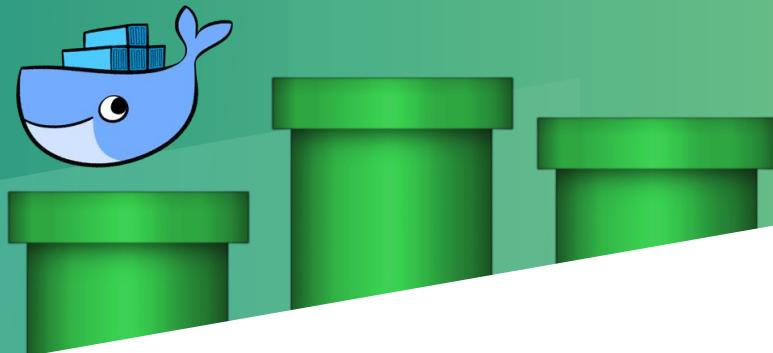


Crea un pipeline con Docker y Luigi



Rodolfo Ferro

<https://rodolfoferro.xyz>

¡Hola!



RODOLFO FERRO PEREZ

Soy Rodo Ferro.

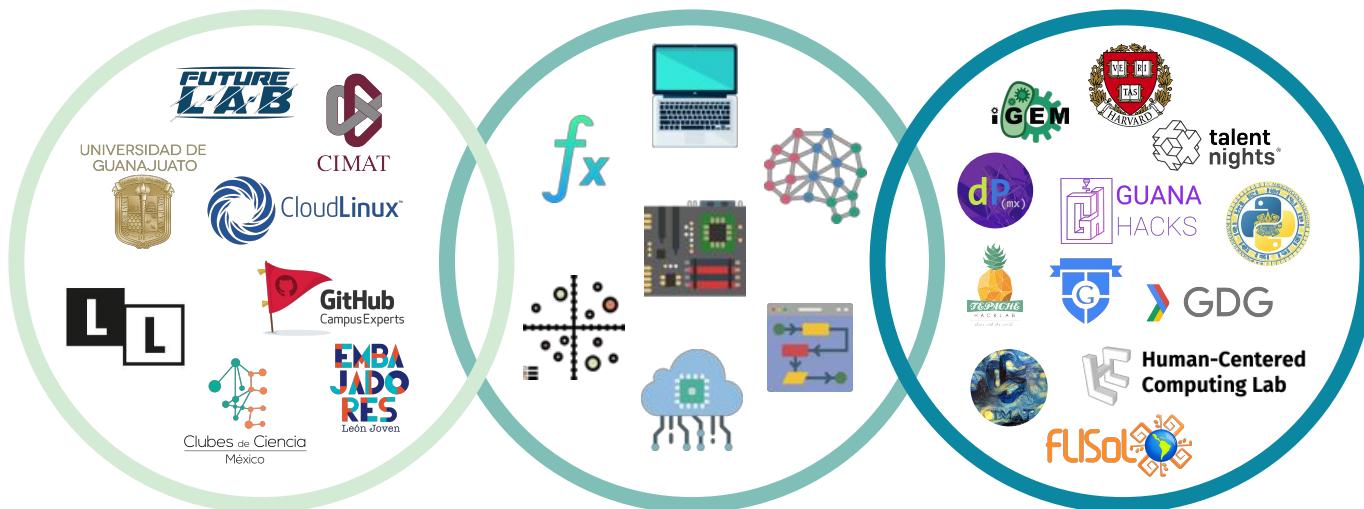
Computer-mathemagician $f[\square]$

Encuéntrame:

 @FerroRodolfo

 @rodo_ferro

Sobre mí:



Objetivos:

- ¿Por qué armar un *Pipeline*?
- Panorama general de la charla
- Contenedorización con Docker
- Pipeline básico con Luigi

