Development Environment

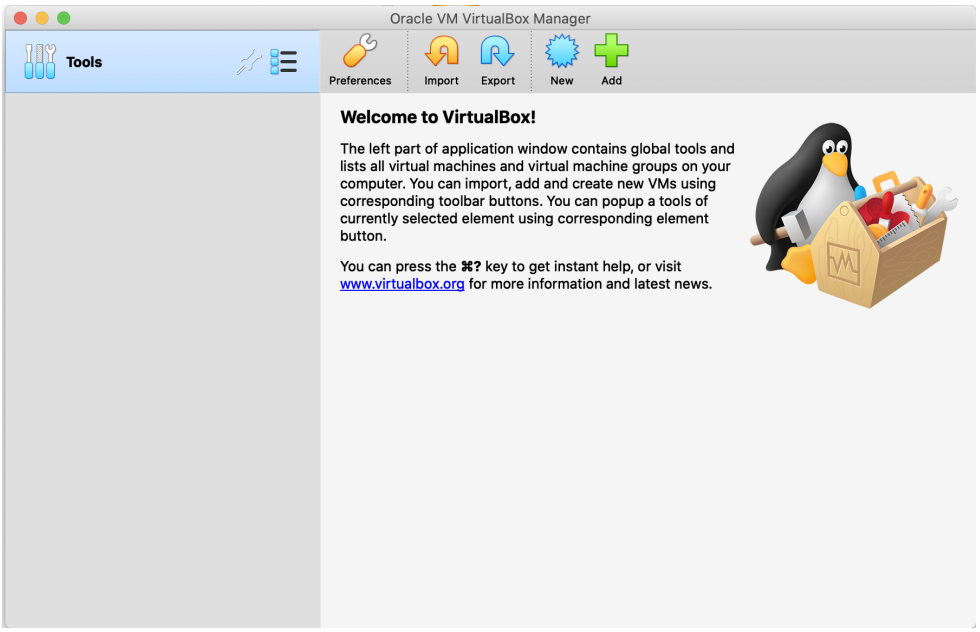
Installing the VM

1. Go to the virtualbox page: https://www.virtualbox.org/wiki/Downloads
2. Select your operating system, to download VirtualBox

   VirtualBox 6.1.16 platform packages

   - Windows hosts
   - OS X hosts
   - Linux distributions
   - Solaris hosts

3. Once it is downloaded, double click on it and follow the instructions to install it on your computer
4. Once installed, open Virtual Box and you should obtain the following output



If you already have Virtual Box installed, make sure your version is up to date (at least 6.1.14)

Well done, you just've installed VirtualBox successfully!

Setting up the Virtual Machine

Now we have Virtual Box, we need to install and set up the virtual machine.

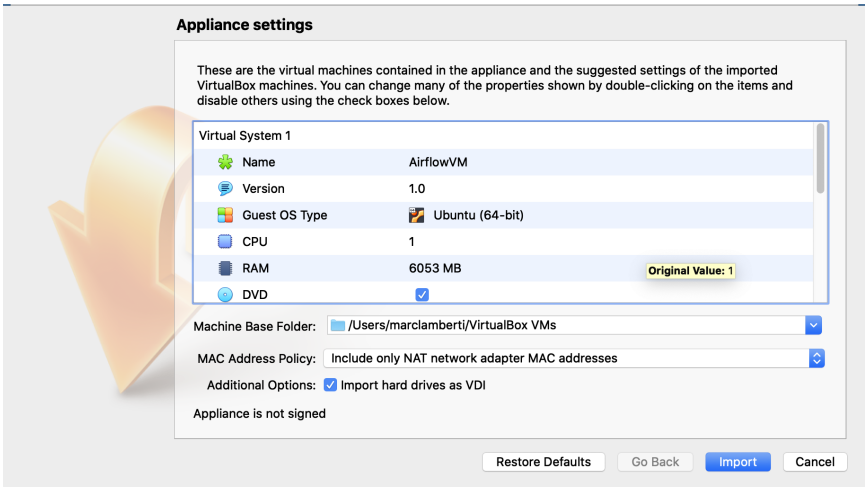
At the end, we will have a Linux distribution (Ubuntu) running in VirtualBox, so inside our computer.

