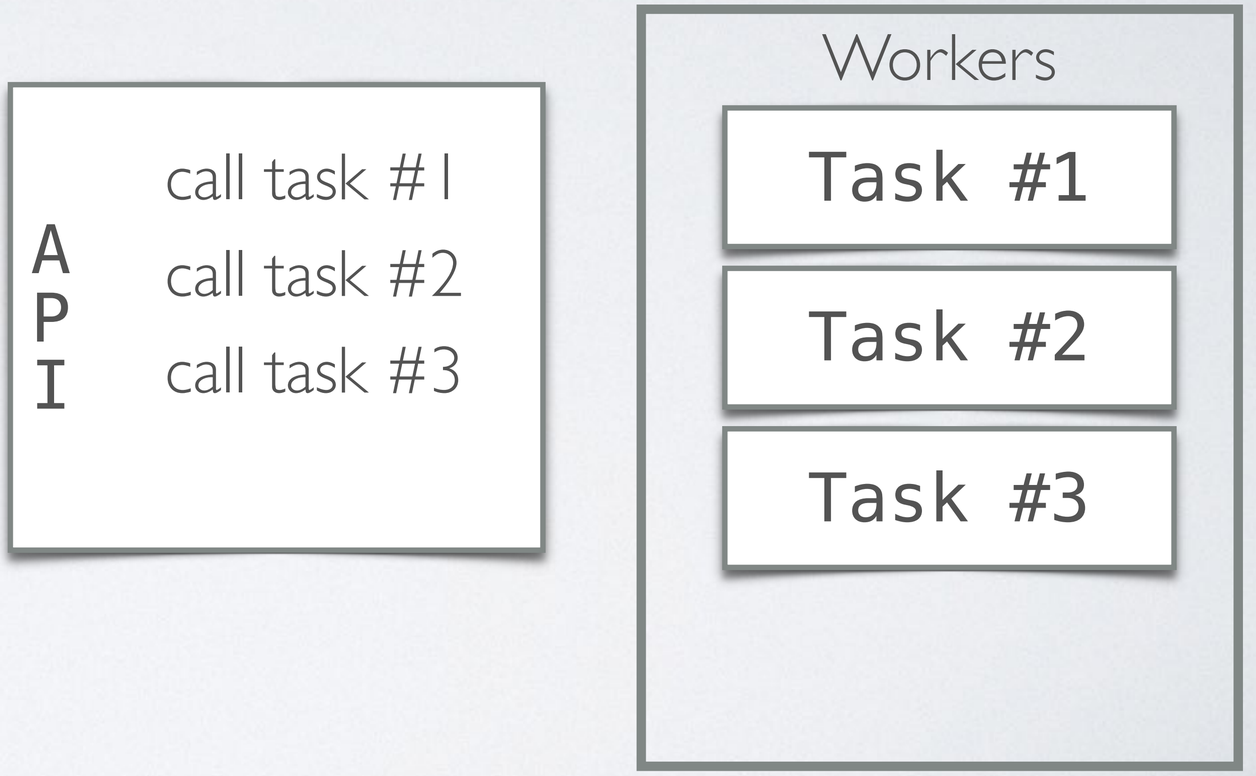


ASYNC TASK EXECUTION



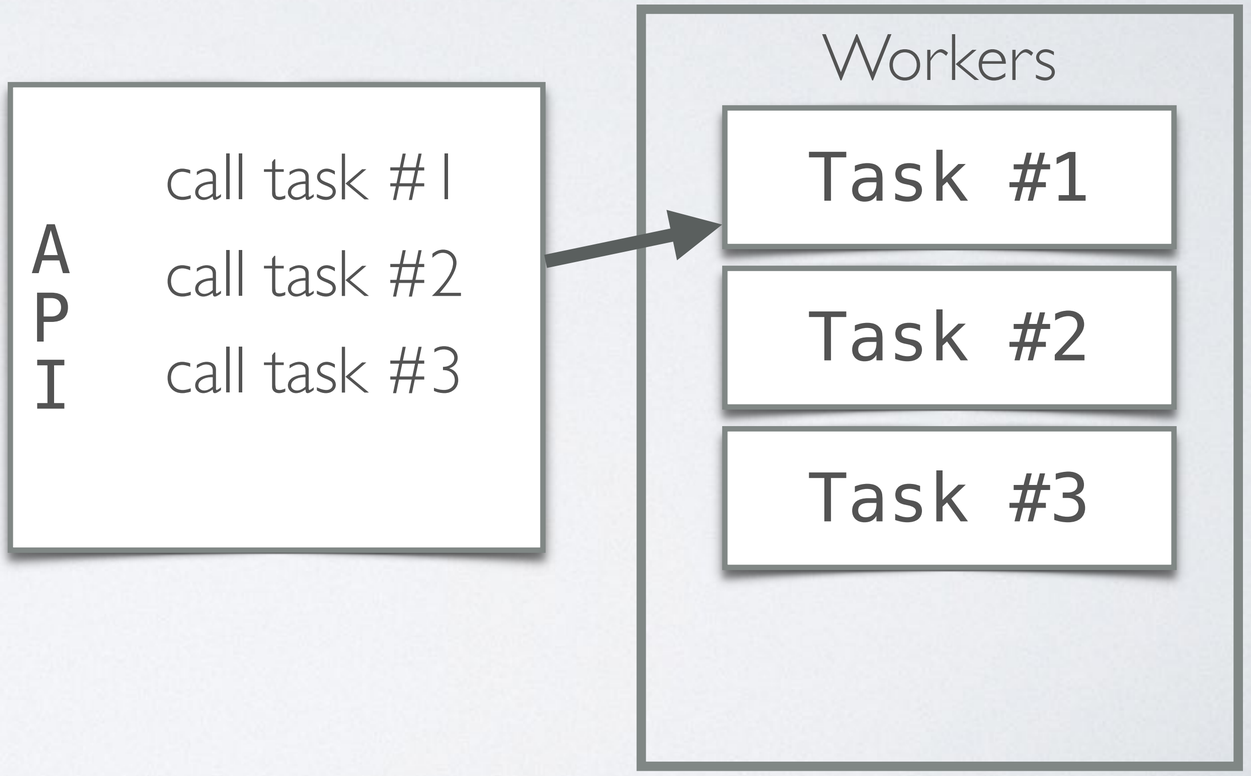
ASYNC TASK EXECUTION

- Asynchronous Procedure Calls
- Queue based task execution
- Client must knows about their dependents (I've to change API every time I need to add a new task)



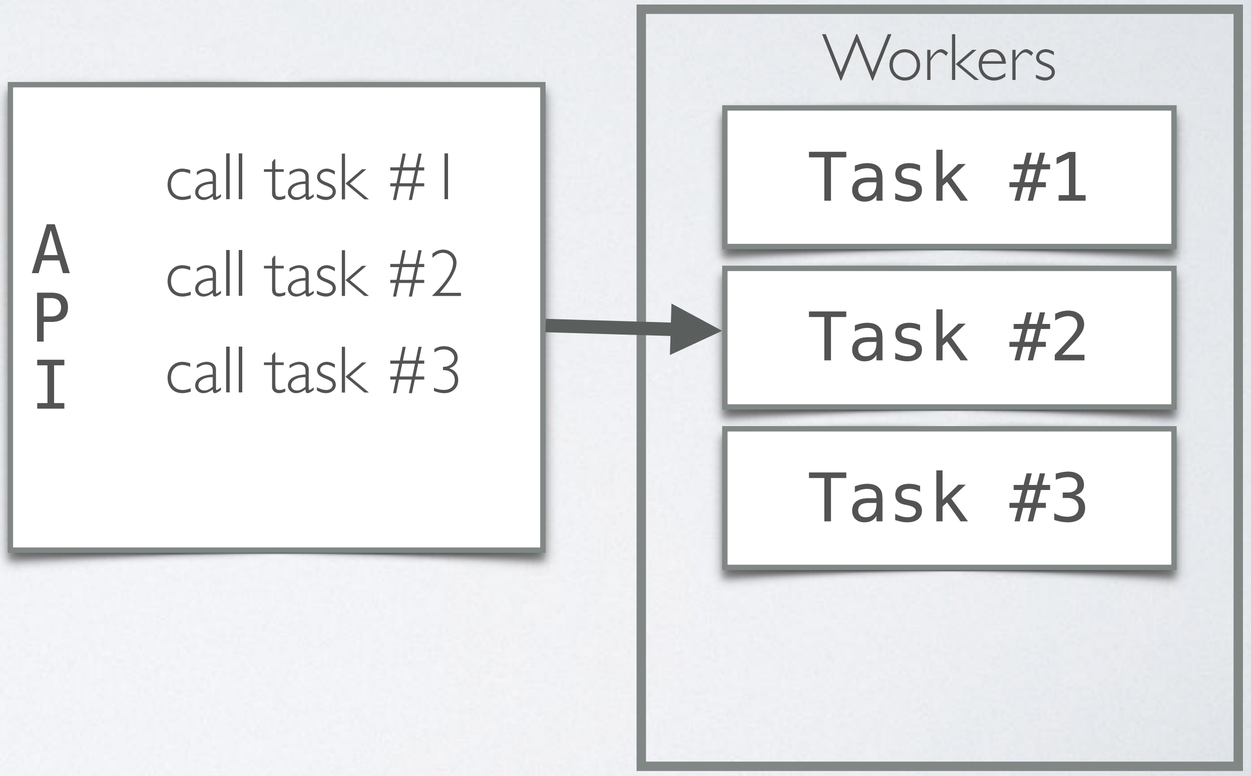
ASYNC TASK EXECUTION

- Asynchronous Procedure Calls
- Queue based task execution
- Client must knows about their dependents (I've to change API every time I need to add a new task)



