

Université Paris Cité

Master 1 Machine Learning pour la Science des Données

Projet Cloud AWS : Titanic Survivors

Rapport de projet

Année Universitaire: 2024/2025

Ouarda BOUMANSOUR

Hamady GACKOU

Omar NAMOUS

Module : Introduction au cloud AWS

Professeur responsable : Meriem OUARET

Table des matières

Table des figures	ii
1 Introduction	iv
Chapitre 1 Configuration initiale	v
1.1 Création de l'utilisateur IAM	v
1.2 Configuration des permissions	vi
Chapitre 2 Stockage and Traitement des données	x
2.1 Téléchargement des données Titanic ,Création du bucket S3 et organisation des dossiers	x
Chapitre 3 Traitement des données	xiv
3.1 Créez un rôle IAM pour la fonction Lambda	xiv
3.2 Création de la fonction Lambda	xvii
3.3 Traitement des données : valeurs manquantes, encodage, statistiques descriptives	xxi
3.4 Test de la fonction lambda et vérification des logs	xxii
3.5 Résultats du traitement dans S3	xxvi
Chapitre 4 Entraînement et Prédition du Modèle	xxvii
4.1 Création d'un role IAM pour le notebook	xxvii
4.2 Séparation en ensembles d'entraînement et de test	xxx
4.3 Entraînement du modèle XGBoost	xxxiii
4.4 Prédictions sur l'ensemble de test	xxxvi
4.5 Évaluation des performances avec l'accuracy	xxxix
Conclusion	xli

Table des figures

1.1	Creation de l'utilisateur AWS pour le projet	v
1.2	L'ajout de permissions gérées par AWS	vi
1.3	AllowRoleCreation : Politique IAM personnalisée pour notre utilisateur	vii
1.4	AllowRoleCreation : Politique IAM personnalisée pour notre utilisateur - suite	vii
1.5	Récapitulatif des autorisations	viii
1.6	Ajout d'un mot de passe au compte utilisateur crée	ix
2.1	Creation de la bucket S3	x
2.2	Creation de la bucket S3 réussi	xi
2.3	Creation du dossier raw	xii
2.4	Ajout du fichier titanic.csv au dossier raw/	xii
3.1	Création de la politique personnalisée	xiv
3.2	Création de la politique personnalisée - suite	xv
3.3	Création de la politique personnalisée - suite	xv
3.4	Création d'un rôle associé à une fonction lambda	xvi
3.5	Création d'un role associé à une fonction lambda réussite	xvi
3.6	Création de la fonction lambda	xvii
3.7	Ajout du rôle à la fonction lambda	xviii
3.8	Ajout du déclencheur à la fonction lambda	xviii
3.9	Ajout d'une couche pour executer lambda avec pandas	xix
3.10	Ajout du code et le déploiement de la fonction lambda	xx
3.11	Ajout d'un événement de test	xxii
3.12	Test réussi pour la fonction lambda	xxiii
3.13	Journal apparu sur cloudWatch après l'exécution du test	xxiii

3.14 Ajout manuel d'un fichier pour surveiller le déclanchement de la fonction lambda	xxiv
3.15 Enregistrement des nouveaux logs dans les journaux après l'ajout d'un fichier dans le bucket	xxv
3.16 Visualisation sur CloudWatch avec graphes	xxv
3.17 Ajout automatiques des résultats dans le dossier processed du bucket	xxvi
4.1 Ajout d'une politique à insérer au role IAM du notebook sagemaker	xxvii
4.2 Ajout d'une politique à insérer au role IAM du notebook sagemaker - suite	xxviii
4.3 Création du role IAM pour le notebook	xxix
4.4 Création du role IAM pour le notebook - suite	xxix
4.5 Création du notebook sagemaker	xxx
4.6 Ajout du rôle créé au notebook	xxxi
4.7 Création d'un notebook jupyter et ajout des librairies nécessaires	xxxi
4.8 Division des données en train/test	xxxii
4.9 Entrainement d'un modèle XGBoost	xxxiii
4.10 Entrainement d'un modèle XGBoost réussi	xxxiv
4.11 Les journaux correspondants à l'entraînement du modèle sur cloudwatch .	xxxiv
4.12 Déploiement du modèle	xxxv
4.13 Chargement du model dans un deuxième notebook pour faire des prédictions	xxxvi
4.14 Récupération des données de test	xxxvi
4.15 Creation du modèle à partir de notre artifact d'entraînement	xxxvii
4.16 Initiation d'un batch transform job pour les prédictions en batch	xxxvii
4.17 Création de transform job réussite	xxxviii
4.18 Résultats d'inférence enregistrés	xxxviii
4.19 Journaux pour l'évenement de batch transform ajoutés à cloudWatch	xxxix
4.20 Évaluation des performances avec l'accuracy	xxxix

1 Introduction

Ce projet a pour objectif de prédire les survivants du Titanic à l'aide des services cloud AWS. En combinant des outils tels que S3 pour le stockage, Lambda pour le traitement des données, et SageMaker pour la construction et l'entraînement du modèle de machine learning, nous avons pu explorer un workflow complet de machine learning sur le cloud.

L'approche suivie inclut plusieurs étapes clés, telles que la configuration d'un environnement sécurisé avec IAM, le traitement des données en utilisant des fonctions Lambda déclenchées par des événements S3, et l'utilisation de SageMaker pour entraîner un modèle XGBoost. Les performances du modèle ont été évaluées à l'aide de prédictions par lot et d'une analyse de l'accuracy. Ce projet met en lumière la puissance et la flexibilité des services AWS pour construire des solutions de machine learning entièrement hébergées.

1

Configuration initiale

1.1 Création de l'utilisateur IAM

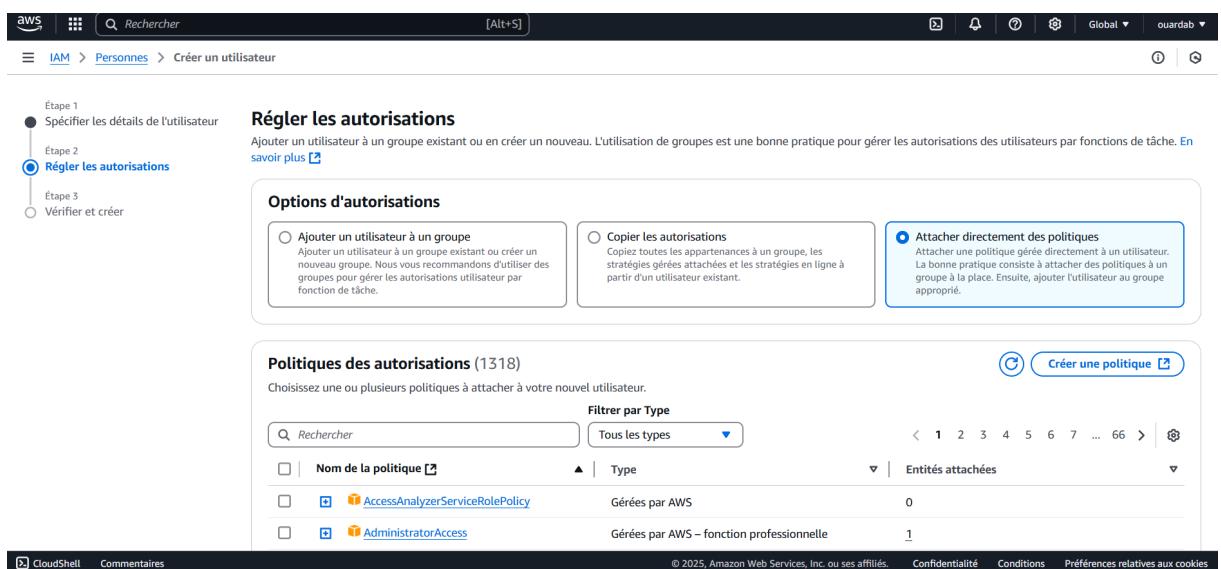


FIGURE 1.1 – Creation de l'utilisateur AWS pour le projet

1.2 Configuration des permissions

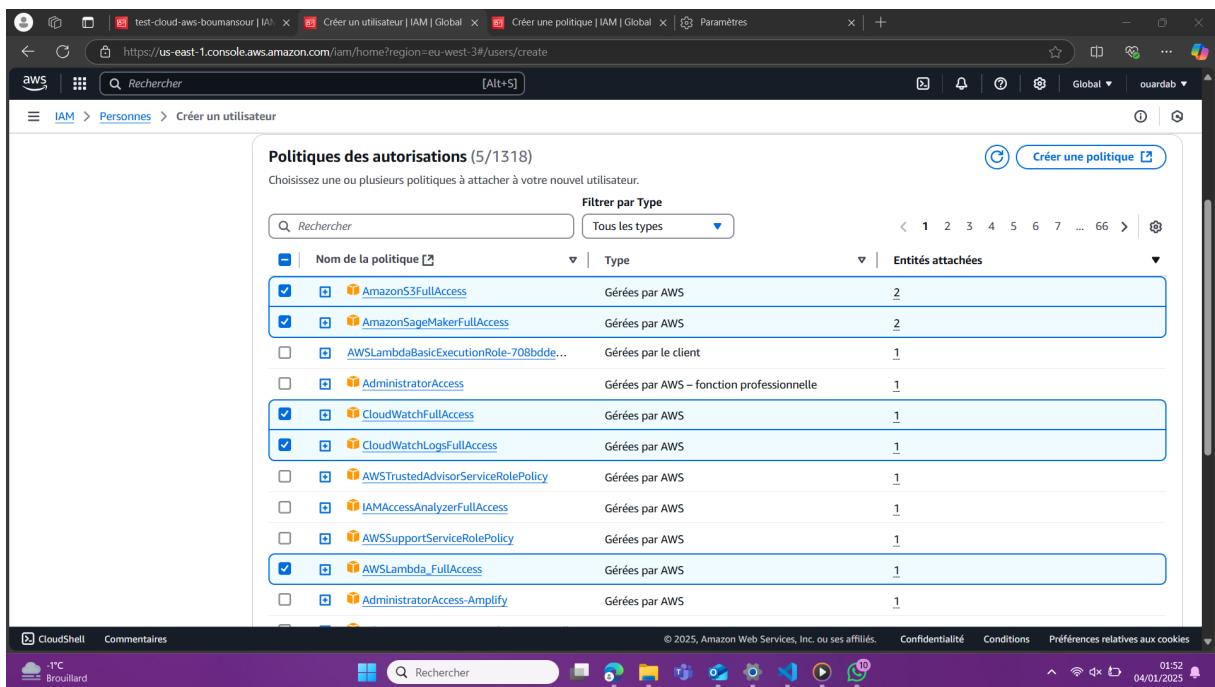


FIGURE 1.2 – L'ajout de permissions gérées par AWS

Dans cette étape, des permissions gérées par AWS ont été attribuées à l'utilisateur IAM pour lui permettre d'interagir avec les services nécessaires au projet. Les politiques suivantes ont été associées à l'utilisateur :

- **AmazonS3FullAccess** : Accès complet à Amazon S3 pour gérer les buckets et objets nécessaires à l'importation des données.
- **AmazonSageMakerFullAccess** : Permet d'utiliser Amazon SageMaker pour créer et entraîner des modèles de machine learning.
- **AWSLambdaFullAccess** : Accès complet aux fonctions Lambda pour automatiser les traitements de données.
- **CloudWatchFullAccess** : Permet de surveiller les performances des services AWS et d'analyser les logs généraux.
- **CloudWatchLogsFullAccess** : Permet un accès complet aux logs générés par les fonctions Lambda et autres services, essentiel pour surveiller et déboguer les traitements en temps réel.

1.2. Configuration des permissions

vii

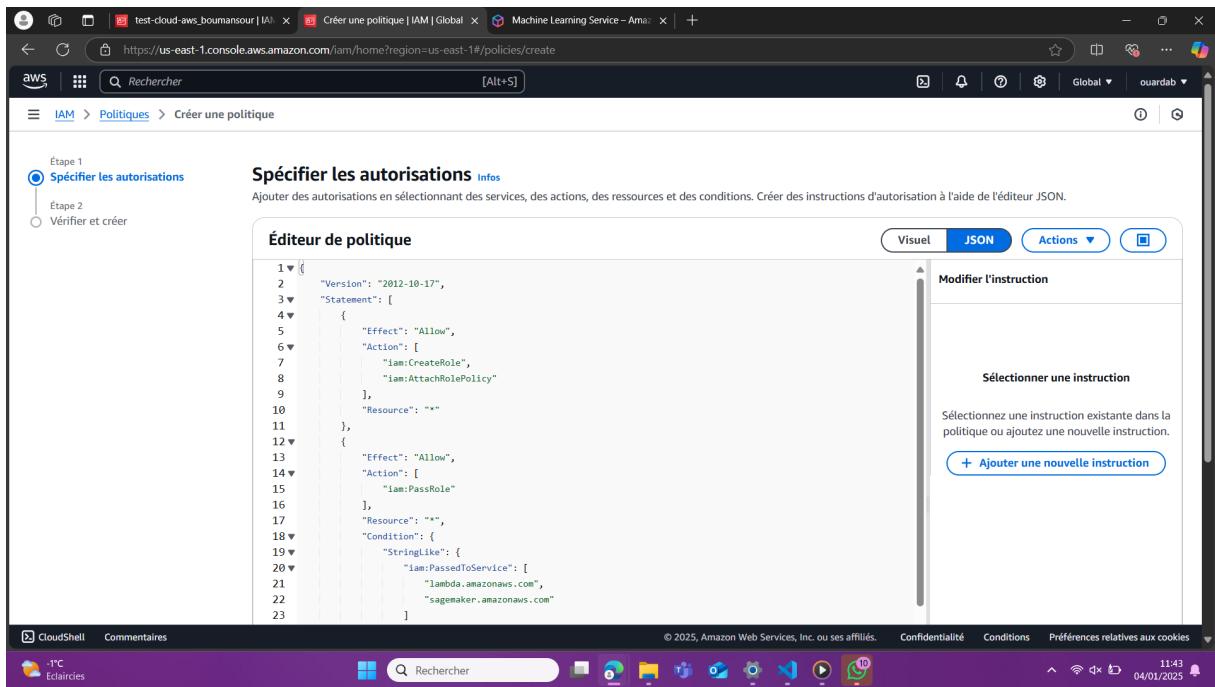


FIGURE 1.3 – AllowRoleCreation : Politique IAM personnalisée pour notre utilisateur

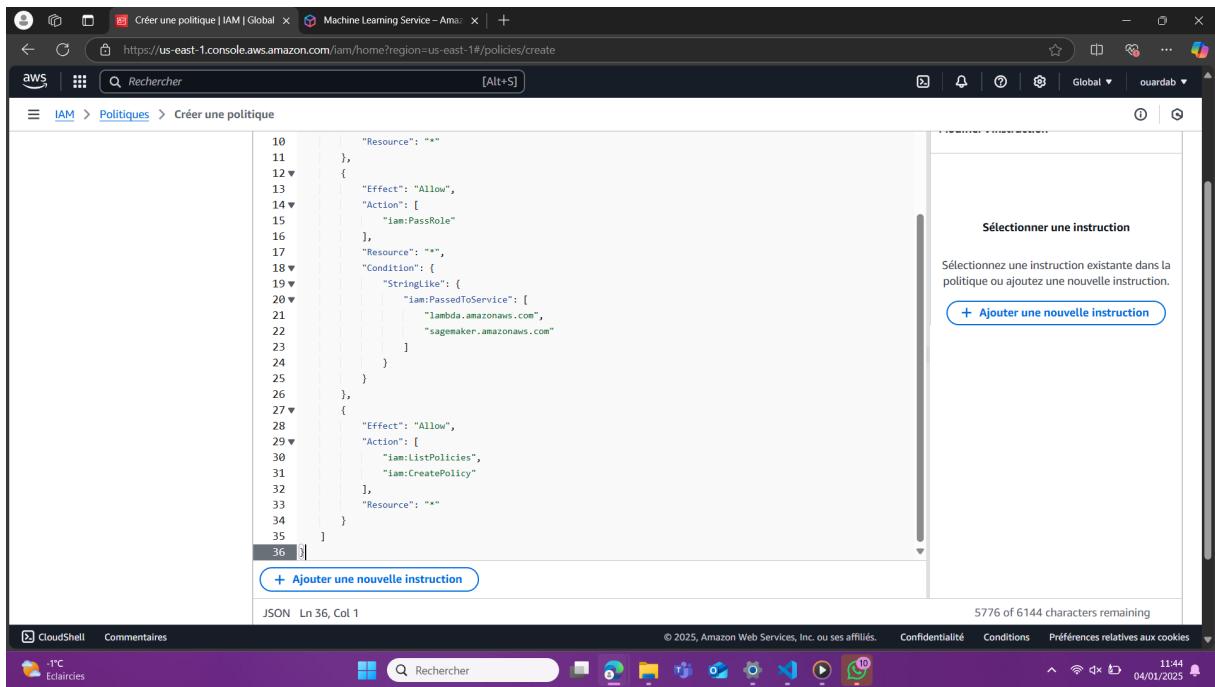


FIGURE 1.4 – AllowRoleCreation : Politique IAM personnalisée pour notre utilisateur - suite

Cette politique IAM a été conçue pour permettre à l'utilisateur de gérer les rôles

nécessaires à l'exécution des services Lambda et SageMaker, comme spécifié dans les étapes du projet. Elle accorde les permissions nécessaires pour créer et attacher des rôles IAM, ainsi que pour transmettre des rôles aux services Lambda et SageMaker tout en respectant une condition spécifique. Cette approche vise à garantir une gestion sécurisée des autorisations tout en respectant le principe du moindre privilège, en limitant les actions aux besoins spécifiques du projet.

