Компоненты Windows WSL.

wsl --unregister Ubuntu

WSL

cd D:\

mkdir WSL

cd WSL

Invoke-WebRequest -Uri https://aka.ms/wsl-ubuntu-1804 -OutFile Ubuntu.appx -UseBasicParsing

Invoke-WebRequest -Uri https://aka.ms/wslubuntu2204 -OutFile Ubuntu.appx -UseBasicParsing

Invoke-WebRequest -Uri <https://aka.ms/wslubuntu2004> -OutFile Ubuntu.appx -UseBasicParsing

move .\Ubuntu.appx .\Ubuntu.zip

Expand-Archive .\Ubuntu.zip

cd .\Ubuntu\

<https://wslstorestorage.blob.core.windows.net/wslblob/wsl_update_x64.msi>

wsl.exe –install wsl.exe –update

sudo apt update

sudo apt upgrade

sudo add-apt-repository ppa:deadsnakes/ppa

sudo apt install python3.8

sudo apt install python3.8-distutils

wget https://bootstrap.pypa.io/get-pip.py

sudo python3.8 get-pip.py

sudo apt-get install mysql-server

sudo /etc/init.d/mysql start

sudo mysql\_secure\_installation

sudo mysql

sudo mysql -u root -p

ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql\_native\_password BY '1';

sudo service mysql stop

sudo service mysql restart

SHOW

DATABASES;

ALTER

USER 'root'@'localhost' IDENTIFIED WITH mysql\_native\_password BY 'root';

sudo nano /etc/mysql/my.cnf

[mysqld]

port = 33061

sudo service mysql restart

sudo python3.8 -m pip install --force-reinstall apache-airflow==2.5.1

CREATE DATABASE Airflow CHARACTER SET UTF8mb3 COLLATE utf8\_general\_ci;

CREATE USER 'Airflow'@'%' IDENTIFIED BY '1';

export AIRFLOW\_HOME=/home/airflow && airflow db init

load\_examples = False

executor = LocalExecutor

sql\_alchemy\_conn = mysql://Airflow:1@localhost:33061/Airflow

catchup\_by\_default = False

sudo apt-get install python3.8-dev libmysqlclient-dev && sudo apt install pkg-config && pip install mysqlclient

chmod -R 777 ./

airflow scheduler -D

airflow webserver -p 8080 -D

airflow users create --role Admin --username admin --email admin --firstname admin --lastname admin --password admin

pip install werkzeug==2.3

sudo apt update

Step 4: Install Dependencies

Install required dependencies for Apache Airflow:

sudo apt install -y python3 python3-pip python3-venv

Step 5: Create a Virtual Environment

Create and activate a virtual environment to isolate the Apache Airflow installation:

python3 -m venv airflow-venv && source airflow-venv/bin/activate

Step 6: Install Apache Airflow

Install Apache Airflow using `pip`:

pip install apache-airflow==2.7.3

pip install apache-airflow==2.7.3 --constraint "https://raw.githubusercontent.com/apache/airflow/constraints-2.7.3/constraints-3.8.txt" --force-reinstall --upgrade

Step 7: Initialize the Database

Initialize the Airflow metadata database:

export AIRFLOW\_HOME=/home/airflow && airflow db init

Step 8: Start the Web Server and Scheduler

Start the Airflow web server and scheduler:

export AIRFLOW\_HOME=/home/airflow && (airflow scheduler & airflow webserver -p 8080)

Кратко об airflow.  
Базовое представление об airflow можно получить в этой статье <https://www.bigdataschool.ru/blog/dag-run-scheduling-in-airflow-some-features.html> От себя добавлю, что преимущество airflow это -

1. возможность видеть все наши скрипты в одном месте.
2. возможность в режиме реального времени видеть, что запущено, что отработано успешно или с ошибкой.
3. видеть длительность работы скриптов и ее динамику.
4. возможность ручного запуска скрипта по требованию через гуи.
5. возможность параллельного и последовательного запуска скриптов, возможность хранить расписание многих скриптов в одном питоновском файле и др.

**Установка:**  
Версия 1.10.15 устанавливается так.  
pip install --trusted-host pypi.org --trusted-host files.pythonhosted.org --force-reinstall apache-airflow==2.2.5  
pip install --trusted-host pypi.org --trusted-host files.pythonhosted.org wtforms==2.3.3  
  
