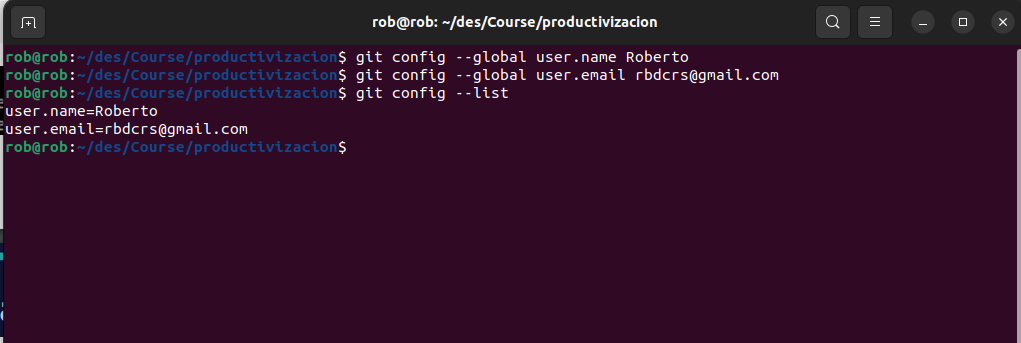
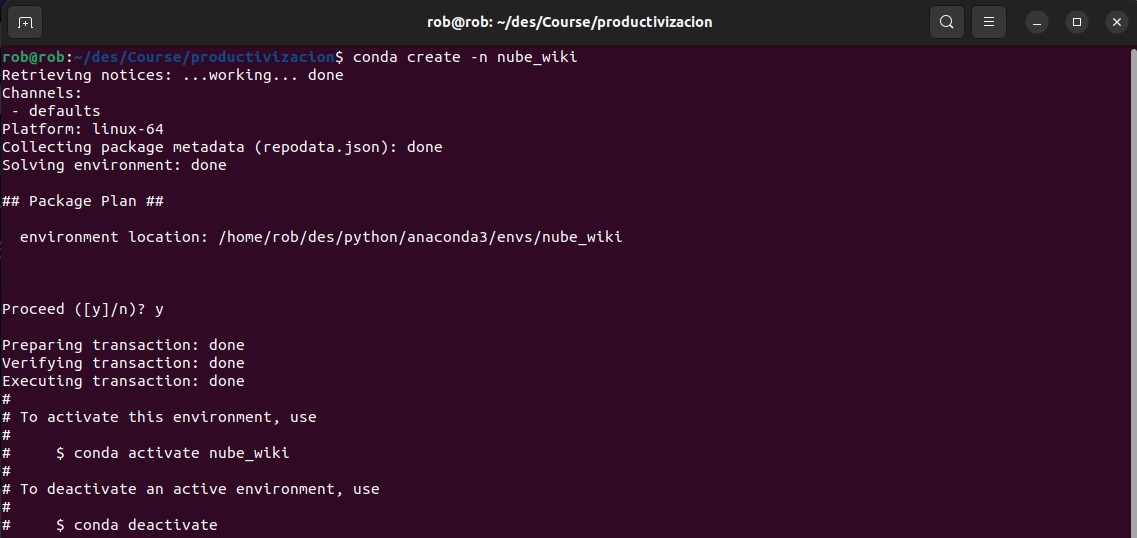
# MEMORIA Ejercicio Git + Environments

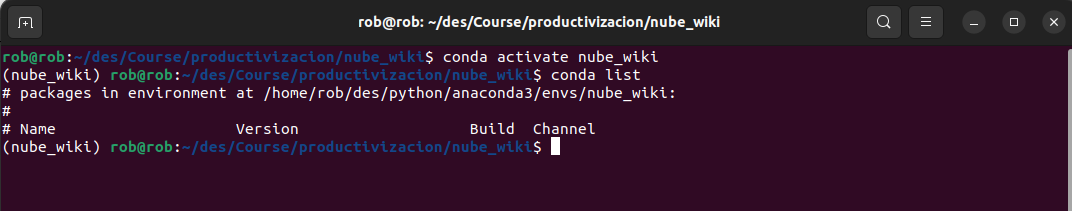
En primer lugar, aunque estos pasos ya se realizaron durante las clases, realizo la configuración del entorno git.