The screenshot shows the AWS IAM 'Create User' interface. The top navigation bar includes the AWS logo, a search bar, and various global settings. The main content area is titled 'Résumé des autorisations' (Authorization Summary). It displays a table with six rows of permissions:

Nom	Type	Utilisé comme
AllowRoleCreation	Gérées par le client	Stratégie des autorisations
AmazonS3FullAccess	Gérées par AWS	Stratégie des autorisations
AmazonSageMakerFullAccess	Gérées par AWS	Stratégie des autorisations
AWSLambda_FullAccess	Gérées par AWS	Stratégie des autorisations
CloudWatchFullAccess	Gérées par AWS	Stratégie des autorisations
CloudWatchLogsFullAccess	Gérées par AWS	Stratégie des autorisations

Below the table, there is a section titled 'Balises - facultatif' (Tags - optional) with instructions about adding resource tags. A button labeled 'Ajouter une nouvelle identification' (Add new tag) is present, along with a note that up to 50 tags can be added. At the bottom right of the page are buttons for 'Annuler' (Cancel), 'Précédent' (Previous), and 'Créer un utilisateur' (Create user).

FIGURE 1.5 – Récapitulatif des autorisations

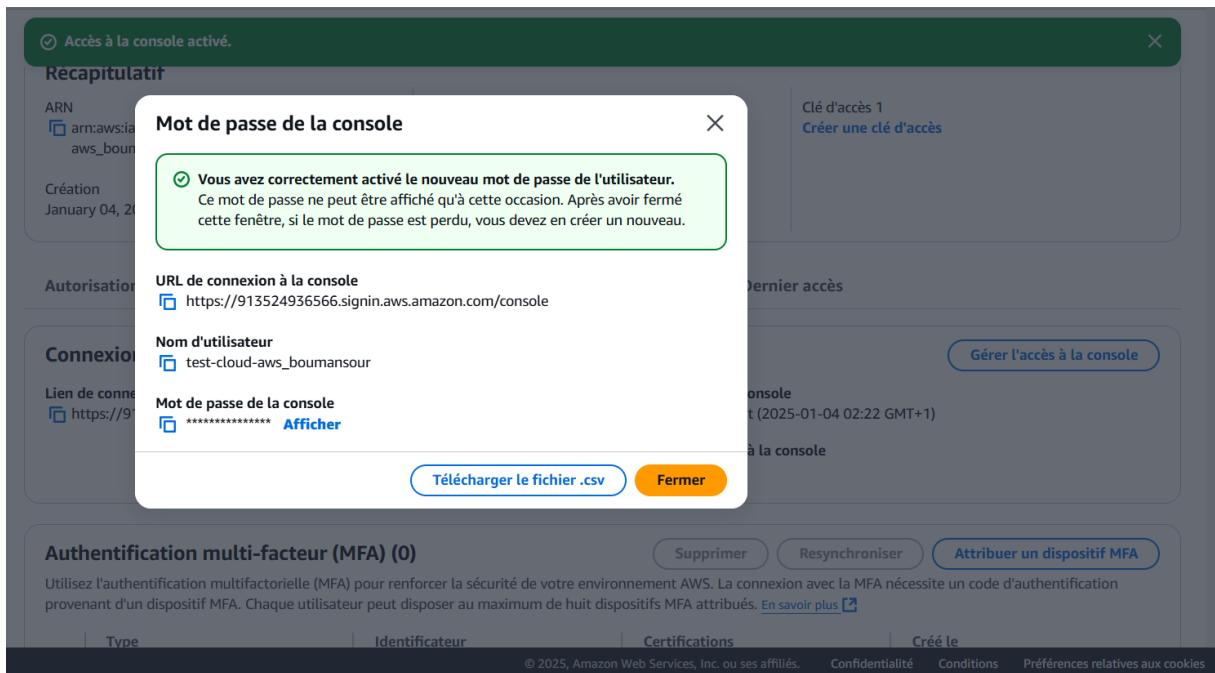


FIGURE 1.6 – Ajout d'un mot de passe au compte utilisateur créé

Dans cette partie on ajoute un mot de passe d'accès à notre utilisateur et voici ses informations :

- **Nom d'utilisateur** : test-cloud-aws_boumansour.
- **Mot de passe** : projetcloud_2025AMSDAWS.
- **URL de connexion** : <https://913524936566.signin.aws.amazon.com/console>

2

Stockage and Traitement des données

2.1 Téléchargement des données Titanic ,Création du bucket S3 et organisation des dossiers

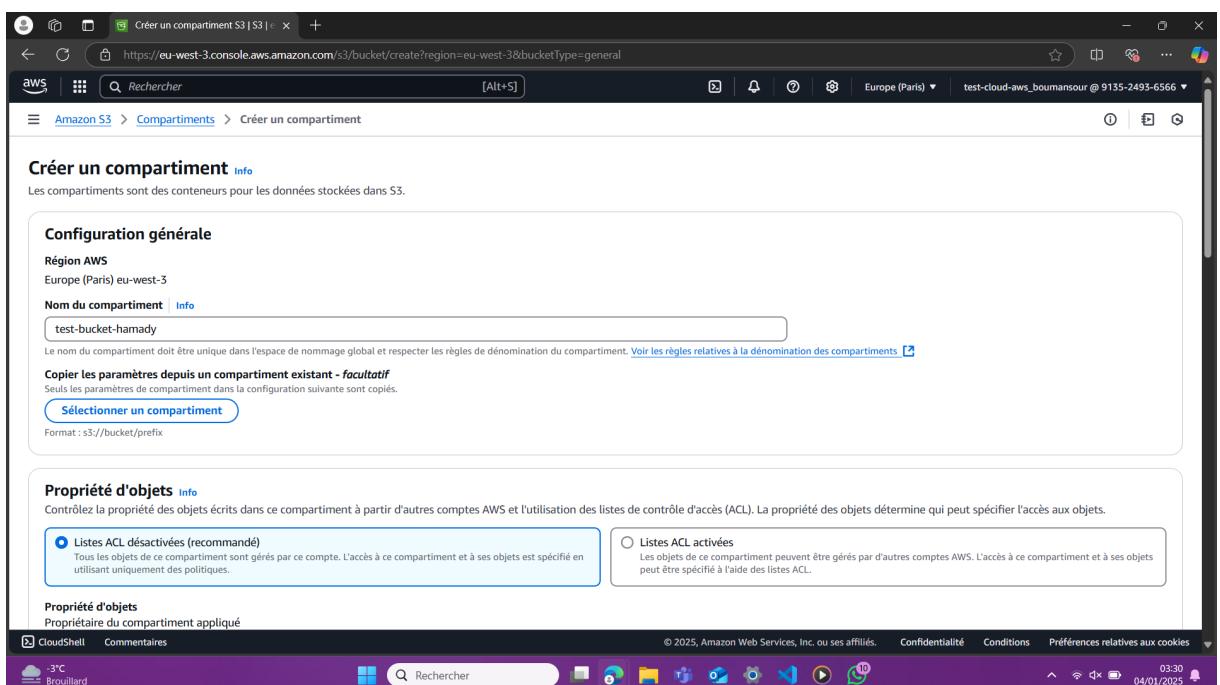


FIGURE 2.1 – Creation de la bucket S3

2.1. Téléchargement des données Titanic ,Création du bucket S3 et organisation des dossiersx1

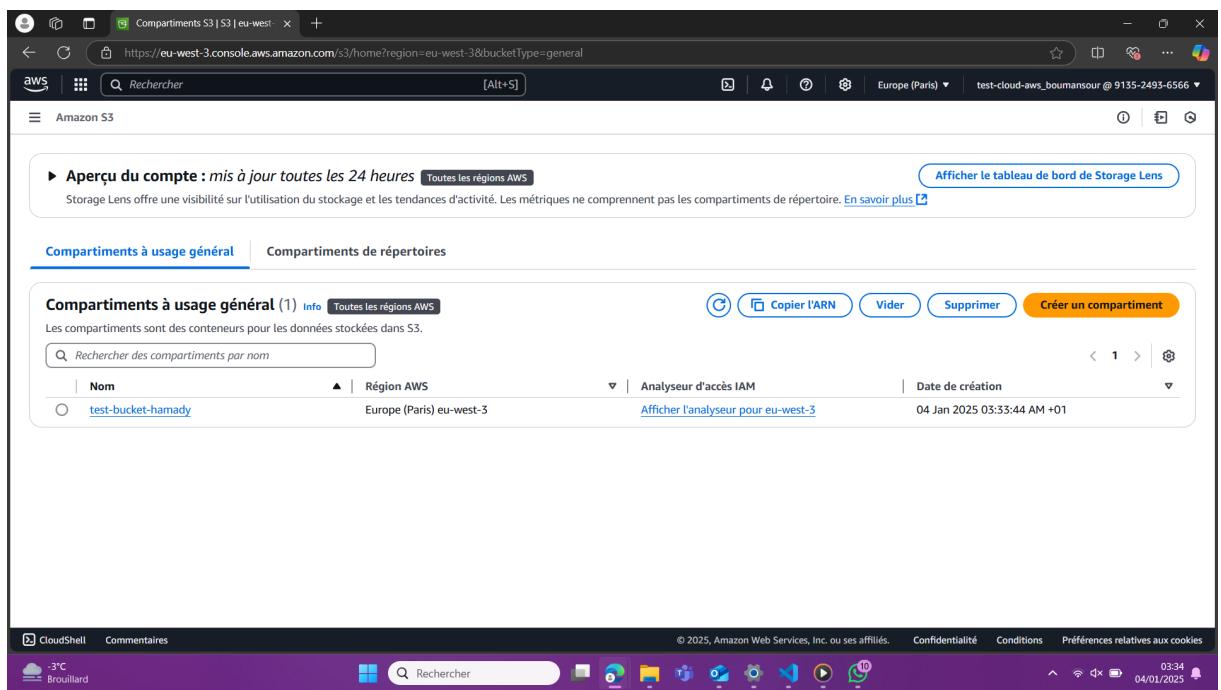


FIGURE 2.2 – Creation de la bucket S3 réussi

Dans cette étape, un bucket S3 nommé test-bucket-hamady a été configuré dans la région AWS Europe (Paris). Le nom du bucket respecte les règles de dénomination globale, et l'option recommandée "Listes ACL désactivées" a été sélectionnée pour gérer les droits d'accès via les politiques IAM. Ce bucket sera utilisé pour stocker les données brutes (raw) du projet Titanic au format CSV.

2.1. Téléchargement des données Titanic ,Création du bucket S3 et organisation des dossierxxii

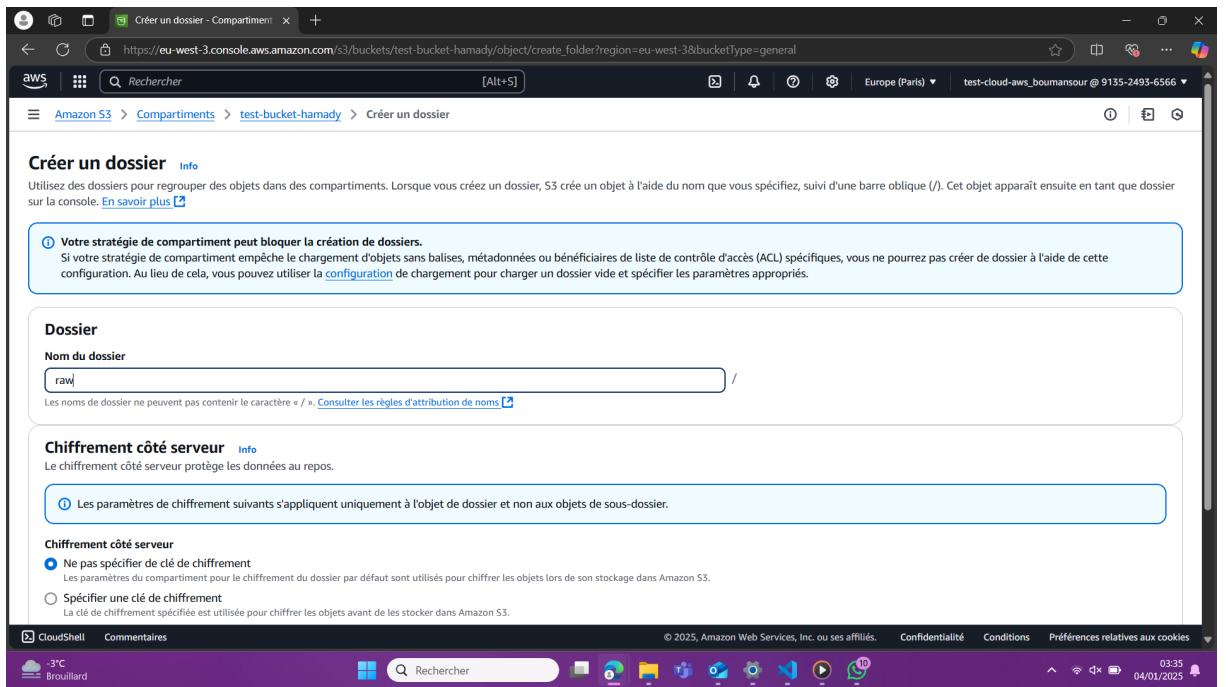


FIGURE 2.3 – Creation du dossier raw

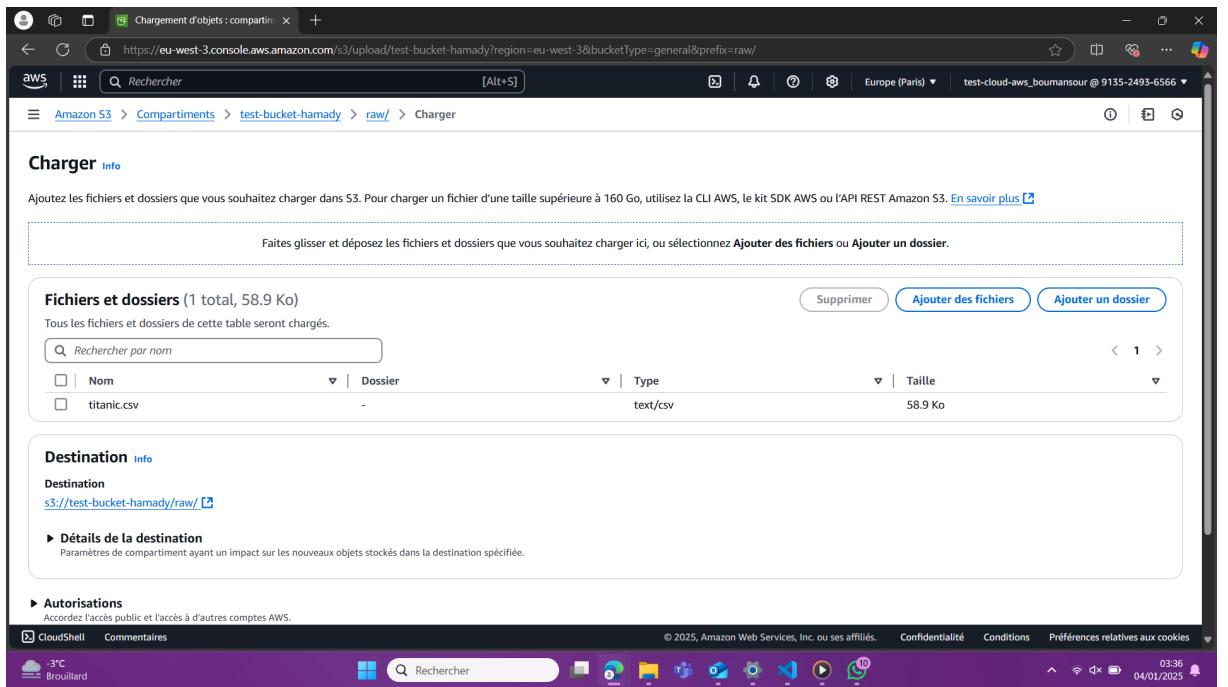


FIGURE 2.4 – Ajout du fichier titanic.csv au dossier raw/

Dans cette étape, les données du Titanic ont été téléchargées au format CSV à partir du lien fourni dans les instructions du projet. Un dossier nommé raw a été créé pour organiser

2.1. Téléchargement des données Titanic ,Création du bucket S3 et organisation des dossiersxiii

les fichiers bruts à l'intérieur du bucket crée. Le fichier CSV contenant les données Titanic a ensuite été téléchargé dans ce dossier. Cette organisation permet une gestion structurée des données dans S3, facilitant ainsi leur traitement et leur manipulation dans les étapes ultérieures du projet.