pip install --trusted-host pypi.org --trusted-host files.pythonhosted.org SQLAlchemy==1.3.23  
pip install --trusted-host pypi.org --trusted-host files.pythonhosted.org flask==0.12.4  
pip install --trusted-host pypi.org --trusted-host files.pythonhosted.org markupsafe==2.0.1  
pip install --trusted-host pypi.org --trusted-host files.pythonhosted.org flask-bcrypt  
После этого аирфлоу должен быть установлен.  
Для версии 2.2.5 достаточно:  
pip install --trusted-host pypi.org --trusted-host files.pythonhosted.org --force-reinstall apache-airflow  
Далее нужно прописать airflow\_home в nano ~/.bashrc  
export AIRFLOW\_HOME=/home/airflow  
В этой папке будет храниться конфиг и скрипты питона, которые будут содержать даги (**Directed Acyclic Graph**) расписания, но об этом позже. Для работы аирфлоу нужна база данных, в этой базе будут хранится даги расписания, юзеры, логи, что отработало успешно, а что нет. База данных может быть встроенная, как sqllite, но в этом случае airflow будет работать в однопоточном режиме, что вызовет проблемы с планировщиком. Поэтому будем хранить данные airflow на нашем mysql сервере. Для этого нужно создать схему и юзера под аирфлоу.  
CREATE DATABASE Airflow;  
CREATE USER 'Airflow'@'%' IDENTIFIED BY '1';  
GRANT USAGE,EXECUTE ON . TO 'Airflow'@'%';  
GRANT ALL PRIVILEGES ON Airflow.\* TO 'Airflow'@'%';  
Для новых версий аирфлоу (2.2.5) схему нужно создавать так:  
CREATE DATABASE airflow CHARACTER SET UTF8mb3 COLLATE utf8\_general\_ci  
После этого, пишем в определенном порядке следующие команды.  
export AIRFLOW\_HOME=/home/airflow && airflow db init  
airflow scheduler -D && airflow webserver -p 8080 -D  
До этого нужно еще написать правильный конфиг, приложу его в этом ишью В этом конфиге нужно прописать  
load\_examples = Falseexecutor = LocalExecutorsql\_alchemy\_conn = mysql://Airflow:пароль@localhost:3306/Airflowcatchup\_by\_default = False и др настройки.  
Параметр catchup\_by\_default - важный параметр, если будет True и даги настроены неправильно со слишком частым обновлением и если их активировать все разом, то это вызовет высокую нагрузку на сервер, где возможно потребуется перезагрузка.  
Установка расписания:\textcolor{blue}{\text{Установка расписания:}}Установка расписания:  
Само расписание живет в питоновском файле, класть его нужно в home/airflow/dags. У себя под каждый скрипт сделал отдельный даг, но каждый даг сам по себе может содержать и запускать несколько задач, которые могут запускаться параллельно или последовательно. Если расписание у задач разное, то их следует распределить по отдельным дагам.  
**Пример одного дага и кода представлен ниже**  
`from airflow import DAG  
from airflow.operators.bash import BashOperator  
from datetime import datetime, timedelta  
import pendulum  
default\_args = {  
'owner': 'AGanshin',  
'depends\_on\_past': False,  
'start\_date': pendulum.datetime(year=2022, month=6, day=1).in\_timezone('Europe/Moscow'),  
'email': ['[A.V.Ganshin@mgts.ru](mailto:A.V.Ganshin@mgts.ru)'],  
'email\_on\_failure': False,  
'email\_on\_retry': False,  
'retries': 0,  
'retry\_delay': timedelta(minutes=5)  
}

dag1 = DAG('AGanshin001',  
default\_args=default\_args,  
description="1528\_Causes\_MTSdts07",  
catchup=False,  
schedule\_interval='0 6 \* \* \*')  
task1 = BashOperator(  
task\_id='1528\_Causes\_MTSdts07',  
bash\_command='python3.6 /home/aganshin/1528\_Causes\_MTSdts07.py',  
dag=dag1)`  
Наиболее простой **способ записать несколько задач** в даг - объединив из на уровне баш. Так по сути в аирфлоу у нас будет одна задача. И тем самым мы можем структурировать большое количество скриптов. Например так:  
dag9 = DAG('Other009',  
default\_args=default\_args,  
description='mrekunchak\_part2',  
catchup=False,  
schedule\_interval='10 5 1 \* \*')  
q9='''  
python3.6 /home/mrekunchak/\_mailers/425\_ote\_otpkk\_otpmr\_repeat.py &  
python3.6 /home/mrekunchak/\_mailers/465\_otpmr\_solved.py &  
python3.6 /home/mrekunchak/reporter\_proj/month\_Collector.py ''' task9 = BashOperator(  
task\_id='mrekunchak\_part2',  
bash\_command=q9.replace("\n", " "),  
dag=dag9)  
Далее **кладем наш скрипт в home/airflow/dags** Переходим по <http://localhost:8080> и вводим логин и пароль.  
По умолчания даги не активированы, **снимаем их с паузы**  
состава задач просто редактируем питоновский скрипт. Аирфлоу автоматически его подхватит.