Para ello defino las variables user.name y user.email que serán las utilizadas por git para publicar las modificaciones en el repositorio.

Antes de crear el directorio del proyecto y comenzar a agregar código genero el entorno virtual en conda: *conda create -n nube\_wiki.*

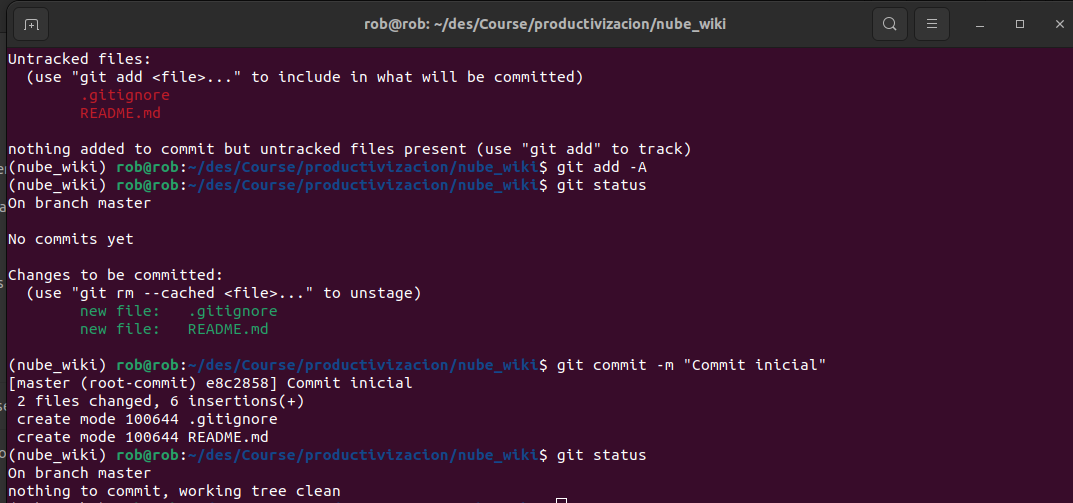
No modifico la ruta por defecto de entornos de conda por lo que el entorno se encuentra en el directorio */home/rob/des/python/anaconda3/envs/nube\_wiki.*



Para comprobar que funciona lo activo *conda activate nube\_wiki.* Por el momento no tiene instalada ninguna librería

Una vez tengo el entorno virtual activado (no es necesario para los siguientes pasos, pero así me aseguro trabajar sobre el entorno virtual y no utilizar el default de la máquina que podría tener otras librerías diferentes) creo el directorio para el proyecto *nube\_wiki* e inicializo el repositorio local: *git init.*

Creo los ficheros iniciales **README.md** con información básica del proyecto y **.gitignore** con los ficheros y directorios que no quiero que se añadan al repositorio, por el momento *.ipynb\_checkpoints*



La practica la voy a realizar con jupyterlab por lo que instalado el paquete en el entorno virtual y realizo el primer export del entorno a fichero .yml:

* *conda install jupyterlab*
* *conda env export -f environment.yml --no-builds -n nube\_wiki*

