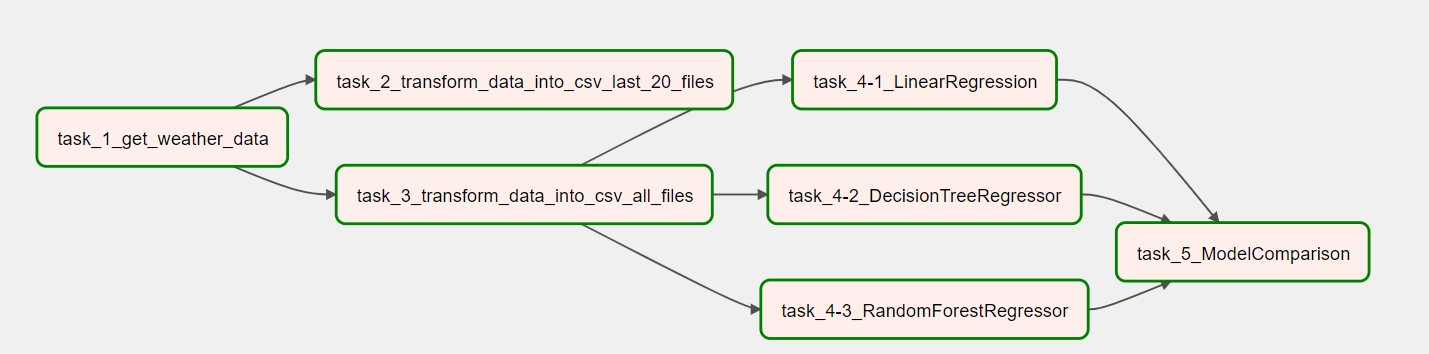
**Test du DAG**

Pour tester le DAG il faut faire les actions suivantes :

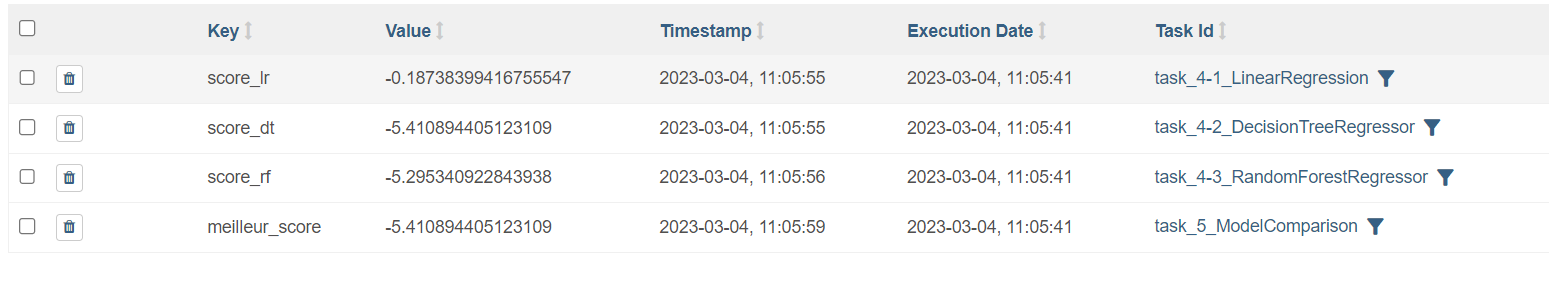
* **Optionnel** : Modifier les variables ci-dessous dans le fichier et le remplacer par les vôtres
  + Api\_url=" "
  + Api\_key=" "
  + cities=['paris', 'london', 'washington']
  + **NB : si le dag est chargée avec ces variables a vide, elles seront automatiquement créées dans airflow avec des valeurs par defaut; vous pouvez les modifier après dans airflow pour mettre vos propres valeurs et ou ajouter une ville ou des villes supplémentaires sans modifier le fichier DAG pour la récolte des données (explications de ce choix dans « choix et paramétrage »)**
* Créer un pool nommé « evaluation\_pool » avec un slot de 100 par exemple (il faut un minimum de 3 puisque les tâches 4’, 4’’ et 4’’’ s’exécutent en parallèle)
* Créer les deux connections suivantes de type File pour vérifier la présence des fichiers crées
  + raw\_files\_fs : pour surveiller « /app/raw\_files/ »
  + clean\_data\_fs : pour surveiller « /app/clean\_data/ »
* Pour initialiser le workflow il faut copier les deux fichiers data.csv et fulldata.csv dans le répertoire « clean\_data » de votre airflow.
* Copier enfin le fichier « openweathermap\_dag.py » dans le répertoire dags de votre installation airflow
* Les premières exécutions seront en erreurs pour les tâches 4’, 4’’ et 4’’’ et 5. Il faut attendre 15 minutes (15 récoltes de fichiers) pour avoir un minimum de données pour l’entrainement des modèles.