3

Traitement des données

3.1 Créez un rôle IAM pour la fonction Lambda

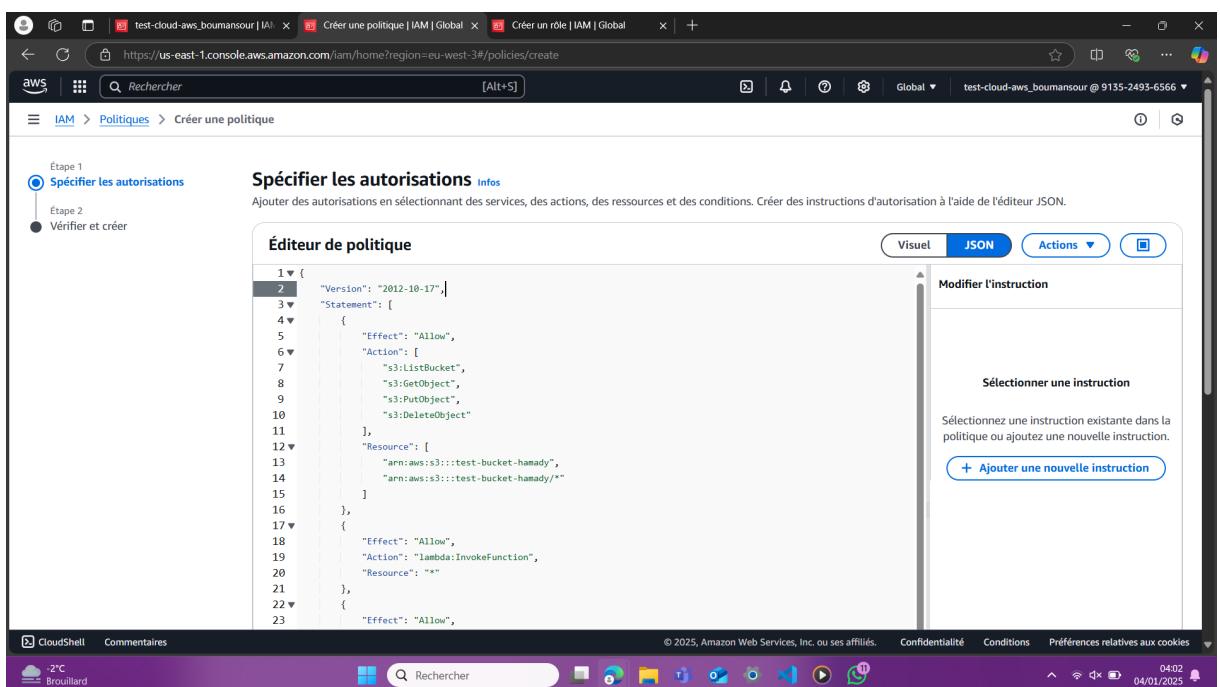


FIGURE 3.1 – Création de la politique personnalisée

On crée la politique AllowLambdaS3 pour la fonction lambda . Cette politique IAM accorde à la fonction Lambda les autorisations nécessaires pour :

S3 : Lister, lire, écrire et supprimer des objets dans le bucket test-bucket-hamady.
Lambda : Appeler d'autres fonctions Lambda (lambda :InvokeFunction). CloudWatch et Logs : Créer et gérer les journaux et métriques pour le monitoring.

3.1. Créez un rôle IAM pour la fonction Lambda

xv

The screenshot shows the AWS IAM 'Create a policy' interface. On the left, there is a large JSON code editor containing a complex policy document. On the right, a sidebar titled 'Sélectionner une instruction' (Select an instruction) displays the message 'Sélectionnez une instruction existante dans la politique ou ajoutez une nouvelle instruction.' (Select an existing instruction in the policy or add a new one). Below this is a blue button labeled '+ Ajouter une nouvelle instruction' (Add new instruction). At the bottom of the JSON editor, there is a note stating '5793 of 6144 characters remaining'. The bottom navigation bar includes links for CloudShell, Commentaires, and other AWS services.

```

    {
        "Version": "2012-10-17",
        "Statement": [
            {
                "Action": [
                    "s3:listBucket",
                    "s3:GetObject",
                    "s3:PutObject",
                    "s3:DeleteObject"
                ],
                "Resource": [
                    "arn:aws:s3:::test-bucket-hamady",
                    "arn:aws:s3:::test-bucket-hamady/*"
                ]
            },
            {
                "Effect": "Allow",
                "Action": "lambda:InvokeFunction",
                "Resource": "*"
            },
            {
                "Effect": "Allow",
                "Action": [
                    "logs:CreateLogGroup",
                    "logs:CreateLogStream",
                    "logs:PutLogEvents"
                ],
                "Resource": "*"
            }
        ]
    }
  
```

FIGURE 3.2 – Création de la politique personnalisée - suite

The screenshot shows the 'Verify and create' step of the AWS IAM policy creation wizard. It includes fields for the policy name ('AllowLambdaS3'), a description ('Permettre à lambda un accès en lecture écriture à s3 et aux logs de cloudwatch aussi'), and a section for defined permissions ('Autorisations définies dans cette politique'). The bottom navigation bar includes links for CloudShell, Commentaires, and other AWS services.

FIGURE 3.3 – Création de la politique personnalisée - suite

3.1. Créez un rôle IAM pour la fonction Lambda

xvi

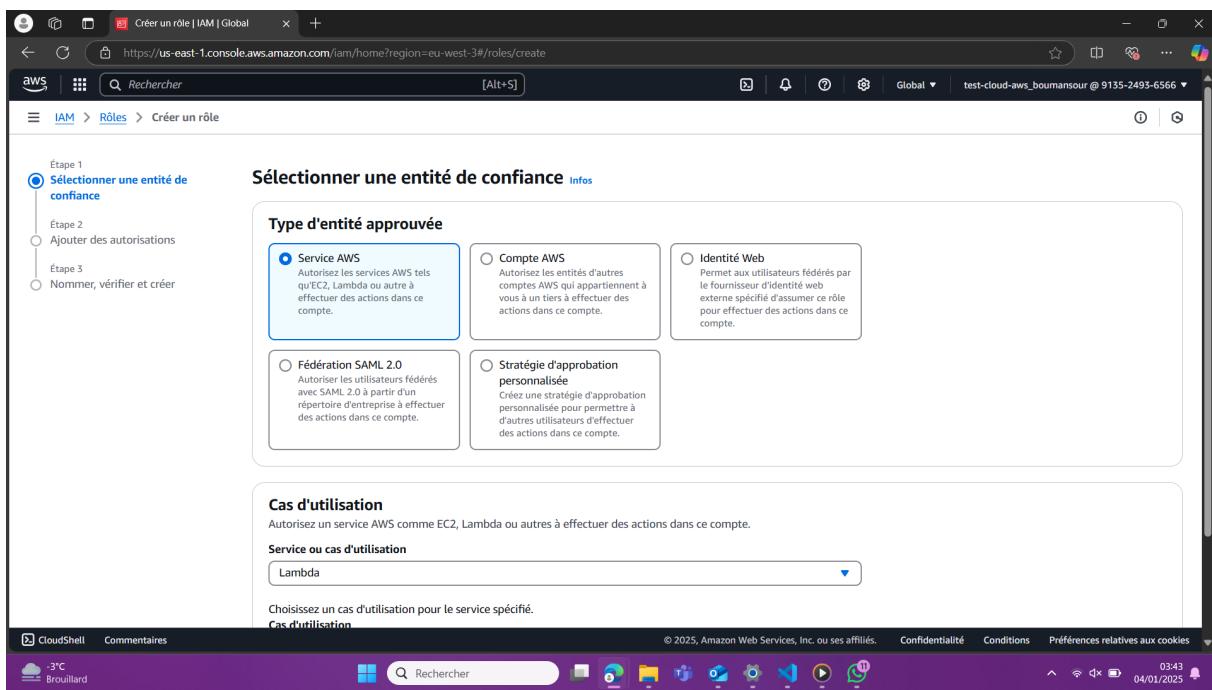


FIGURE 3.4 – Création d'un rôle associé à une fonction lambda

On crée par la suite un Rôle pour la fonction lambda auquel on associe la politique précédemment créée .

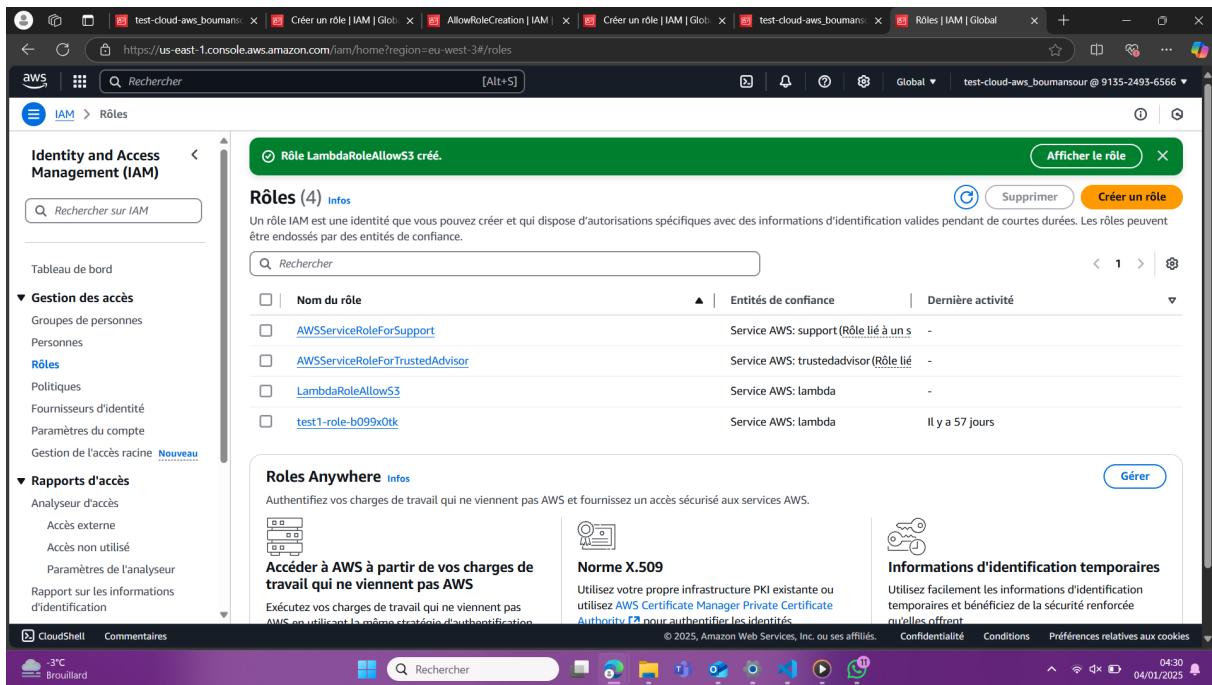


FIGURE 3.5 – Création d'un role associé à une fonction lambda réussite

Ce rôle garantit une intégration sécurisée et efficace entre la fonction Lambda et les services AWS utilisés dans le projet.

3.2 Crédit de la fonction Lambda

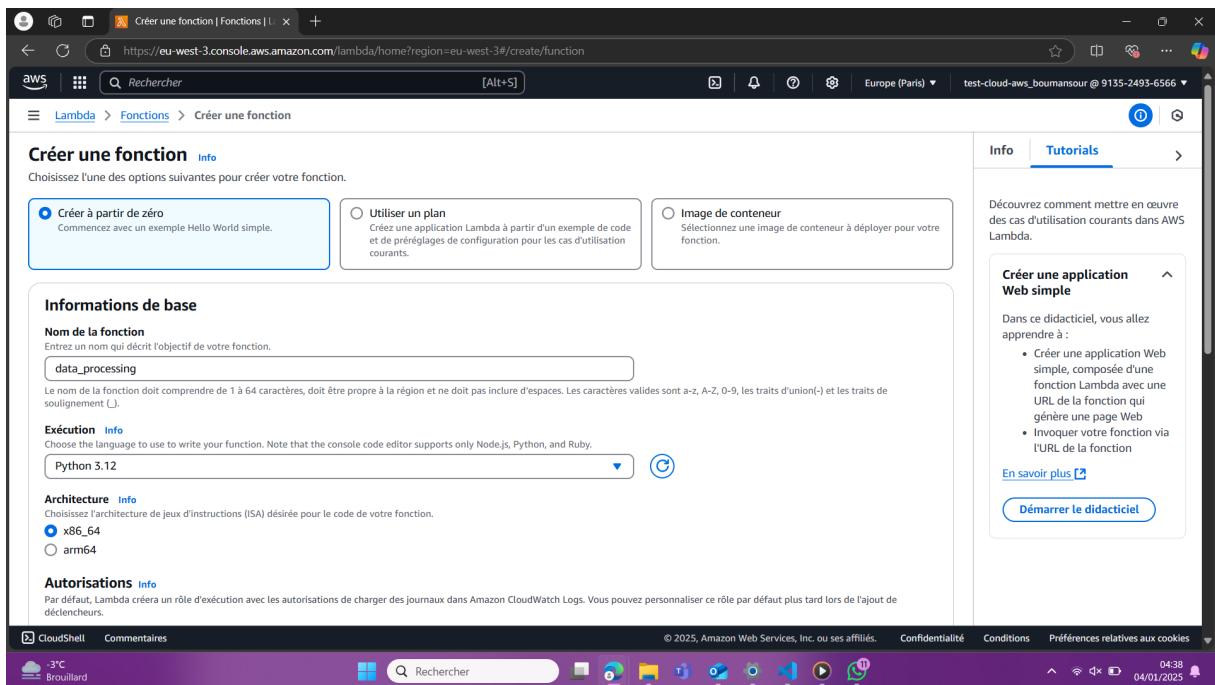


FIGURE 3.6 – Crédit de la fonction lambda

On crée la fonction lambda dans laquelle on fait le traitement de nos données Titanic brutes enregistrées dans notre bucket S3 .

3.2. Création de la fonction Lambda

xviii

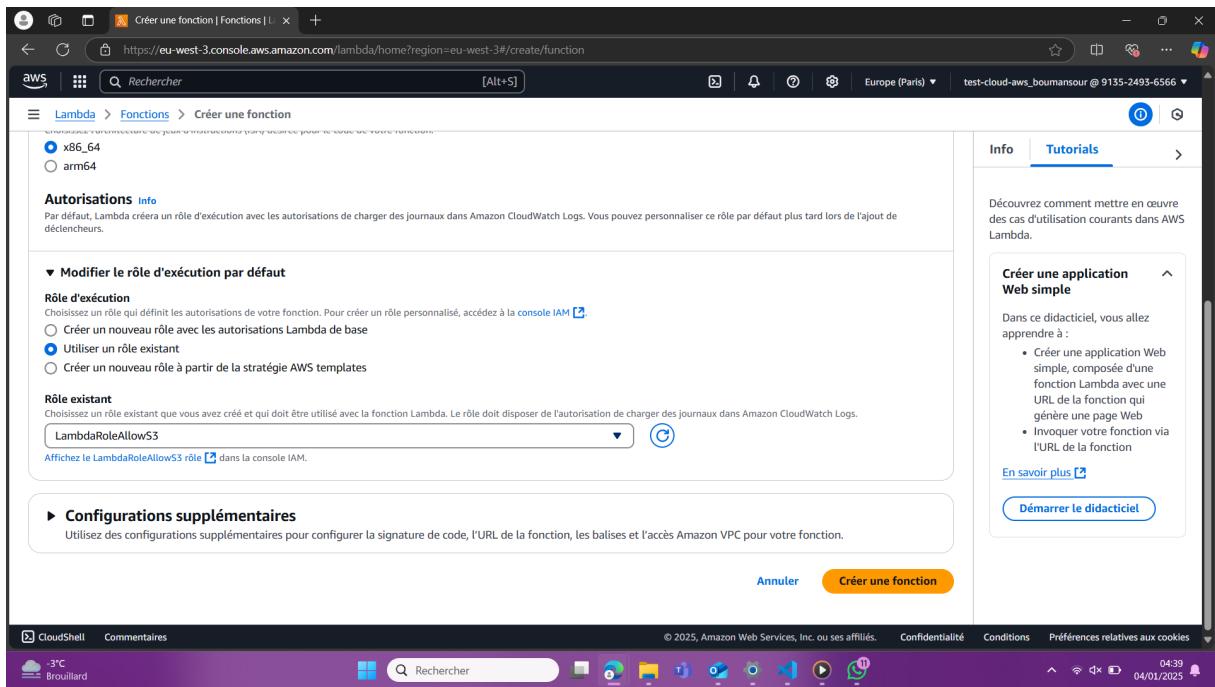


FIGURE 3.7 – Ajout du rôle à la fonction lambda

On ajoute par la suite le rôle qu'on a crée précédemment à notre fonction lambda afin qu'elle puisse accéder à la bucket S3 qu'on a crée et aussi pour pouvoir suivre ses logs sur Cloudwatch