To do this, go to https://marclamberti.wetransfer.com/downloads/b1057cf63e657b355cdf4eae70e65e66202101131250 28/663de9

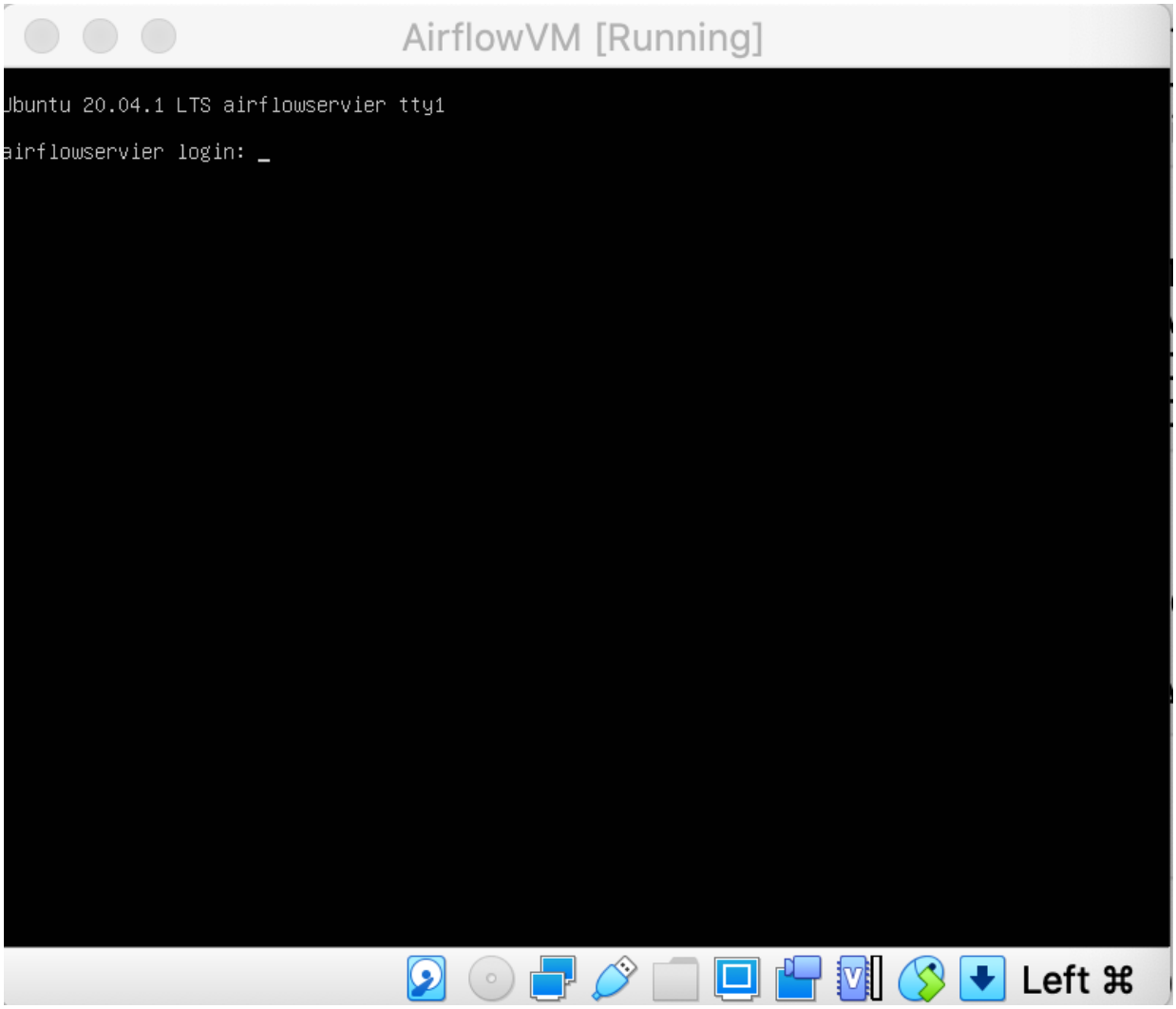
Once you have downloaded the Virtual Machine, you should have a file called **AirflowVM.ova**

- Double click on it

You might NEED to uncheck Import hard drives as VDI if case you get an error after importing it related to 'medium'



- Click on "Start" and wait for the VM to start until you get the following output from it



Well done, the VM is ready!