**Choix et paramétrage**

* Pour la définition du DAG nous avons utilisé un start\_date initialisé toujours à la date du jours ('start\_date': days\_ago(0, minute=1)) à partir de minuit et nous choisissons avec l’option catchup=False de ne pas exécuter les minutes perdus depuis 00h01 par rapport à l’heure d’exécution.
* Nous avons également défini les arguments communs aux tâches dans l’argument default\_args.
* Les noms des tâches (toutes les tâches parents doivent bien se terminer avant que les fils ne s’exécutent) ; cela permet également d’empêcher des exécutions concurrentes d’écraser les données (fichiers data.csv ou fulldata.csv) pour cela nous avons utilisé une variable commune a toutes les tâches ('trigger\_rule':'all\_success') :
  + (1) : **task\_1\_get\_weather\_data**
  + (2) : **task\_2\_transform\_data\_into\_csv\_last\_20\_files**
  + (3) : **task\_3\_transform\_data\_into\_csv\_all\_files**
  + (4) :
    - (4’) : **task\_4-1\_LinearRegression**
    - (4’’) : **task\_4-2\_DecisionTreeRegressor**
    - (4’’’) : **task\_4-3\_RandomForestRegressor**
  + (5) : **task\_5\_ModelComparison**

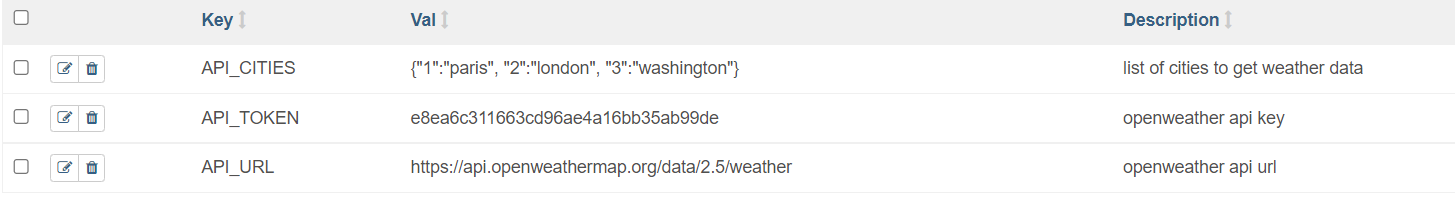


* Nous avons utilisé XComs pour transmettre les valeurs des différents modèles d’entrainements à la tâche de comparaison. Une fois la comparaison faite nous enregistrons également la valeur du meilleur score dans un XComs et l’affichons également dans les logs.



* Pour pouvoir changer facilement l’URL de l’api, la clé de l’api ainsi que la liste des villes nous avons choisi de créer des variables dans airflow. Dans le DAG nous avons choisi de les créer automatiquement lorsqu’elles n’existent pas (au lancement si les variables n’existent pas elles seront automatiquement créées).

