AWS Academy Cloud Security Foundations

Lab 3.1 Report: Using Resource -Based Policies to Secure an S3 Bucket

TAKI Oussama

3ACI Info

1. Présentation du laboratoire et objectifs

L'objectif principal de ce laboratoire était de comprendre la différence entre les politiques basées sur l'identité (IAM identity-based policies) et les politiques basées sur les ressources, notamment celles appliquées aux buckets S3. Nous avons également examiné la manière dont ces politiques interagissent pour fournir un contrôle d'accès granulaire.

Objectifs clés :

- Comprendre la distinction entre politiques basées sur l'identité et basées sur les ressources.
- Démontrer comment un utilisateur IAM peut assumer un rôle (assume role) afin de modifier ou d'étendre ses permissions.
- Analyser l'impact des différentes politiques sur la visibilité et l'exécution des actions dans la console AWS.

Architecture :

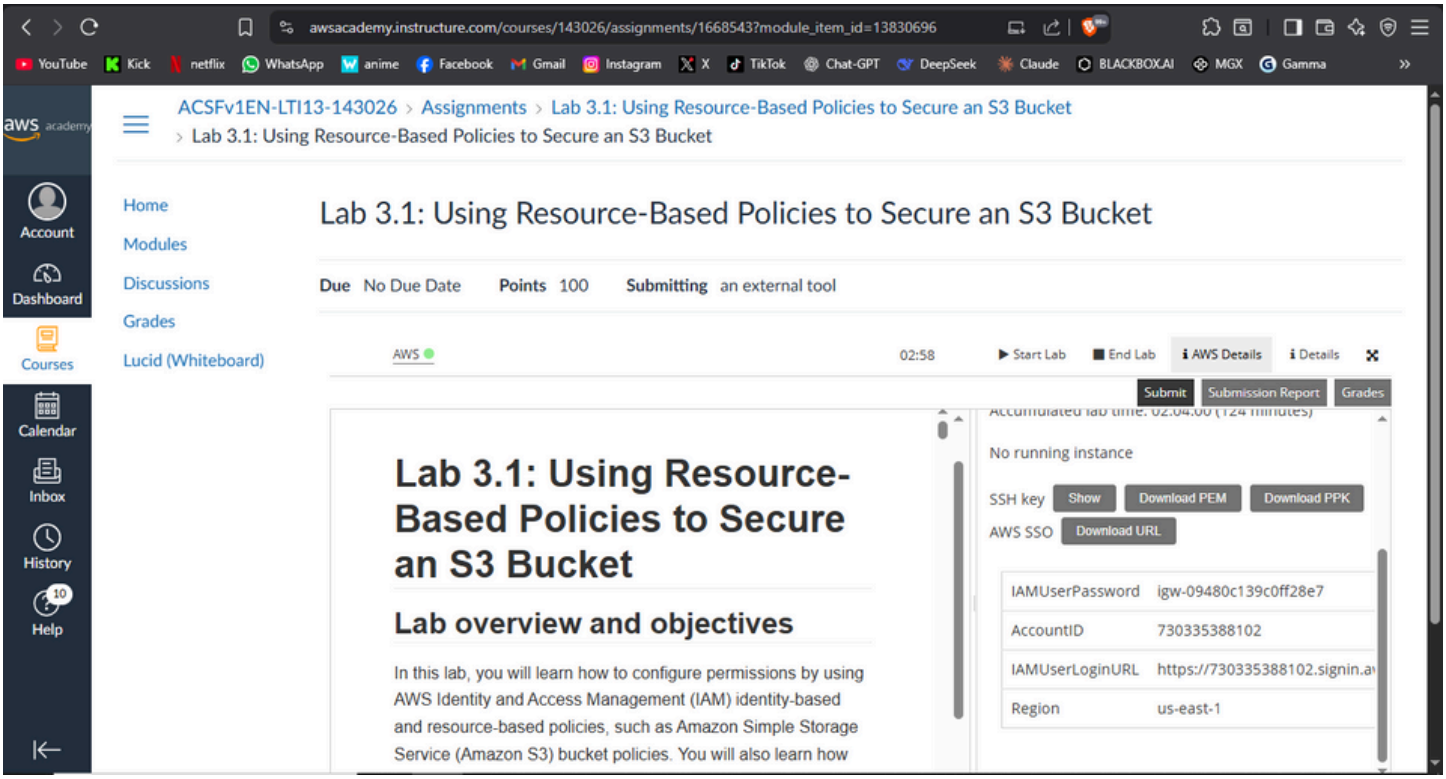
- **Architecture initiale :**
  Le laboratoire débute avec un utilisateur IAM nommé *devuser*, appartenant au groupe *DeveloperGroup*.