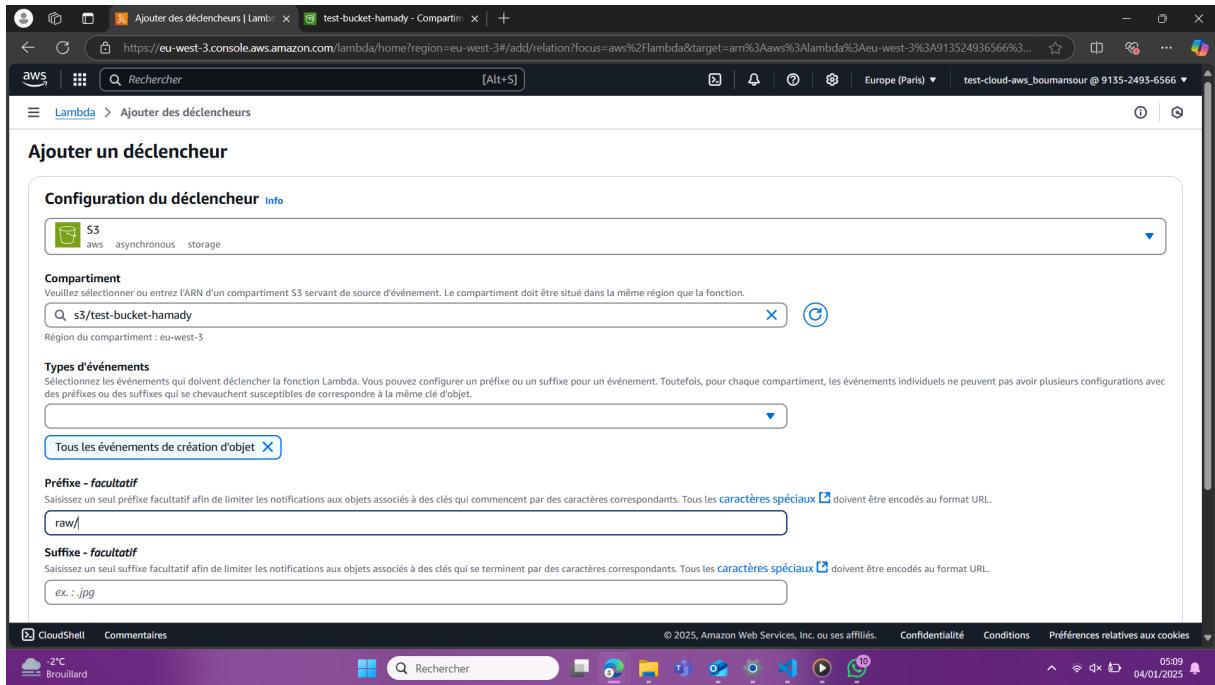


FIGURE 3.8 – Ajout du déclencheur à la fonction lambda

On ajoute un déclencheur pour garantir que cette fonction Lambda est déclenchée automatiquement lorsqu'un fichier est téléchargé dans le dossier `raw/` du bucket S3 `test-bucket-hamady`.

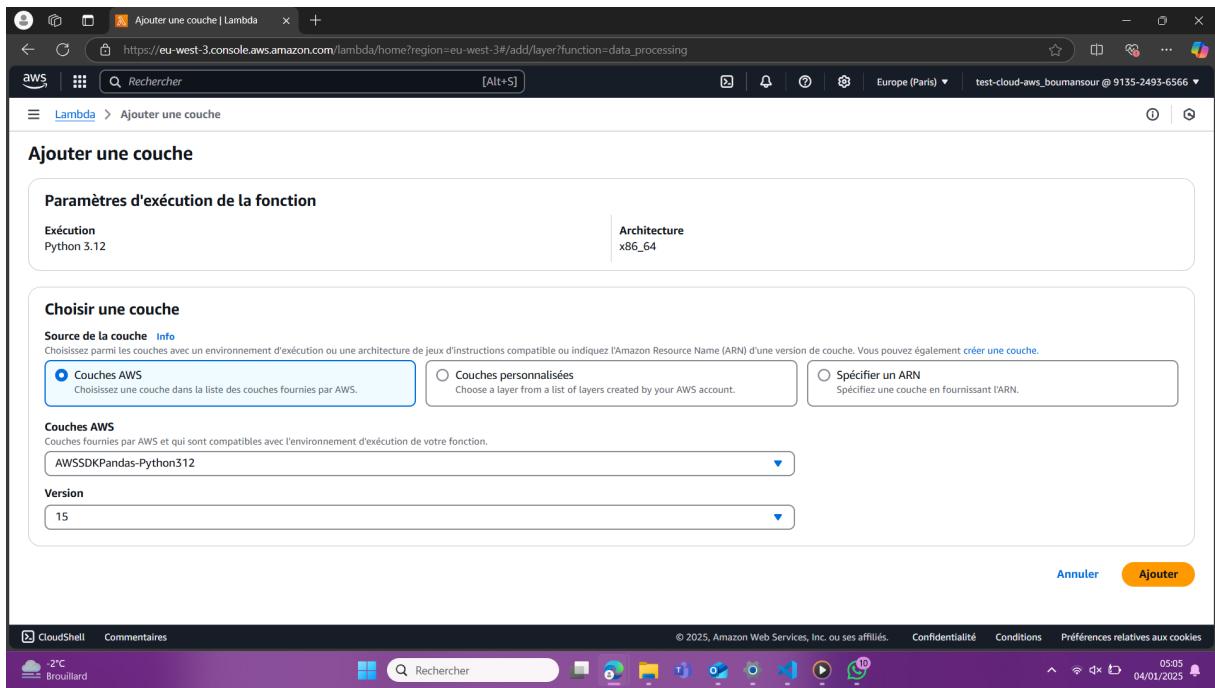


FIGURE 3.9 – Ajout d'une couche pour executer lambda avec pandas

La figure 3.9 illustre l'ajout d'une couche dans AWS Lambda pour permettre l'utilisation de la bibliothèque Pandas dans une fonction Lambda. La couche sélectionnée, `AWSSDKPandas-Python312`, est compatible avec l'environnement d'exécution Python 3.12. Cette configuration est nécessaire pour effectuer des traitements avancés sur des données tabulaires dans le cadre de l'exécution de la fonction Lambda.

3.2. Création de la fonction Lambda

xx

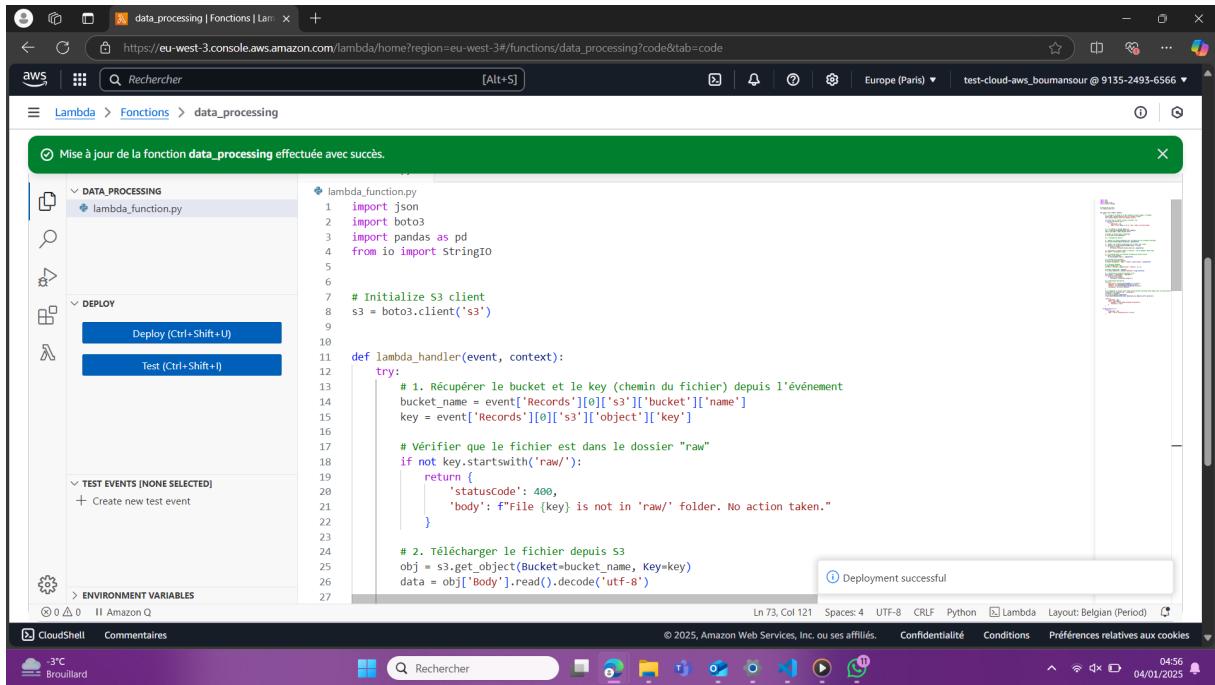


FIGURE 3.10 – Ajout du code et le déploiement de la fonction lambda

La fonction lambda exécute les étapes suivantes :

1. Validation et récupération du fichier :

- Vérifie que le fichier se trouve dans le dossier `raw/`.
- Télécharge le fichier CSV depuis S3.

2. Traitement des données :

- Remplit les valeurs manquantes pour les variables numériques avec leur médiane et pour les variables catégoriques (sauf `Cabin`) avec leur mode.
- Supprime la colonne `Cabin` en raison du nombre élevé de valeurs manquantes.
- Exclut les colonnes inutiles (`PassengerId`, `Name`, `Ticket`).
- Encode les variables catégoriques (`Sex` avec *Label Encoding* et `Embarked` avec *One-Hot Encoding*).
- Convertit les colonnes booléennes (`Embarked_Q`, `Embarked_S`) en entiers.

3. Statistiques descriptives :

- Calcule les statistiques de base (moyenne, écart-type, etc.), l'asymétrie (*skewness*), la kurtosis et les corrélations des données traitées.

4. Enregistrement des résultats :

- Enregistre le fichier traité dans le dossier `processed/` du bucket S3.

5. Retour d'information :

- Retourne un message confirmant le succès du traitement, accompagné des statistiques descriptives calculées.

Cette fonction automatise la préparation et la transformation des données tout en garantissant une gestion efficace des fichiers dans le bucket S3 pour les étapes ultérieures du projet.

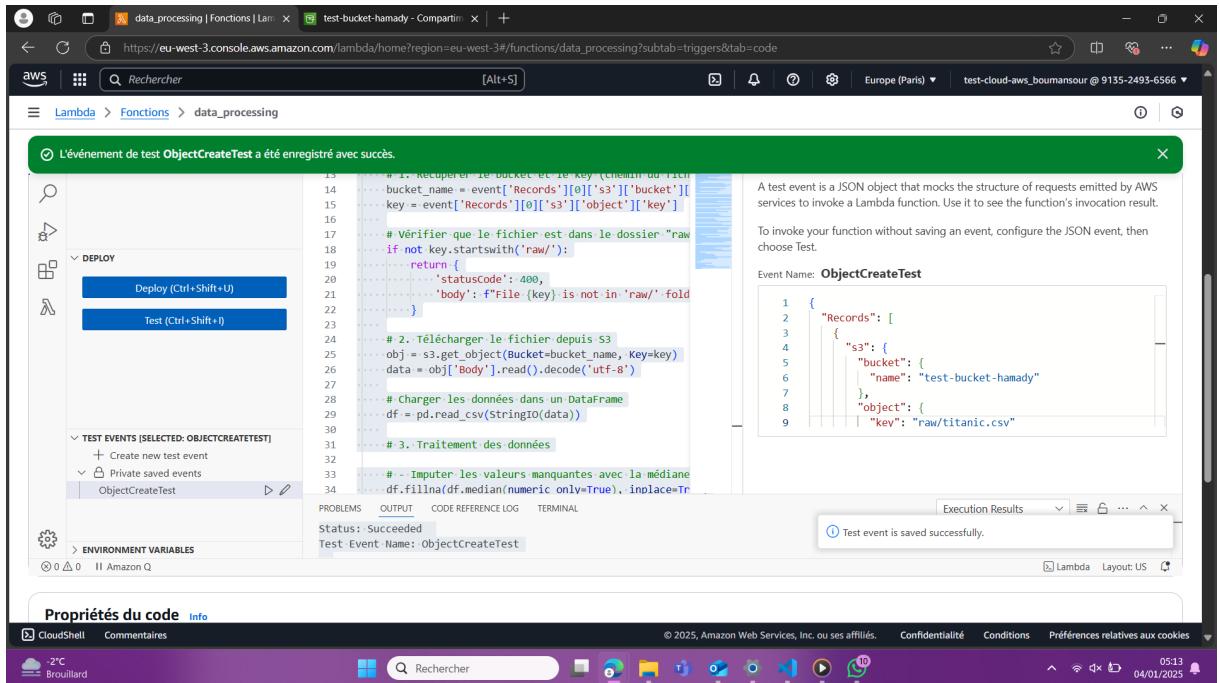
3.3 Traitement des données : valeurs manquantes, encodage, statistiques descriptives

Le pré-traitement des données commence par l'imputation des valeurs manquantes. Les variables numériques ont leurs valeurs manquantes remplacées par la médiane, tandis que les variables catégorielles (sauf la colonne "Cabin") sont remplies avec la mode de chaque colonne. La colonne "Cabin" est ensuite transformée en une valeur booléenne, où True indique un manque de valeur et False indique une présence. Étant donné que "Cabin" présente un taux élevé de valeurs manquantes, elle est finalement supprimée. D'autres colonnes inutiles, telles que "PassengerId", "Name" et "Ticket", sont également supprimées pour alléger le dataset.

L'encodage des variables suit ensuite. La variable "Sex" est convertie en format numérique, où "male" devient 0 et "female" devient 1. La variable "Embarked" est transformée à l'aide d'un encodage one-hot, créant des colonnes binaires pour chaque catégorie possible de "Embarked".

Enfin, des statistiques descriptives sont générées. Cela inclut le calcul des mesures de tendance centrale (comme la moyenne et la médiane), des mesures de forme de la distribution des données (asymétrie et kurtose), ainsi que des corrélations entre les variables numériques.

3.4 Test de la fonction lambda et vérification des logs



3.4. Test de la fonction lambda et vérification des logs

xxiii

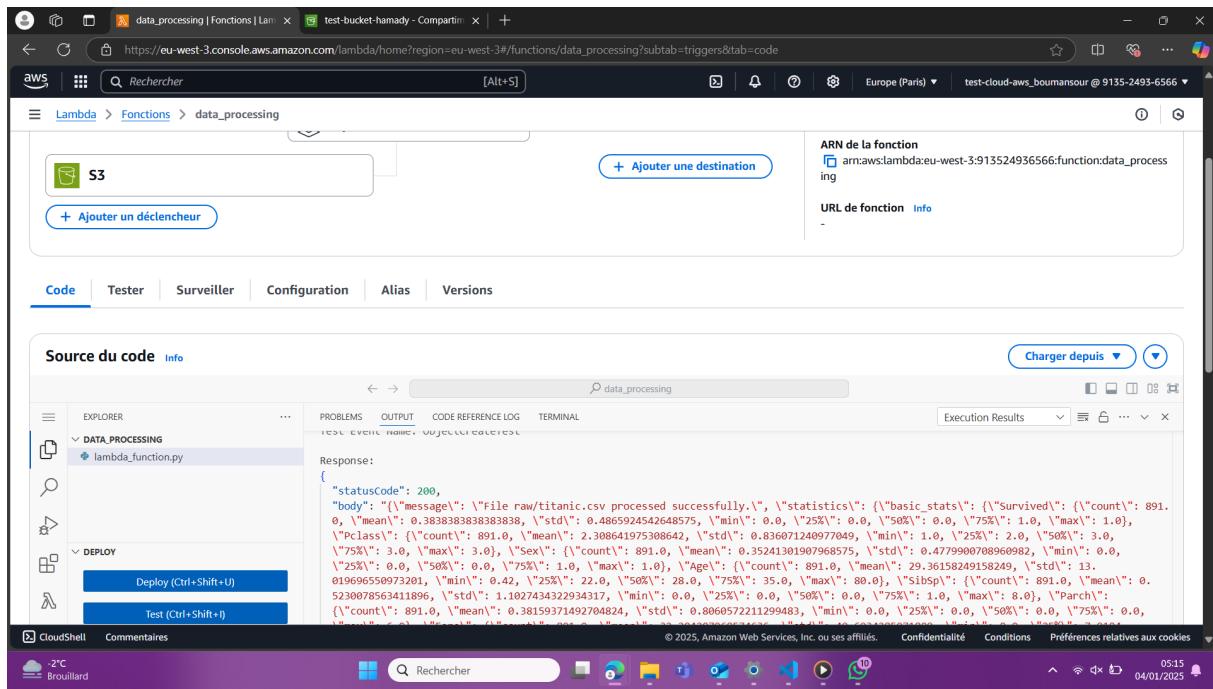


FIGURE 3.12 – Test réussi pour la fonction lambda

Grace à ce résultat on s'assure que notre fonction lambda parvient à être déclenchée à chaque ajout de nouveau fichier .