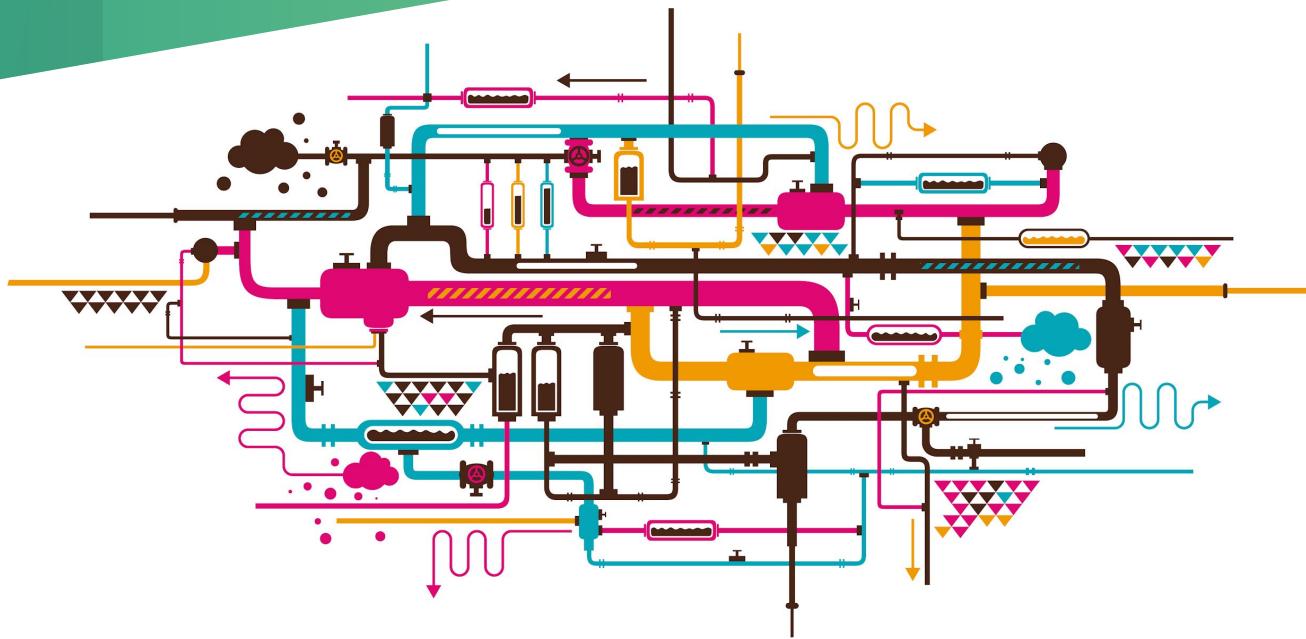
Charla → Código

1.

¿Por qué armar un *Pipeline*?

Eficientemos trabajo

Pipeline

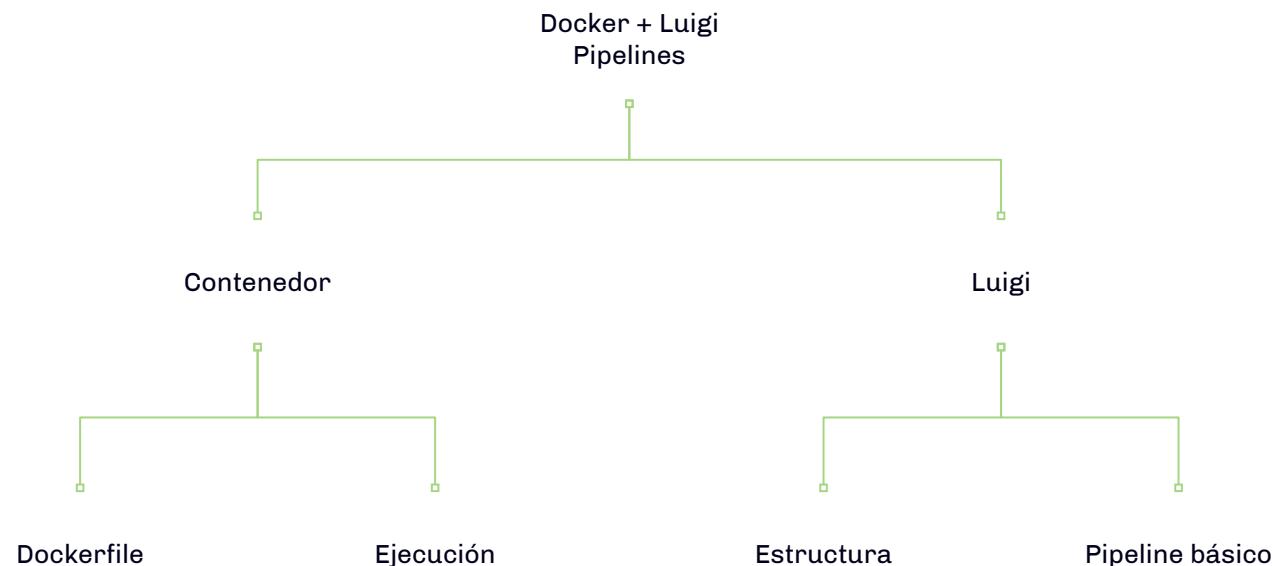


2.

Panorama general de la charla

¿De qué va todo esto?

Flujo de la charla



3.

