Installing Airflow on EMR

This document contains the steps to install and configure Airflow on your EMR cluster. Make sure that you are using the “hadoop” user while following these instructions.

Enter the following command to become the superuser

```
sudo su
```

NOTE: Please proceed with caution while running commands in superuser/ root user mode as the changes made cannot be undone.

Installing Docker:

The airflow installation will make use of docker images. To check if the docker software is installed, use the following command

```
yum list installed | grep docker
```

You’ll get a output below:

```
[root@ip-172-31-71-166 airflow]# yum list installed | grep docker
containerd.x86_64              1.6.19-1.amzn2.0.1          @amzn2extra-docker
docker.x86_64                  20.10.23-1.amzn2.0.1        @amzn2extra-docker
emr-docker-apps.x86_64         1.11.0-1                    @emr-apps
runc.x86_64                    1.1.4-1.amzn2.0.1           @amzn2extra-docker
[root@ip-172-31-71-166 airflow]#
```

Run the following command to find the docker version

```
docker --version
```

The following output will be shown

```
[root@ip-172-31-71-166 airflow]# docker --version
Docker version 20.10.23, build 7155243
[root@ip-172-31-71-166 airflow]#
```

If the docker package is not installed or you get an error while running the above command, install docker by running the following command

```
yum install docker
```

Press ‘y’ when prompted

```
[root@ip-172-31-71-166 airflow]# yum install docker
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
3 packages excluded due to repository priority protections
Resolving Dependencies
--> Running transaction check
---> Package docker.x86_64 0:20.10.23-1.amzn2.0.1 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
 Package     Arch        Version                   Repository              Size
================================================================================
Installing:
 docker      x86_64      20.10.23-1.amzn2.0.1      amzn2extra-docker       41 M

Transaction Summary
================================================================================
Install  1 Package

Total download size: 41 M
Installed size: 163 M
Is this ok [y/d/N]: y
```

```
Installing:
 docker      x86_64      20.10.23-1.amzn2.0.1      amzn2extra-docker       41 M

Transaction Summary
================================================================================
Install  1 Package

Total download size: 41 M
Installed size: 163 M
Is this ok [y/d/N]: y
Downloading packages:
docker-20.10.23-1.amzn2.0.1.x86_64.rpm                    |  41 MB   00:00
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : docker-20.10.23-1.amzn2.0.1.x86_64                          1/1
  Verifying  : docker-20.10.23-1.amzn2.0.1.x86_64                          1/1

Installed:
  docker.x86_64 0:20.10.23-1.amzn2.0.1

Complete!
[root@ip-172-31-71-166 airflow]#
```

Get pip3 if not already installed

```
yum install python3-pip
```

Install Docker-Compose through Pip

```
pip3 install --user docker-compose
```

Download the docker-compose binaries

```
sudo curl -L
https://github.com/docker/compose/releases/latest/download/docker-compose-$(uname
-s)-$(uname -m) -o /usr/local/bin/docker-compose
```

Change the permissions of the docker-compose binaries

```
sudo chmod +x /usr/local/bin/docker-compose
```

Create a system link between the docker-compose binaries

```
ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose
```

Verify docker compose version:

```
docker-compose version
```

Enable docker service at AMI boot time:

```
sudo systemctl enable docker.service
```

Start the Docker service:

```
sudo systemctl start docker.service
```

To get the docker service status on your AMI instance, run:

```
sudo systemctl status docker.service
```

The following output will be shown on the terminal.

```
[root@ip-172-31-71-166 airflow]# sudo systemctl status docker.service
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; vendor prese
t: disabled)
   Active: active (running) since Thu 2023-04-27 09:01:29 UTC; 6s ago
     Docs: https://docs.docker.com
  Process: 5585 ExecStartPre=/usr/libexec/docker/docker-setup-runtimes.sh (code=
exited, status=0/SUCCESS)
  Process: 5569 ExecStartPre=/bin/mkdir -p /run/docker (code=exited, status=0/SU
CCESS)
 Main PID: 5593 (dockerd)
    Tasks: 26
   Memory: 28.1M
   CGroup: /system.slice/docker.service
           ├─5593 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/cont...
           ├─5772 /usr/bin/docker-proxy -proto tcp -host-ip 0.0.0.0 -host-por...
           └─5780 /usr/bin/docker-proxy -proto tcp -host-ip :: -host-port 944...

Apr 27 09:01:28 ip-172-31-71-166 dockerd[5593]: time="2023-04-27T09:01:28.04...c
Apr 27 09:01:28 ip-172-31-71-166 dockerd[5593]: time="2023-04-27T09:01:28.04...c
Apr 27 09:01:28 ip-172-31-71-166 dockerd[5593]: time="2023-04-27T09:01:28.12..."
Apr 27 09:01:28 ip-172-31-71-166 dockerd[5593]: time="2023-04-27T09:01:28.25..."
Apr 27 09:01:28 ip-172-31-71-166 dockerd[5593]: time="2023-04-27T09:01:28.37..."
Apr 27 09:01:29 ip-172-31-71-166 dockerd[5593]: time="2023-04-27T09:01:29.22..."
Apr 27 09:01:29 ip-172-31-71-166 dockerd[5593]: time="2023-04-27T09:01:29.27...3
Apr 27 09:01:29 ip-172-31-71-166 dockerd[5593]: time="2023-04-27T09:01:29.27..."
Apr 27 09:01:29 ip-172-31-71-166 systemd[1]: Started Docker Application Cont....
Apr 27 09:01:29 ip-172-31-71-166 dockerd[5593]: time="2023-04-27T09:01:29.29..."
Hint: Some lines were ellipsized, use -l to show in full.
[root@ip-172-31-71-166 airflow]#
```