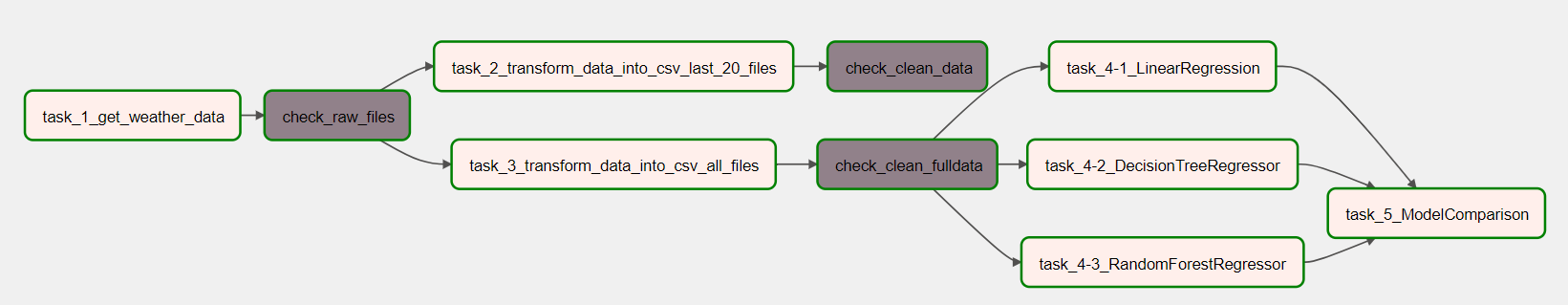




* Nous avons choisi de crée un pool de 20 slots et nous l'avons défini dans le DAG ; dans default\_args

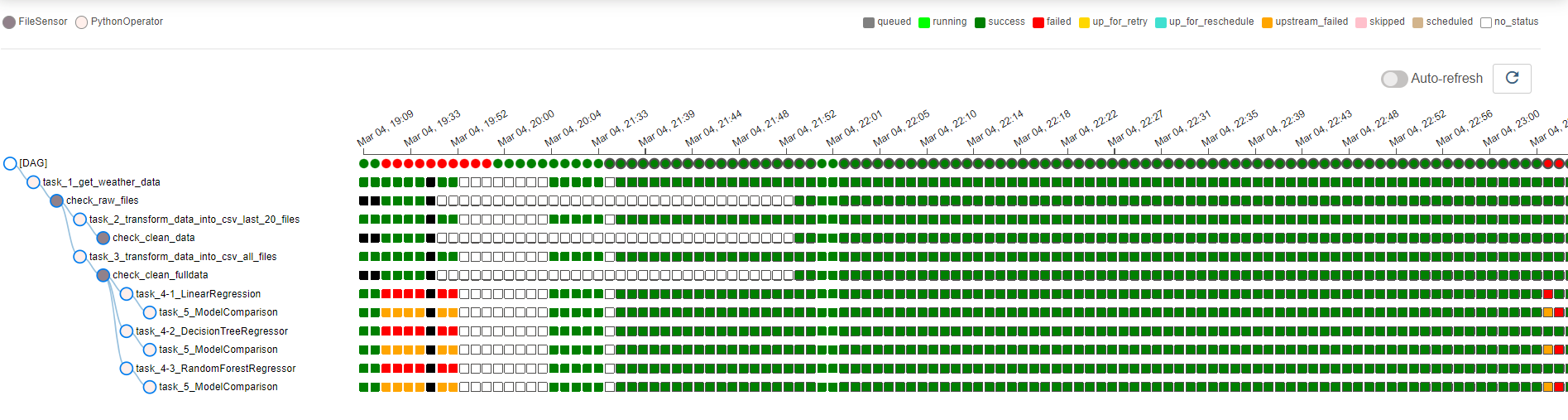


* + Comme toutes les tâches dépendent de la première tâche (1) : **task\_1\_get\_weather\_data.** Nous avons choisi de le relancer 5 fois avec un intervalle de 30 secondes lorsqu’elle echoue.
* Nous avons utilisé un FileSensor pour vérifier que les fichiers existent bien dans les différents répertoires ; Si les fichiers ne sont pas trouvés, la tâche est relancée 5 fois avant de sortir en erreur. Pour le tester vous pouvez essayer de supprimer le fichier data.csv ou fulldata.csv et de renommer le nouveau fichier qui sera créer par exemple en data2.csv ou datafull2.csv ou de faire en sorte que le répertoire raw\_files soit vide

