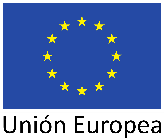




MANUAL TÉCNICO



Programa Operativo Fondo Europeo de Desarrollo Regional Aragón 2014-2020

*Construyendo Europa desde Aragón*

Contenido

[1. Requisitos del entorno 4](#_Toc54627417)

[2. Creación del entorno 5](#_Toc54627418)

[2.1 Instalación de Python 3 5](#_Toc54627419)

[2.1.1 Python 5](#_Toc54627420)

[2.1.2 Descarga de paquetes 5](#_Toc54627421)

[2.1.3 Instalación *offline* de paquetes 8](#_Toc54627422)

[2.1.4 Ejecución y verificación 9](#_Toc54627423)

[2.2 Instalación de Apache Airflow 10](#_Toc54627424)

[2.2.1 Apache Airflow 10](#_Toc54627425)

[2.2.2 Esquema de funcionamiento 11](#_Toc54627426)

[2.2.3 Despliegue e instalación 12](#_Toc54627427)

[2.2.4 Ejecución y verificación 15](#_Toc54627428)

[2.2.5 Versiones 15](#_Toc54627429)

[2.3 Despliegue sobre entorno de desarrollo 15](#_Toc54627430)

[3. Desarrollos realizados 16](#_Toc54627431)

[3.1 Ingesta de los datos 16](#_Toc54627432)

[3.1.1 Ingesta de los artículos del BOE 16](#_Toc54627433)

[3.1.2 Ingesta de los artículos de Aragón 16](#_Toc54627434)

[3.2 Conversión de los documentos 16](#_Toc54627435)

[3.2.1 De PDF a TXT 16](#_Toc54627436)

[3.2.2 De XML a TXT 16](#_Toc54627437)

[3.2.3 De HTML a TXT 16](#_Toc54627438)

[3.3 Extracción de la información 16](#_Toc54627439)

[3.3.1 Extracción mediante reglas 16](#_Toc54627440)

[3.3.2 Extracción mediante el modelo NER 16](#_Toc54627441)

[3.3.3 Modelo jerárquico 16](#_Toc54627442)

[3.4 Inserción en la base de datos 16](#_Toc54627443)

[4. Base de datos 17](#_Toc54627444)

[4.1 Estructura 17](#_Toc54627445)

[4.2 Tablas 17](#_Toc54627446)

[4.3 Campos 17](#_Toc54627447)

Control de versiones

|  |  |  |  |
| --- | --- | --- | --- |
| Versión | Fecha | Autor | Descripción |
| 1.0 |  | Hiberus | Primera versión (incompleta) del documento |

# Requisitos del entorno

En esta sección se van a enumerar los diferentes requisitos que ha de cumplir el entorno para el correcto funcionamiento del programa.

# Creación del entorno

En esta sección se van a detallar las instalaciones realizadas para crear el entorno descrito en la sección 1.

## Instalación de Python 3

### Python

Python es un lenguaje de programación interpretado y multiparadigma que soporta programación imperativa, orientada a objetos y funcional. Es uno de los lenguajes más utilizados de la actualidad (Octubre de 2020), ya que se diseñó para aumentar la productividad y la legibilidad. Además, a lo largo de los años se han publicado diferentes módulos y *frameworks* que, unido a su sencillez y su velocidad de desarrollo, lo han convertido en una de las mejores opciones a la hora de desarrollar aplicaciones relacionadas con el campo del dato.

Las versiones principales de Python son Python 2 y Python 3. La versión 2 se quedó sin soporte cuando empezó el año 2020, mientras que la 3 sigue desarrollándose y mejorándose. De manera general, los sistemas operativos basados en Linux llevan preinstaladas ambas versiones por defecto, pero no tiene por qué ser así en distribuciones CentOS o Red Hat.

El objetivo de esta sección es describir el proceso de instalación de Python 3.6.9 en un servidor CentOS/Red Hat sin conexión a Internet. Al estar aislados, estos servidores no pueden descargarse los archivos de instalación, por lo que estos deberán obtenerse previamente a través de un sistema *online*, y, después, añadirlos al sistema *offline* para que este pueda llevar a cabo el proceso de instalación.