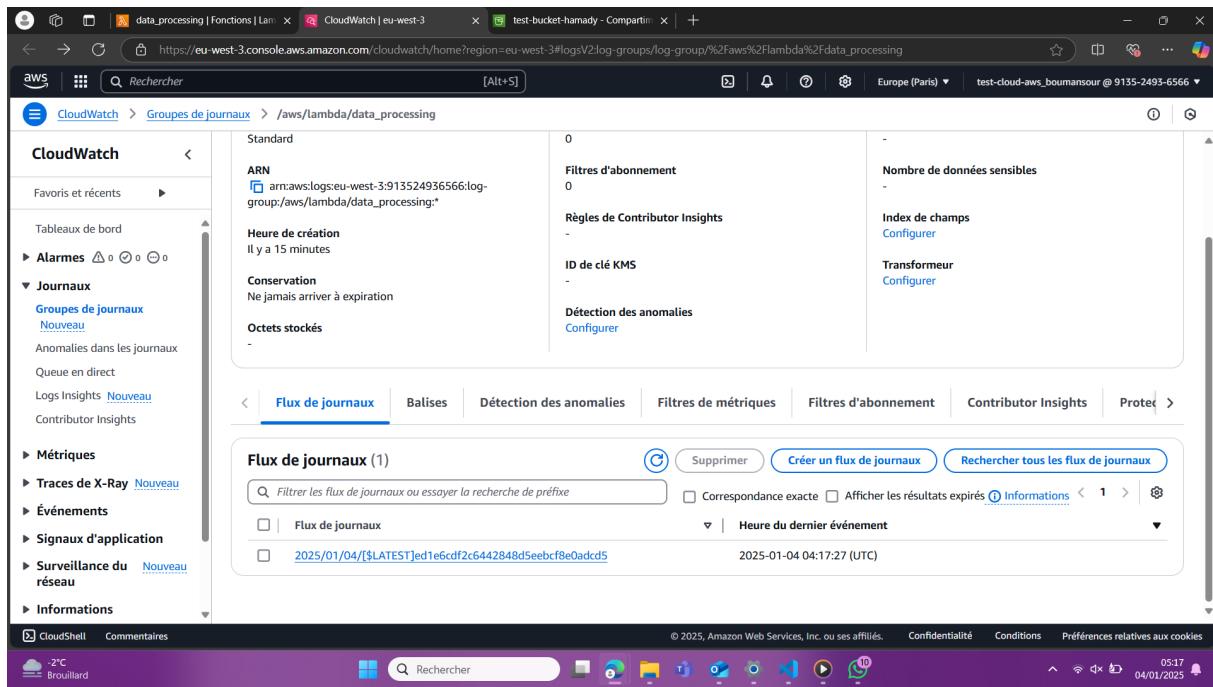


FIGURE 3.13 – Journal apparue sur cloudWatch après l'exécution du test

Lors de l'exécution du test de traitement un flux de journaux a été automatiquement généré et capturé dans AWS CloudWatch. On voit dans la figure 3.13 montrer l'interface de CloudWatch où ce flux de journaux est visible. Le journal est associé au groupe de journaux nommé aws/lambda/dataprocessing et inclut les détails liés à l'exécution, tels que l'ARN, la durée de conservation des journaux (14 jours), et les paramètres de configuration des abonnements.

Ce flux de journaux fournit des informations essentielles pour le suivi, la détection des anomalies, et le débogage des fonctions Lambda. La structure de CloudWatch permet une analyse approfondie des événements grâce à ses outils de recherche et de filtrage. Le flux confirme le bon fonctionnement des mécanismes de journalisation après le déploiement de la fonction.

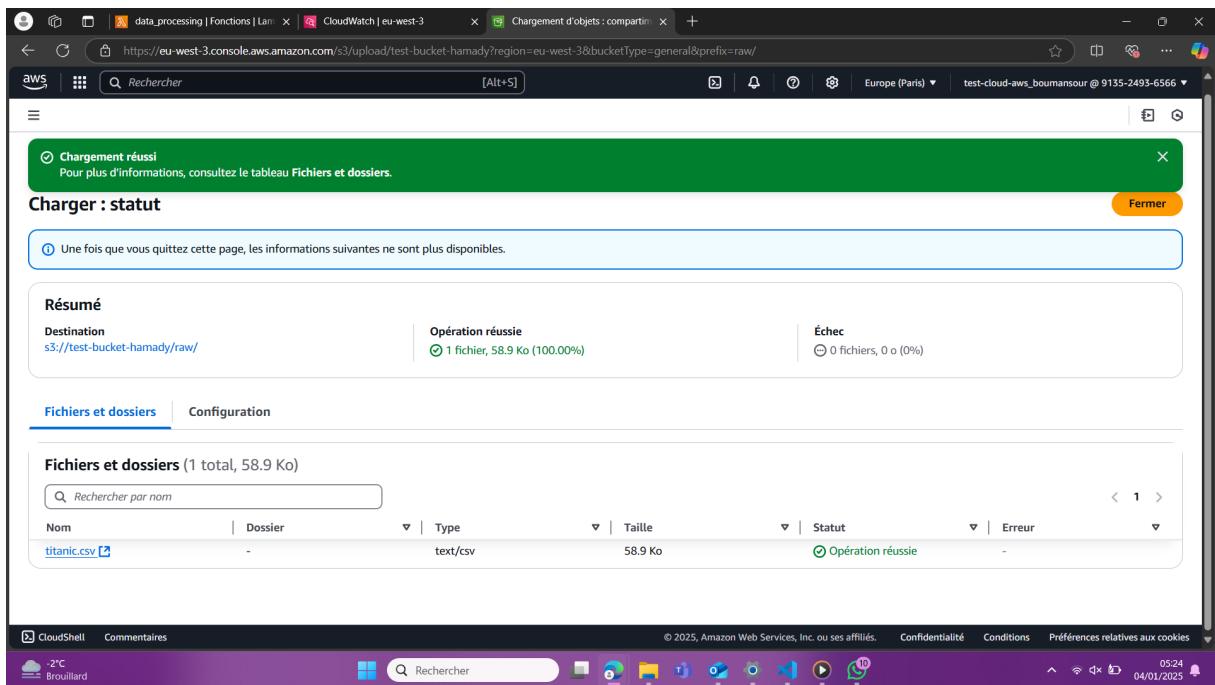


FIGURE 3.14 – Ajout manuel d'un fichier pour surveiller le déclenchement de la fonction lambda

On refait un test manuel également en ajoutant un fichier dans le bucket créé .

3.4. Test de la fonction lambda et vérification des logs

xxv

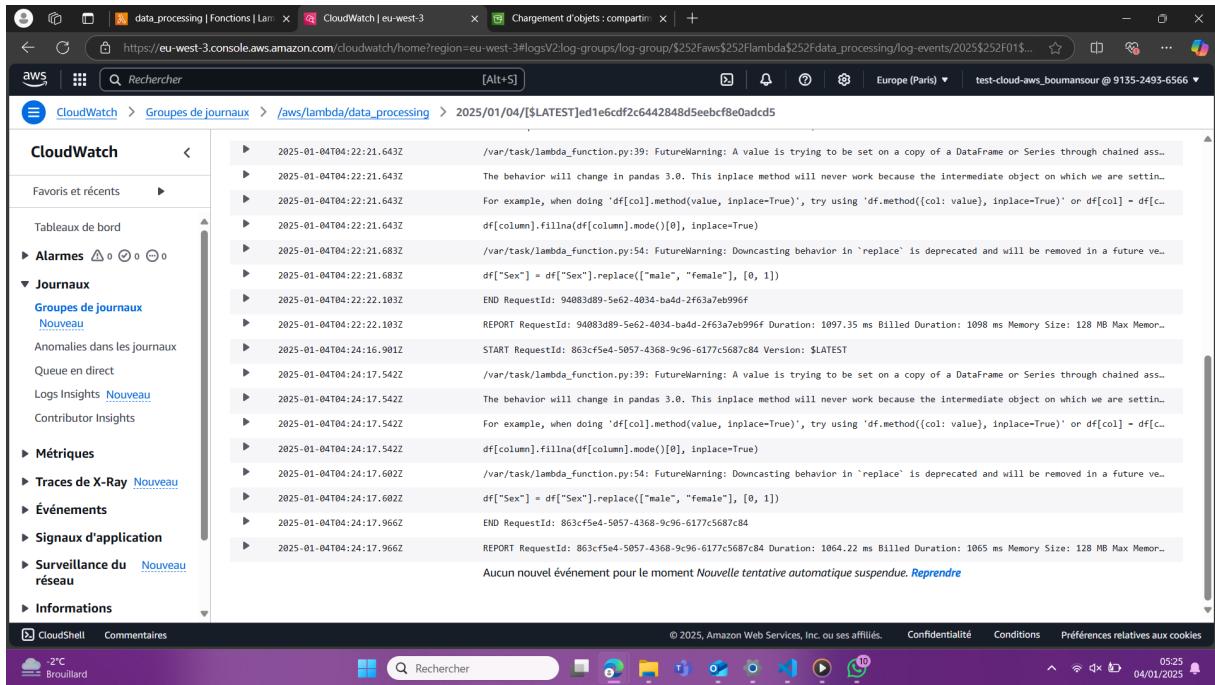


FIGURE 3.15 – Enregistrement des nouveaux logs dans les journaux après l'ajout d'un fichier dans le bucket

On retrouve de nouveaux logs ce qui témoigne le bon fonctionnement de notre lambda function .

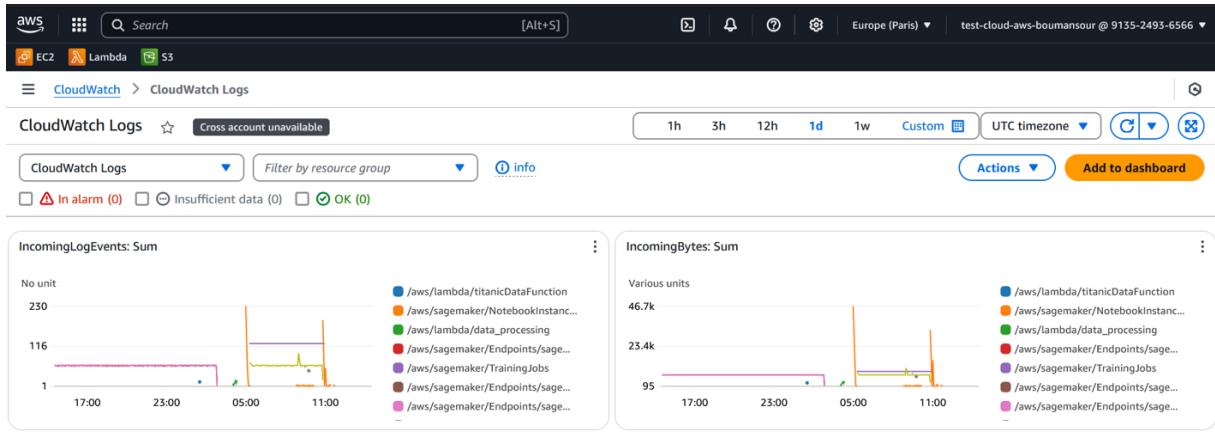


FIGURE 3.16 – Visualisation sur CloudWatch avec graphes

La figure 3.16 montre les graphiques générés par CloudWatch pour suivre l'activité des fonctions et services AWS. Le graphique **IncomingLogEvents : Sum** présente le volume des événements journaux, tandis que **IncomingBytes : Sum** affiche la taille des données entrantes. Ces visualisations permettent d'identifier les pics d'activité et d'analyser les

tendances pour optimiser les ressources. On retrouve le dossier correspondant à notre fonction en vert .

3.5 Résultats du traitement dans S3

The screenshot shows the AWS S3 console interface. The URL in the browser is <https://eu-west-3.console.aws.amazon.com/s3/buckets/test-bucket-hamady?region=eu-west-3&bucketType=general&prefix=processed/&showversions=false>. The left sidebar shows the navigation path: Amazon S3 > Compartiments > test-bucket-hamady > processed/. The main content area displays the 'processed/' folder. There is one object listed: 'titanic.csv' (CSV type, 21.9 Ko). The 'Actions' menu is open above the object list, showing options like Copier l'URI S3, Copier l'URL, Télécharger, Ouvrir, Supprimer, Actions, Crée un dossier, and Charger.

Nom	Type	Dernière modification	Taille	Classe de stockage
titanic.csv	csv	04 Jan 2025 05:24:18 AM +01	21.9 Ko	Standard

FIGURE 3.17 – Ajout automatiques des résultats dans le dossier processed du bucket

Enfin , On retrouve nos données traitées pretes pour la prédiction dans le dossier processed/ de notre bucket .

4

Entraînement et Prédiction du Modèle

4.1 Création d'un rôle IAM pour le notebook

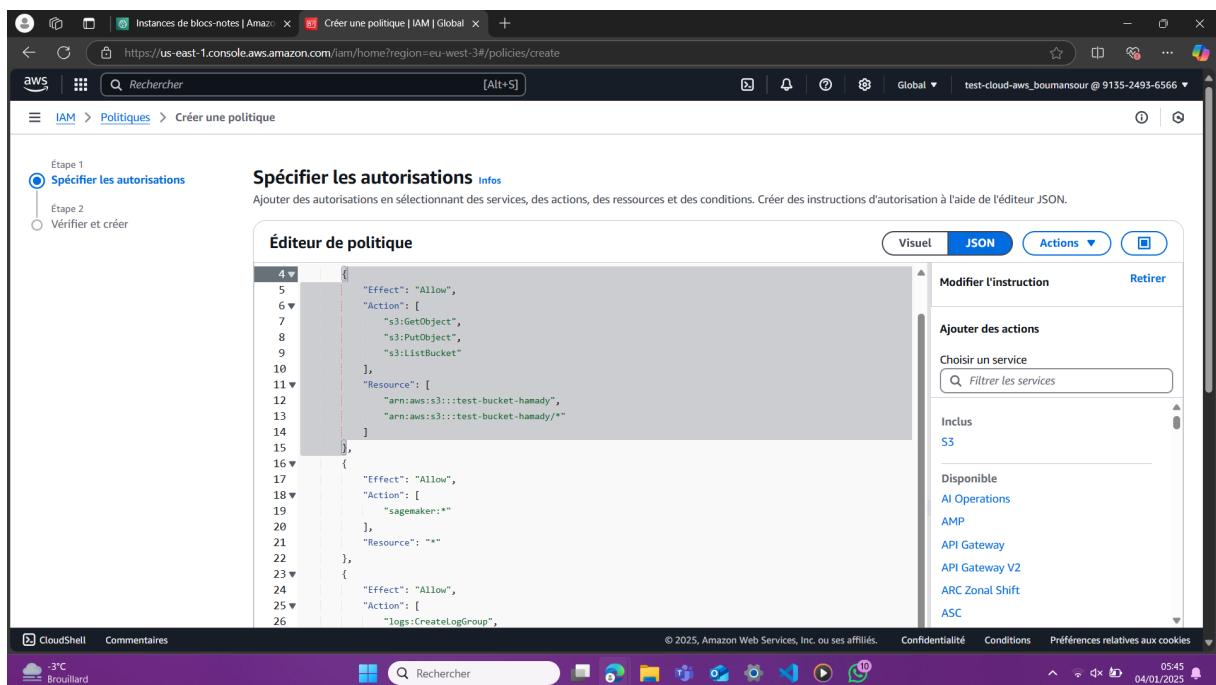


FIGURE 4.1 – Ajout d'une politique à insérer au rôle IAM du notebook sagemaker

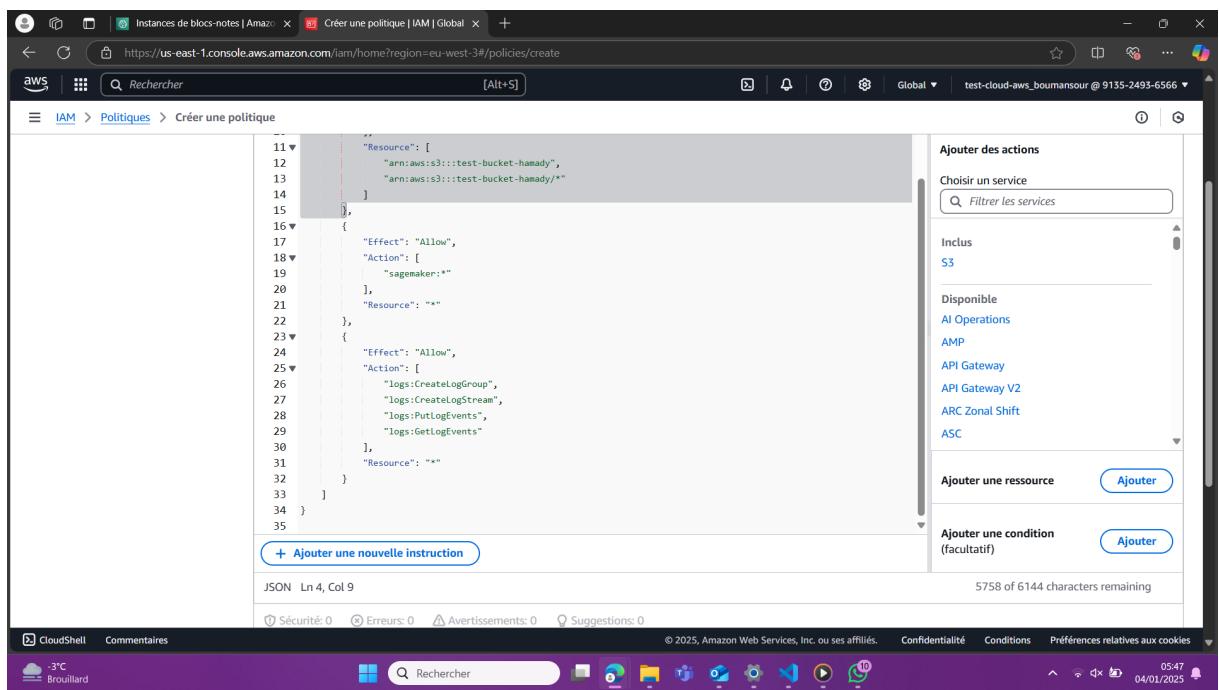


FIGURE 4.2 – Ajout d'une politique à insérer au rôle IAM du notebook sagemaker - suite

On définit une politique personnalisée IAM en JSON pour un rôle associé au notebook SageMaker. Cette politique accorde des autorisations spécifiques, comme l'accès aux actions S3, CloudWatch et un accès total aux actions sagemaker , nécessaires pour exécuter les tâches du notebook. Elle garantit que les ressources, telles que le bucket S3 et les journaux CloudWatch, sont accessibles en toute sécurité tout en respectant les meilleures pratiques en matière de gestion des droits Tout en gardant le principe de moindre privilège en laissant accès qu'à des ressources spécifiques .