Quick check: Open your web browser and go to localhost:8080. If you don't get an empty page, you might already having something running on that port. You need to stop it as we will use that port for Airflow

SSH and Visual Studio Code

VirtualBox is installed, the VM is set up, there is one more step.

Access the VM in SSH and connect Visual Studio Code through it.

If you don't know what SSH is, it's a protocol to connect two machines over a network in a secured way.

- Open your terminal and type: *ssh -p 2222 airflow@localhost*

You should obtain the following output:

```
→  ~ ssh -p 2222 airflow@localhost
The authenticity of host '[localhost]:2222 ([127.0.0.1]:2222)' can't be established.
ECDSA key fingerprint is SHA256:jtJ/kw1lF1ZfzcrqZR2yBNXYlZT/fQgSOZkT99IbYJs.
Are you sure you want to continue connecting (yes/no/[fingerprint])?
```

Type: *yes*

Then you need to enter the password: *airflow*

And you should be connected to the VM as shown below

```
airflow@localhost's password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Mon Nov 30 09:11:31 UTC 2020

  System load:  0.0               Processes:               128
  Usage of /:   34.5% of 18.57GB  Users logged in:          0
  Memory usage: 6%                IPv4 address for docker0: 172.17.0.1
  Swap usage:   0%                IPv4 address for enp0s3:  10.0.2.15


81 updates can be installed immediately.
0 of these updates are security updates.
To see these additional updates run: apt list --upgradable


Last login: Mon Nov 30 09:03:54 2020 from 10.0.2.2
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

airflow@airflowvm:~$
```

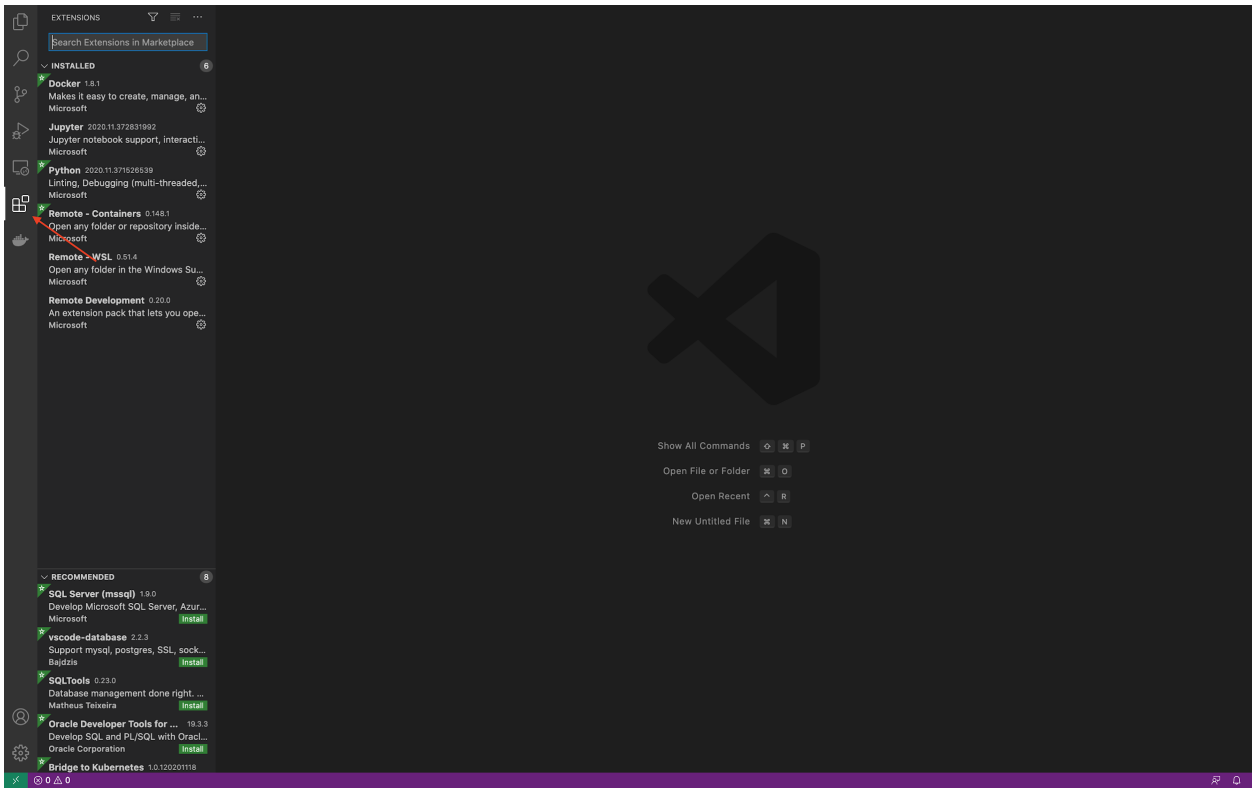
At this point you are **inside** the VM connected through the user **airflow**

Hit control-D to exit the VM.