### Descarga de paquetes

En este apartado se describen diferentes formas de obtener los paquetes RPM requeridos (.rpm) de repositorios fiables para instalar python3.6 en sistemas con las siguientes características:

* Sistemas Linux basados en CentOS/Red Hat/similares.
* Sistemas con arquitectura x86\_64.
* Sistemas sin versiones ya instaladas de Python3.6.

Todo el contenido de este apartado debe ser ejecutado en sistemas *online*, con conexión a Internet.

#### Paquetes principales

Los programas generalmente poseen dependencias con otros paquetes/programas. Esto significa que la ejecución principal se apoya total o parcialmente en otro programa.

La lista de dependencias principales (no las sub-dependencias) de Python 3.6 en un sistema CentOS recién instalado son las que se enumeran a continuación:

**Python**

* rh-python36-python-libs
* rh-python36
* rh-python36-python
* rh-python36-python-devel
* rh-python36-python-setuptools
* rh-python36-python-pip
* rh-python36-runtime
* rh-python36-python-virtualenv

**Sistema**

* libc.so.6
* glibc
* scl-utils-build
* dwz
* perl-srpm-macros
* python-srpm-macros
* redhat-rpm-config

Estos paquetes tienen, a su vez, diferentes dependencias que deben estar disponibles en el sistema. Generalmente, estas ya están disponibles en la instalación base de los SO, pero si se desea descargar todas las sub-dependencias se puede seguir el proceso descrito en el apartado 2.1.2.4.

#### Descarga mediante gestor de paquetes yum

Yum es el gestor de paquetes disponible por defecto en sistemas Linux basados en CentOS/Red Hat/similares y RPM. A continuación se muestran los comandos que, ejecutándolos en bash, descargan (sin instalar) los paquetes mencionados en el apartado 2.1.2.1 y sus dependencias principales, guardándolos en la carpeta packages:

$ yum install yum-utils centos-release-scl epel-release

$ mkdir ./packages

$ yumdownloader --destdir=./packages/ --resolve \

libc.so.6 \

glibc \

scl-utils-build \

dwz \

perl-srpm-macros \

python-srpm-macros \

redhat-rpm-config \

rh-python36-python-libs \

rh-python36 \

rh-python36-python \

rh-python36-python-devel \

rh-python36-python-setuptools \

rh-python36-python-pip \

rh-python36-runtime \

rh-python36-python-virtualenv

Aunque en el presente documento se detallan más opciones de descarga, este ha sido el procedimiento seguido para instalar Python 3.6 en el servidor biv-aodback01 del entorno de desarrollo.

Los detalles de la descarga, en Octubre de 2020, son los siguientes:

|  |  |  |  |
| --- | --- | --- | --- |
| Paquete | Arquitectura | Versión | Repositorio |
| nss-softokn-freebl | x86\_64 | 3.44.0-8.el7\_7 | base |
| glibc | x86\_64 | 2.17-307.el7.1 | base |
| scl-utils-build | x86\_64 | 20130529-19.el7 | base |
| dwz | x86\_64 | 0.11-3.el7 | base |
| perl-srpm-macros | noarch | 1-8.el7 | base |
| python-srpm-macros | noarch | 3-32.el7 | base |
| redhat-rpm-config | noarch | 9.1.0-88.el7.centos | base |
| rh-python36 | x86\_64 | 2.0-1-el7 | centos-sclo-rh |
| rh-python36-python | x86\_64 | 3.6.9-2.el7 | centos-sclo-rh |
| rh-python36-python-libs | x86\_64 | 3.6.9-2.el7 | centos-sclo-rh |
| rh-python36-python-devel | x86\_64 | 3.6.9-2.el7 | centos-sclo-rh |
| rh-python36-python-setuptools | noarch | 36.5.0-1.el7 | centos-sclo-rh |
| rh-python36-python-virtualenv | noarch | 15.1.0-2.el7 | centos-sclo-rh |
| rh-python36-python-pip | noarch | 9.0.1-2.el7 | centos-sclo-rh |
| rh-python36-runtime | x86\_64 | 2.0-1.el7 | centos-sclo-rh |

#### Descarga desde los repositorios

Yum facilita el acceso a los diferentes repositorios que almacenan los paquetes instaladores. Sin embargo, la descarga puede efectuarse directamente sobre los sitios web utilizando cualquier navegador (Firefox, Google Chrome, Opera, etc.). Es importante verificar la identidad y la oficialidad, o no oficialidad, de los sitios de descarga.