4.1. Création d'un rôle IAM pour le notebook

xxix

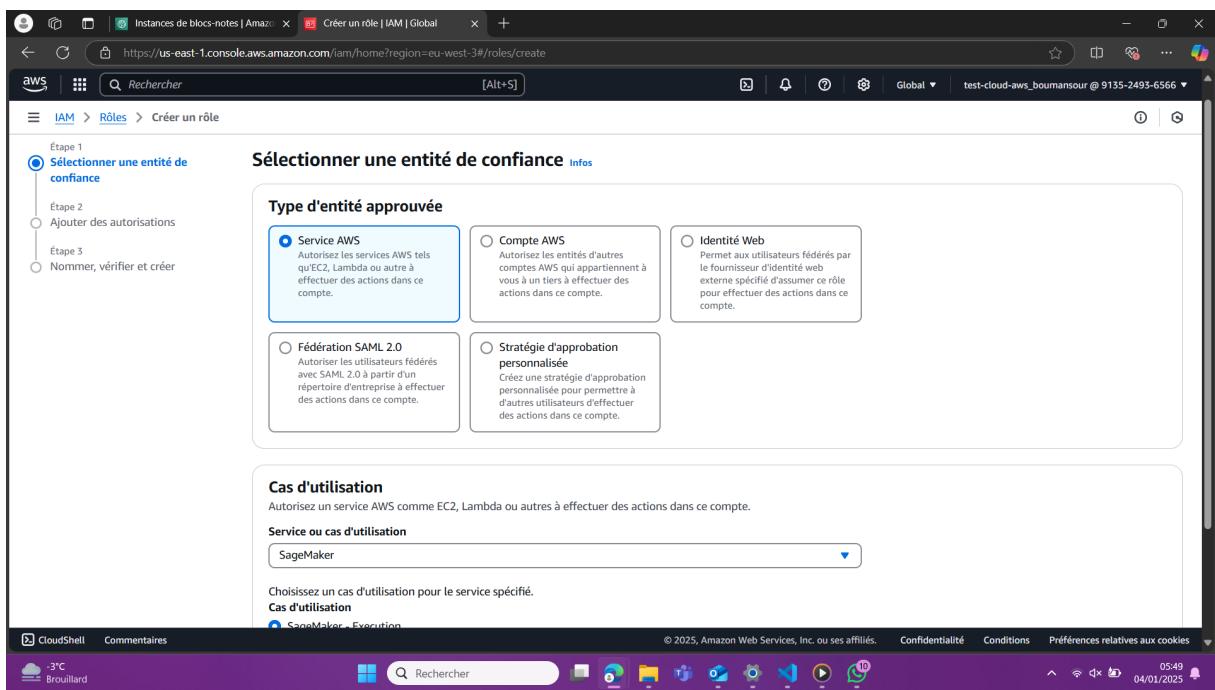


FIGURE 4.3 – Création du rôle IAM pour le notebook

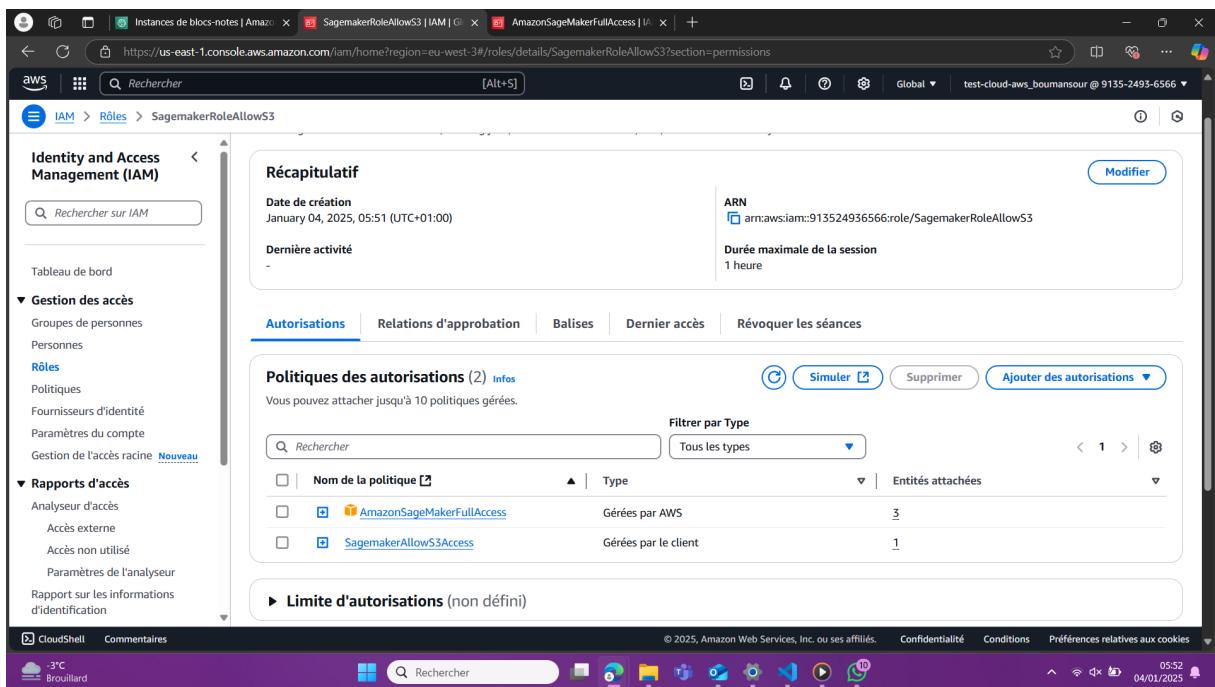


FIGURE 4.4 – Création du rôle IAM pour le notebook - suite

On crée un rôle IAM avec la politique créée précédemment ainsi que la politique gérée par AWS : AmazonSagemakerFullAccess

4.2 Séparation en ensembles d'entraînement et de test

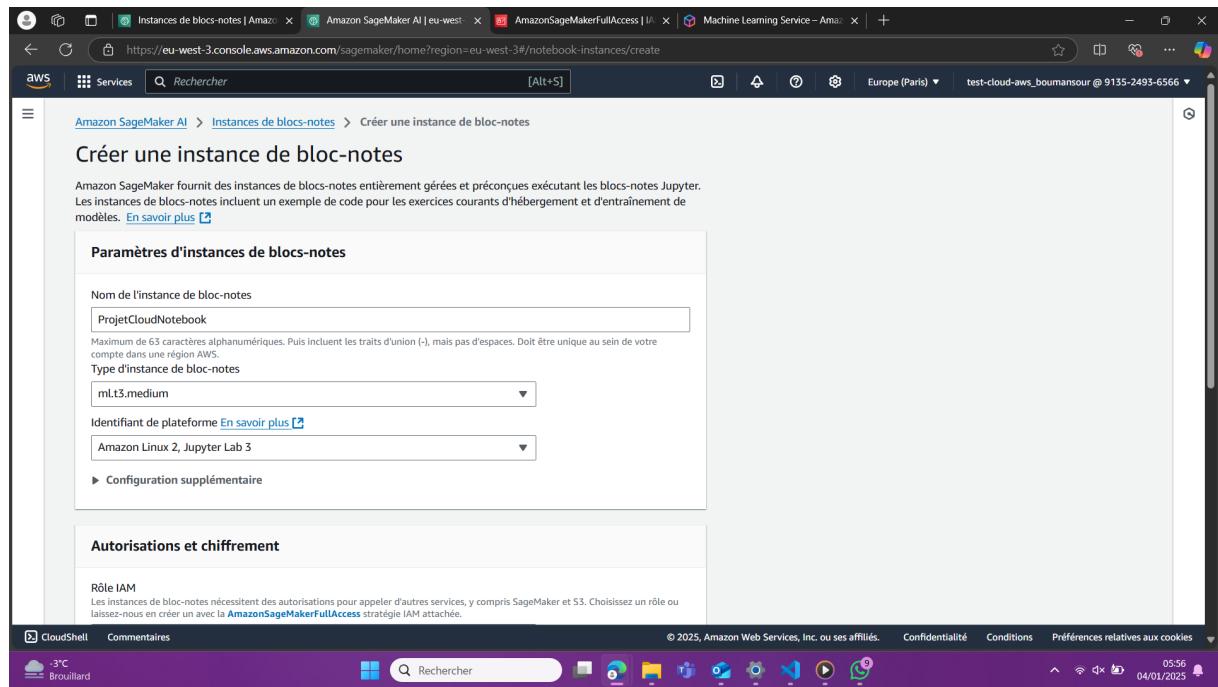


FIGURE 4.5 – Creation du notebook sagemaker

Pour cette partie du projet on crée une instance de notebook sur Amazon SageMaker AI avec ml.t3.medium qui correspond à celle de l'offre gratuite de AWS .

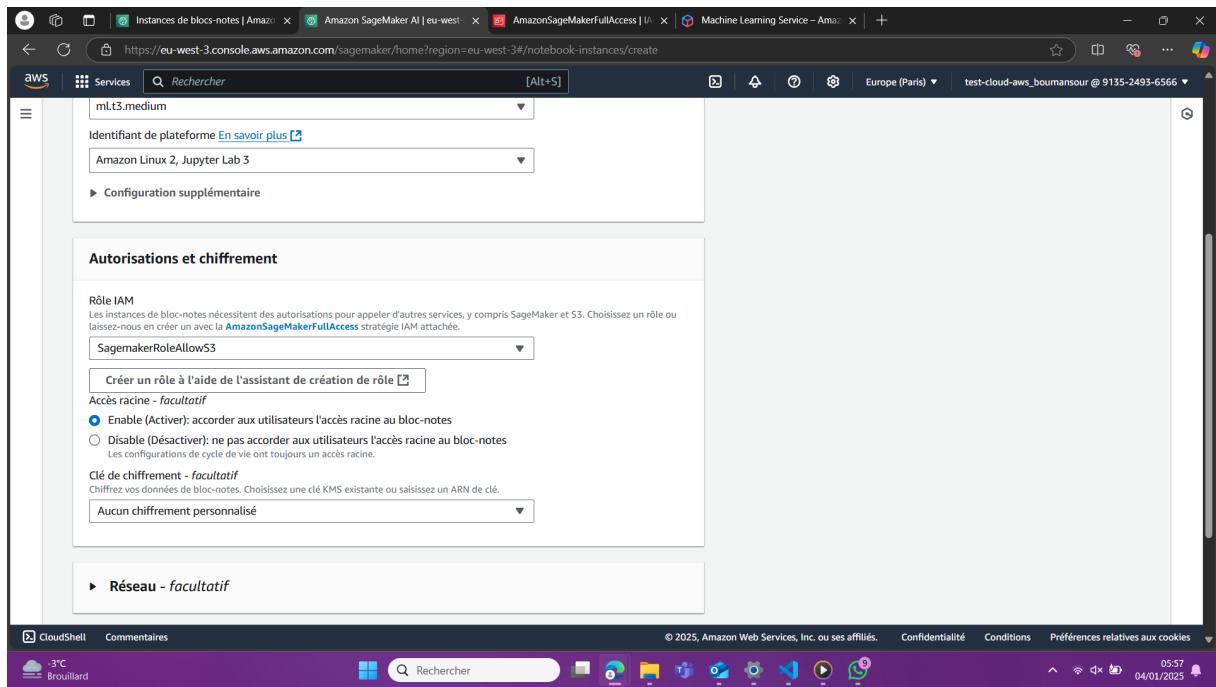


FIGURE 4.6 – Ajout du rôle créé au notebook

On ajoute le rôle créé précédemment pour donner les accès nécessaires à ce groupe pour récupérer les données de S3 .

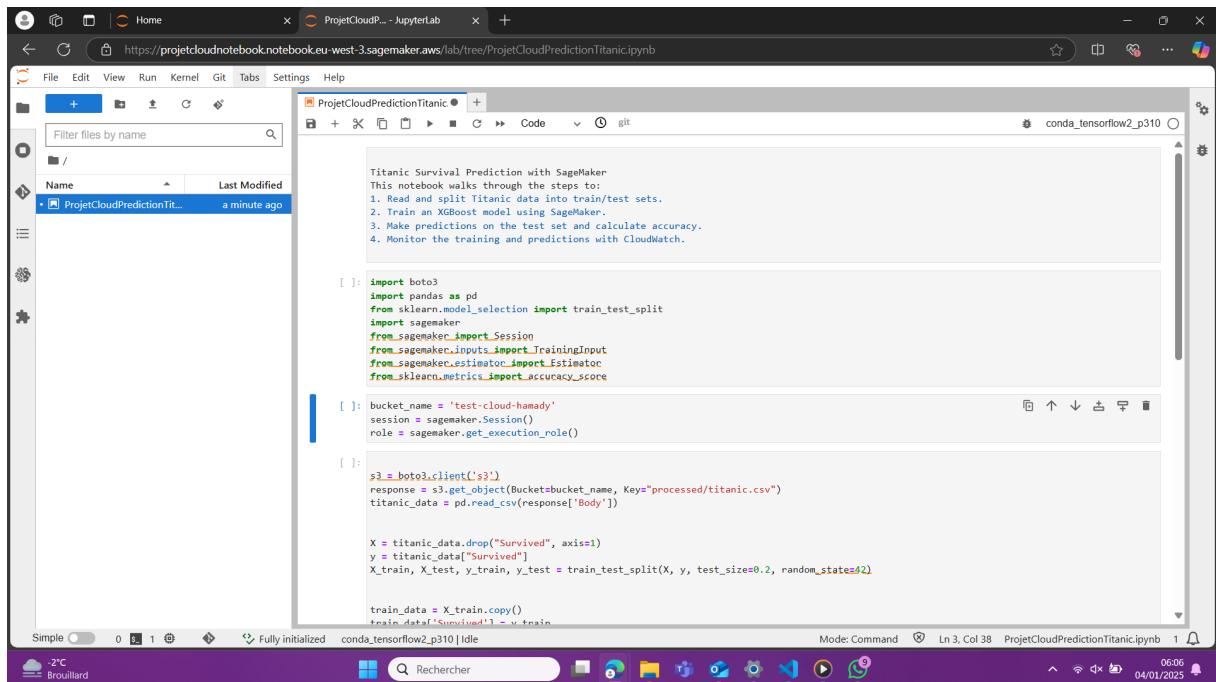
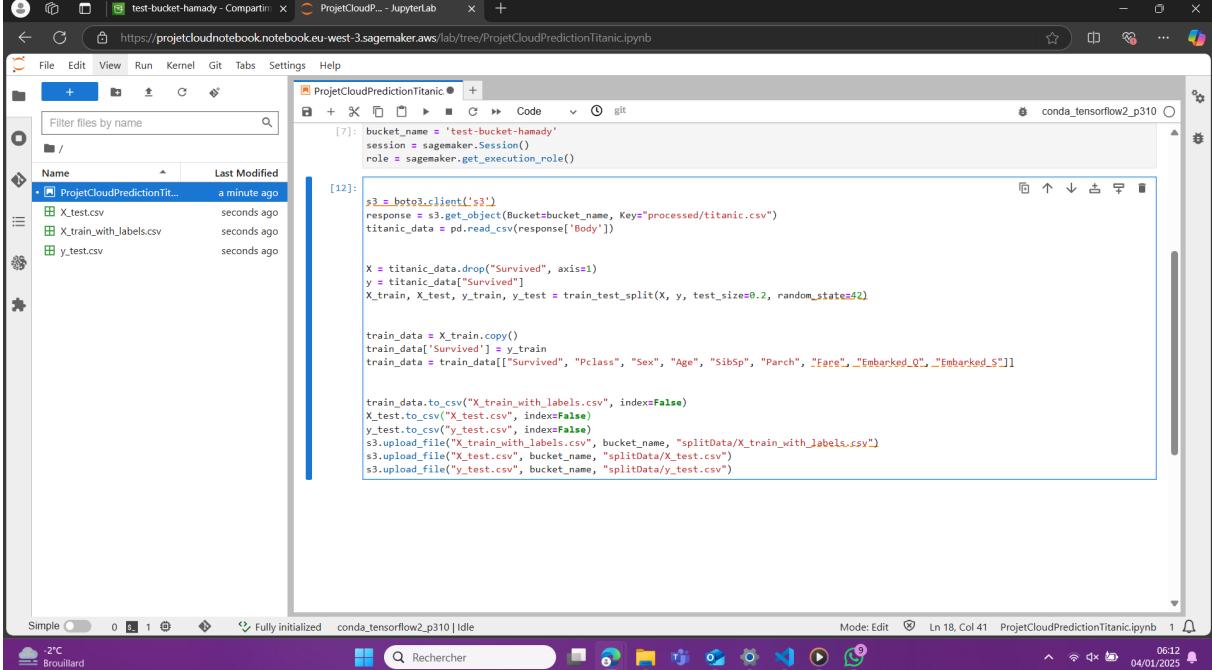