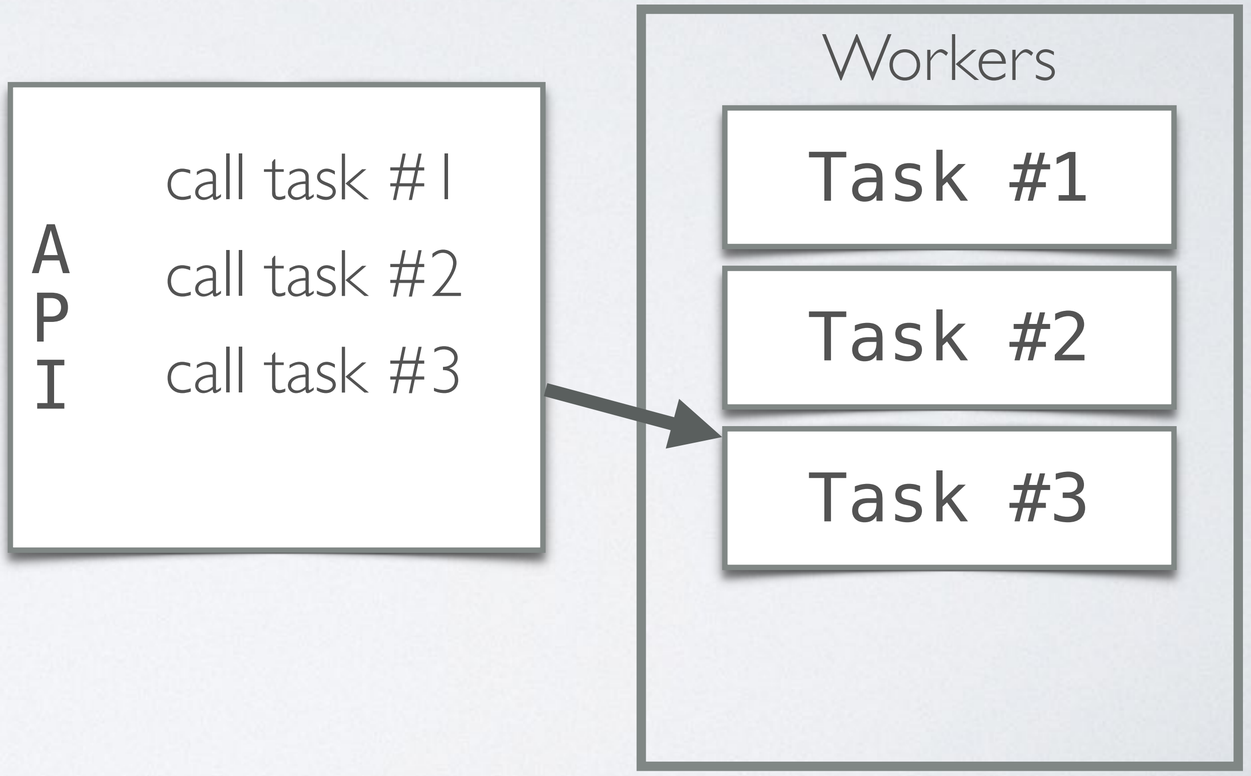
ASYNC TASK EXECUTION

- Asynchronous Procedure Calls
- Queue based task execution
- Client must knows about their dependents (I've to change API every time I need to add a new task)



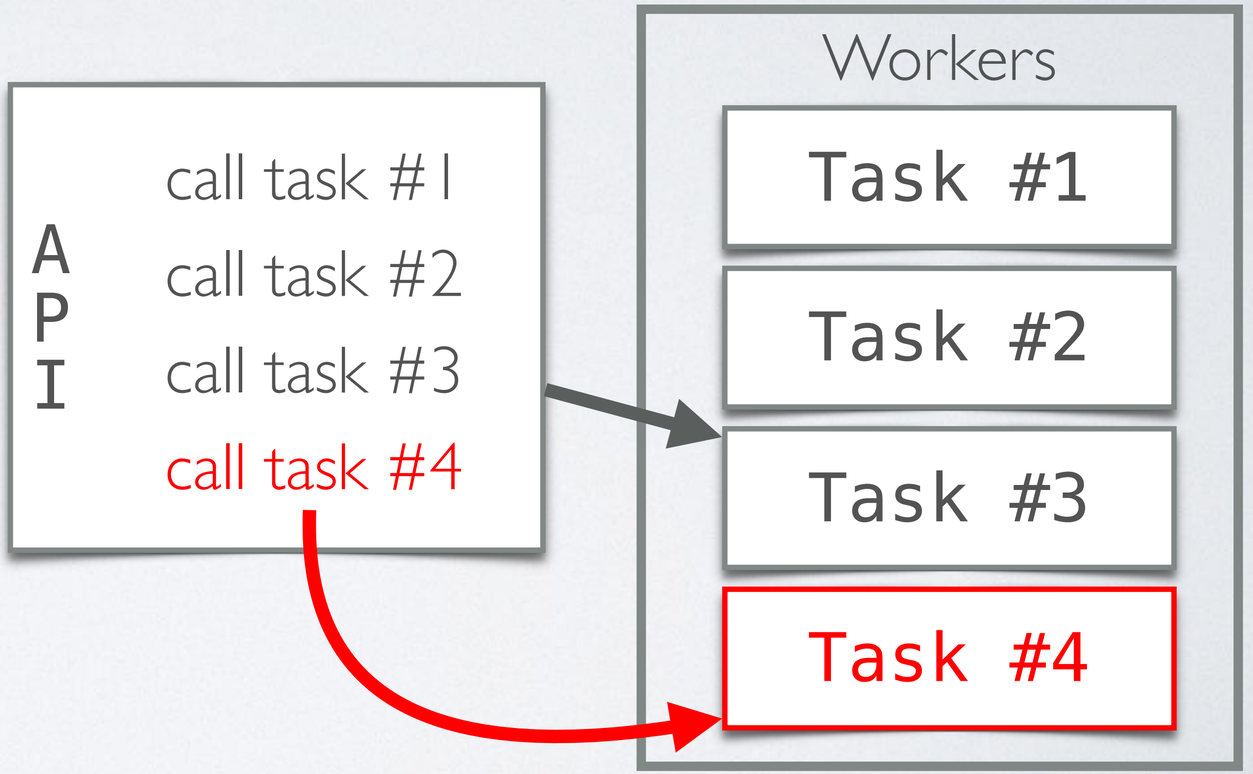
ASYNC TASK EXECUTION

- Asynchronous Procedure Calls
- Queue based task execution
- Client must knows about their dependents (I've to change API every time I need to add a new task)



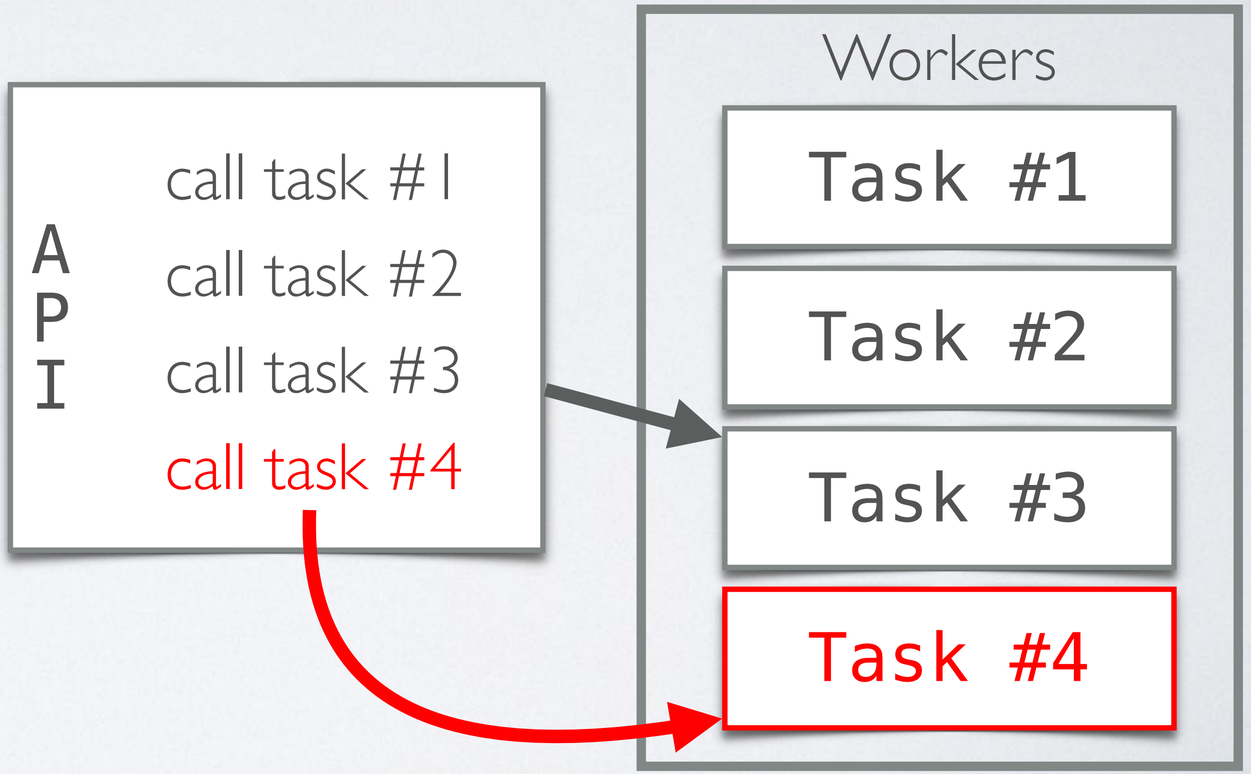
ASYNC TASK EXECUTION

- Asynchronous Procedure Calls
- Queue based task execution
- Client must knows about their dependents (I've to change API every time I need to add a new task)