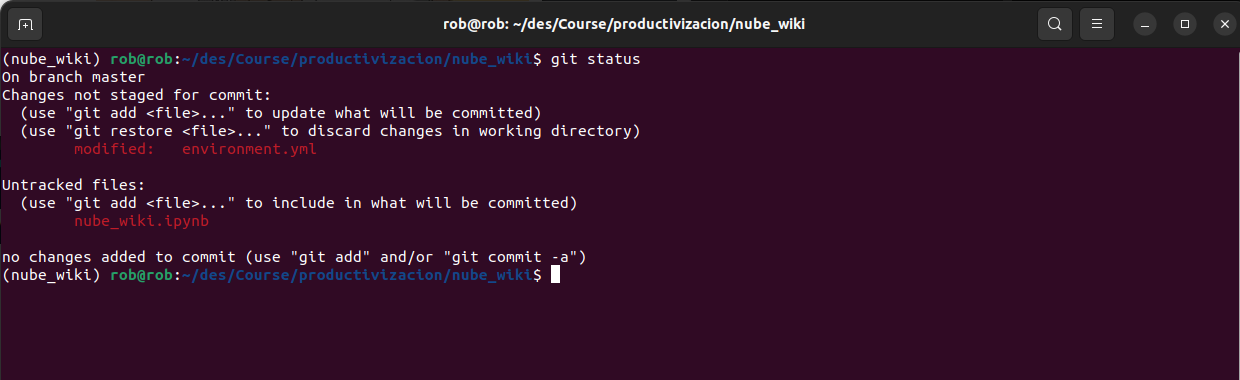
Añado el fichero *environment.yml* al repositorio.

Los siguientes pasos los realizo desde la interfaz de jupiterlab. En este paso se genera el directorio *.ipynb\_checkpoints* que filtre en *.gitignore*

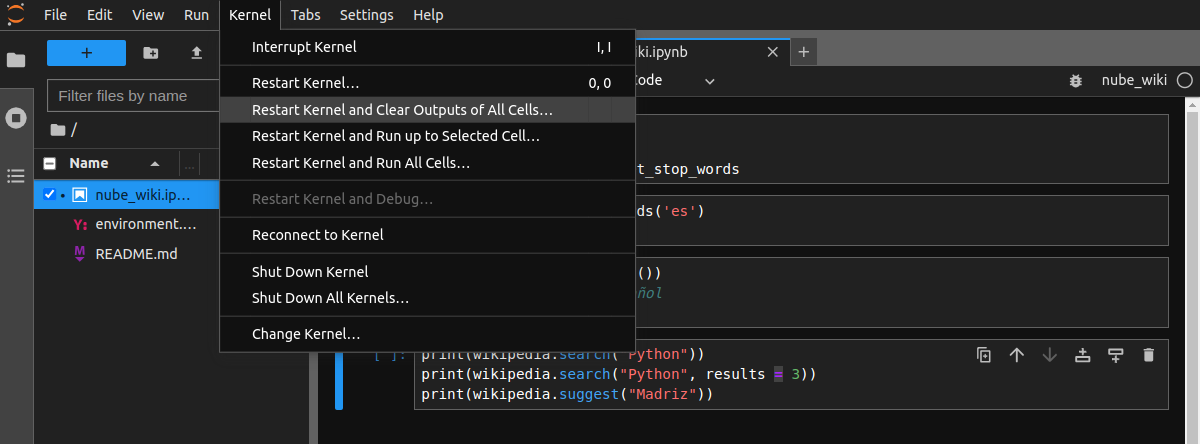
* Creo el notebook y añado las primeras líneas de código
* Al ejecutar faltan librerías por instalar. Son necesarias las librerías stop\_words y Wikipedia, que instalo, compruebo que el notebook se ejecuta correctamente y actualizo el fichero environment.