Airflow Docker Installation

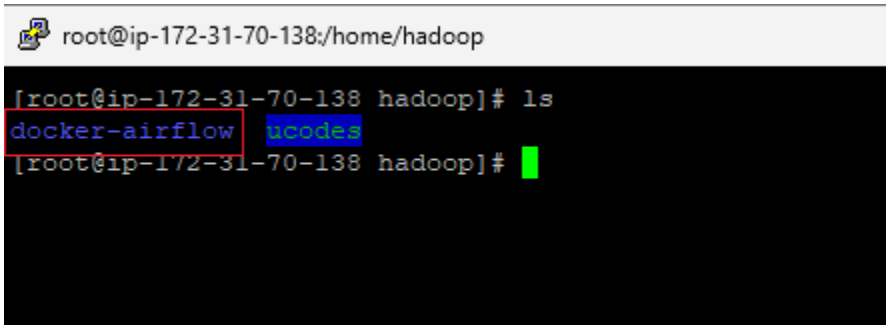
Download the repository from the following Github page - https://github.com/neylsoncrepalde/docker-airflow

```
git clone https://github.com/neylsoncrepalde/docker-airflow.git
```

Using the following command, verify if the files have been downloaded correctly

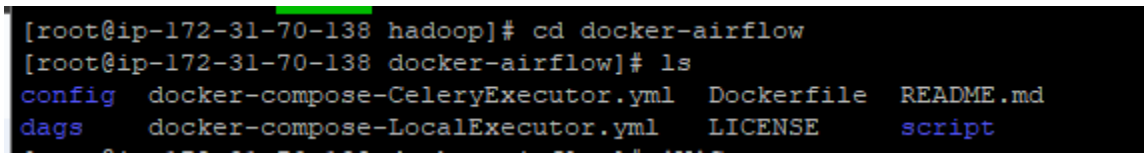
```
ls
```

The git files should be visible in the terminal session



Change directory use the cd command to see the list of files in the directory

```
root@ip-172-31-70-138 hadoop]# ls
root@ip-172-31-70-138 hadoop]# cd docker-airflow
```



You'll see two docker-compose files. The '**docker-compose-CeleryExecutor.yml**' corresponds to the docker-compose file for the Celery Exectuor whereas the '**docker-compose-LocalExecutor.yml'** corresponds to the docker-compose file for the LocalExecutor.
The **dags** directory contains all the custom dags. This dags folder will be mounted to the airflow dags folder when you launch the docker container through the docker-compose file.
The **config** directory contains the configuration files for Airflow.

The airflow service by default runs in the 8080 port but in the AWS EMR cluster, the port 8080 is reserved for Tez UI (ref)

JupyterHub	https://*master-public-dns-name*:9443/
Livy	http://*master-public-dns-name*:8998/
Spark HistoryServer	http://*master-public-dns-name*:18080/
Tez	http://*master-public-dns-name*:8080/tez-ui
YARN NodeManager	http://*coretask-public-dns-name*:8042/
YARN ResourceManager	http://*master-public-dns-name*:8088/
Zeppelin	http://*master-public-dns-name*:8890/

For this reason, we'll be updating the ports in which Airflow can be accessed by editing the dockerfile and docker-compose files.

First edit **docker-compose-CeleryExecutor.yml**

Using the vim editor, open the yaml file for Celery Executor

```
vi docker-compose-CeleryExecutor.yml
```

Scroll down to the airflow webserver section, update the port section

```
            - postgres
            - redis
        environment:
            - LOAD_EX=n
            - FERNET_KEY=46BKJoQYlPPOexq0OhDZnIlNepKFf87WFwLbfzqDDho=
            - EXECUTOR=Celery
            # - POSTGRES_USER=airflow
            # - POSTGRES_PASSWORD=airflow
            # - POSTGRES_DB=airflow
            # - REDIS_PASSWORD=redispass
        volumes:
            - ./dags:/usr/local/airflow/dags
            # Uncomment to include custom plugins
            # - ./plugins:/usr/local/airflow/plugins
        ports:
            - "8080:8080"
        command: webserver
        healthcheck:
            test: ["CMD-SHELL", "[ -f /usr/local/airflow/airflow-webserver.pid ]"]
            interval: 30s
            timeout: 30s
            retries: 3

"docker-compose-CeleryExecutor.yml" 92L, 2851B                    1,1           Top
```

Update the ports mapping to below:

```
8100:8080
```

This maps the port 8080 of the docker container to the 8100 port of the EMR Cluster.

```
    depends_on:
        - postgres
        - redis
    environment:
        - LOAD_EX=n
        - FERNET_KEY=46BKJoQYlPPOexq0OhDZnIlNepKFf87WFwLbfzqDDho=
        - EXECUTOR=Celery
        # - POSTGRES_USER=airflow
        # - POSTGRES_PASSWORD=airflow
        # - POSTGRES_DB=airflow
        # - REDIS_PASSWORD=redispass
    volumes:
        - ./dags:/usr/local/airflow/dags
        # Uncomment to include custom plugins
        # - ./plugins:/usr/local/airflow/plugins
    ports:
        - "8100:8080"
    command: webserver
    healthcheck:
        test: ["CMD-SHELL", "[ -f /usr/local/airflow/airflow-webserver.pid ]"]
        interval: 30s
        timeout: 30s
        retries: 3
```

Type **:wq!** and save the docker-compose file.