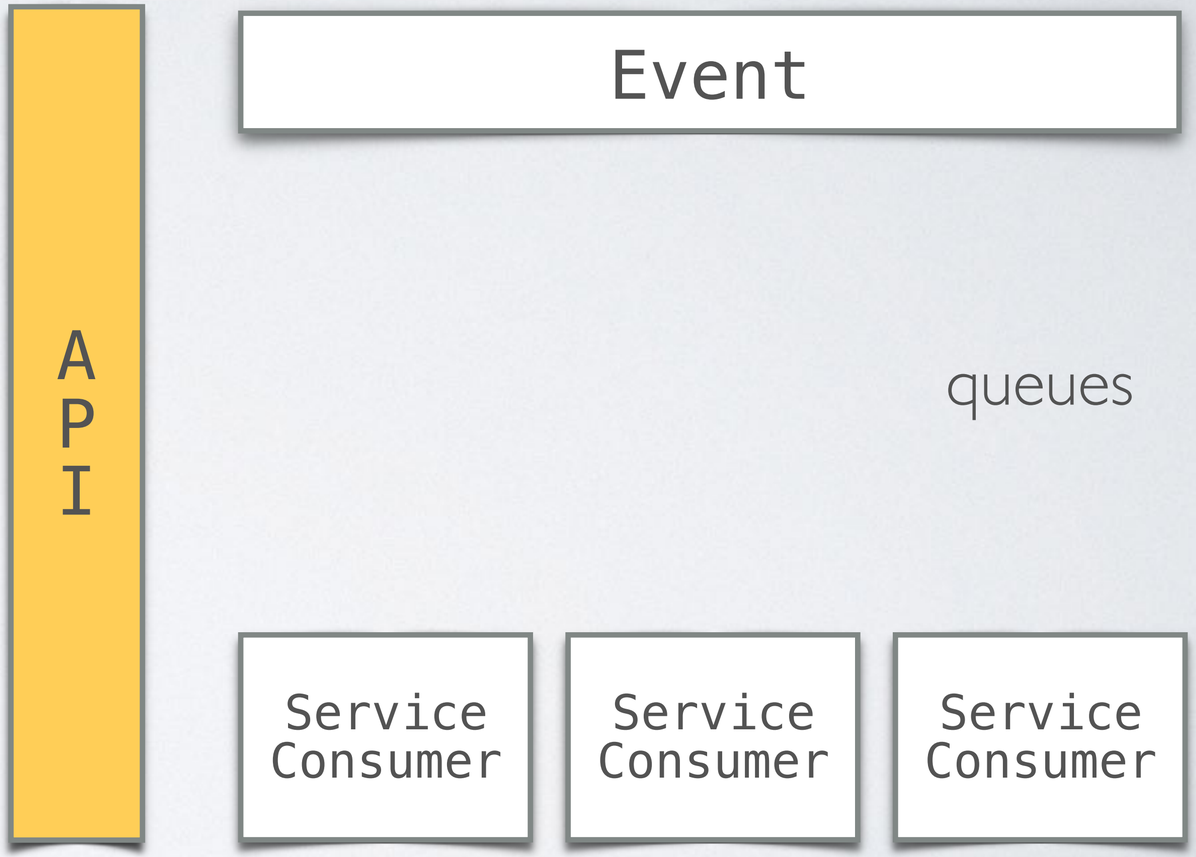


ASYNC TASK EXECUTION

- Asynchronous Procedure Calls
- Queue based task execution
- Client must knows about their dependents (I've to change API every time I need to add a new task)
- **No decoupling ➡ No modularity**
- **No safety* (eg. data loss on deployment failures)**

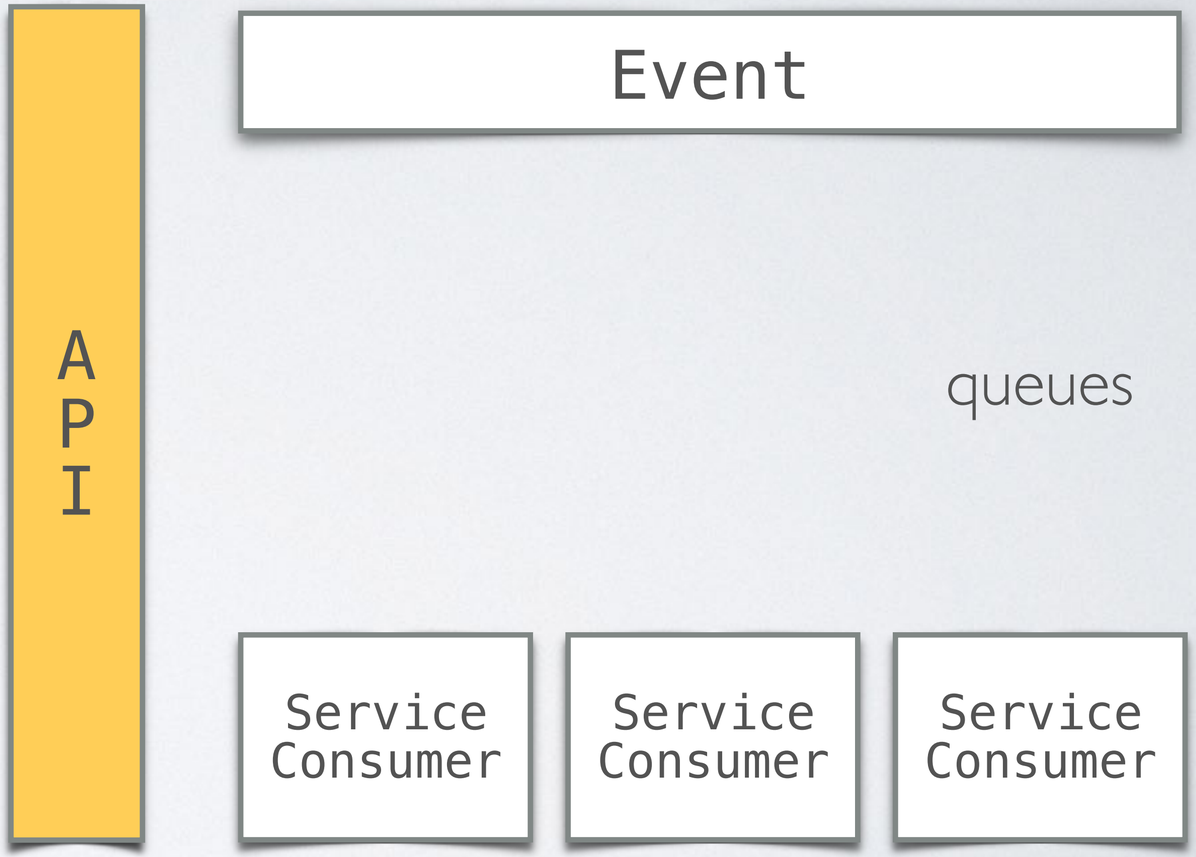


MESSAGE EVENT TRIGGERING



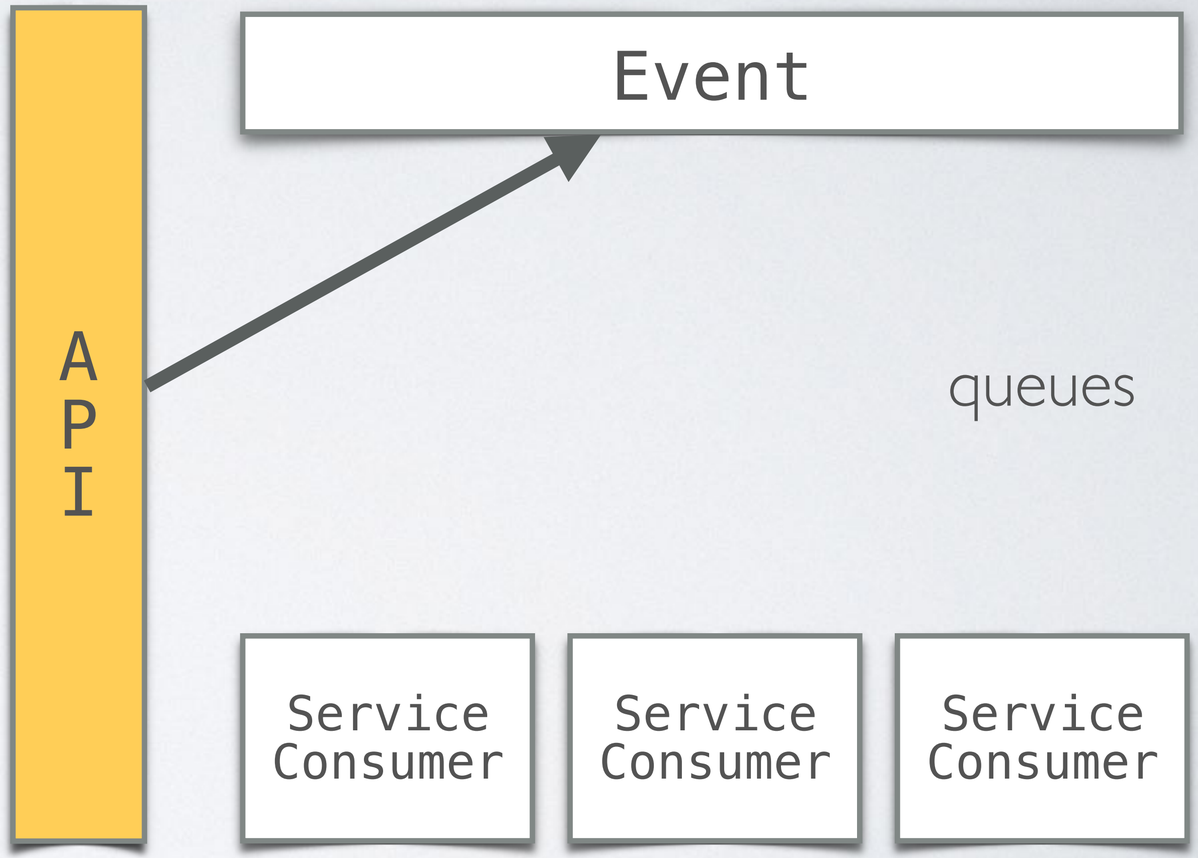
MESSAGE EVENT TRIGGERING

- Action Event triggering
- Queue based message event handling
- Event as messages