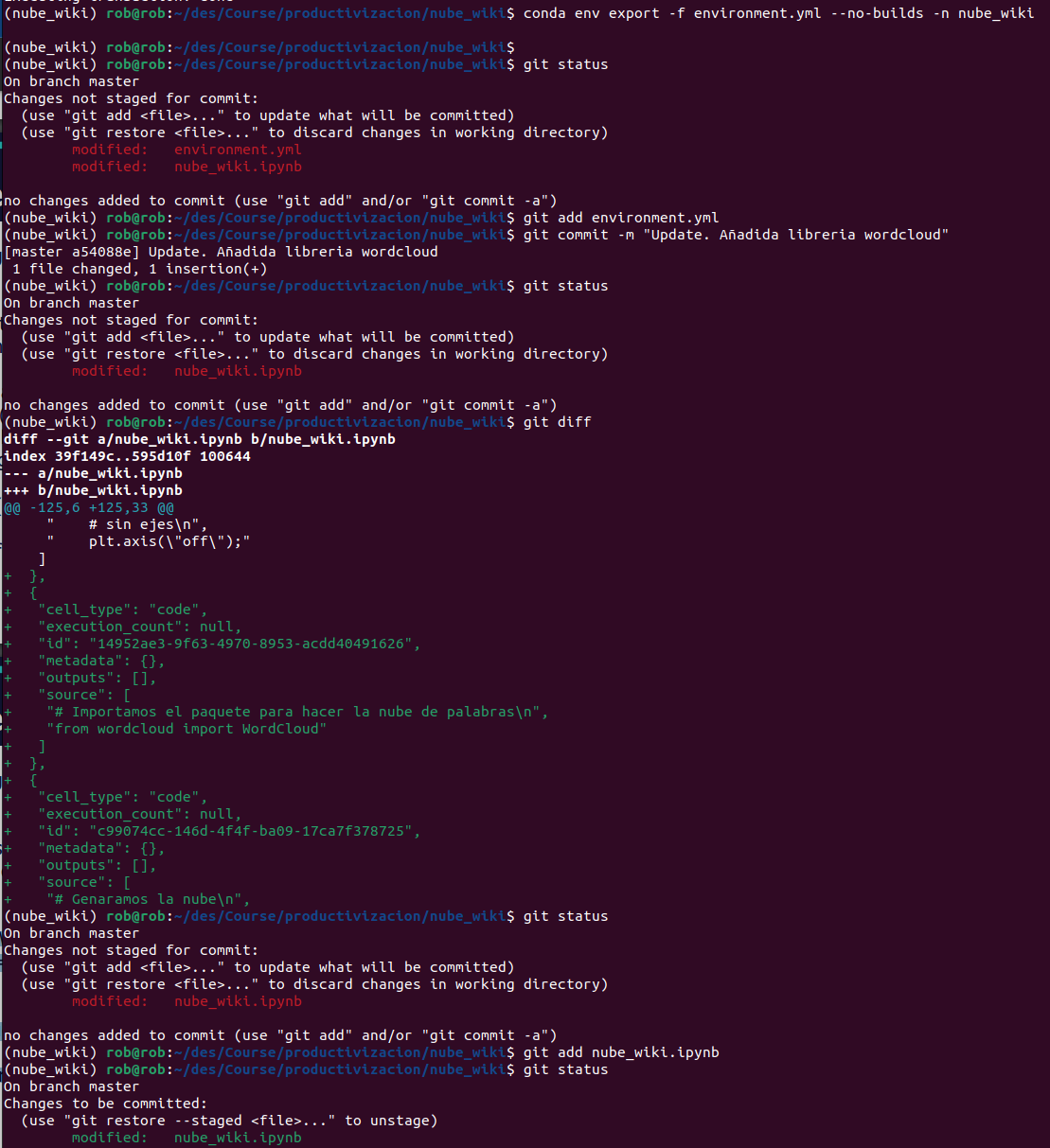
*conda install conda-forge::wikipedia*

*conda install conda-forge::stop-words*

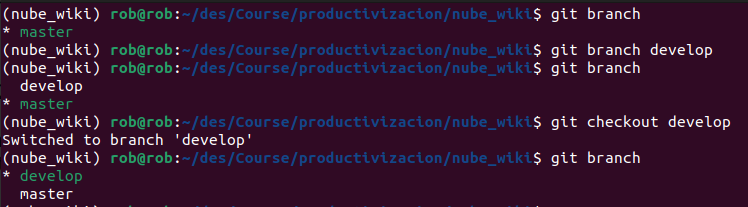


* Hay dos cambios en el proyecto, los imports y las nuevas librerías. Añado los cambios al repositorio en dos commits independientes