A continuación se detallan las características de los repositorios a los que se ha accedido, así como opciones adicionales y comúnmente utilizadas (en sistemas CentOS/Red Hat la carpeta */etc/yum.repos.d* contiene, generalmente, la información de todos los repositorios a los que se tiene acceso).

|  |  |  |
| --- | --- | --- |
| Nombre | URL Base | Oficial |
| CentOS-7 – Base [base] | http://mirror.centos.org/centos/7/os/x86\_64/ | Si |
| Extra Packages for Enterprise  Linux 7 – x86\_64 [epel] | http://download.fedoraproject.org/pub/epel/7/x86\_64 | Si |
| CentOS-7 – SCLo sclo  [centos-sclo-sclo] | http://mirror.centos.org/centos/7/sclo/x86\_64/sclo | Si |
| CentOS-7 – SCLo rh  [centos-sclo-rh] | http://mirror.centos.org/centos/7/sclo/x86\_64/rh | Si |

#### Descargar todas las dependencias

En apartados previos se ha explicado como instalar Python 3.6 junto con sus dependencias directas. Sin embargo, es posible descargar todas las dependencias simultáneamente. Los siguientes comandos bash descargan todos los paquetes RPM en la carpeta ./all/packages. La carpeta ./all/install-root se le indica como argumento para que yum no detecte que ya existen paquetes preinstalados, ya que en tal caso no los descargaría.

$ yum install yum-utils centos-release-scl epel-release

**$ mkdir /your/path/all**

**$ mkdir /your/path/all/packages /your/path/all/install-root**

**$ yum install --downloadonly \**

**--installroot=/your/path/all/install-root/ \**

**--releasever=7Server \**

**--downloaddir=/your/path/all/packages/ \**

**rh-python36**

El valor para –-releasever es 7 para CentOS 7, mientras que en Red Hat puede utilizarse 7Server.

En este caso la carpeta *packages* contendrá todos los instalables asociados a las dependencias y sub-dependencias del paquete principal rh-python36 (~120 en CentOS 7).

Se recomienda utilizar esta opción con prudencia, ya que las dependencias de más bajo nivel suelen ser programas instalados por defecto en las distribuciones de Linux (bash, curl, gcc, etc), y una reinstalación/actualización de estos puede afectar directamente al funcionamiento general.

### Instalación *offline* de paquetes

Una vez descargados los archivos RPM, estos pueden instalarse en el servidor *offline* utilizando, de nuevo, el gestor de paquetes yum. Para ello es necesario copiar/encapsular los paquetes obtenidos y disponibilizarlos mediante copiado o cualquier método de compartición de archivos en el servidor que no dispone de conexión a Internet. Una vez que los instaladores están todos en la ruta

*@offline-node:/path/to/packages/,* estos pueden instalarse mediante:

$ sudo yum –-nogpgcheck localinstall /path/to/packages/\*

Si el proceso se ha ejecutado correctamente, el sistema CentOS/Red Hat debería de disponer de un intérprete de Python 3.6(.9) alojado en la ruta:

$ /opt/rh/rh-python36/root/usr/bin/python

Junto a él deberían estar disponibles los ejecutables de pip y virtualenv, así como pyvenv y *aliases* de los ejecutables.

**NOTA:** de manera general,para poder realizar la instalación se requieren permisos de administrador.

### Ejecución y verificación

En sistemas CentOS/Red Hat existen programas básicos que requieren de Python 2 para funcionar. Por ello, se recomienda instalar el nuevo intérprete a través de las *Software Collections* (siguiendo el proceso descrito en este documento), y después ejecutarlo utilizando la herramienta *scl*. De esta forma se puede activar/desactivar Python 3 y cambiar las variables de entorno de manera sencilla y sin afectar al comportamiento general del sistema:

$ scl enable rh-python36 bash

A continuación se muestra un ejemplo de cómo cambiar de versión de Python tras el proceso de instalación en un sistema Red Hat:

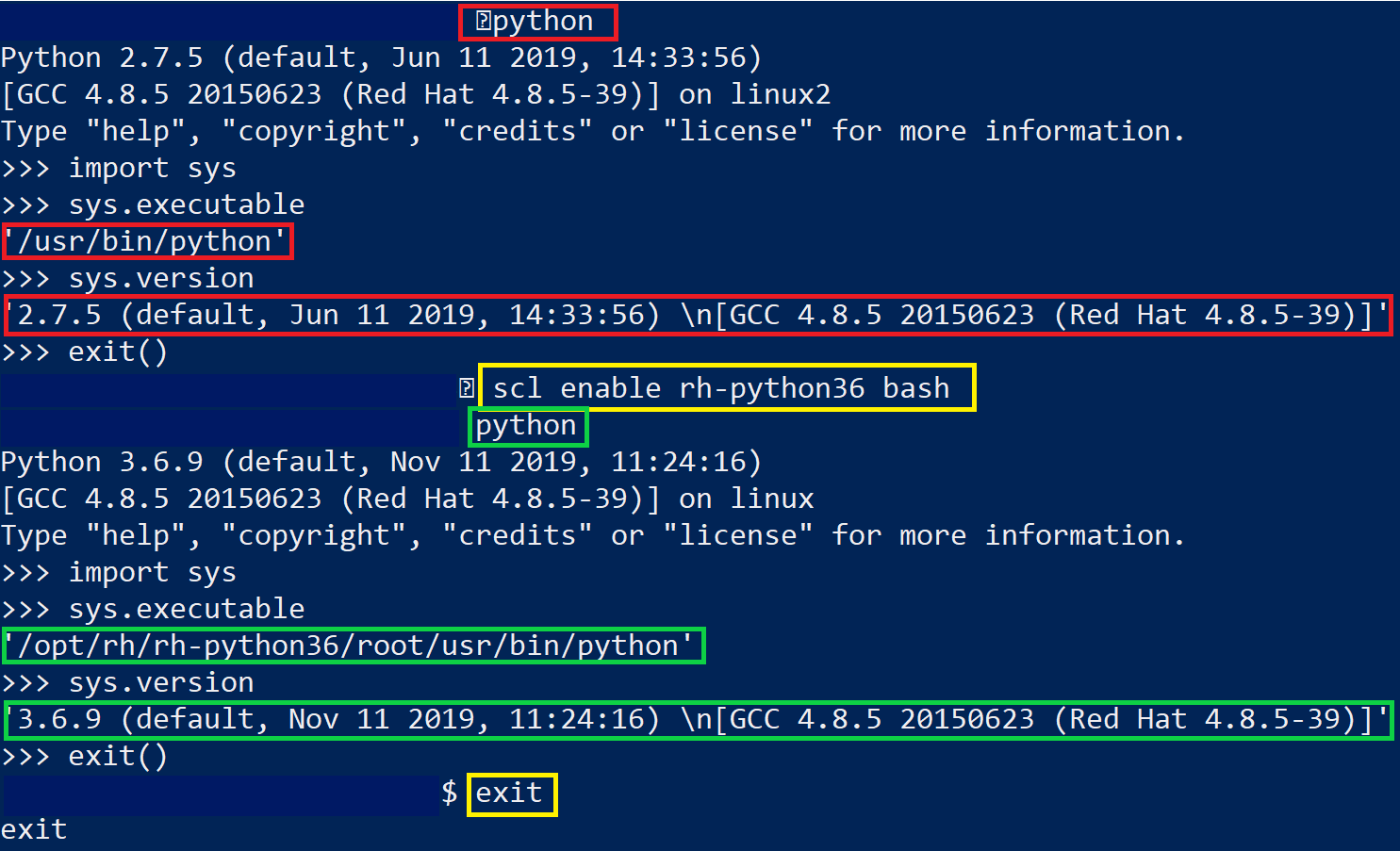


Ilustración 1: ejecución de Python 3 con scl

<explicar la foto>

## Instalación de Apache Airflow

### Apache Airflow

Apache Airflow es una plataforma *open-source* escalable y dinámica utilizada para diseñar, programar y orquestar tareas, ETLs o procesos. Su funcionamiento se basa en ejecutar DAGs (grafos acíclicos dirigidos), definidos como un conjunto de tareas que deben ejecutarse en un orden predefinido sin lazos de realimentación (p. ej. ETL: extracción 🡪 transformación 🡪 carga). Airflow fue desarrollado y liberado por la empresa Airbnb, pero hoy en día la mantiene la organización Apache.