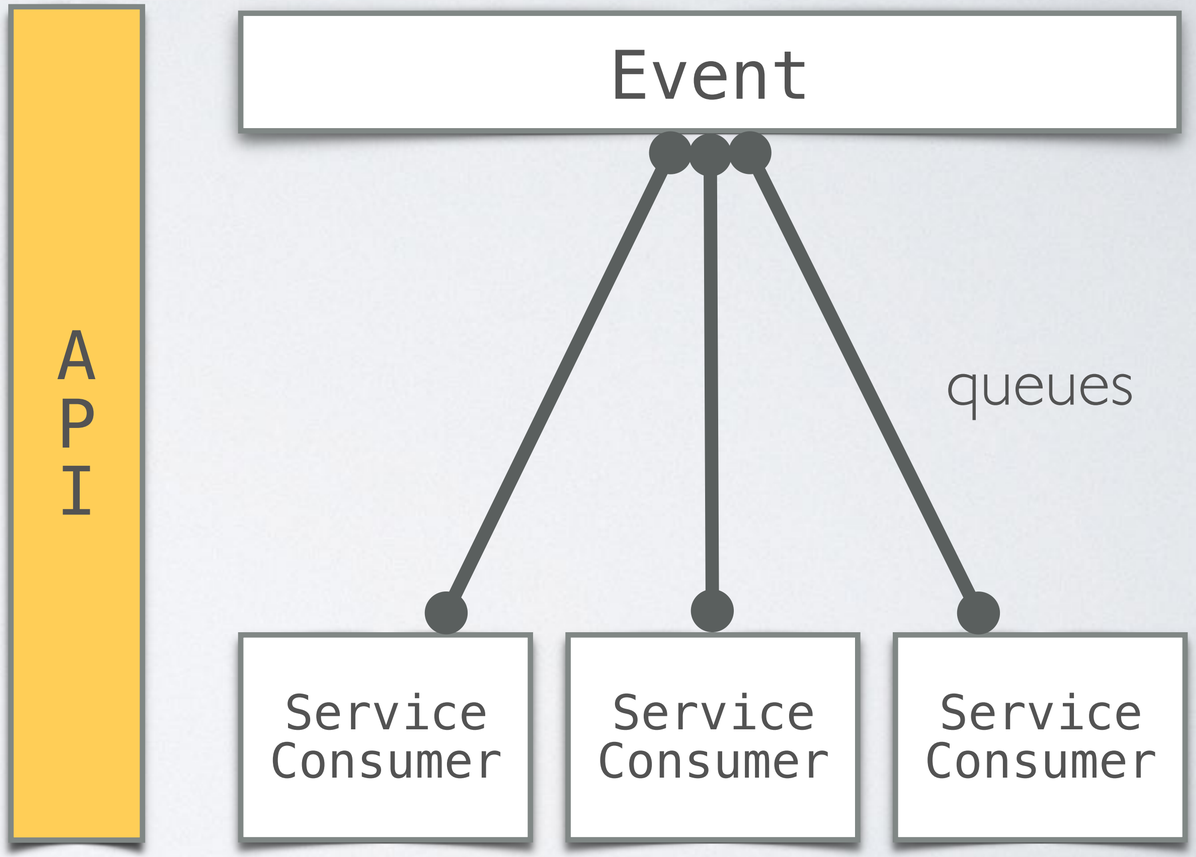
MESSAGE EVENT TRIGGERING

- Action Event triggering
- Queue based message event handling
- Event as messages



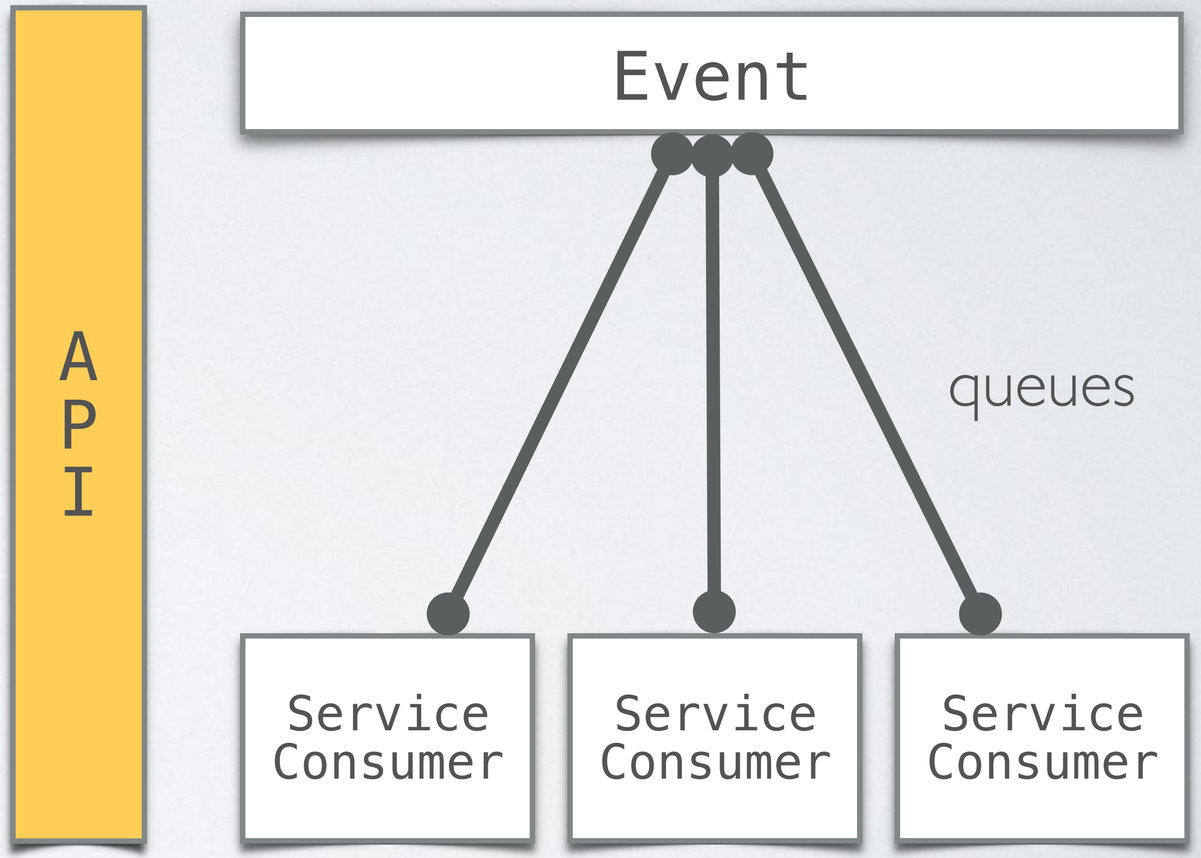
MESSAGE EVENT TRIGGERING

- Action Event triggering
- Queue based message event handling
- Event as messages



MESSAGE EVENT TRIGGERING

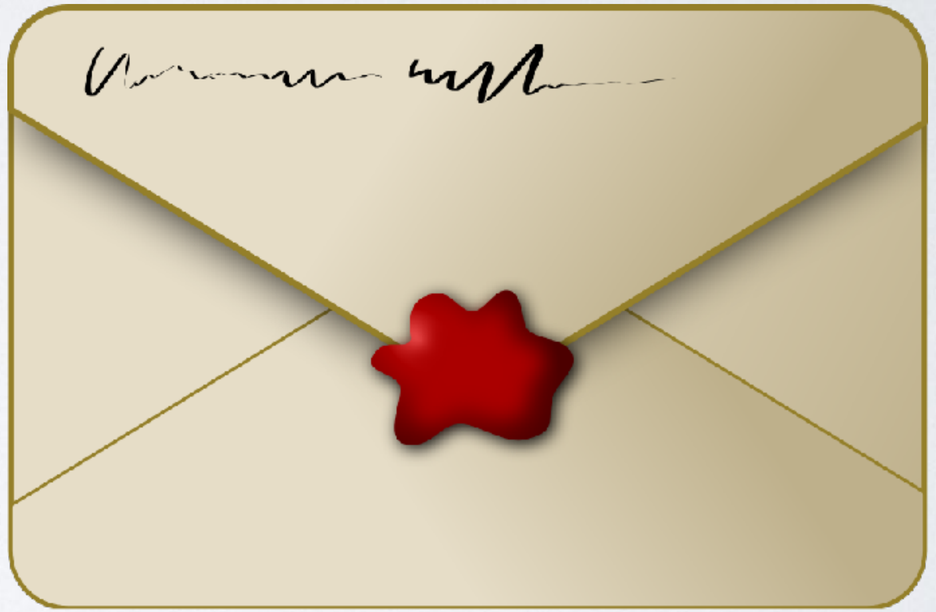
- Action Event triggering
- Queue based message event handling
- Event as messages
- **It works!**