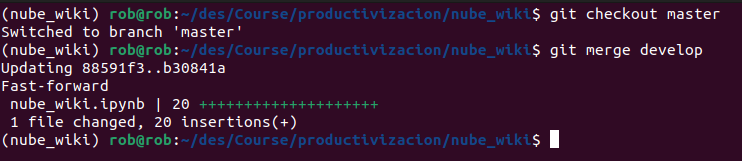
En el primer commit del fichero *nube\_wiki.ipynb*, cometí el error de no realizar una limpieza tanto del kernel como de los resultados de las celdas, en posteriores actualizaciones aplico el siguiente proceso previo al guardado.

En la siguiente secuencia se observa el trabajo realizado en aquellos cambios con nuevas librerías añadidas al environment y nueva funcionalidad añadida en el notebook.

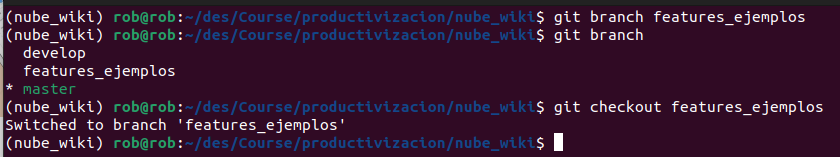
Comienzo con el workflow en diferentes ramas. Siempre compruebo en que rama estoy situado antes de realizar cualquier acción. En la siguiente secuencia creo la rama develop y me sitúo en ella para realizar los cambios.

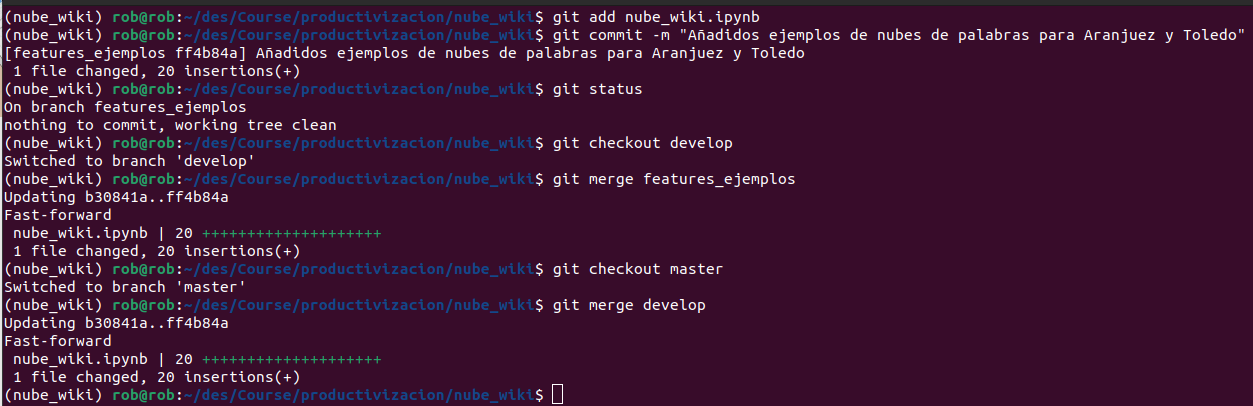


Añado el método *crea\_nube* y actualizo develop.