SImilarly, open the docker-compose-LocalExecutor.yml by typing the following command

```
vi docker-compose-LocalExecutor.yml
```

```
        restart: always
        depends_on:
            - postgres
            - redis
        environment:
            - LOAD_EX=n
            - FERNET_KEY=46BKJoQYlPPOexq0OhDZnIlNepKFf87WFwLbfzqDDho=
            - EXECUTOR=Celery
            # - POSTGRES_USER=airflow
            # - POSTGRES_PASSWORD=airflow
            # - POSTGRES_DB=airflow
            # - REDIS_PASSWORD=redispass
        volumes:
            - ./dags:/usr/local/airflow/dags
            # Uncomment to include custom plugins
            # - ./plugins:/usr/local/airflow/plugins
        ports:
            - "8080:8080"
        command: webserver
        healthcheck:
            test: ["CMD-SHELL", "[ -f /usr/local/airflow/airflow-webserver.pid ]"]
            interval: 30s
            timeout: 30s
            retries: 3

"docker-compose-CeleryExecutor.yml" 92L, 2851B                    1,1           Top
```

BEFORE PORT MAP UPDATE

And update the port mappings in the airflow webserver section of the YAML file as shown below

```
        restart: always
        depends_on:
            - postgres
        environment:
            - LOAD_EX=n
            - EXECUTOR=Local
        logging:
            options:
                max-size: 10m
                max-file: "3"
        volumes:
            - ./dags:/usr/local/airflow/dags
            # - ./plugins:/usr/local/airflow/plugins
        ports:
            - "8100:8080"
        command: webserver
        healthcheck:
            test: ["CMD-SHELL", "[ -f /usr/local/airflow/airflow-webserver.pid ]"]
            interval: 30s
            timeout: 30s
            retries: 3
```

AFTER PORT MAP UPDATE

Type **:wq!** and save the docker-compose file.

Usage:
We'll use Local Executor for working with the airflow dag scripts.
For LocalExecutor:

```
docker-compose -f docker-compose-LocalExecutor.yml up -d
```

If you want to run another executor, use the other docker-compose.yml files provided in this repository.
For example, for CeleryExecutor:

```
docker-compose -f docker-compose-CeleryExecutor.yml up -d
```

You can verify the successful creation by using the following command

```
docker images
```

```
[root@ip-172-31-70-138 docker-airflow]# docker images
REPOSITORY              TAG                 IMAGE ID       CREATED          SIZE
airflow_image           latest              ba19c0bd14cf   10 seconds ago   1.79GB
python                  3.7-slim-buster     00f71ed3d81c   2 weeks ago      116MB
emr/jupyter-notebook    6.4.0               ca763ad1e352   12 months ago    2.63GB
puckel/docker-airflow   latest              ce92b0f4d1d5   3 years ago      797MB
[root@ip-172-31-70-138 docker-airflow]#
```

To see the status and details of the docker container, type the following command

```
docker ps
```

```
[root@ip-172-31-68-229 docker-airflow]# docker ps
CONTAINER ID   IMAGE                                  COMMAND                  CREATED             STATUS                       PORTS
                                                  NAMES
a64949f49445   neylsoncrepalde/airflow-docker:latest  "/entrypoint.sh webs…"   About a minute ago  Up About a minute (healthy)  5555/tcp, 8793/tcp, 0.0.0.0:8100->808
0/tcp, :::8100->8080/tcp   docker-airflow-webserver-1
3eabd43a1e7e   postgres:9.6                           "docker-entrypoint.s…"   About a minute ago  Up About a minute            5432/tcp
                                                  docker-airflow-postgres-1
80a966eb5f71   emr/jupyter-notebook:6.4.0             "tini -g -- jupyterh…"   57 minutes ago      Up 57 minutes                8888/tcp, 0.0.0.0:9443->9443/tcp, :::
9443->9443/tcp             jupyterhub
[root@ip-172-31-68-229 docker-airflow]#
```

In the above terminal output, note the container_id of the image 'airflow_image'

Execute the following command /bin/bash along with the <container_id> to enter the Airflow CLI/ the shell session for the airflow_image docker container.

```
docker exec -it <container_id> /bin/bash
```

```
[root@ip-172-31-71-166 airflow]# docker exec -it 4b16c40acbbd /bin/bash
airflow@4b16c40acbbd:~$
```

As you can see, the shell session is independent previous session

```
airflow@4b16c40acbbd:~$ whoami
airflow
airflow@4b16c40acbbd:~$
```

Print the current path where you are

```
pwd
```

```
airflow@4b16c40acbbd:~$ pwd
/opt/airflow
airflow@4b16c40acbbd:~$
```

To see the files in the current directory, use the command

```
ls
```

The **airflow.cfg** is the configuration file for the Airflow Webserver.
The dags in airflow contain all the DAG files that you'll use in Airflow.

```
airflow@4b16c40acbbd:~$ ls
airflow-webserver.pid  airflow.cfg  airflow.db  dags  logs  webserver_config.py
airflow@4b16c40acbbd:~$
```

NOTE:
You'll be required to install the PyPI packages for working with Sqoop, Spark and Hive.
Enter the Airflow CLI using the command,

```
docker exec -it <container_id> /bin/bash
```

And install the required packages using the following commands:

```
pip install apache-airflow-providers-apache-sqoop
pip install apache-airflow-providers-apache-spark
pip install apache-airflow-providers-common-sql
pip install hmsclient
pip install pyhive
pip install sasl
pip install thrift
pip install apache-airflow-providers-apache-hive
```

NOTE: To exit from the airflow shell session in the terminal, press Ctrl + D

```
airflow@4b16c40acbbd:~$
exit
[root@ip-172-31-71-166 airflow]#
```