FIGURE 4.7 – Crédit d'un notebook jupyter et ajout des librairies nécessaires

On importe les libraires nécessaires pour nos traitement comme sklearn pour la séparation des données en données d'entraînements et données de tests .



```

File Edit View Run Kernel Git Tabs Settings Help
File + ☰ Filter files by name
Name Last Modified
ProjectCloudPredictionTit... a minute ago
X_test.csv seconds ago
X_train_with_labels.csv seconds ago
y_test.csv seconds ago
[7]: bucket_name = 'test-bucket-hamady'
session = sagemaker.Session()
role = sagemaker.get_execution_role()

[112]: s3 = boto3.client('s3')
response = s3.get_object(Bucket=bucket_name, Key="processed/titanic.csv")
titanic_data = pd.read_csv(response['Body'])

X = titanic_data.drop("Survived", axis=1)
y = titanic_data["Survived"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

train_data = X_train.copy()
train_data["Survived"] = y_train
train_data = train_data[["Survived", "Pclass", "Sex", "Age", "SibSp", "Parch", "Fare", "Embarked_Q", "Embarked_S"]]

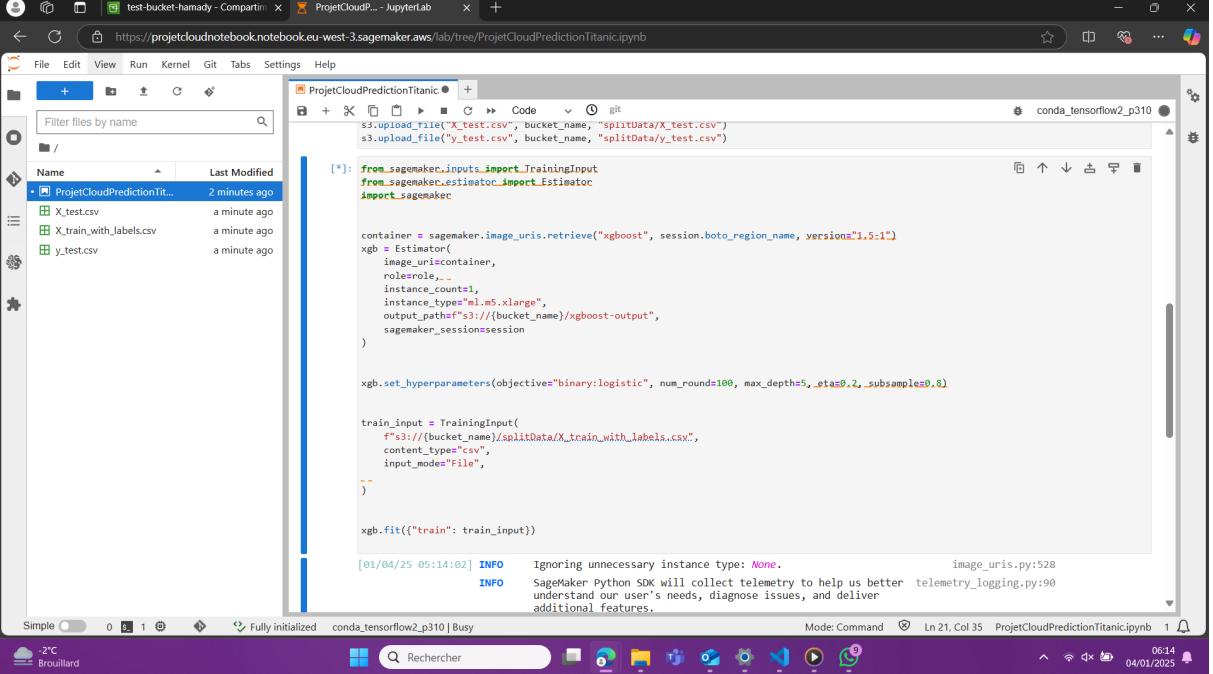
train_data.to_csv("X_train_with_labels.csv", index=False)
X_test.to_csv("X_test.csv", index=False)
y_test.to_csv("y_test.csv", index=False)
s3.upload_file("X_train_with_labels.csv", bucket_name, "splitData/X_train_with_labels.csv")
s3.upload_file("X_test.csv", bucket_name, "splitData/X_test.csv")
s3.upload_file("y_test.csv", bucket_name, "splitData/y_test.csv")

```

FIGURE 4.8 – Division des données en train/test

On divise les données en gardant vingt pourcent pour le test . On sauvegarde le tous dans un dossier appelé splitData . On ajoute la columne du label aux données de training et on les sauvegarde combinées car par défaut le modèle XGBoost de AWS nécessite que le label soit dans les données d'entraînements .

4.3 Entraînement du modèle XGBoost



The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** test-bucket-hamady - Compartir | ProjetCloudP... - JupyterLab
- File Explorer:** Shows files in the 'ProjetCloudPredictionTitanic' directory, including 'X_test.csv', 'X_train_with_labels.csv', and 'y_test.csv', all updated 'a minute ago'.
- Code Cell:**

```

[*]: from sagemaker.inputs import TrainingInput
from sagemaker.estimator import Estimator
import sagemaker

container = sagemaker.image_uris.retrieve("xgboost", session.boto_region_name, version="1.5-1")
xgb = Estimator(
    image_url=container,
    role=role,
    instance_count=1,
    instance_type="ml.m5.xlarge",
    output_path=f"s3://{bucket_name}/xgboost-output",
    sagemaker_session=session
)

xgb.set_hyperparameters(objective="binary:logistic", num_round=100, max_depth=5, eta=0.2, subsample=0.8)

train_input = TrainingInput(
    f"s3://{bucket_name}/splitData/X_train_with_labels.csv",
    content_type="csv",
    input_mode="File",
)

```
- Output Panel:** Displays log messages:


```
[01/04/25 05:14:02] INFO Ignoring unnecessary instance type: None. image_uris.py:528
[01/04/25 05:14:02] INFO SageMaker Python SDK will collect telemetry to help us better understand our user's needs, diagnose issues, and deliver additional features. telemetry_logging.py:90
```
- System Tray:** Shows weather (2°C), Brouillard, and system icons.
- Bottom Bar:** Shows the notebook is 'Fully initialized' and the kernel is 'conda_tensorflow2_p310 | Busy'. The status bar indicates 'Ln 21, Col 35' and the date '04/01/2025'.

FIGURE 4.9 – Entraînement d'un modèle XGBoost

On configure un workflow pour entraîner un modèle XGBoost sur Amazon SageMaker. On commence par importer les bibliothèques nécessaires, notamment `TrainingInput` et `Estimator`, qui facilitent la gestion des données et l'entraînement du modèle. Ensuite, l'image préconstruite de XGBoost (version 1.5-1) est récupérée à partir des conteneurs SageMaker, adaptée à la région AWS en cours d'utilisation. L'estimateur XGBoost est ensuite configuré avec des paramètres essentiels : un rôle IAM permettant l'accès aux ressources AWS, une instance de type `ml.m5.xlarge` pour l'entraînement, et un bucket S3 pour stocker les résultats. Les hyperparamètres du modèle sont également définis, notamment l'objectif de classification binaire (`binary :logistic`), un nombre de 100 itérations (`numround`), une profondeur maximale de 5 pour les arbres, un taux d'apprentissage (`eta`) de 0.2, et une sous-échantillonnage de 80 pourcent des données.

Enfin, un objet `TrainingInput` est créé pour spécifier l'emplacement des données d'entraînement dans un fichier CSV stocké sur S3, avec une configuration en mode fichier pour faciliter l'accès lors de l'entraînement. Ce workflow garantit une mise en place rapide et efficace pour l'entraînement du modèle XGBoost dans un environnement SageMaker.

4.3. Entraînement du modèle XGBoost

xxxiv

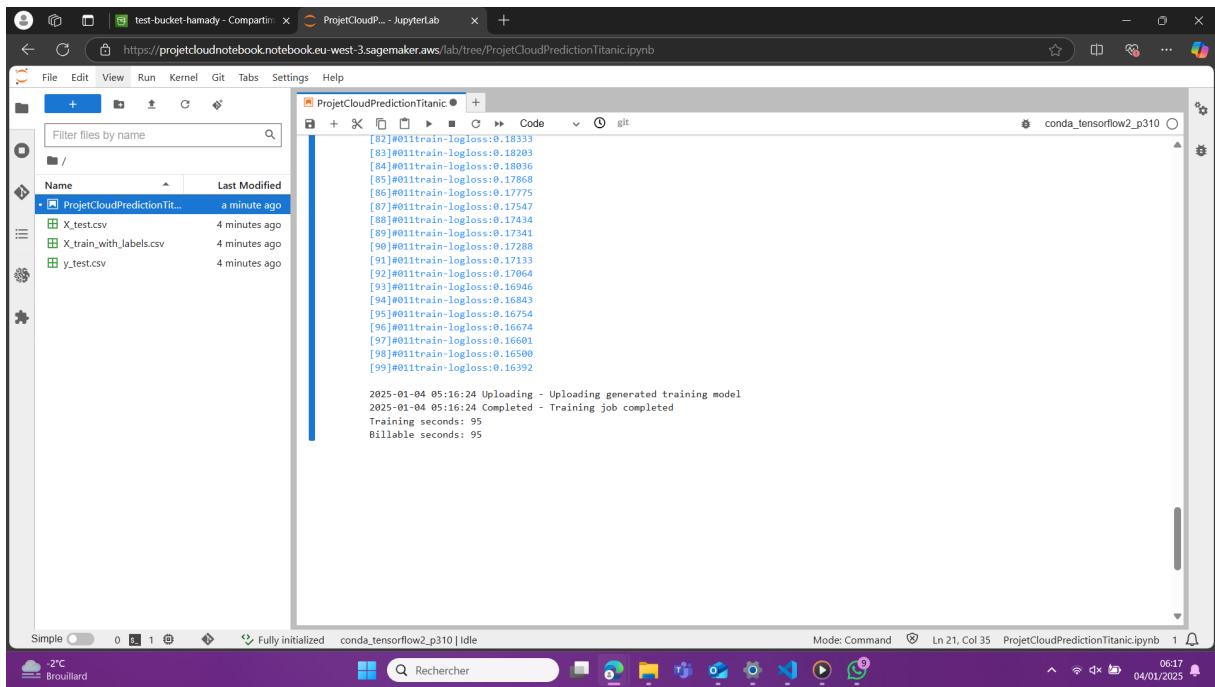


FIGURE 4.10 – Entraînement d'un modèle XGBoost réussi

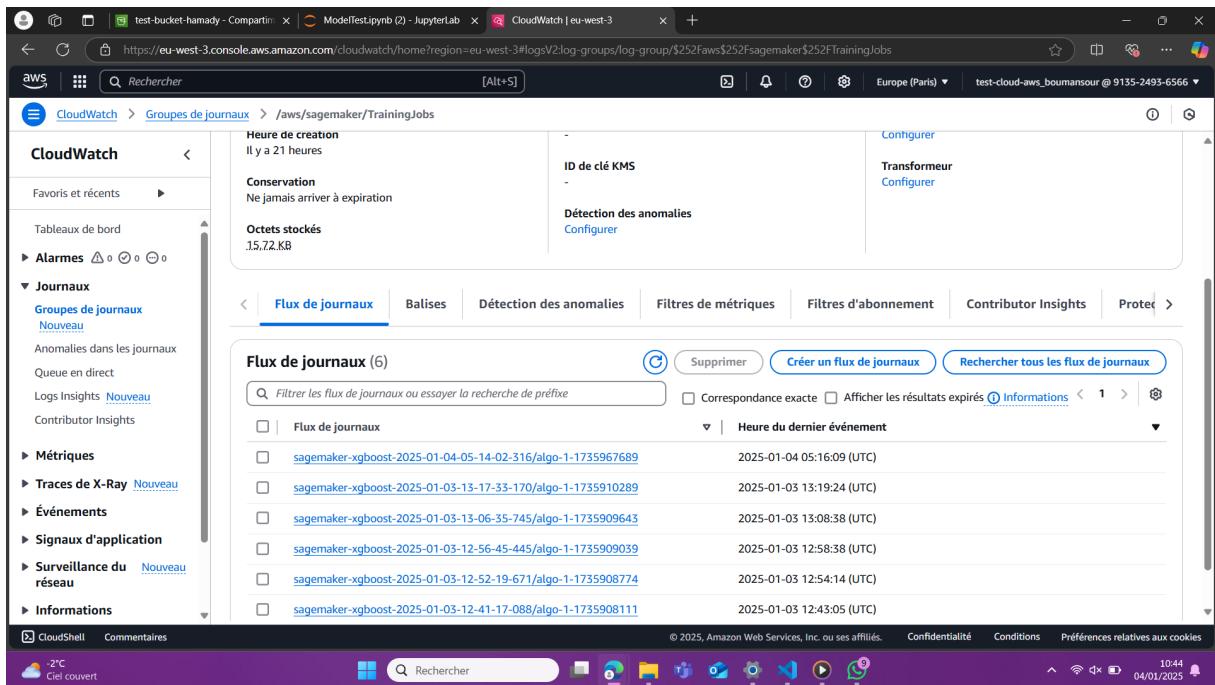


FIGURE 4.11 – Les journaux correspondants à l'entraînement du modèle sur cloudwatch

On surveille les journaux correspondants à l'entraînement du modèle sur cloudWatch , qui montrent un entraînement réussi . On retrouve aussi un training job dans l'onglet

trainig jobs de AWS Sagemaker AI .