BASIC **BUILDING BLOCKS**

MESSAGES

- Also known as Resource in REST context
- Follow a contract (schema)
- Can be enveloped with metadata (eg. SNS/SQS metadata)

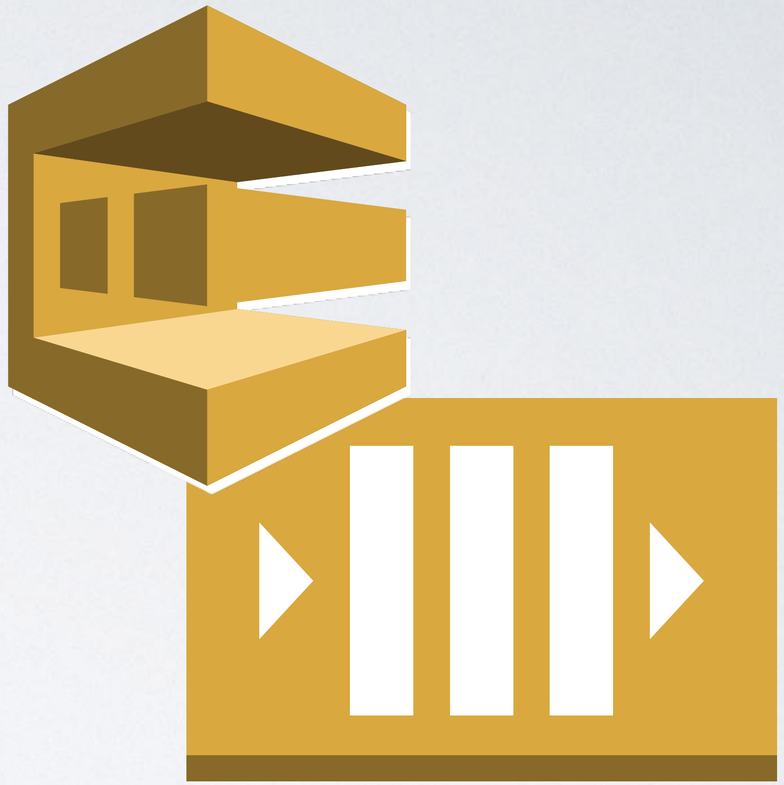


GLOBAL TOPICS

- Publisher in PubSub Pattern
- Global topics for *message* publication
- Topics belong to the system (or architecture) and not to a (micro)service
- We use AWS SNS
  - Kafka was not available on our hosting provider (Heroku) at the time we choose SNS

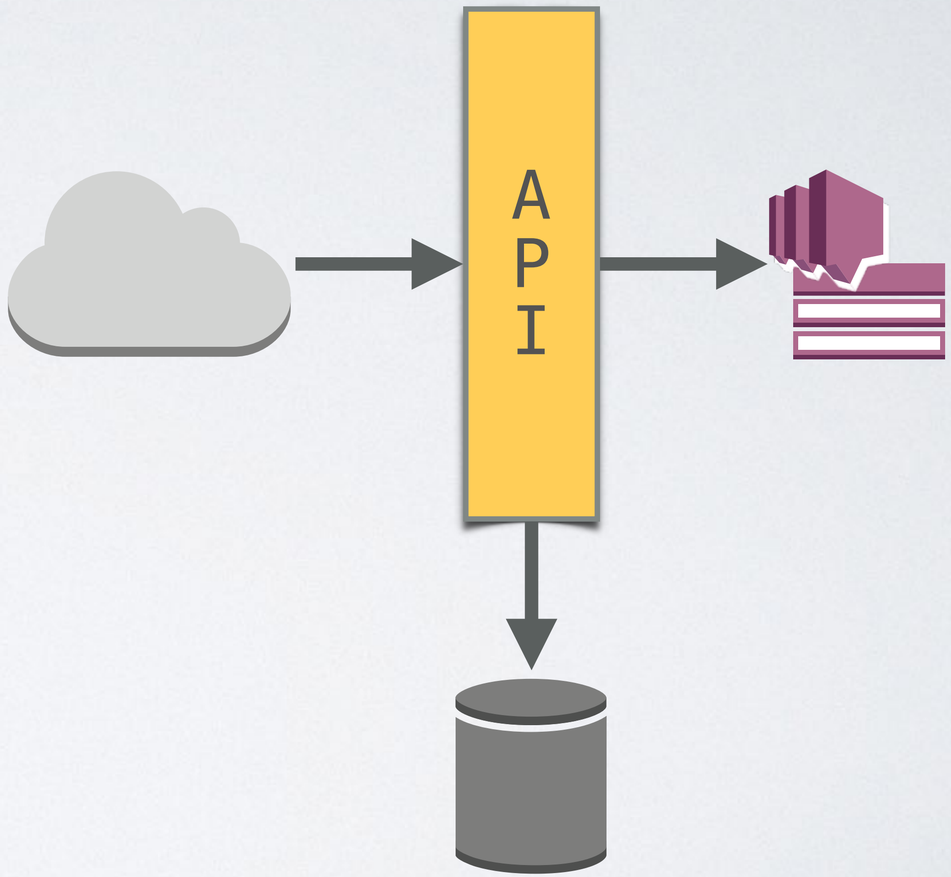
SERVICE QUEUES

- Subscribers in PubSub Pattern
- Queues subscribe topics
- One queue belong exclusively to one (micro)service
- We use AWS SQS
  - SQS can be used as a SNS subscriber



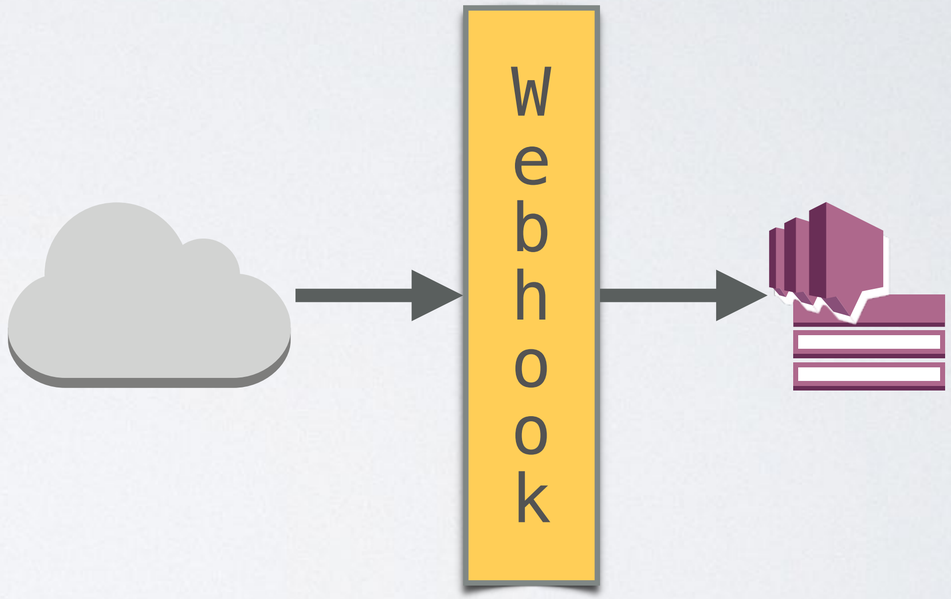
API

- Main Data Entry Point
- Data Validation
- Data Persistence
- Workflow Management (eg. status and state machine)
- Event Triggering
- Idempotency handling (eg. discard duplicated requests returning a HTTP 304 Not Modified)
- Python 3, Django REST Framework



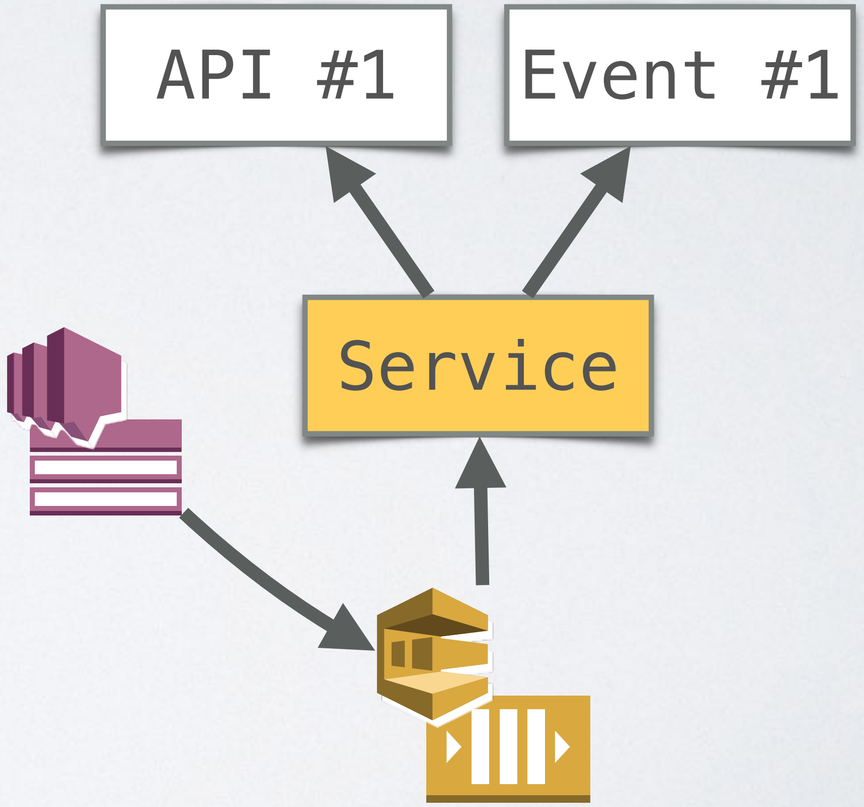
WEBHOOK

- Data Entry Point
- No Data Persistence
- Proxy HTTP ➡ SNS
- Event Triggering
- Python 3, Django REST Framework