NOTE: The docker-compose file mounts a local folder with the airflow container.

```
  webserver:
      image: neylsoncrepalde/airflow-docker:latest
      restart: always
      depends_on:
          - postgres
      environment:
          - LOAD_EX=n
          - EXECUTOR=Local
      logging:
          options:
              max-size: 10m
              max-file: "3"
      volumes:
          - ./dags:/usr/local/airflow/dags
          - ./data:/usr/local/airflow/data
          # - ./plugins:/usr/local/airflow/plugins
```

This corresponds to the ‘dags’ and ‘data’ folder in the original directory.

```
[hadoop@ip-172-31-68-229 docker-airflow]$ ls
config  docker-compose-CeleryExecutor.yml  LICENSE          scr
dags    docker-compose-LocalExecutor.yml   README.md
data    Dockerfile                         requirements.txt
[hadoop@ip-172-31-68-229 docker-airflow]$ pwd
/home/hadoop/air2/docker-airflow
[hadoop@ip-172-31-68-229 docker-airflow]$
```

Any dag files will be automatically synced with the dags folder in the airflow container.

```
->9443/tcp                jupyterhub
[root@ip-172-31-68-229 docker-airflow]# docker exec -it a64949f49445 /bin/bash
airflow@a64949f49445:~$ pwd
/usr/local/airflow
airflow@a64949f49445:~$ ls
airflow.cfg            dags  logs              unittests.cfg
airflow-webserver.pid  data  requirements.txt  webserver_config.py
airflow@a64949f49445:~$
```

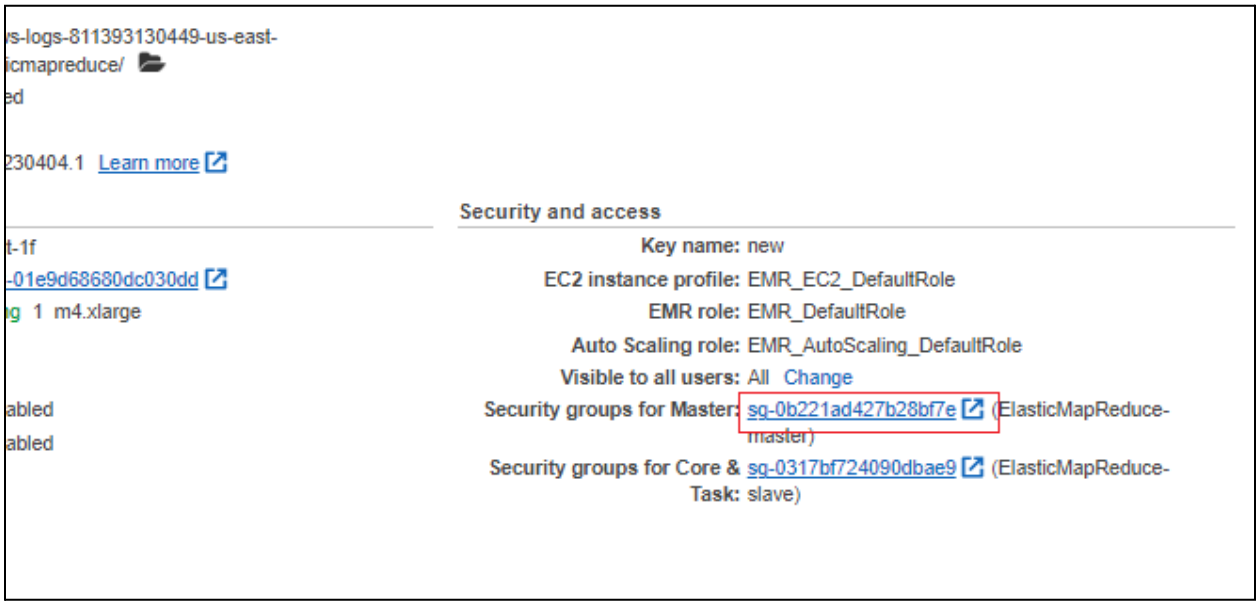
```
airflow@a64949f49445:~$ cd dags
airflow@a64949f49445:~/dags$ ls
sample_bash.py  sample_python.py  sample_spark.py
airflow@a64949f49445:~/dags$
```

SSH Tunneling

To access the Airflow UI, the usual link is localhost:8080. But, as mentioned earlier, the port 8080 is reserved for the Tez UI in the EMR cluster and hence when you try running the docker container through the default port(8080), you'll receive an error message that says that the port is already in use.
Therefore, to resolve this issue we're going to map an unused port (8100 in this case) to the default Airflow port (8080) to access the Airflow UI page via a concept known as **SSH tunnelling**, which allows us to access the UI page through the port 8100.
First you'll need to update the Security Group of the EMR cluster and perform SSH tunnelling through the terminal (Putty/Terminal)

Updating Security Group

In the Summary page of the EMR cluster, navigate to the Security group for the master node.