Una de sus características principales, y que le diferencian de otros orquestadores de *workflows,* es que todas las tareas ejecutadas por la plataforma son programadas y desarrolladas con el lenguaje Python. Además, incluye módulos para ejecutar diferentes acciones (conexiones SSH, FTP, etc.), soporta integración junto a otros servicios (HDFS, Spark, PostgreSQL, GCP, AWS, Azure, etc.) y dispone de diferentes modos de ejecución que le permiten ajustarse al sistema sobre el que está desplegado. Por ello, Apache Airflow tiene importantes aplicaciones en proyectos relacionados con el procesamiento, ingesta o manipulación de datos.

Para interactuar con Apache Airflow, este dispone de dos interfaces principales. La primera consiste en una interfaz por medio de la consola de comandos, CLI (*command line interface*). Con la CLI de Airflow se pueden listar los DAGs activos, añadir o eliminarlos, controlar los servicios que componen el ecosistema de Airflow, ver logs o el estado de las tareas ejecutadas, etc.

La segunda es una interfaz web gráfica (WebUI). Airflow incluye un *webserver* desarrollado sobre Flask y gunicorn que permite acceder al estado de los DAGs y las tareas, a sencillos *dashboards*, a logs o al propio código Python de los DAGs. Es una interfaz muy accesible y cómoda que permite al usuario tener una rápida visualización del funcionamiento del sistema.