SERVICE

- Event Handling / Message Processing
- Business Logic
- Some service could trigger some events but it's not so usual
- Service Types:
  - **Dequeuer** - process messages from one queue
  - **Broker** - process messages from multiple queues
- Python 3, Loafer / Asyncio

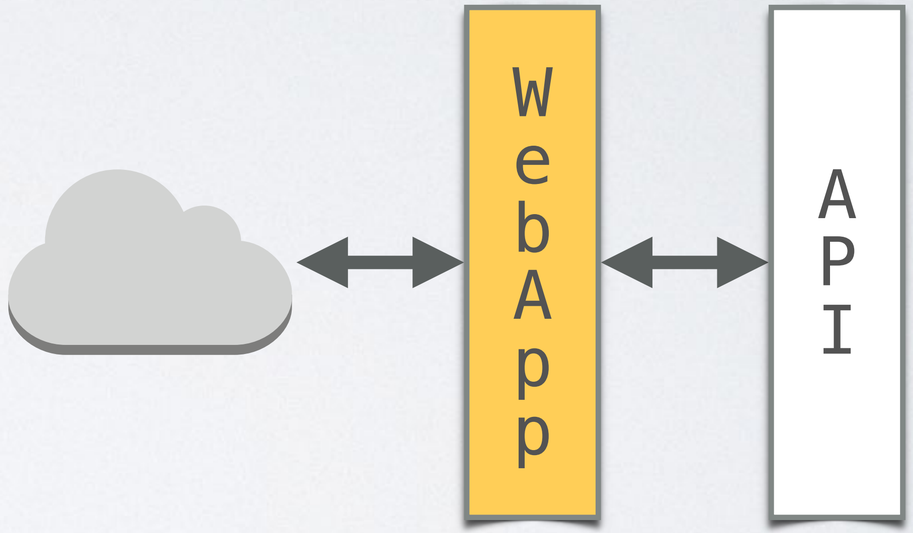


JOB

- Scheduled Job
- No Persistence
- Event Triggering
- Python 3, etc.

CLIENT

- Web or Mobile Applications
- No Persistence (or basic persistence)
- Web presentation of APIs' resources
- Python 3, Django



LIBRARIES

- Common Libraries — common utilities for APIs and Services (eg. event triggering/topic publishing)
- Client Libraries — libraries to connect our APIs
- Auth Library — library to handle authentication basics
- Open Source Libraries — useful libraries for community (eg. correios)

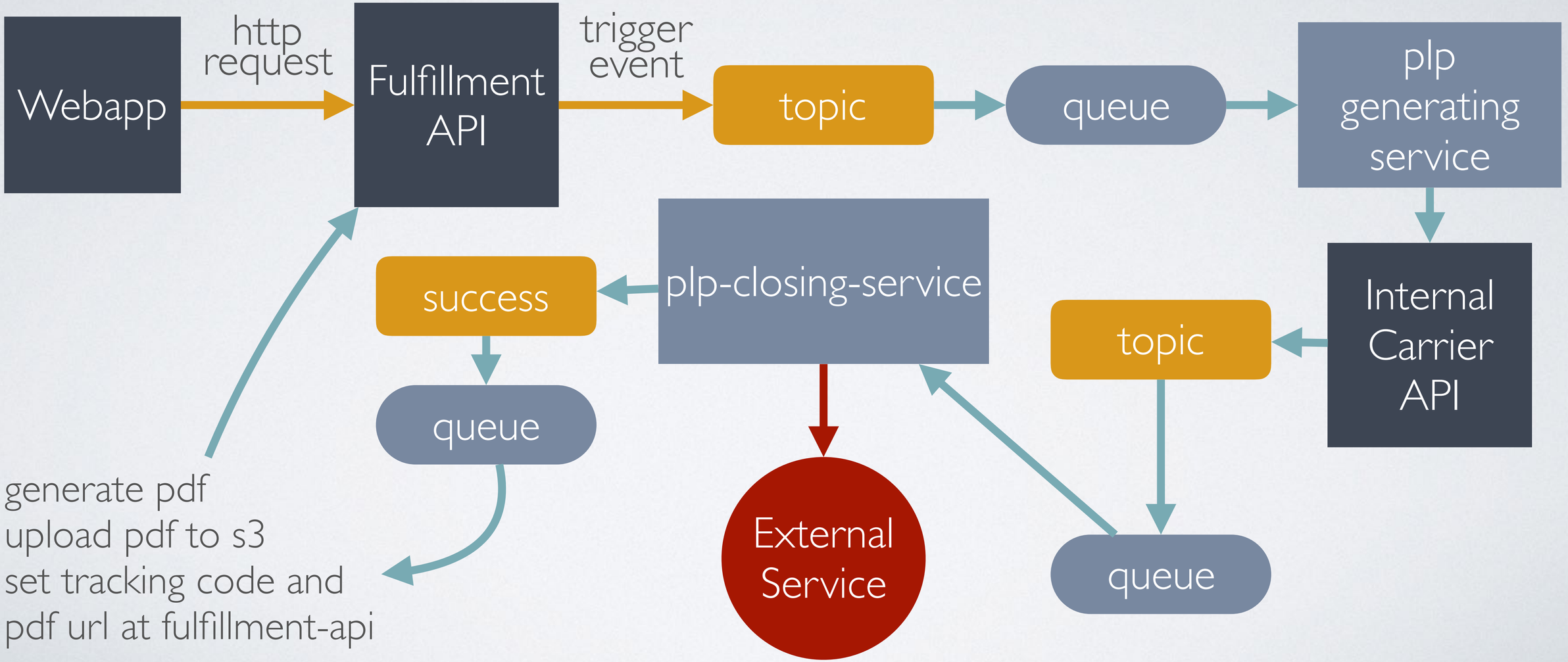
TOOLS

- Legacy Data Migrator — tool that connects in all databases and provides a small framework for data migration. It uses Kenneth Reiz's records library. It was initially used to migrate data from the old version of our application.

- Toolbelt — tool that provide basic management commands to interact with our APIs, partner APIs and to make ease to manage SQS queues or trigger some events in SNS Topics.