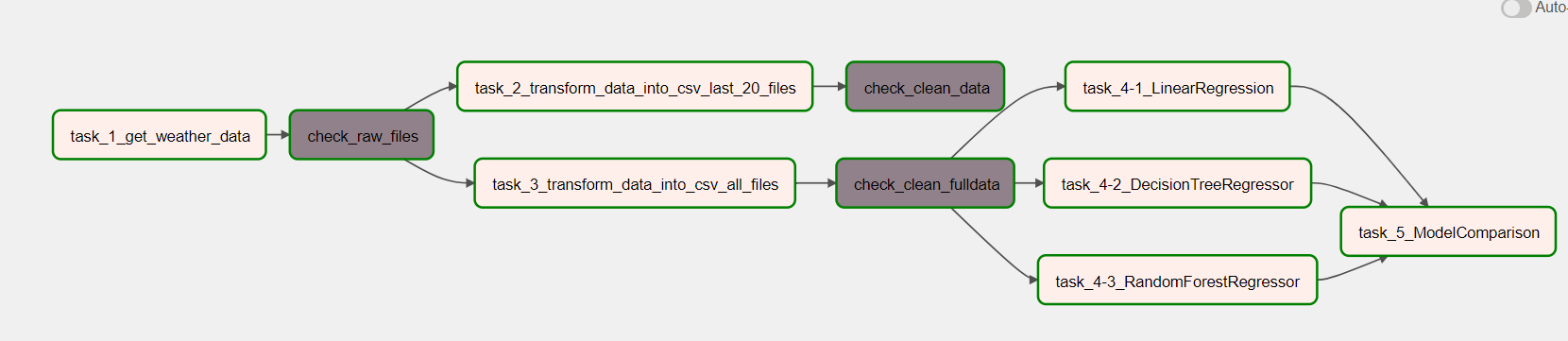


**Quelques captures d’écrans**

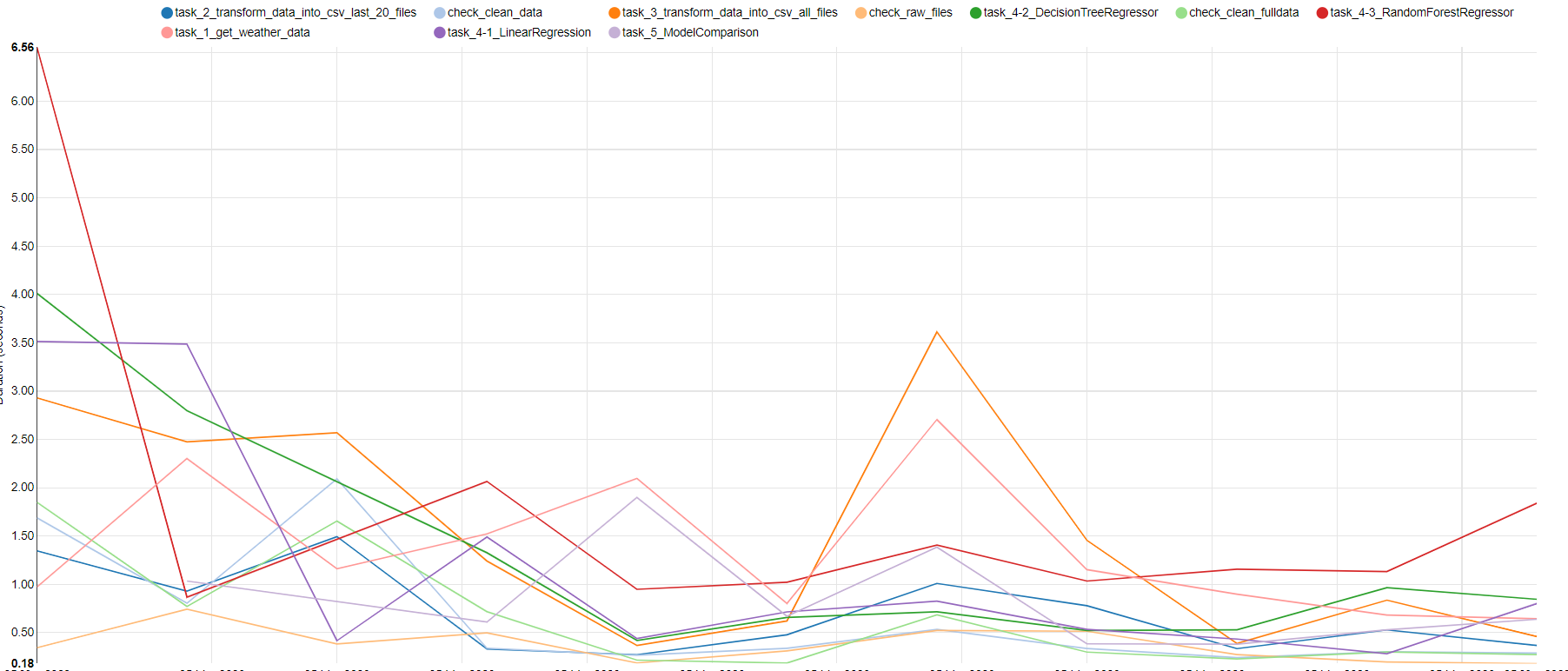