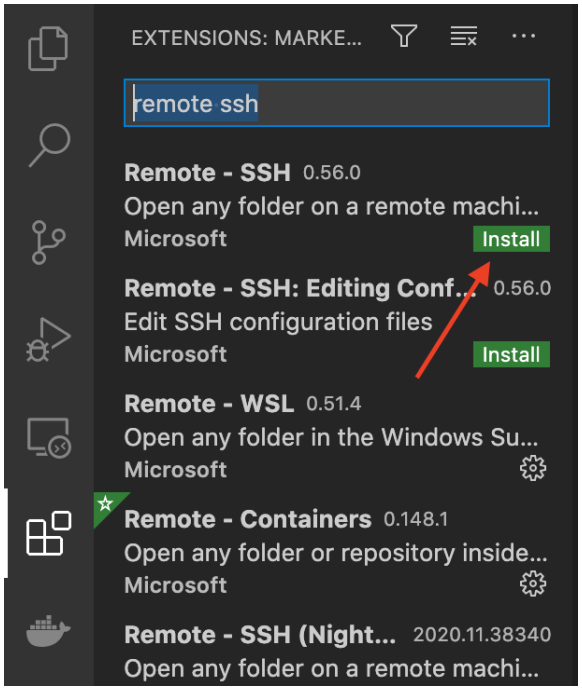
Ok, last step, set up Visual Studio Code to edit files/folders in the VM.

Open Visual Studio Code (If you don't have it, you should 😌
https://code.visualstudio.com/

Click on "Extensions"



And in the search bar on the top, look for "**remote ssh**" and Install it



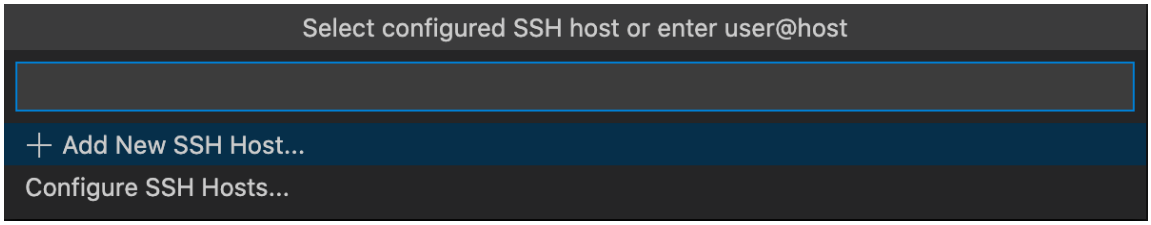
Once the plugin is installed, open it by hitting

- Cmd-P (on Mac)

- F1 (on Windows)

and type *>remote-ssh*

Then hit enter, and select "Add New SSH host..."



Enter the following command

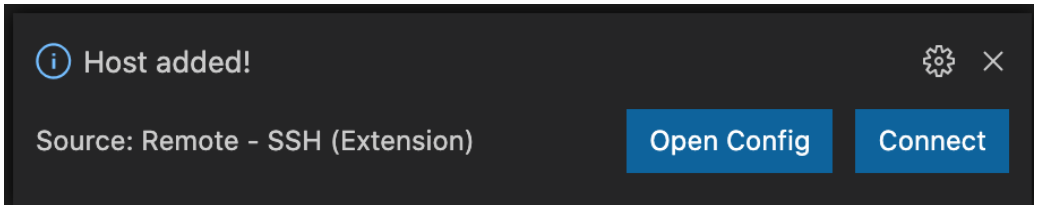
ssh -p 2222 airflow@localhost

and hit enter.