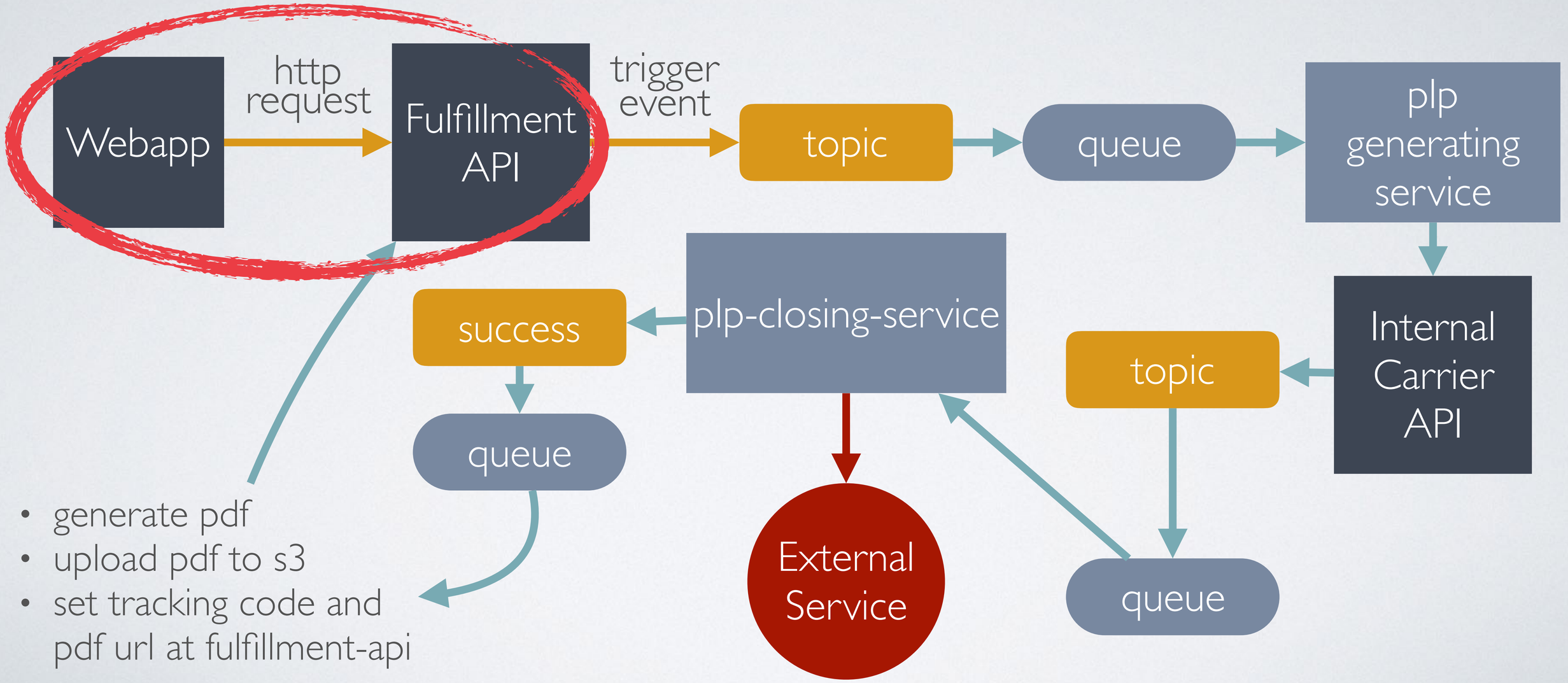
WORKING SAMPLE

posting list and shipping label generation



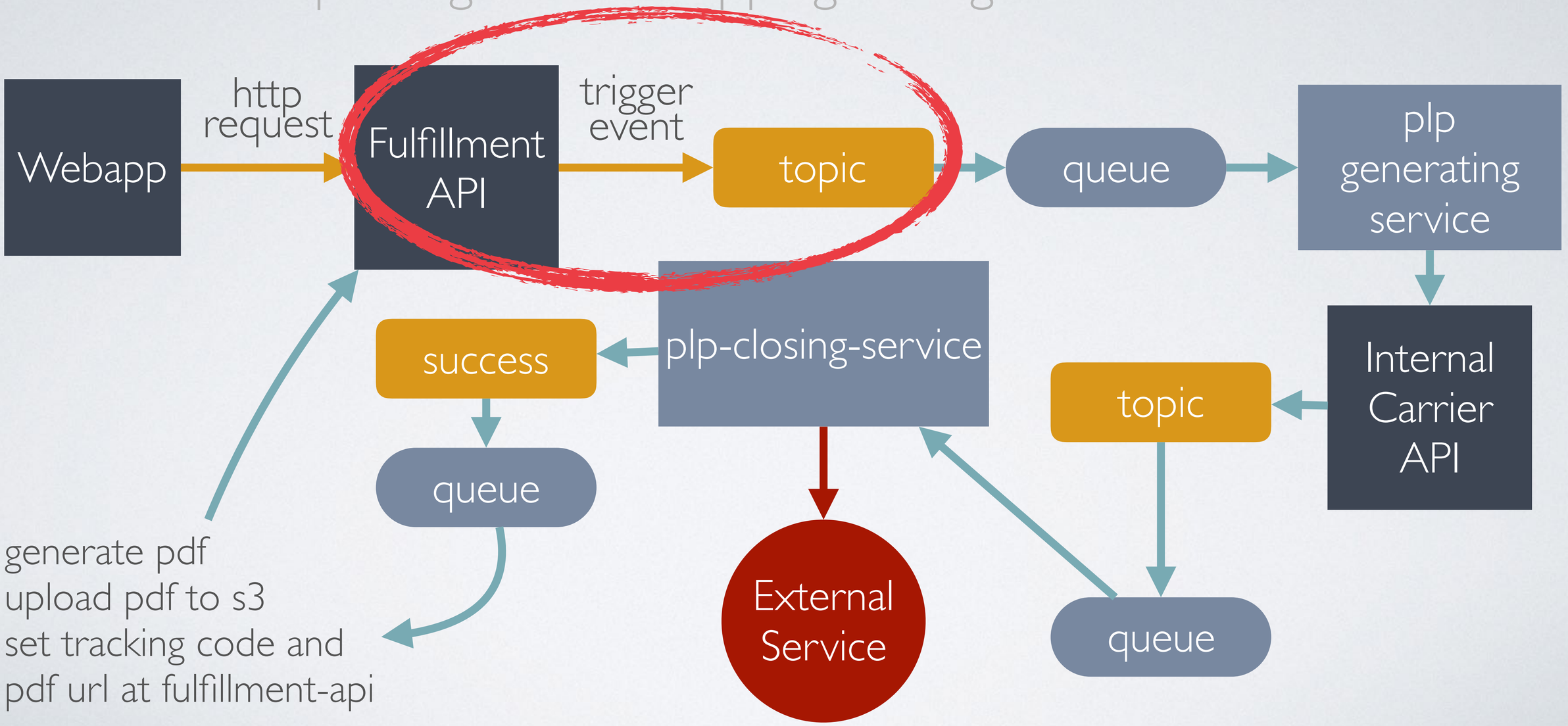
WORKING SAMPLE

posting list and shipping label generation



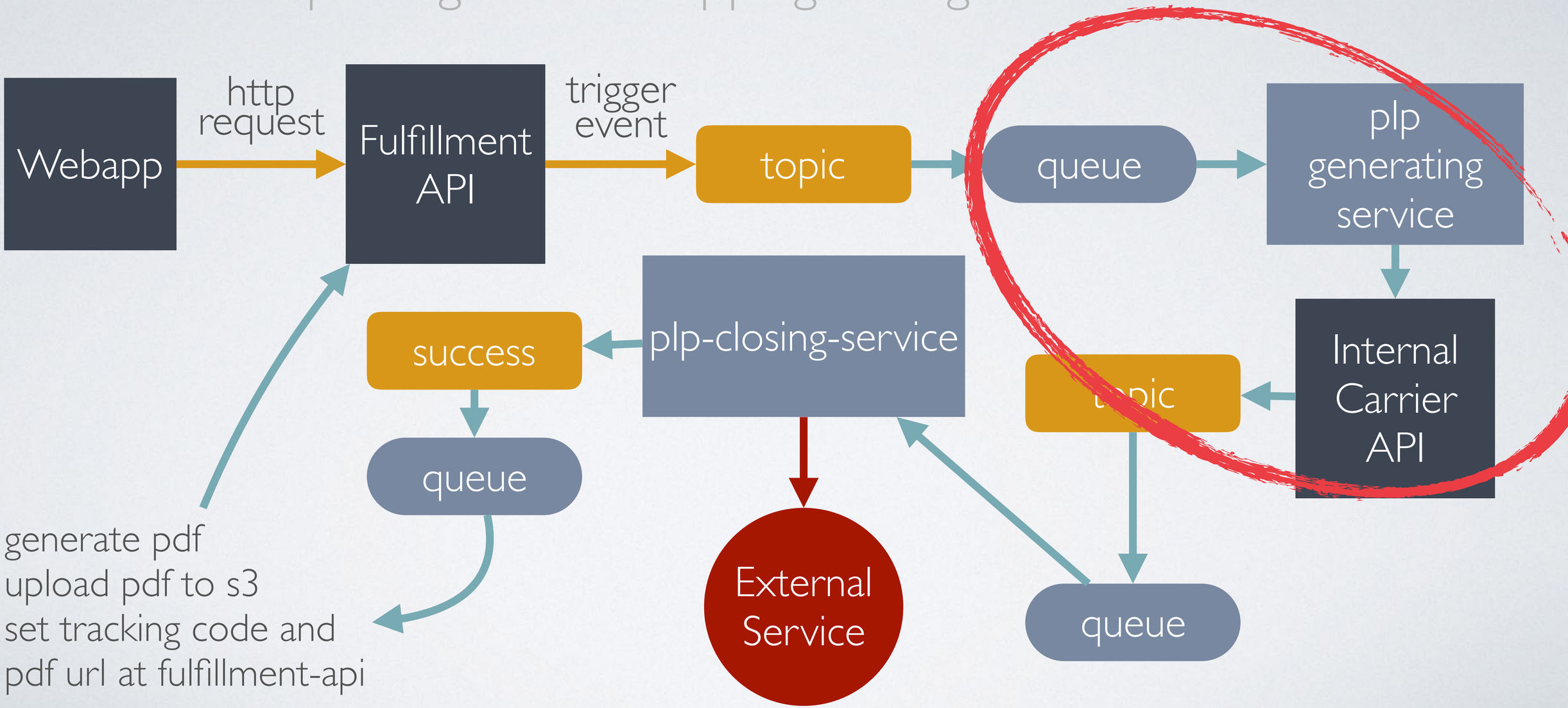
WORKING SAMPLE

posting list and shipping label generation



WORKING SAMPLE

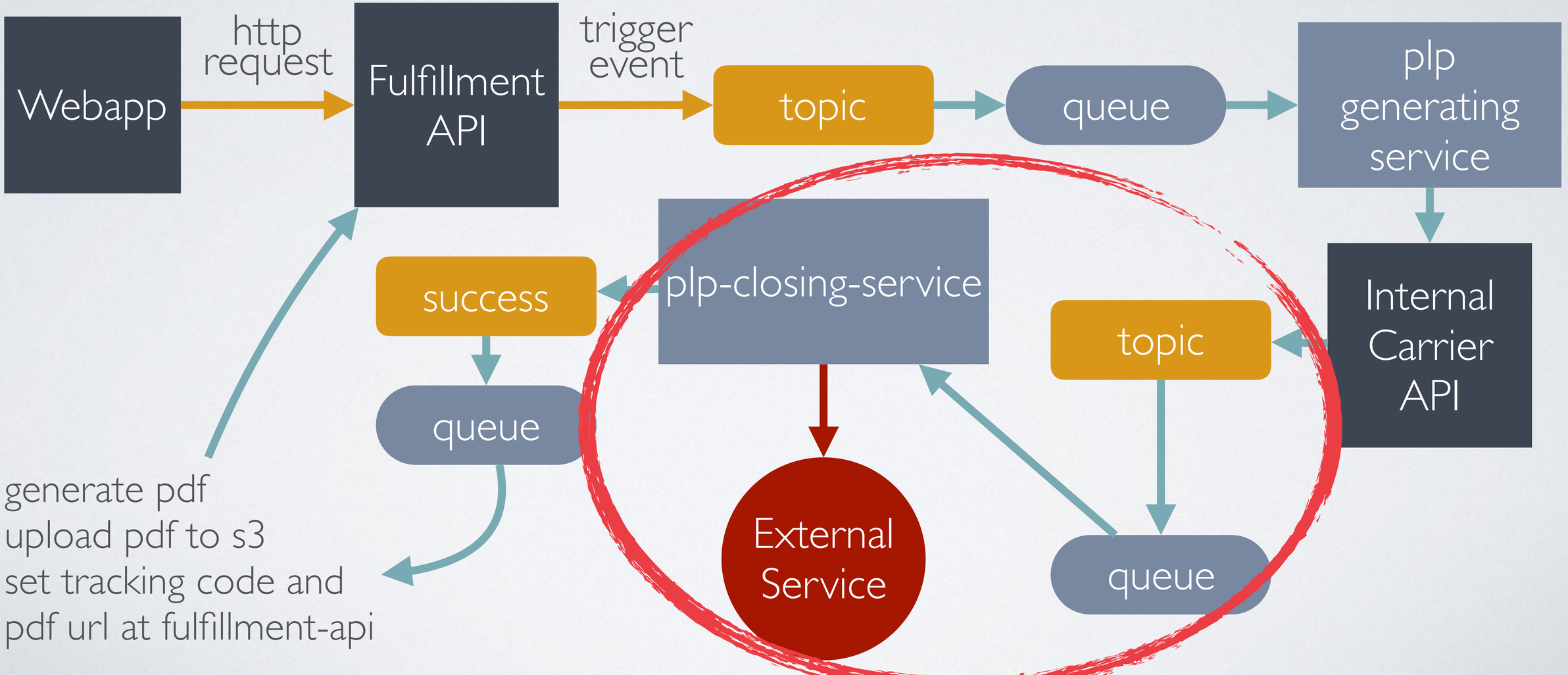
posting list and shipping label generation