* Tree view



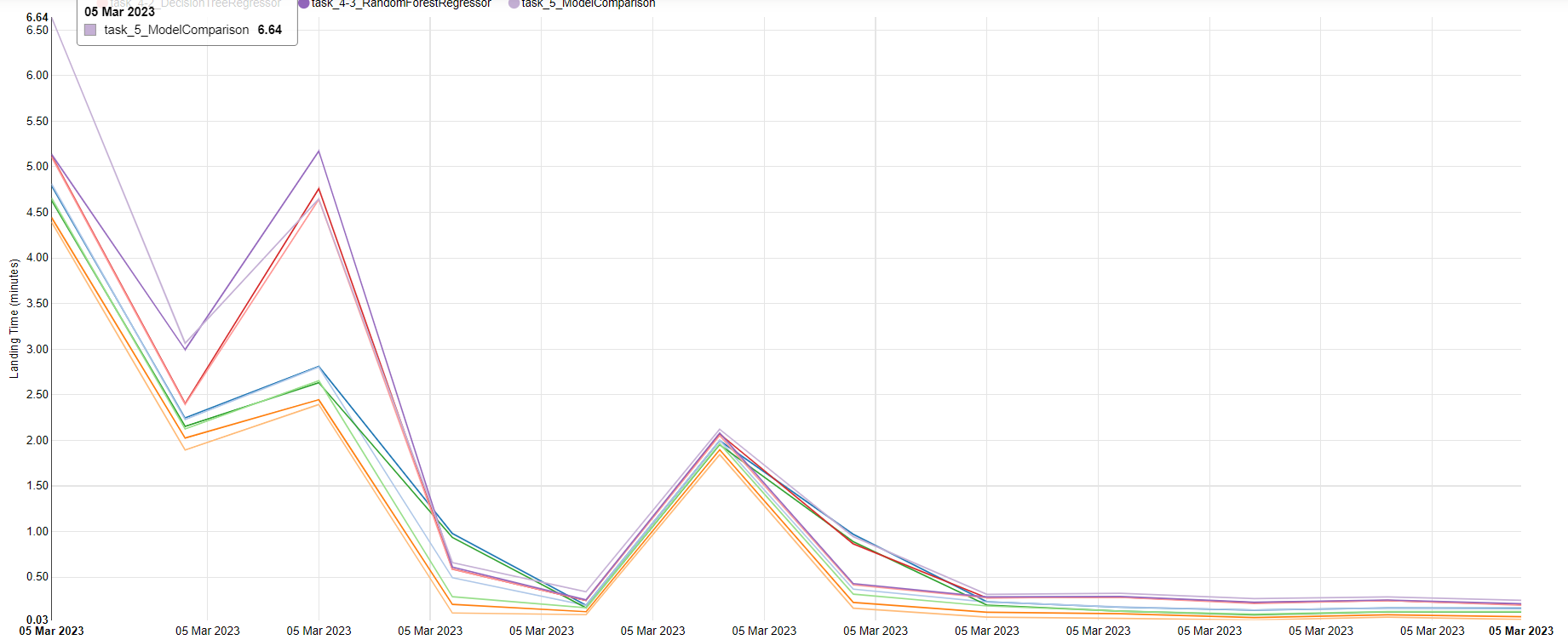
* Graph view



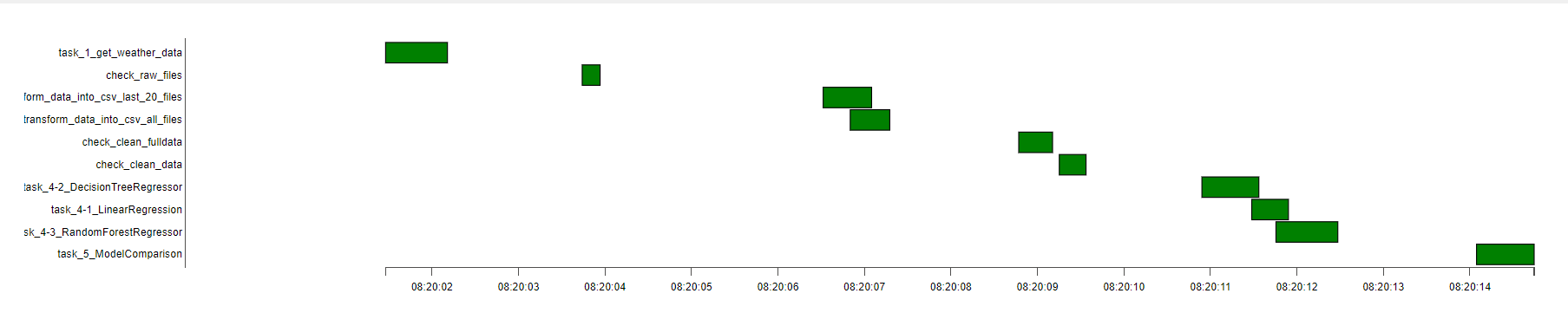
* Task duration