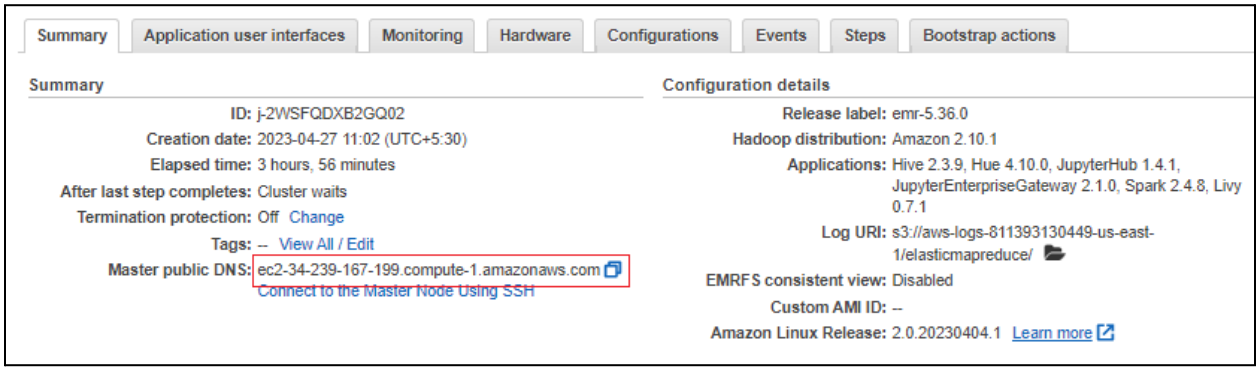


Add the ports to the inbound connections: 8080 and 8100 using the MyIP Source

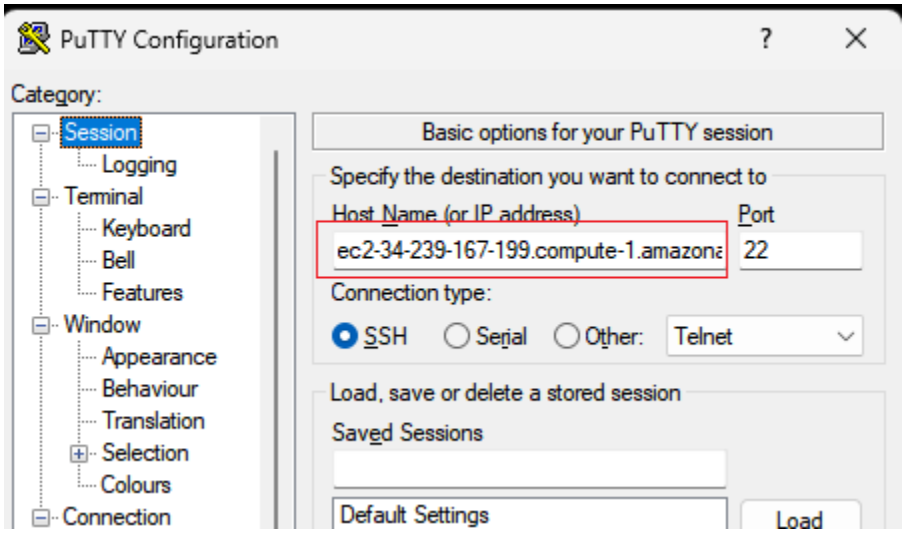
Click on the **Save Rules** button.

SSH Tunneling

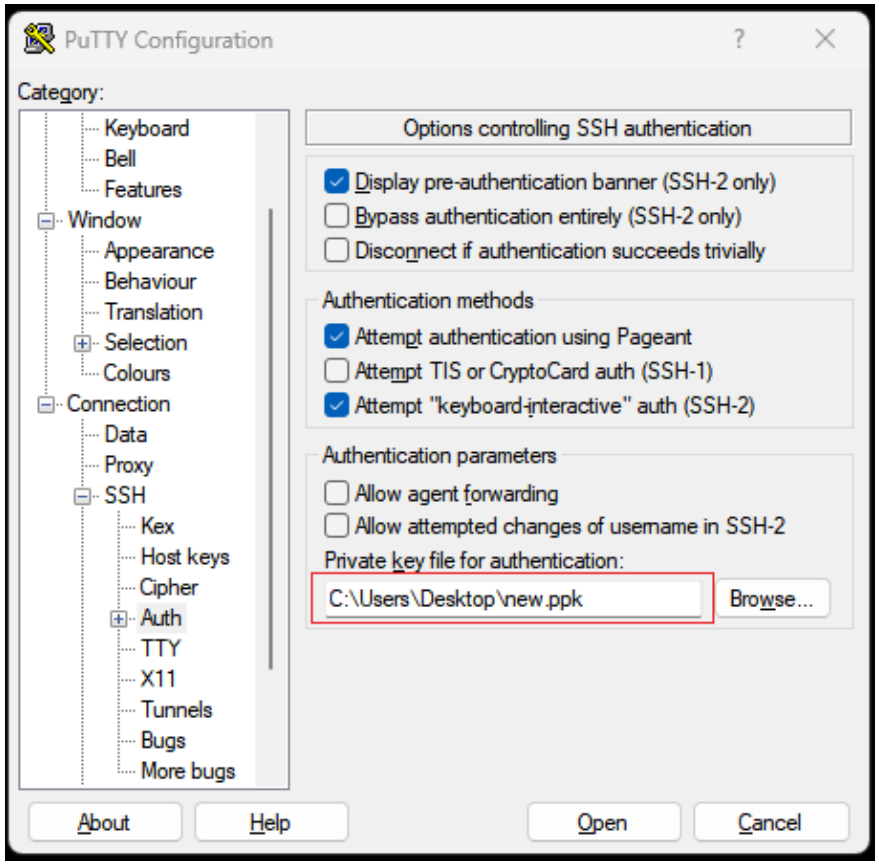
Copy the Master Public DNS of the EMR Cluster to the terminal.