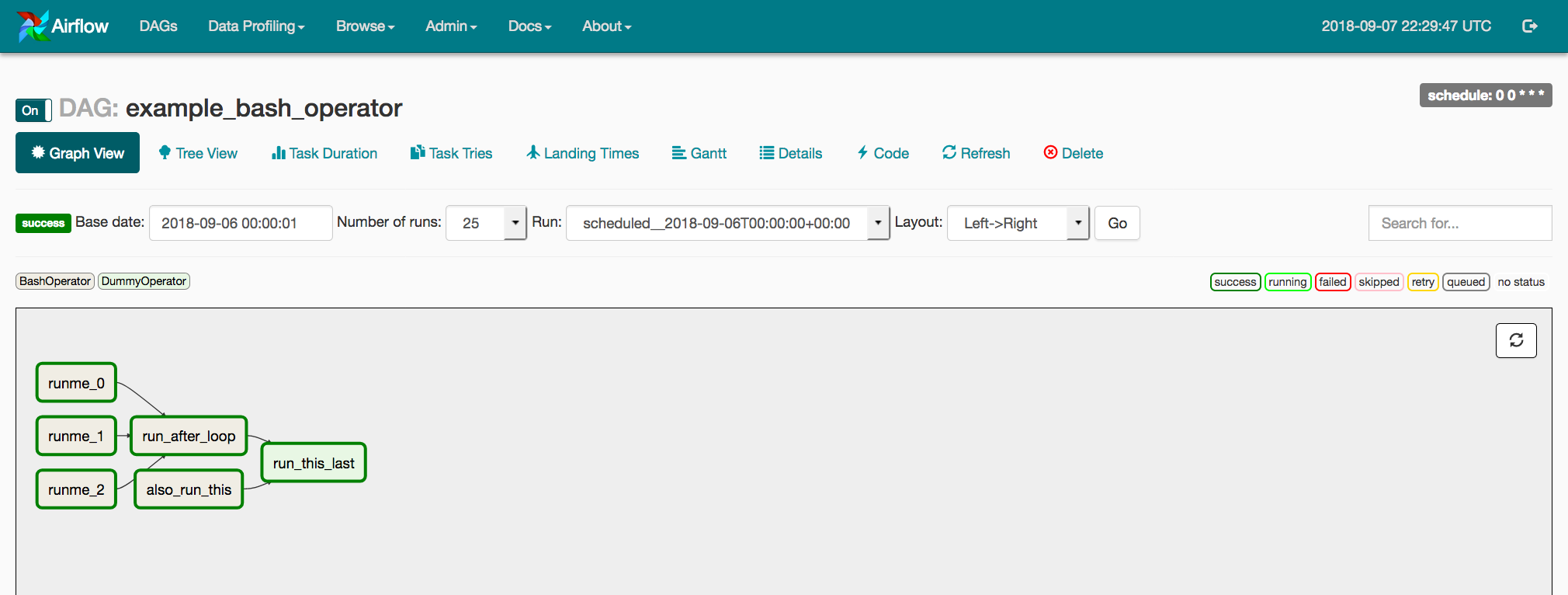


Ilustración 2: ejemplo de visualización de DAG en WebUI de Airflow

### Esquema de funcionamiento

Apache Airflow posee una arquitectura modular, en la que cada componente es el encargado de ejecutar o administrar una parte del proceso de orquestación. Así, un componente de Airflow se encarga de almacenar las tareas programadas, otro de ejecutarlas en el instante temporal que corresponda, otro de almacenar los resultados y el estado de cada tarea ejecutada, otro mantiene el servicio de visualización y seguimiento del sistema, etc.

Tal y como se ha mencionado en el apartado 2.2.1, Airflow puede ejecutarse en diferentes modos. El modo de ejecución y despliegue debe ser escogido teniendo en cuenta diferentes factores, asociados siempre al proyecto al que se vincula. Algunos de ellos son:

* Recursos hardware de los que se dispone y/o se puede llegar a disponer.
* Tiempo, CPU y memoria requerido por las tareas ejecutadas.
* Aplicación/sistema donde debe integrarse, y qué servicios componen el ecosistema del proyecto.
* Objetivo. En caso de que se desee disponer de Airflow para un proyecto en concreto, deben evaluarse los requisitos de las tareas involucradas en tal proyecto, y tomar una decisión en base a ello. Por el contrario, si se busca instalar un orquestador transversal, utilizado por diferentes proyectos, se recomienda evitar el modo (*executor*) Sequential y el Local, y valorar la ejecución distribuida de Airflow (Celery, etc.)
* Etc.

En el contexto de este proyecto (Expte. CUS\_CMN2020\_06), las tareas definidas pueden implementarse sobre un despliegue local de Airflow, siendo opciones válidas tanto el LocalExecutor como el SequentialExecutor. A pesar de ello, se recomienda evitar el modo SequentialExecutor ya que es el despliegue más sencillo de Airflow, y no aprovecha realmente todo el potencial del sistema sobre el que se ejecuta. A continuación se describen brevemente ambos modos de ejecución.

#### SequentialExecutor

Está asociado a un despliegue local no distribuído de Apache Airflow. Se caracteriza por su sencillez y por no admitir concurrencia ni paralelismo, lo cual implica que solo una tarea puede ejecutarse de manera simultánea. Además, es el único que puede utilizarse con *sqlite* como *meta-database,* ya que esta no soporte múltiples conexiones. Puede ser adecuado en entornos no productivos o en entornos de pruebas. Es la configuración por defecto que incluye la instalación.

<https://airflow.apache.org/docs/stable/executor/sequential.html>

#### LocalExecutor

También está asociado a un despliegue local no distribuído de Apache Airflow. La diferencia fundamental con el SequentialExecutor es que sí permite paralelismo y concurrencia en la ejecución de las tareas. Por ello, se requiere un sistema de bases de datos diferente a sqlite, pudiendo implementarse tanto con PostgreSQL como con MySQL para ejercer el papel de *meta-database*. La arquitectura de Airflow con LocalExecutor puede ser escalado verticalmente.

<https://airflow.apache.org/docs/stable/executor/local.html>

### Despliegue e instalación

En esta sección se explica y detalla cómo instalar la versión 1.10.12 de Airflow en un servidor *offline* utilizando Python 3.6.9 como intérprete y LocalExecutor como sistema de ejecución. Esta sección está dividida en:

* 2.2.4.1 Obtención de paquetes con pip: se detalla la forma de instalar Airflow en un entorno virtual, junto con las posibles dependencias que se requieran para el proyecto.
* 2.2.4.2 Configuración de PostgreSQL: se detalla la configuración requerida en PostgreSQL para poder funcionar como sistema de bases de datos de Airflow.
* 2.2.4.3 Configuración de Airflow: se explica como manipular el archivo de configuración autogenerado y los pasos a seguir para iniciar Airflow.