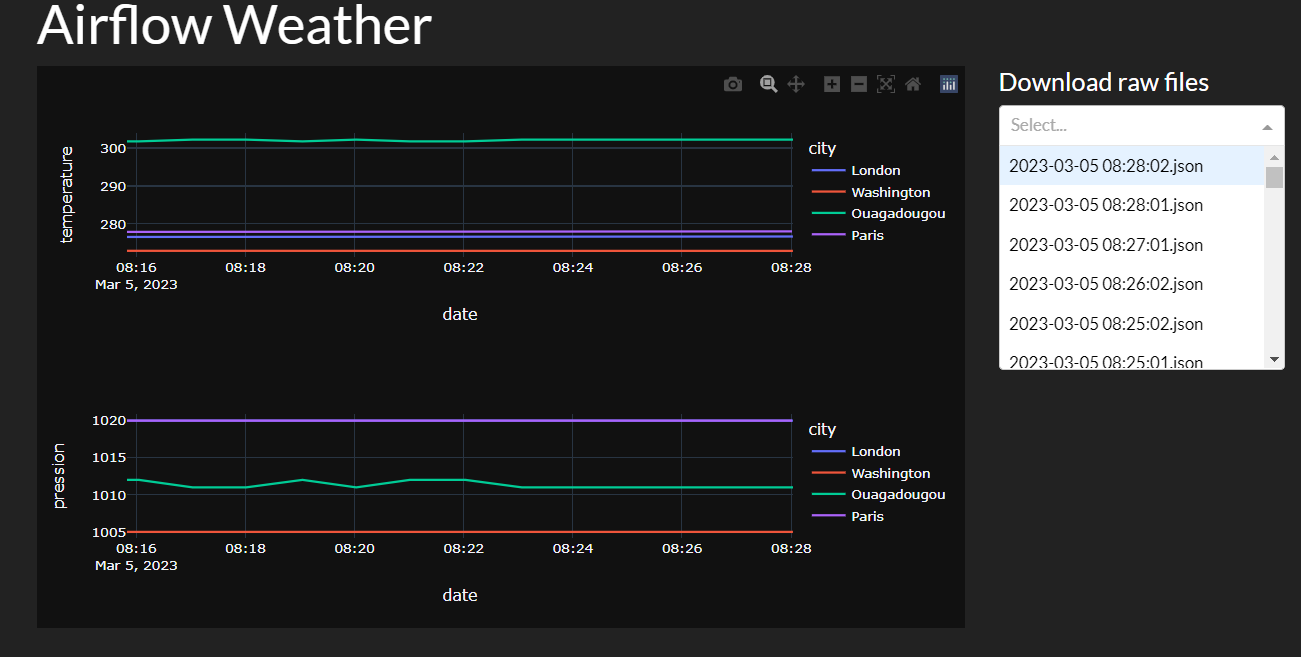
* Landing times



* Grant



* Airflow weather



* Résultat training (XComs)