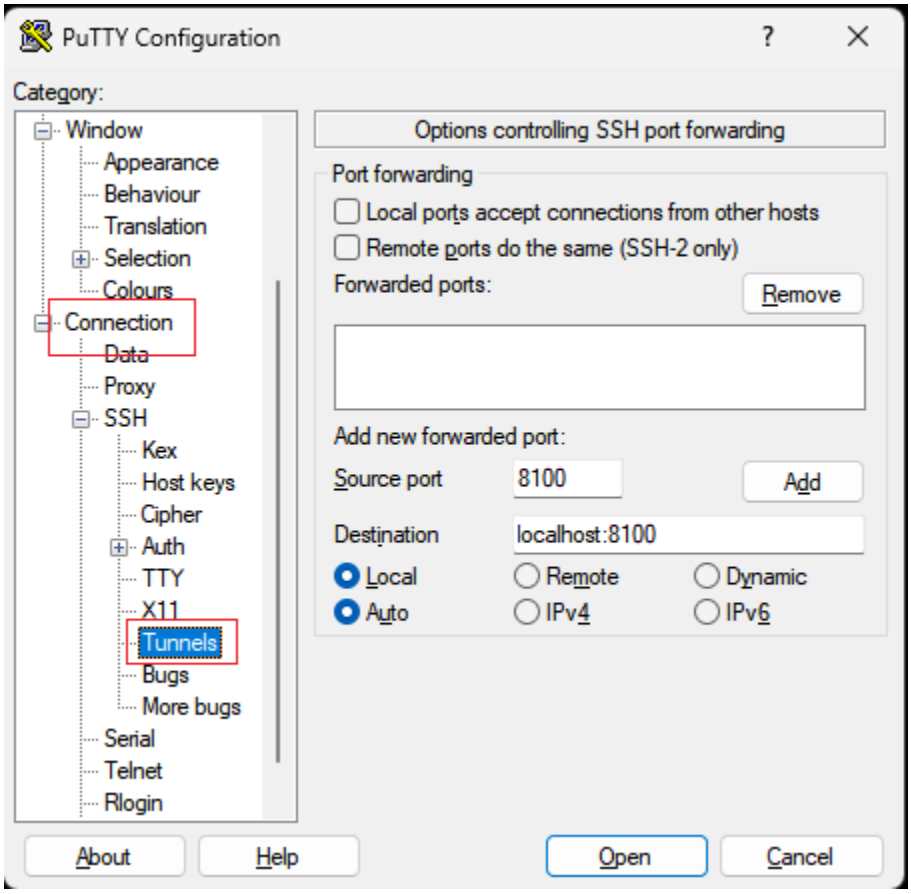
Enter the DNS details in a new terminal session. The following commands are valid for PuTTy in a Windows machine.



Enter the authentication .ppk file



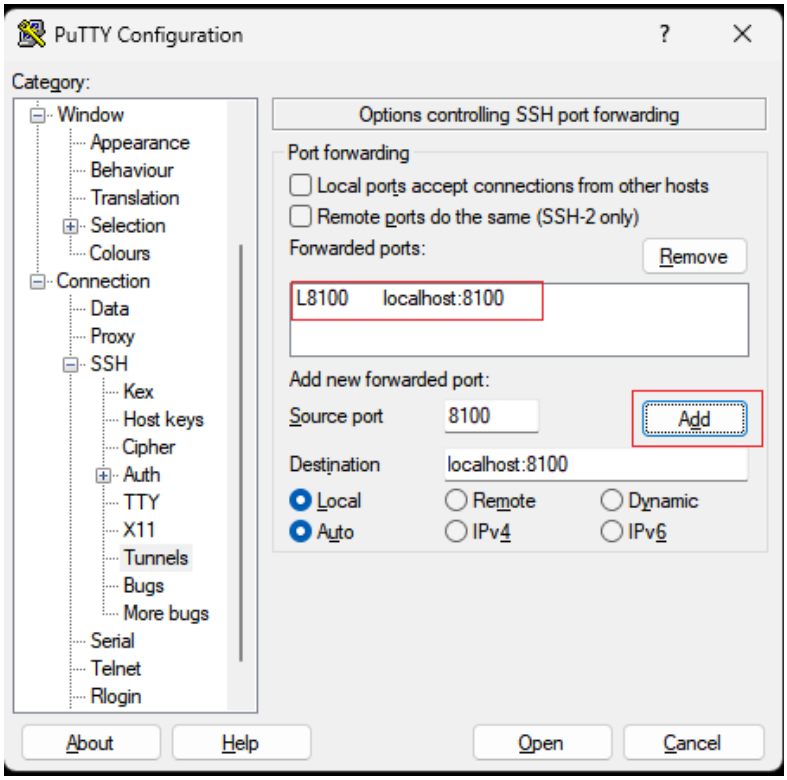
Navigate to SSH option in Connections



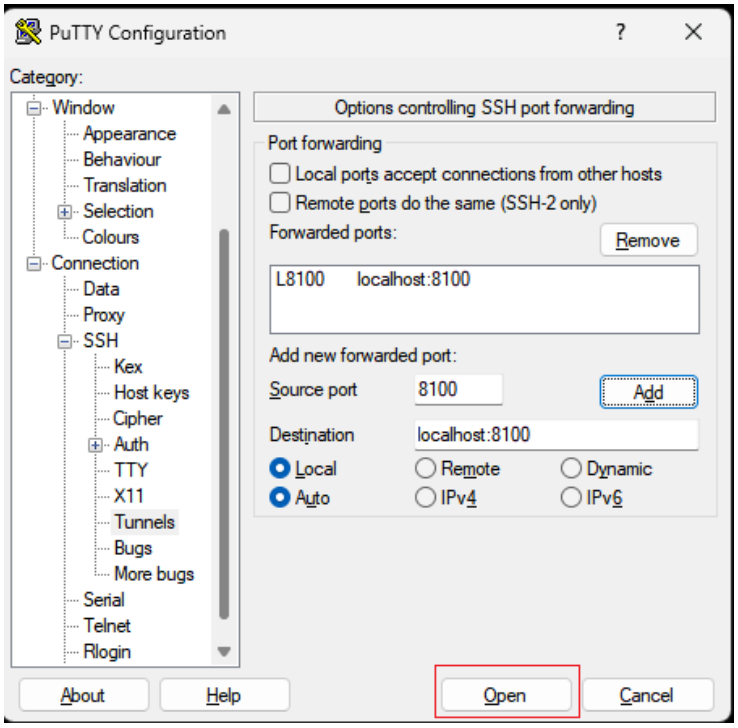
Enter the following details in Source port and Destination

Source port: 8100
Destination: localhost:8100

Keep/ select **'Local'** and **'Auto'** options below the destination.
Click on the 'Add' button. You'll see the following screen when done correctly.