Contenedорización con Docker

Docker salvando el mundo

Dockerfile

```
# We will use Ubuntu for our image:  
FROM ubuntu:latest  
  
# Updating Ubuntu packages:  
RUN apt-get update && yes|apt-get upgrade && \  
    apt-get install -y wget && \  
    apt-get install -y sudo && \  
    apt-get install -y vim && \  
    apt-get install -y nano && \  
    apt-get install -y htop  
  
# Set locales for encoding:  
RUN apt-get update && apt-get install -y locales && rm -rf /var/lib/apt/lists/* \  
    && localedef -i en_US -c -f UTF-8 -A /usr/share/locale/locale.alias en_US.UTF-8  
ENV LANG en_US.utf8  
  
# Add working directory:  
WORKDIR /docker-luigi-pipes  
  
# Copy files into our container:  
COPY . /docker-luigi-pipes
```

Dockerfile

```
# Install Anaconda:  
RUN wget https://repo.continuum.io/archive/Anaconda3-5.0.1-Linux-x86_64.sh && \  
    bash Anaconda3-5.0.1-Linux-x86_64.sh -b && \  
    rm Anaconda3-5.0.1-Linux-x86_64.sh  
  
# Set path to conda:  
ENV PATH /root/anaconda3/bin:$PATH  
  
# Update Anaconda packages:  
RUN conda update conda && \  
    conda update anaconda && \  
    conda update --all  
  
# Install conda packages:  
RUN conda install -c conda-forge luigi scipy numpy pillow tesseract && \  
    conda install -c menpo opencv && \  
    conda install -c jim-hart pytesseract
```

A correr...

Construimos el contenedor:



```
$ docker build --no-cache -t pipe:latest .
```

Lo corremos:



```
$ docker run -it pipe
```

4.

Pipeline básico con Luigi

¿Luigi es mejor que Mario?

About Luigi



Spotify®



Luigi is a Python (2.7, 3.6, 3.7 tested) package that helps you build complex pipelines of batch jobs. It handles dependency resolution, workflow management, visualization, handling failures, command line integration, and much more.

Estructura

```
import luigi

class MyTask(luigi.Task):
    filename = luigi.Parameter(default='..../assets/test.png')

    def requires(self):
        # Returns a list of requirements (other tasks):
        return []

    def output(self):
        # Returns the output of the task into a Target object:
        return luigi.LocalTarget('my_output_file.txt')

    def run(self):
        # Runs main action of task, outputs results:
        with self.output().open('w') as out:
            out.write(new_name)
```

A correr...

Ejecutamos la tarea:



```
python luigi_script.py --local-scheduler MyTask
```

Por hacer:

- Extender tareas
- Ejecutar visualizador
- ¿Hacer **PR** al repo?
- ...

Visualizador

Luigi Task Visualiser x
localhost:8082/static/visualiser/index.html#

Luigi Task Status

Task List Dependency Graph Workers

Icon	Status	Count
Yellow play button	PENDING TASKS	6679
Blue play button	RUNNING TASKS	0
Green checkmark	DONE TASKS	6946
Red X	FAILED TASKS	88
Pink warning sign	UPSTREAM FAILED	351
Grey minus sign	DISABLED TASKS	0
Grey warning sign	UPSTREAM DISABLED	0

Displaying tasks of family **CreateReportingUsage**.

Show 10 entries Filter table: Filter on Server

Name	Details
DONE CreateReportingUsage	(test=False, date=2015-06-10, parallel)
DONE CreateReportingUsage	(test=False, date=2015-06-11, parallel)
PENDING CreateReportingUsage	(test=False, date=2015-06-13, parallel)
PENDING CreateReportingUsage	(test=False, date=2015-06-12, parallel)
UPSTREAM_FAILED CreateReportingUsage	(test=False, date=2015-06-14, parallel)
UPSTREAM_FAILED CreateReportingUsage	(test=False, date=2015-06-15, parallel)

Showing 1 to 6 of 6 entries (filtered from 14,064 total entries) Previous 1 Next

The screenshot shows the Luigi Task Visualiser interface at localhost:8082. The top navigation bar includes tabs for 'Task List', 'Dependency Graph', and 'Workers'. Below the navigation is a summary section with five large boxes: PENDING TASKS (6679), RUNNING TASKS (0), DONE TASKS (6946), FAILED TASKS (88), and UPSTREAM FAILED (351). A message below the summary indicates it's displaying tasks of the 'CreateReportingUsage' family. The main area features a table with columns for Name and Details, listing six entries: three DONE tasks, two PENDING tasks, and one UPSTREAM FAILED task. The table includes filtering and pagination controls.

¡Gracias!



RODOLFO FERRO PEREZ



¿Preguntas?

Rodolfo Ferro:

 @FerroRodolfo

 @rodo_ferro

<https://rodolfoferroperez.xyz>