Password is also airflow

Select the first proposition for the SSH config file

You should see this message box on the bottom right of VSCode:



Now the connection has been added, open the plugin again

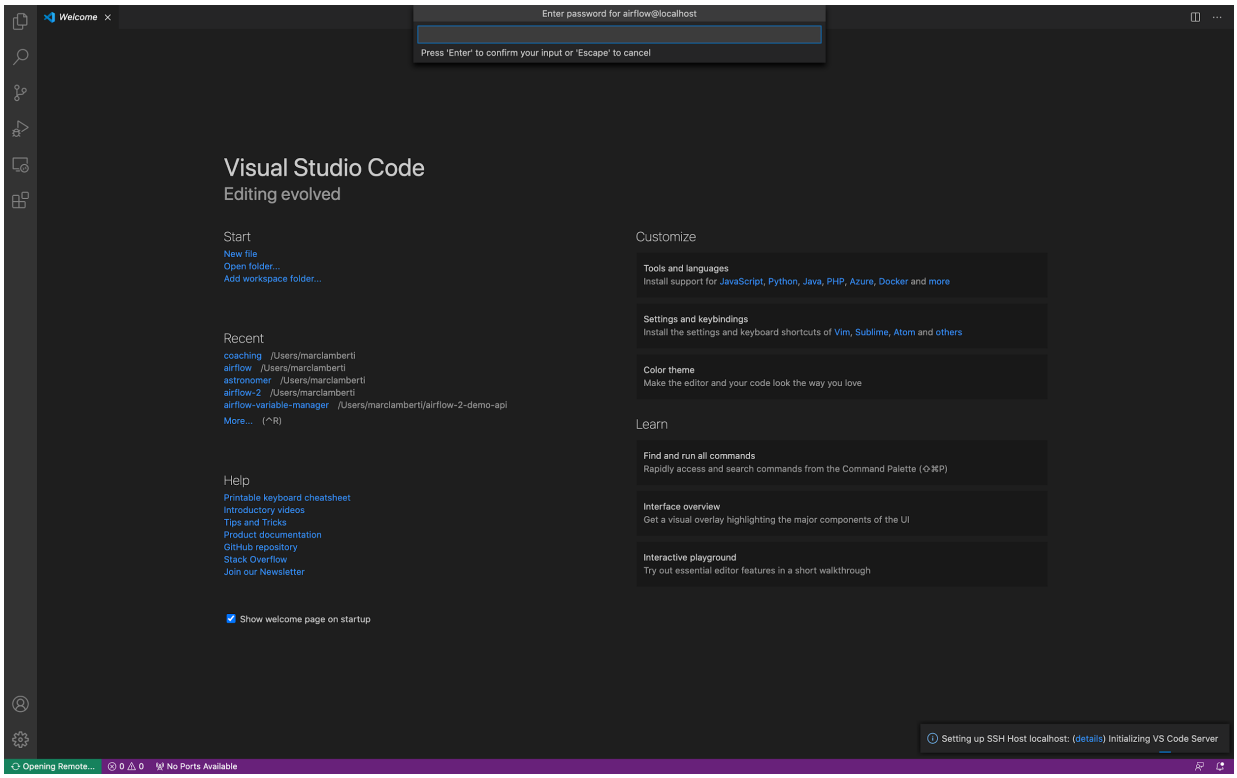
- Cmd-P (on Mac)

- F1 (on Windows)

and type *>remote-ssh*

Then choose "localhost" by hitting enter.

A new Visual Studio Code window opens, type the password *airflow*



And YOU ARE CONNECTED! Congrats! 😎

python3 -m venv sandbox
source sandbox/bin/activate

pip3 install apache-airflow==2.1.0 --constraint
https://gist.githubusercontent.com/marclamberti/742efaef5b2d94f44666b0aec020be7c/raw/21c8
8601337250b6fd93f1adceb55282fb07b7ed/constraint.txt

Initialize the Metastore DB

airflow db init

Start the Web Server

airflow webserver --port 8080

Create Admin User

airflow users create -u admin -p admin -f Marc -l saurav -r Admin -e admin@airflow.com

Start the Scheduler

airflow scheduler

IGNORE BELOW HIGHLIGHTED IN YELLOW FOR NOW

Start a Worker Node
If you are in distributed mode (Celery)
airflow worker

Print the List of Active DAGs
airflow list_dags

Print the List of Tasks of the dag_id
airflow list_tasks dag_id
Exemple:
airflow list_tasks hello_world

Print the Hierarchy of Tasks in the dag_id
airflow list_tasks dag_id --tree
Exemple:
airflow list_tasks hello_world --tree

Test your Tasks in your DAG
airflow test dag_id task_id execution_date
Exemple:
airflow test hello_world hello_task 2018-10-05

Airflow 2.0 is composed of multiple separated but connected packages with a Core package apache-airflow and providers.