- generate pdf
- upload pdf to s3
- set tracking code and pdf url at fulfillment-api

WORKING SAMPLE

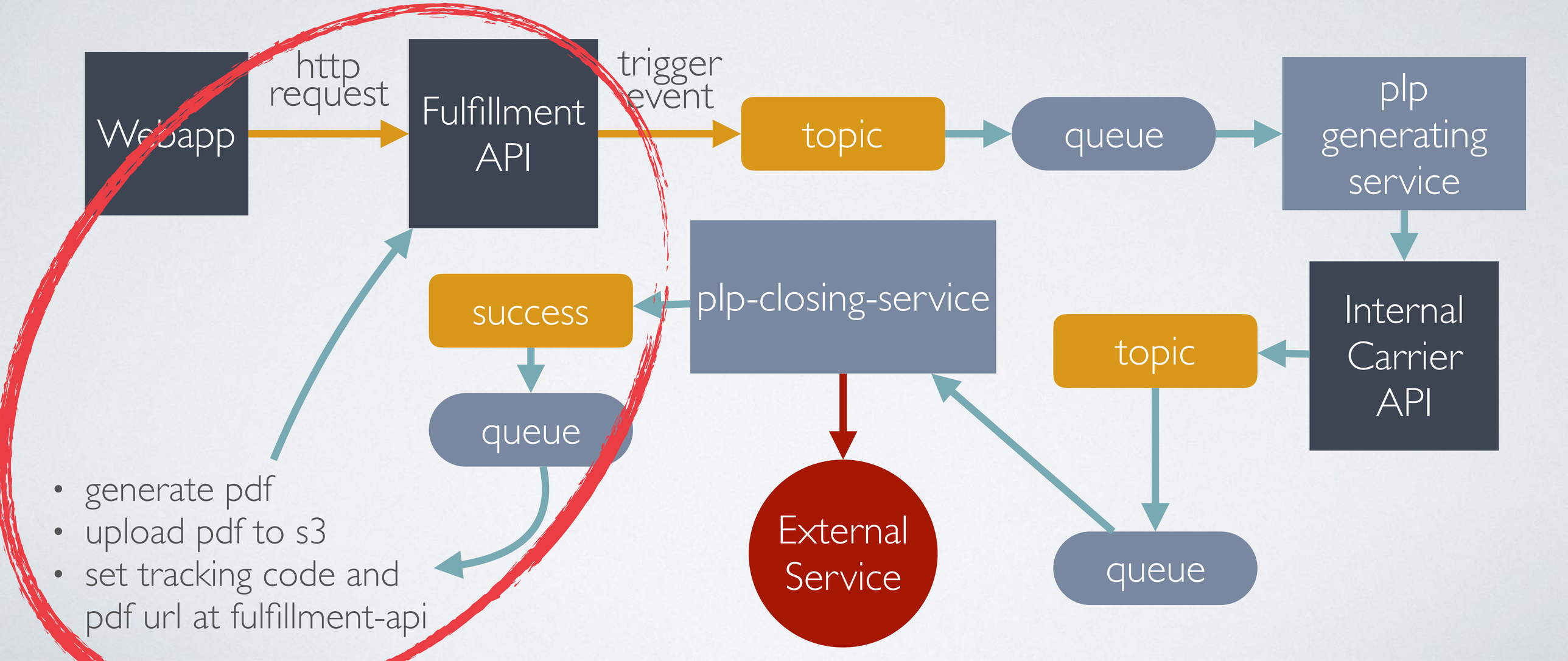
posting list and shipping label generation



- generate pdf
- upload pdf to s3
- set tracking code and pdf url at fulfillment-api

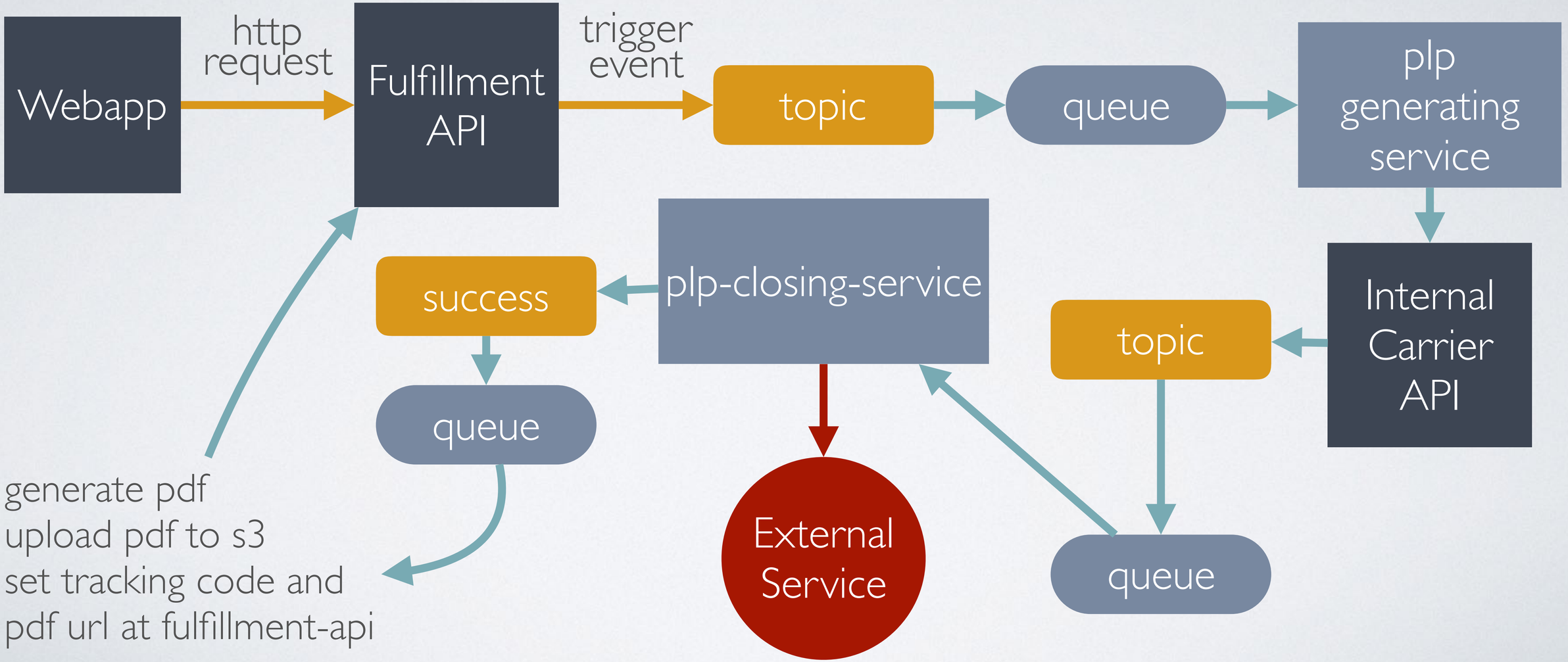
WORKING SAMPLE

posting list and shipping label generation



WORKING SAMPLE

posting list and shipping label generation



CHALLENGES

- It's hard to make evolution of message/resource contracts between services
- All sequential process must be splitted over multiple (small) services
- Denormalization of data can easily lead to problems of data consistency if we do not take certain precautions
  - Information needed for one API must be replicated through services and stored locally
  - Data migration or refactoring in several services requires the development of an specific application

DEVELOPMENT

REMOTE TEAM

DEVELOPMENT TOOLS

- Github - code management
- CircleCI - continuous integration
- vim / PyCharm / SublimeText / etc - development

COMMUNICATION TOOLS

- JIRA - project management
- Confluence - documentation
- Google Apps - Mail, Calendar, Docs
- Slack - chat and monitoring integrations
- Mumble - voice conference
- tmux, ngrok, vim (and mumble) - pair programming

DEPLOYMENT

HEROKU

- Hosted on Heroku PaaS (our secret weapon!)
  - Easy deployment, configuration management, log handling, etc
- Heroku PostgreSQL Database
  - Easy deployment, easy configuration and setup, easy backup, replica and foreign data wrappers
- Other services and tools
  - Logentries, Sentry and New Relic

PERGUNTAS?

https://engineering.olist.com/

ESTAMOS CONTRATANDO!

https://bit.ly/olist-webdev
https://bit.ly/olist-newdev