Хоть без дополнительных демонов(daemons) линукса airflow работает стабильно, но сбои возможны. При перезагрузке mysql сервера, при израсходовании Ram. Поэтому для обеспечения стабильной работы необходимо установить специальный daemon для airflow, который в случае остановки процессов планировщика и вебсервера перезапустит их автоматически.  
Для **установки демона** сделал:  
Три команды на **создание пустых файлов**.  
sudo touch /usr/lib/systemd/system/airflow-webserver.service  
sudo touch /usr/lib/systemd/system/airflow-scheduler.service  
sudo touch /etc/sysconfig/airflow  
Далее через nano вписываем  
nano /usr/lib/systemd/system/airflow-webserver.service  
**Для вебсервера**  
[Unit]  
Description=Airflow webserver daemon  
After=network.target postgresql.service mysql.service redis.service rabbitmq-server.service  
Wants=postgresql.service mysql.service redis.service rabbitmq-server.service

[Service]  
EnvironmentFile=/etc/sysconfig/airflow  
User=root  
Group=root Type=simple  
ExecStart=/usr/local/bin/airflow webserver --pid /run/airflow/webserver.pid  
Restart=on-failure  
RestartSec=5s  
PrivateTmp=true

[Install]  
WantedBy=multi-user.target

nano /usr/lib/systemd/system/airflow-scheduler.service  
**Для планировщика**  
[Unit]  
Description=Airflow scheduler daemon  
After=network.target postgresql.service mysql.service redis.service rabbitmq-server.service  
Wants=postgresql.service mysql.service redis.service rabbitmq-server.service

[Service]  
EnvironmentFile=/etc/sysconfig/airflow  
User=root  
Group=root  
Type=simple ExecStart=/usr/local/bin/airflow scheduler  
Restart=always  
RestartSec=5s

[Install]  
WantedBy=multi-user.target

nano /etc/sysconfig/airflow  
**Указываем пусть к airflow\_home**  
AIRFLOW\_CONFIG=/home/airflow/airflow.cfg  
AIRFLOW\_HOME=/home/airflow  
**Создаем папку ран для PIDs сервера**  
mkdir /run/airflow  
chown root:root /run/airflow  
chmod 0777 airflow -R  
**Останавливаем все процессы аирфлоу**  
kill $(ps -ef | grep "airflow webserver" | awk '{print $2}') && kill $(ps -ef | grep "airflow scheduler" | awk '{print $2}') && airflow scheduler -D && airflow webserver -p 8080 -D  
и проверяем чтобы было чисто  
sudo lsof -i tcp:8080  
**Подгружаем наш демон для аирфлоу**  
sudo systemctl daemon-reload  
и запускаем аирфлоу  
export AIRFLOW\_HOME=/home/airflow && airflow db init  
systemctl start airflow-scheduler && systemctl start airflow-webserver  
Проверяем **статус аирфлоу**.  
systemctl status airflow-scheduler && systemctl status airflow-webserver  
Если все хорошо, должно быть что-то вроде этого  
**Для остановки** аирфлоу пишем:  
systemctl stop airflow-scheduler && systemctl stop airflow-webserver  
**Для перезапуска** пишем:  
systemctl restart airflow-scheduler&& systemctl restart airflow-webserver  
Чтобы **посмотреть логи** аирфлоу пишем:  
journalctl -u airflow-scheduler -n 50  
journalctl -u airflow-webserver -n 50  
Полную перезагрузку аирфлоу следует осуществлять следующей командой.  
systemctl stop airflow-scheduler && systemctl stop airflow-webserver && export AIRFLOW\_HOME=/home/airflow && airflow resetdb && systemctl start airflow-scheduler && systemctl start airflow-webserver  
Установка email оповещения о неудачах отработки скриптов:\textcolor{blue}{\text{Установка email оповещения о неудачах отработки скриптов:}}Установка email оповещения о неудачах отработки скриптов:  
Для настройки указанного оповещения, необходимо отредактировать общие настройки дагов в питоновском скрипте по следующему образцу.  
default\_args = {  
'owner': 'AGanshin',  
'depends\_on\_past': False,  
'start\_date': pendulum.datetime(year=2022, month=6, day=1).in\_timezone('Europe/Moscow'),  
**'email': [''],**  
**'email\_on\_failure': True,**  
'email\_on\_retry': False,  
'retries': 0,  
'retry\_delay': timedelta(minutes=5)  
}  
nano ~/.bashrc

export SPARK\_HOME=/home/spark && export PATH=$PATH:$SPARK\_HOME/bin:$SPARK\_HOME/sbin

source ~/.bashrc

sudo apt-get install openjdk-8-jdk

pip install pyspark==3.2.4

pip install pandas==1.5.3

pip install SQLAlchemy==1.4.46

pip install apache-airflow-providers-telegram

<https://api.telegram.org/botайдибота/getUpdates>

{"ok":true,"result":[{"update\_id":794148997,

"message":{"message\_id":5,"from":{"id":920746911,"is\_bot":false,"first\_name":"Alex","username":"A4815162342","language\_code":"en"},"chat":{"id":920746911,"first\_name":"Alex","username":"A4815162342","type":"private"},"date":1706545512,"text":"\u0442\u0435\u0441\u0442"}}]}

/start

/newbot

/tocken

/mybots