A provider is an independent python package that brings everything your need to interact with a service or a tool such as Spark or AWS.

It contains connection types, operators, hooks and so on.

By default, some operators are pre installed by default such as the PythonOperator and the BashOperator but for the others you will have to install the corresponding prodiver.

Now, you can install only the operators you need (no more 200 different dependencies to deal with whereas you just use 3 or 4 operators). If there is a new version of your operator, you just have to update the provider and not your Airflow instance like before. On top of that, it's never been easy to create your own provider.

Much better isn't it?

That being said, in the next video you will use the SimpleHTTPOperator. That operator isn't installed by default so you to install the providers that contains it. For that, go to https://airflow.apache.org/docs/ and under Provider Packages, you get the list of all Airflow integrations.

Click on Hypertext Transfer Protocol.

You should land on that page
https://airflow.apache.org/docs/apache-airflow-providers-http/stable/index.html

Scroll down until you reach the section Installation.

To install the provider, in your python virtual environment, execute the command

pip install apache-airflow-providers-http==2.0.0

Once the install is done, you ar nowe able to import and use the SimpleHTTOperator. In addition, the connection type HTTP becomes available.

Each time you want to interact with a service or a tool, take a look at the provider list.

Also, to know which providers are already installed, type

airflow providers list

airflow tasks test user_processing creating_table 2020-01-01

airflow tasks test user_processing is_api_available 2020-01-01

```
from airflow.models import DAG
from airflow.providers.sqlite.operators.sqlite import SqliteOperator
from airflow.providers.http.sensors.http import HttpSensor
from airflow.providers.http.operators.http import SimpleHttpOperator
from airflow.operators.python import PythonOperator
from airflow.operators.bash_operator import BashOperator
from datetime import datetime
#from sqlalchemy import create_engine
import json
import os
import pandas as pd

default_args = {
    'start_date': datetime(2020, 1, 1),
}

def _processing_user(ti):
    users = ti.xcom_pull(task_ids=['extracting_user'])
    if not len(users) or 'results' not in users[0]:
        raise ValueError('User is empty')
    user = users[0]['results'][0]
    processed_user = pd.json_normalize({
        'firstname': user['name']['first'],
        'lastname': user['name']['last'],
        'country': user['location']['country'],
```

```
        'username': user['login']['username'],
        'password': user['login']['password'],
        'email': user['email']
    })
    processed_user.to_csv('/tmp/processed_user.csv', index=None,
header=False)


with DAG('user_processing', schedule_interval='@daily',
         default_args=default_args,
         catchup=False) as dag:
    # define tasks/operators
    create_table = SqliteOperator(
        task_id='creating_table',
        sqlite_conn_id='db_sqlite',
        sql='''
            CREATE TABLE users (
                firstname TEXT NOT NULL,
                lastname TEXT NOT NULL,
                country TEXT NOT NULL,
                username TEXT NOT NULL,
                password TEXT NOT NULL,
                email TEXT NOT NULL PRIMARY KEY
            );
        ''',
    )


    is_api_available = HttpSensor(
        task_id='is_api_available',
        http_conn_id='user_api',
        endpoint='api/'
    )

    extracting_user = SimpleHttpOperator(
        task_id='extracting_user',
        http_conn_id='user_api',
        endpoint='api/',
        method='GET',
        response_filter=lambda response: json.loads(response.text),
```

```
        log_response=True
    )

    processing_user = PythonOperator(
        task_id='processing_user',
        python_callable=_processing_user
    )

    #storing_user = PythonOperator(
   #     task_id='storing_user',
  #      python_callable=_csvToSql
   # )

    storing_user = BashOperator(
        task_id='storing_user',
        bash_command='echo -e ".separator ","\n.import
/tmp/processed_user.csv users" | sqlite3 /home/airflow/airflow/airflow.db'
        #python_callable=_csvToSql
    )









create_table >> is_api_available >> extracting_user >> processing_user >>
storing_user
```