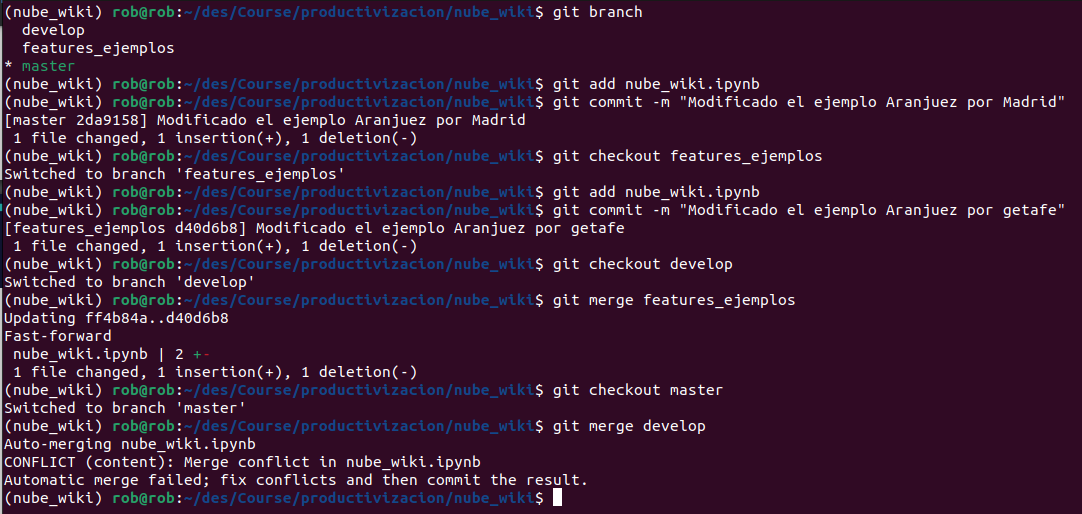
Cambio a master y realizo el merge con develop.

Creo la nueva rama features\_ejemplos y me sitúo en ella.

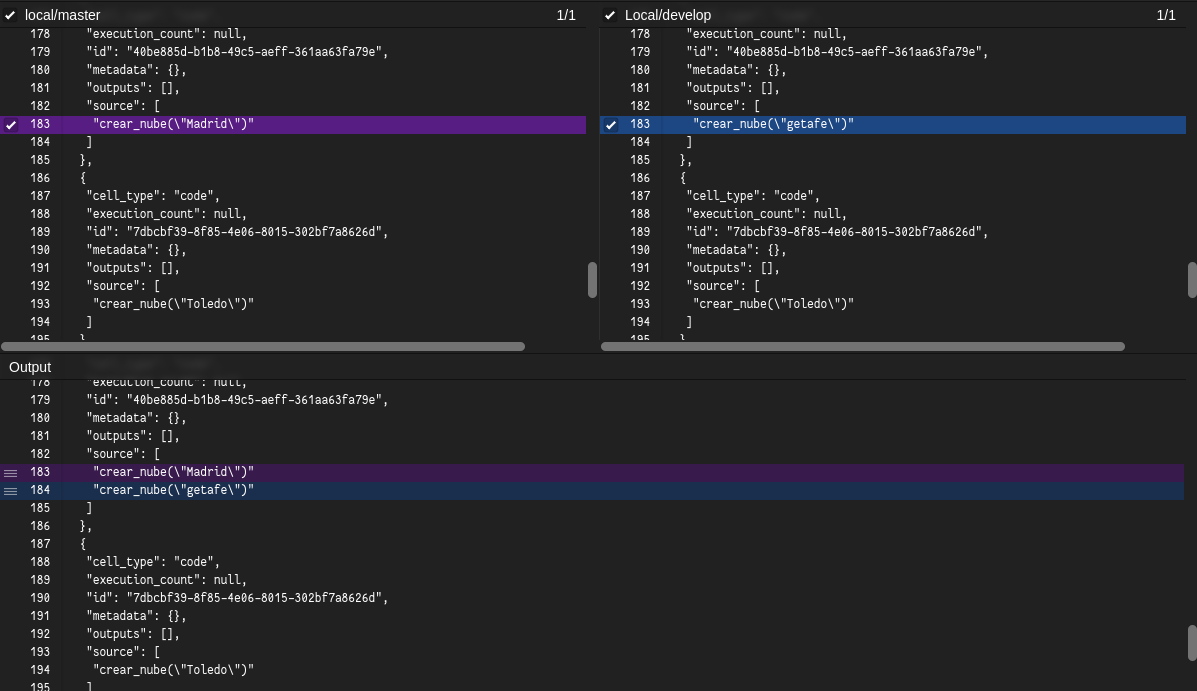


La siguiente secuencia contiene el gitflow desde features hasta master, como no hay conflictos en los cambio el proceso es automático.