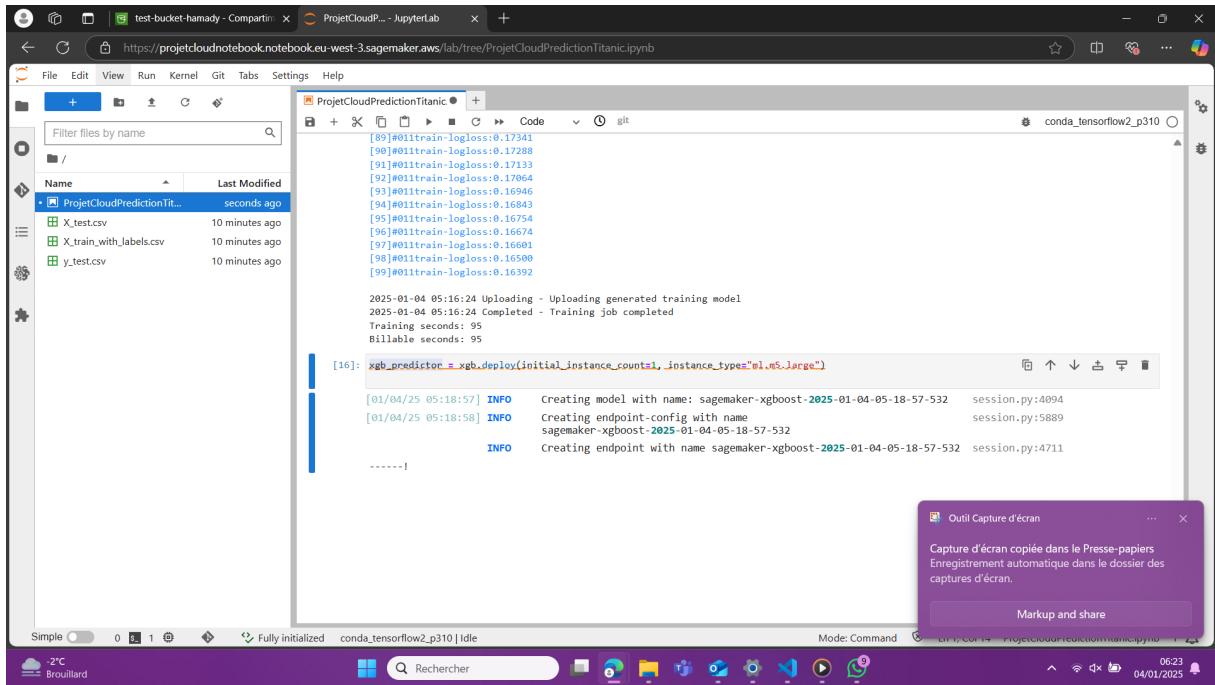
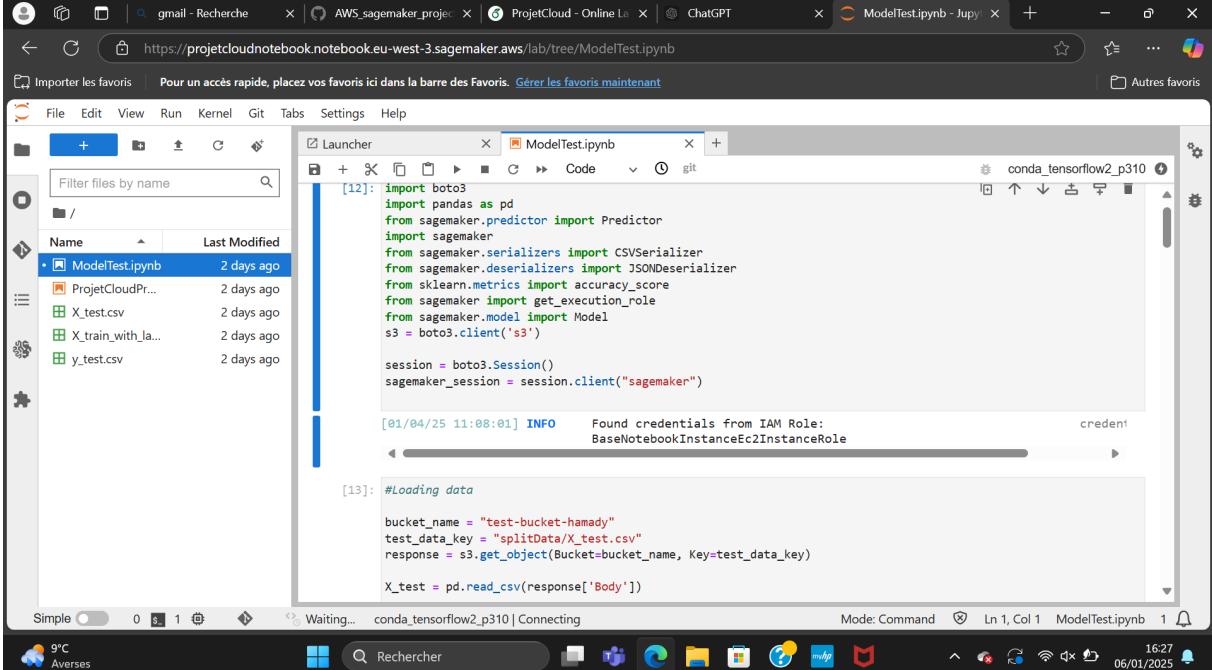


FIGURE 4.12 – Déploiement du modèle

On a également testé le déploiement du modèle même si c'est destiné aux inférences realtime et ça a été réussi , à noter que les inférences en realtime sont coûteuses au dépassement d'un seuil.

4.4 Prédictions sur l'ensemble de test



```
[12]: import boto3
import pandas as pd
from sagemaker.predictor import Predictor
import sagemaker
from sagemaker.serializers import CSVSerializer
from sagemaker.deserializers import JSONDeserializer
from sklearn.metrics import accuracy_score
from sagemaker import get_execution_role
from sagemaker.model import Model
s3 = boto3.client('s3')

session = boto3.Session()
sagemaker_session = session.client("sagemaker")

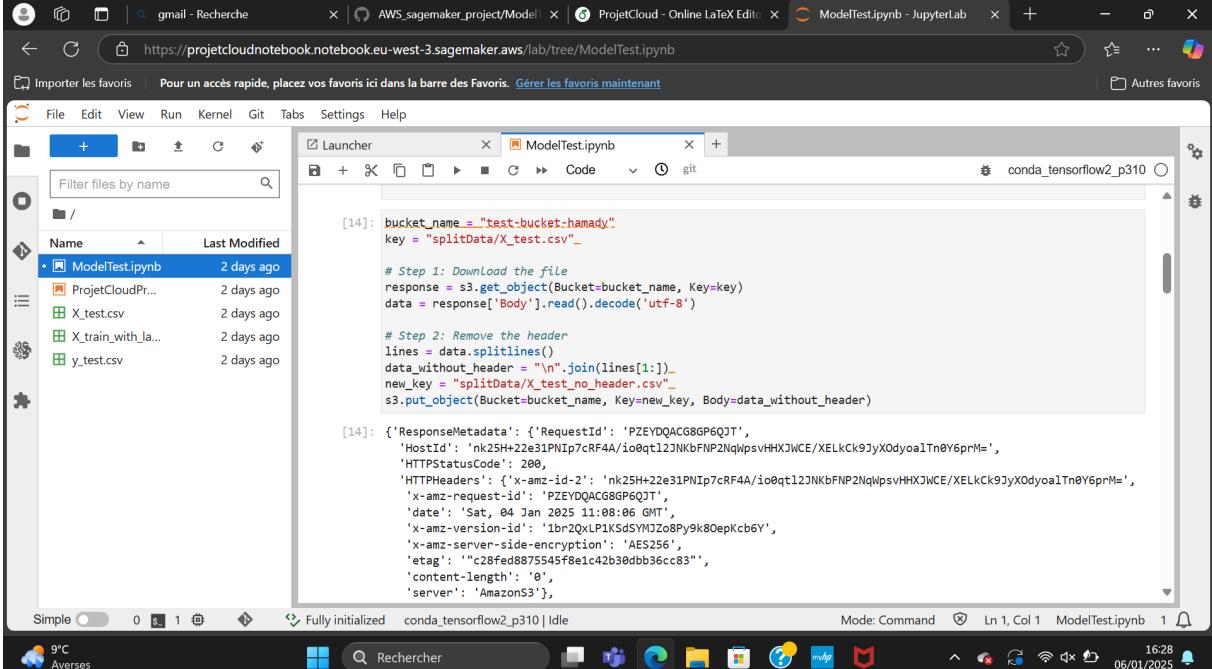
[01/04/25 11:08:01] INFO    Found credentials from IAM Role:
BaseNotebookInstanceEc2InstanceRole
```

[13]: #Loading data

```
bucket_name = "test-bucket-hamady"
test_data_key = "splitData/X_test.csv"
response = s3.get_object(Bucket=bucket_name, Key=test_data_key)

X_test = pd.read_csv(response['Body'])
```

FIGURE 4.13 – Chargement du modèle dans un deuxième notebook pour faire des prédictions



```
[14]: bucket_name = "test-bucket-hamady"
key = "splitData/X_test.csv"

# Step 1: Download the file
response = s3.get_object(Bucket=bucket_name, Key=key)
data = response['Body'].read().decode('utf-8')

# Step 2: Remove the header
lines = data.splitlines()
data_without_header = "\n".join(lines[1:])
new_key = "splitData/X_test_no_header.csv"
s3.put_object(Bucket=bucket_name, Key=new_key, Body=data_without_header)

[14]: {'ResponseMetadata': {'RequestId': 'PZEYDQACG8GP6QJT',
'HostId': 'nk25H+22e31PNiP7cRF4A/ie0qt12JNkbFNP2NqkpsvHXJWCE/XELkCk9JyXOdyoa1Tn0Y6prM=',
'HTTPStatusCode': 200,
'HTTPHeaders': {'x-amz-id-2': 'nk25H+22e31PNiP7cRF4A/ie0qt12JNkbFNP2NqkpsvHXJWCE/XELkCk9JyXOdyoa1Tn0Y6prM=',
'x-amz-request-id': 'PZEYDQACG8GP6QJT',
'date': 'Sat, 04 Jan 2025 11:08:08 GMT',
'x-amz-version-id': '1br2QxLP1KsdSMZ2o8Py9k80epKcb6Y',
'x-amz-server-side-encryption': 'AES256',
'etag': '"c28Fed8875545f8e1c42b3ddb36cc83"',
'content-length': '0',
'server': 'AmazonS3'},
```

FIGURE 4.14 – Récupération des données de test

```

[25]:
sagemaker_client = boto3.client("sagemaker", region_name="eu-west-3")

model_name = "xgboost-model-for-batch-transform"
model_data = "s3://test-bucket-hamady/xgboost-output/sagemaker-xgboost-2025-01-04-05-14-02-316/output/model.tar.gz"

container = sagemaker.image_uris.retrieve("xgboost", region="eu-west-3", version="1.5-1")

response = sagemaker_client.create_model(
    ModelName=model_name,
    PrimaryContainer={
        "Image": container,
        "ModelDataUrl": model_data,
    },
    ExecutionRoleArn="arn:aws:iam::913524936566:role/SagemakerRoleAllowS3"
)

print(f"Model created with name: {model_name}")

```

Fri 04/25 11:24:32 TINFO Tenoring unnecessary instance type: None.

FIGURE 4.15 – Creation du modèle à partir de notre artifact d'entraînement

```

[26]:
import boto3

sagemaker_client = boto3.client("sagemaker", region_name="eu-west-3")

# Get the model name after training
model_name = "xgboost-model-for-batch-transform"

# Define job parameters
transform_job_name = "titanicdata-batch-transform-job"
input_data_location = "s3://test-bucket-hamady/splitData/X_test_no_header.csv"
output_data_location = "s3://test-bucket-hamady/inference-output/"
# Create the transform job
response = sagemaker_client.create_transform_job(
    TransformJobName=transform_job_name,
    ModelName=model_name,
    TransformInput={
        "DataSource": {
            "S3DataSource": {
                "S3DataType": "S3Prefix",
                "S3Uri": input_data_location,
            }
        },
        "ContentType": "text/csv",
    },
    TransformOutput={}
)

```

Fri 04/25 11:24:32 TINFO Model created with name: xgboost-model-for-batch-transform

FIGURE 4.16 – Initiation d'un batch transform job pour les prédictions en batch

```

1020 |     "Code"
1021 |
1022 |
1023 |     error_class = self.exceptions.from_code(error_code)
1024 |     raise error_class(parsed_response, operation_name)
1025 |
1026

ResourceInUse: An error occurred (ResourceInUse) when calling the CreateTransformJob operation: Job name within an AWS account and region, and a job with this name already exists
(arn:aws:sagemaker:eu-west-3:913524936566:transform-job/titanicdata-batch-transform-job)

[27]: response = sagemaker_client.describe_transform_job(TransformJobName=transform_job_name)
job_status = response["TransformJobStatus"]
print(f"Transform Job Status: {job_status}")

if "FailureReason" in response:
    print(f"Failure Reason: {response['FailureReason']}")

Transform Job Status: Completed

[28]: #--- Paramètres S3 et SageMaker ---
bucket_name = "test-bucket-hamady"

```

FIGURE 4.17 – Création de transform job réussite

Objets (1) Info										
<input type="button" value="Copier l'URI S3"/> <input type="button" value="Copier l'URL"/> <input type="button" value="Télécharger"/> <input type="button" value="Ouvrir"/> <input type="button" value="Supprimer"/> <input type="button" value="Actions"/> <input type="button" value="Créer un dossier"/> <input type="button" value="Charger"/>										
Les objets sont les entités fondamentales stockées dans Amazon S3. Vous pouvez utiliser l'inventaire Amazon S3 pour obtenir une liste de tous les objets de votre compartiment. Pour que d'autres personnes puissent accéder à vos objets, vous devez leur accorder explicitement des autorisations. En savoir plus										
<input type="text" value="Rechercher des objets en fonction du préfixe"/> <input type="checkbox" value="Afficher les versions"/>										
<table border="1"> <thead> <tr> <th>Nom</th> <th>Type</th> <th>Dernière modification</th> <th>Taille</th> <th>Classe de stockage</th> </tr> </thead> <tbody> <tr> <td>X_test_no_header.csv.out</td> <td>out</td> <td>04 Jan 2025 10:45:30 AM +01</td> <td>3.4 Ko</td> <td>Standard</td> </tr> </tbody> </table>	Nom	Type	Dernière modification	Taille	Classe de stockage	X_test_no_header.csv.out	out	04 Jan 2025 10:45:30 AM +01	3.4 Ko	Standard
Nom	Type	Dernière modification	Taille	Classe de stockage						
X_test_no_header.csv.out	out	04 Jan 2025 10:45:30 AM +01	3.4 Ko	Standard						

FIGURE 4.18 – Résultats d'inférence enregistrés

4.5. Évaluation des performances avec l'accuracy

xxxix

FIGURE 4.19 – Journals pour l'événement de batch transform ajoutés à cloudWatch

4.5 Évaluation des performances avec l'accuracy

```

prediction_files = [content['Key'] for content in response.get('Contents', [])]

# Télécharger et charger les fichiers des prédictions
predictions = []
for file_key in prediction_files:
    obj = s3.get_object(Bucket=bucket_name, Key=file_key)
    batch_predictions = pd.read_csv(obj['Body'], header=None)
    predictions.append(batch_predictions)

# Fusionner les prédictions
predictions_df = pd.concat(predictions, axis=0)

# --- Étape 2 : Charger les vraies étiquettes ---
obj = s3.get_object(Bucket=bucket_name, Key=y_test_key)
y_test = pd.read_csv(obj['Body'], header=None) # Aucun en-tête

# il faut ignorer la première ligne (index 0)
y_test = y_test[1:].reset_index(drop=True)

# --- Étape 3 : Convertir les prédictions continues en classes binaires ---
# Appliquer un seuil de 0.5 pour transformer les prédictions en classes binaires
y_pred = (predictions_df[0] > 0.5).astype(int) # Si > 0.5, prédiction = 1, sinon = 0

# Convertir y_test en int_
y_test = y_test[0].astype(int)

# --- Étape 4 : Calculer l'accuracy ---
accuracy = accuracy_score(y_test, y_pred)

# --- Afficher les résultats ---
print(f"Accuracy: {accuracy:.2f}")

```

FIGURE 4.20 – Évaluation des performances avec l'accuracy

Pour la partie prédiction on a opté pour un workflow d’inférence par lot en utilisant un modèle **XGBoost** avec SageMaker. Les étapes sont résumées ci-dessous :

1. **Chargement des données d’entrée** : Les données de test sont téléchargées depuis un bucket S3, pré-traitées pour retirer l’en-tête.
2. **Création du modèle SageMaker** : Un modèle **XGBoost** est configuré à partir des artefacts générés lors de l’entraînement. L’image SageMaker correspondante est utilisée avec l’URI des artefacts et le rôle IAM nécessaire.
3. **Lancement du job d’inférence par lot** : Un job **Batch Transform** est initié en utilisant les données de test pré-traitées comme entrée et en spécifiant l’emplacement des résultats sur S3. L’inférence est exécutée sur une instance SageMaker **ml.m5.xlarge** avec une stratégie de traitement **MultiRecord**.
4. **Vérification du statut du job** : Le statut du job d’inférence est surveillé pour identifier son succès ou d’éventuelles erreurs.
5. **Récupération et traitement des résultats** : Pour calculer l’accuracy , Les fichiers de prédiction générés sont récupérés depuis S3 et fusionnés en un **DataFrame**. Les vraies étiquettes (**y_test**) sont également récupérées depuis S3.
6. **Post-traitement des prédictions** : Les scores continus sont transformés en classes binaires avec un seuil de 0.5. Les vraies étiquettes et les prédictions sont converties en entiers pour le calcul de la métrique.
7. **Évaluation des performances** : L’accuracy est calculée en comparant les prédictions et les vraies étiquettes, avec un résultat final de 83 pourcent.

Ce workflow illustre l’utilisation efficace de SageMaker pour exécuter un processus d’inférence par lot, combinant le traitement des données, la gestion des modèles, et l’évaluation des performances de manière automatisée et évolutive.

Conclusion

Ce projet a permis de mettre en œuvre un workflow complet de machine learning sur AWS, de la préparation des données au déploiement du modèle. L'utilisation de services comme S3, Lambda et SageMaker a démontré leur capacité à simplifier des tâches complexes, tout en garantissant une scalabilité et une efficacité remarquables.

Les résultats obtenus, notamment l'accuracy du modèle XGBoost, ont montré la pertinence de l'approche adoptée. Ce projet a également offert une expérience pratique avec le cloud AWS, renforçant les compétences dans l'utilisation de ses outils pour des cas d'usage de data science.

En conclusion, cette expérience souligne la valeur ajoutée d'un écosystème cloud intégré pour développer des solutions de machine learning, tout en ouvrant des perspectives pour des projets futurs plus complexes et collaboratifs