Click on the ‘Open’ button to create a new terminal session.

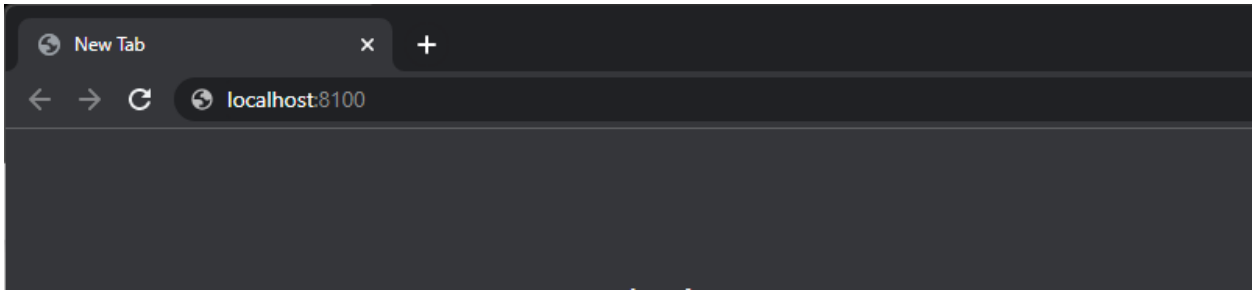


Enter the username as ‘hadoop’ to get a new terminal session. We’ve successfully tunnelled through the port for the airflow docker container.

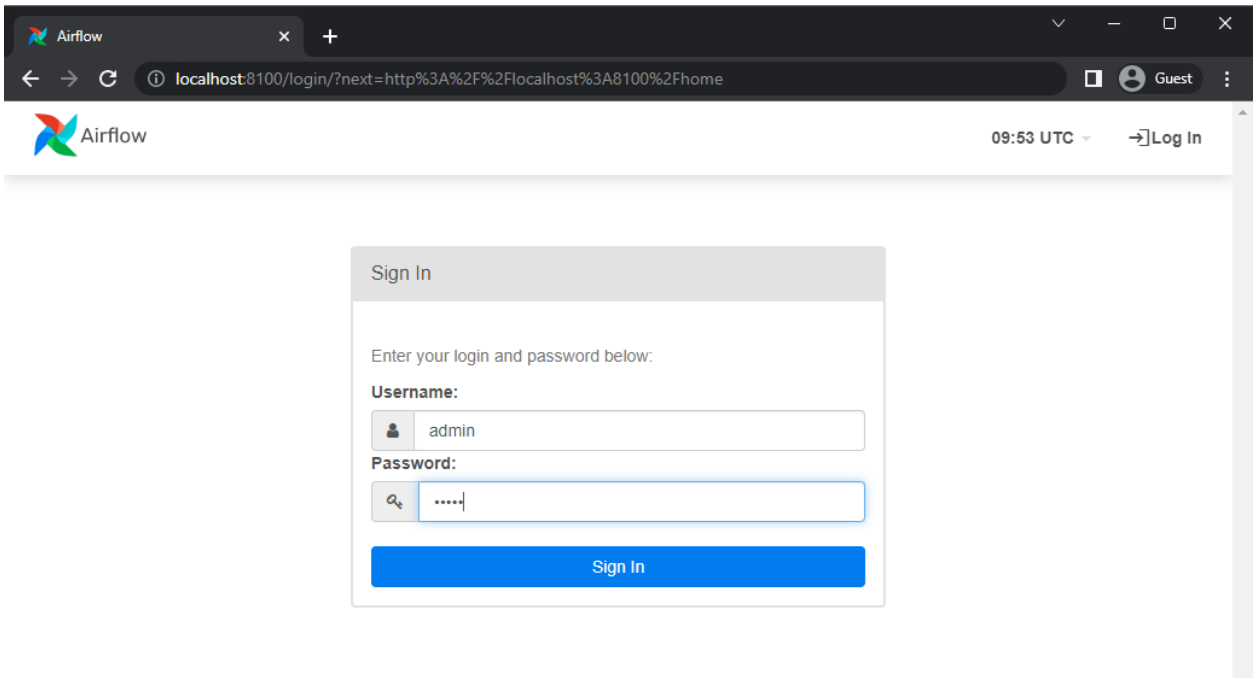
Accessing the Airflow UI Page

Open a new tab/window on a browser and navigate to the following URL:

localhost:8100



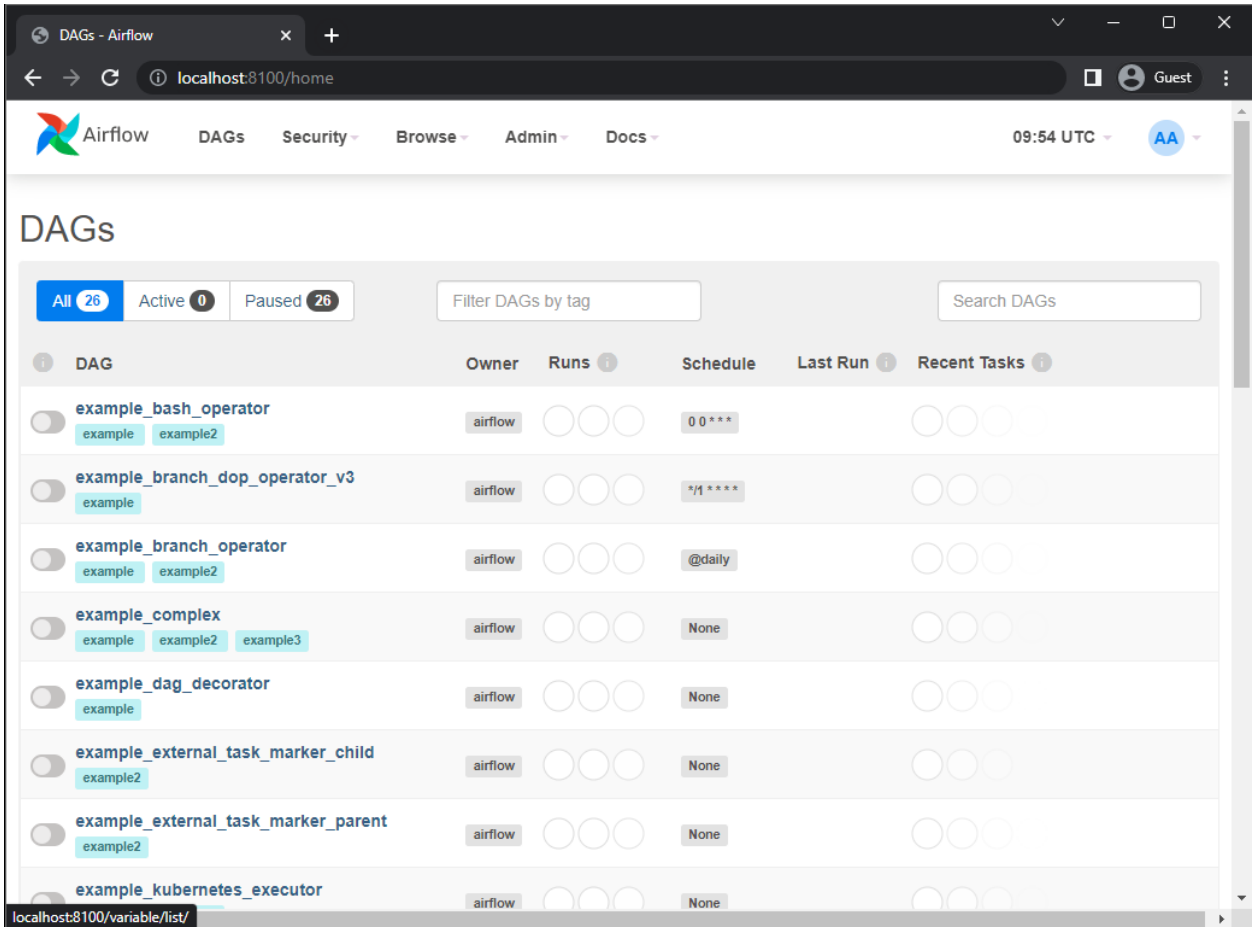
You'll now be redirected to the Airflow UI page as shown below (**NOTE**: If there's no login page visible, you'll be redirected directly to the Airflow UI page)



Enter the following credentials when prompted for the username and password and click on the **'Sign In'** button.

```
Username:airflow
Password: airflow
```

You can now see the Airflow UI Home page as shown in the image below



Useful Docker Commands

The following are some useful commands in Docker.

List docker images

The following command lists the active docker images installed on your system

```
docker images
```

```
[root@ip-172-31-71-166 airflow]# docker images
REPOSITORY              TAG         IMAGE ID       CREATED         SIZE
airflow-basic           latest      2b8a3a07d71d   3 hours ago     945MB
airflow1                latest      2b8a3a07d71d   3 hours ago     945MB
python                  3.8-slim    015fc0fe8ec7   2 weeks ago     124MB
emr/jupyter-notebook    6.4.0       ca763ad1e352   11 months ago   2.63GB
[root@ip-172-31-71-166 airflow]#
```

The following command lists the active docker containers running on your system

```
docker ps
```

```
[root@ip-172-31-71-166 airflow]# docker ps
CONTAINER ID   IMAGE                        COMMAND                  CREATED       STATUS
     PORTS                                                    NAMES
5f4864cd0622   emr/jupyter-notebook:6.4.0   "tini -g -- jupyterh…"   3 hours ago   Up 5 minut
es   8888/tcp, 0.0.0.0:9443->9443/tcp, :::9443->9443/tcp   jupyterhub
[root@ip-172-31-71-166 airflow]#
```

Running a container

First get the list of active docker containers using the command

```
docker ps
```

Then run the following command to run the container

```
docker exec -it <container_id>
```

The following command can be used for removing the docker images that have been created.

```
docker image rm <image-name>
```

```
[root@ip-172-31-71-166 airflow]# docker images
REPOSITORY              TAG         IMAGE ID       CREATED         SIZE
airflow1                latest      2b8a3a07d71d   3 hours ago     945MB
python                  3.8-slim    015fc0fe8ec7   2 weeks ago     124MB
emr/jupyter-notebook    6.4.0       ca763ad1e352   11 months ago   2.63GB
[root@ip-172-31-71-166 airflow]# docker image rm airflow1
```

Stopping a Docker container:

Don't forget to stop your Airflow docker container with the commands

```
docker ps
```

Then copy the container id of the docker container (here, it’s airflow_image) and then execute:

```
docker stop <container_id>
```