#### Obtención de paquetes con pip

Apache Airflow está disponible en PyPI, por lo que es fácilmente descargable con pip. Actualmente (Octubre de 2020) la versión estable es la 1.10.12.

Debido a que existen diferentes versiones de Python 3 y Airflow requiere de más paquetes de PyPI para funcionar correctamente, es recomendable descargar las versiones específicas y adecuadas al intérprete que va a ejecutar Airflow. Para la versión 3.6 de Python y la 1.10.12 de Airflow, se puede obtener el fichero de constraints de la siguiente forma:

**NOTA:** los siguientes comandos son válidos en un sistema con conexión a Internet.

$ wget -O constraint.txt \

[**https://raw.githubusercontent.com/apache/airflow/constraints-**](https://raw.githubusercontent.com/apache/airflow/constraints-)**1.10.12/constraints-3.6.txt**

El contenido del fichero incluye las versiones apropiadas para las dependencias Python de Airflow 1.10.12 y Python 3.6. Sus primeras líneas deberían ser semejantes a:

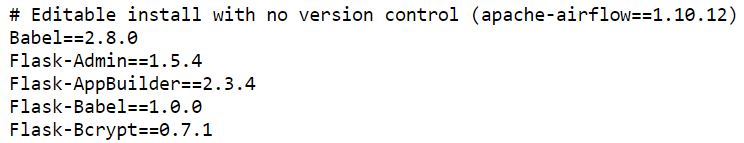


Ilustración 3: segmento del fichero constraints.txt para Airflow 1.10.12 + Python 3.6.X

Una vez se disponga del fichero constraints.txt, pueden descargarse los paquetes de Apache Airflow con los siguientes comandos:

$ pip install apache-airflow[<extras>]==1.10.12 \

**--constraint “path/to/constraint.txt”**

Donde <extras> representa los paquetes adicionales a instalar y que permitirán posteriormente interactuar con otros sistemas o servicios. Algunos ejemplos de <extras> pueden ser postgres, mysql, aws o gcp. Para ver la lista completa: <airflow.apache.org/docs/stable/installation.html#extra-packages>

Se recomienda llevar a cabo la instalación de los paquetes sobre un entorno virtual, para posteriormente encapsularlo y distribuirlo sobre los sistemas *offline* en los que se desee integrar Airflow. En el apartado 2.3 se detalla el procedimiento completo seguido para desplegar el entorno en el servidor de desarrollo biv-aodback-01.aragon.local.

#### Configuración de PostgreSQL

Como se ha explicado en apartados anteriores, Airflow requiere de un sistema de bases de datos que admita conexiones concurrentes. Para ello puede hacerse uso de PostgreSQL o MySQL.

Si se requiere instalarlo, los RPM de PostgreSQL 9.6.19 pueden obtenerse de la siguiente manera:

**$ mkdir ./packages**

# añadir el repositorio oficial de PostgreSQL

$ yum install [https://download.postgresql.org/pub/repos/yum/9.6/redhat/rhel-7-x86\_64/pgdg-redhat-repo-latest.noarch.rpm](https://download.postgresql.org/pub/repos/yum/9.6/redhat/rhel-7-x86_64/pgdg-redhat-repo-latest.noarch.rpm´)

# instalar yum-utils

$ yum install yum-utils

# descargar los archivos RPM

**$ yumdownloader –destdir=./packages/ --resolve \**

**postgresql96 \**

**postgresql96-server \**

**postgresql96-contrib \**

**postgresql96-libs \**

**postgresql96-devel**

Y trasladando los paquetes al servidor destino:

# instalar los paquetes

$ yum --nogpgcheck localinstall ./packages/\*

En caso de que ya se disponga de un sistema PostgreSQL funcional, airflow requiere tener acceso a su propia base de datos, por lo que es necesario crear tanto un rol como una nueva base de datos. En referencia al rol, es recomendable que el nombre del usuario sea airflow y que no disponga de permisos salvo para acceder a su propia base de datos:

# ingresar en PostgreSQL como administrador

$ su - postgres

# crear rol

$ createuser --interactive --pwprompt

# crear BBDD

# createdb -O <rol creado> <nombre BBDD>

$ createdb -O airflow airflow

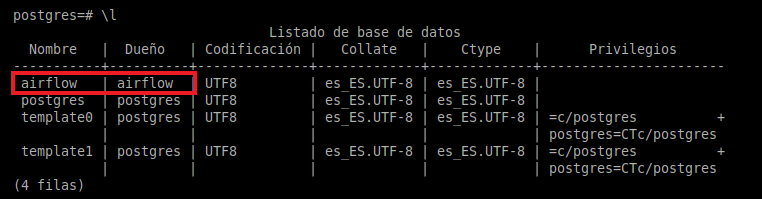


Ilustración 4: usuario(rol) y BBDD en PostgreSQL para Airflow

**NOTA**: el sistema de autenticación en PostgreSQL puede verse en el archivo de configuración pg\_hba.conf. Para evitar la creación de un usuario en el sistema operativo asociado a Airflow, el sistema de autenticación para conexiones sobre el host debe ser establecida a un valor diferente de PEER. El administrador puede ver y modificar el archivo utilizando los comandos siguientes:

# localizar el archivo de configuración

**$ sudo find / -iname pg\_hba.conf**

# modificar el archivo de configuración

$ sudo vi </path/to/pg\_hba.conf>

# actualizar configuración sin reiniciar servicio

Además, es necesario cambiar el sistema de autenticación en PostgreSQL para las conexiones locales. Por defecto, cuando un usuario se conecta desde la terminal a psql, este interpreta que el usuario es el mismo que el que ha iniciado la sesión. Este comportamiento por defecto puede cambiarse para abstraer a Airflow del usuario que lo ha ejecutado, para ello hay que cambiar el archivo de configuración pg\_hba.conf.

cd airflow-env

grep -ilr “/home/aciriano/opendta-env” . | xargs -d ‘\n’ sed -i ‘s/\/home\/aciriano\/opendata-env/\/export\/home\/hiberus\/airflow-env/g’

airflow connections --add --conn\_id ‘ssh\_front’ –conn\_uri ‘ssh://hiberus@172.27.38.119:22/?key\_file=/export/home/hiberus/.ssh/id\_rsa’

#### Configuración de Airflow

Con esta configuración, el string de conexión requerido en la configuración de Airflow es (línea 75):

sql\_alchemy\_conn = postgresql+psycopg2://airflow:airflow@localhost/airflow

load\_Examples = False

sql\_alchemy\_conn = postgresql+psycopg2://airflow:airflow@localhost/airflow

result\_backend = db+postgresql://airflow:airflow@localhost/airflow

parallelism =

### Ejecución y verificación

#### Entornos virtuales

<en este apartado se definen los paquetes a instalar con pip y se explican las dependencias de airflow>

--sistema con conexión a internet--

scl enable rh-python36 bash

python -m venv ~/opendata-env

<instalar airflow con los extras y con los constraints>

<instalar paquetes específicos del proyecto “requirements.txt”>

--sistema sin conexión a internet—

scl enable rh-python36 bash

python -m venv ~/dev-env

Copiar la carpeta ~/dev-env/lib/python3.6/site-packages a su análogo en el sistema remoto destino offline

source ~/dev-env/bin/actívate

<lanzar airflow>

airflow initdb

airflow scheduler &

airflow webserver

<cambiar el archivo de configuración creado en ~/airflow>

Sustituir las líneas indicadas arriba para que se conecte a postgres

Relanzar airflow

#### Prueba de funcionamiento

<en este apartado se muestra un código muy sencillo que puede usarse para verificar el funcionamiento>

### Versiones

## Despliegue sobre entorno de desarrollo

# Desarrollos realizados

En esta sección se van a detallar los desarrollos realizados para el presente proyecto.

## Ingesta de los datos

### Ingesta de los artículos del BOE

### Ingesta de los artículos de Aragón

## Conversión de los documentos

### De PDF a TXT

### De XML a TXT

### De HTML a TXT

## Extracción de la información

### Extracción mediante reglas

### Extracción mediante el modelo NER

### Modelo jerárquico

Este modelo es el que decide de cuál de los dos extractores se va a recoger la información de cada campo.

## Inserción en la base de datos

# Base de datos

## Estructura

## Tablas

## Campos