En los siguientes pasos de la práctica se modifica en Master y en features la misma línea de código por lo que al realizar el gitflow como en el paso anterior se produce un conflicto entre la rama Master y los cambios realizados en features que posteriormente pasan a develop



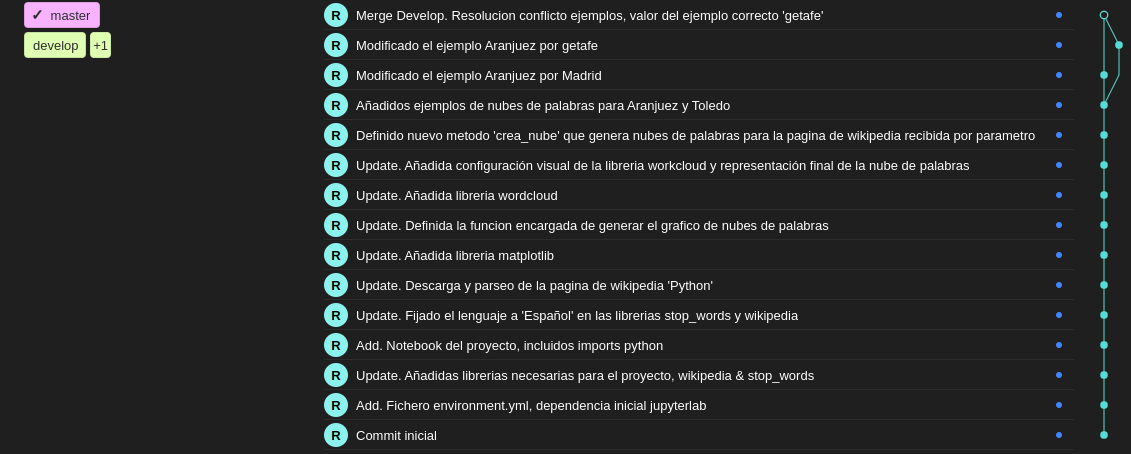
Git no es capaz de resolver el conflicto de manera automática por lo que para determinar que cambio es el correcto hay que resolverlo de manera manual, bien desde un editor grafico como gitfiend o directamente en el fichero que contiene el conflicto



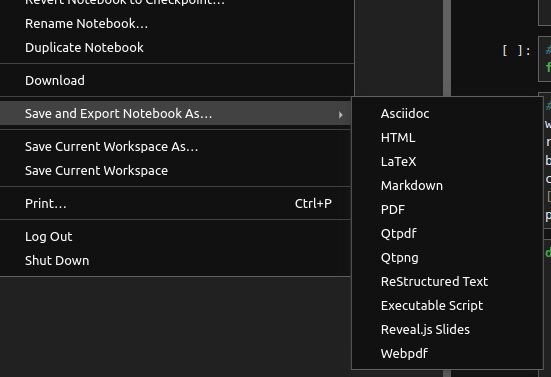


En este caso me quedo con el cambio que incluye el literal ‘getafe’ y completo el merge indicando la resolución en el commit.

El aspecto final del repositorio git seria el siguiente



Por último genero el ejecutable del proyecto exportando desde jupyter lab como script ejecutable que me genera un fichero .py que añado al repositorio.



Añado el fichero .py y el docx con la memoria al repositorio en un nuevo commit. Para evitar que se suban ficheros temporales de python y de cambios en el docx actualizo el .gitignore

\*.pyc

\*.swp

.~\*