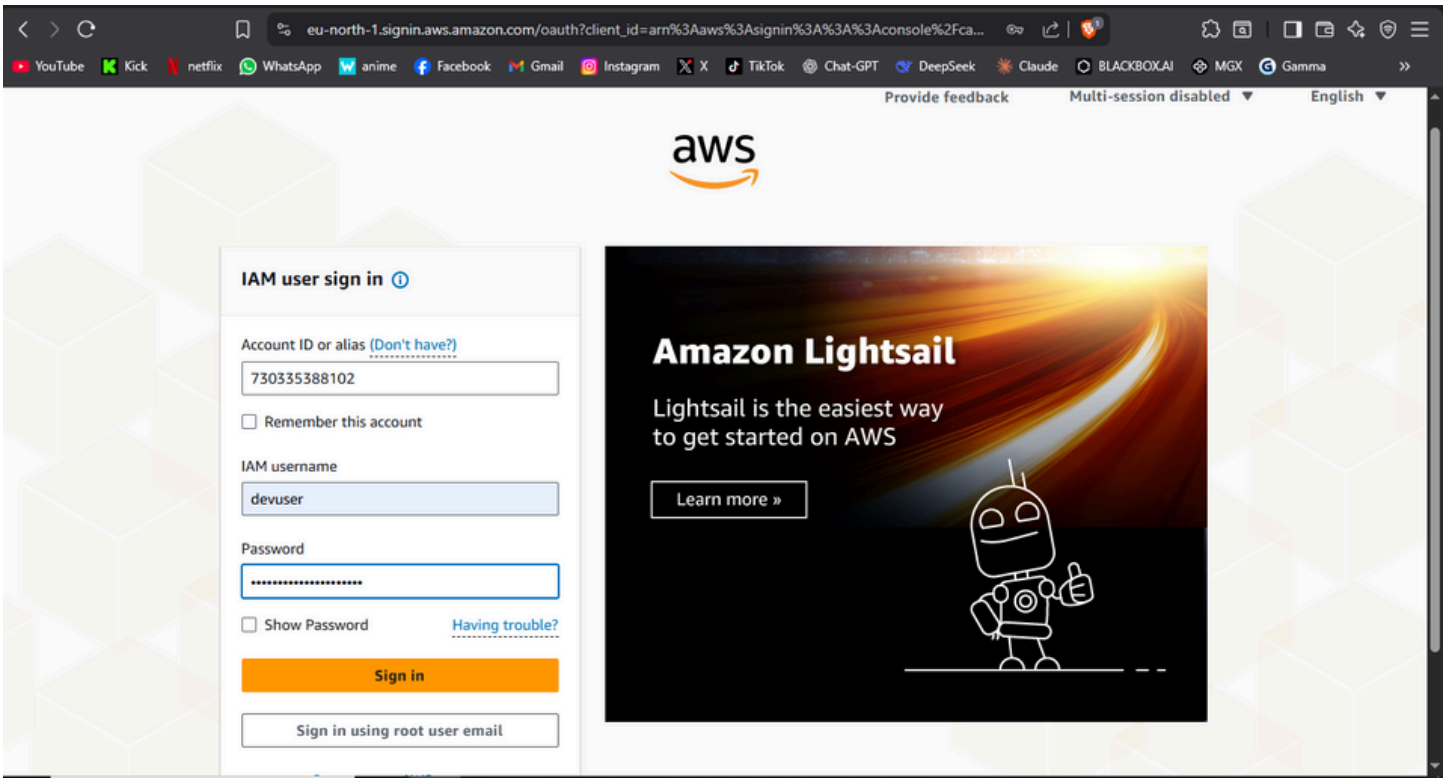
- **Architecture cible :**
  À la fin de l'exercice, nous utilisons des rôles IAM spécifiques (*BucketAccessRole* et *OtherBucketAccessRole*) permettant d'interagir avec des buckets S3 soumis à des restrictions d'accès.

2. Description détaillée des tâches

Tâche 1 : Accès à la console

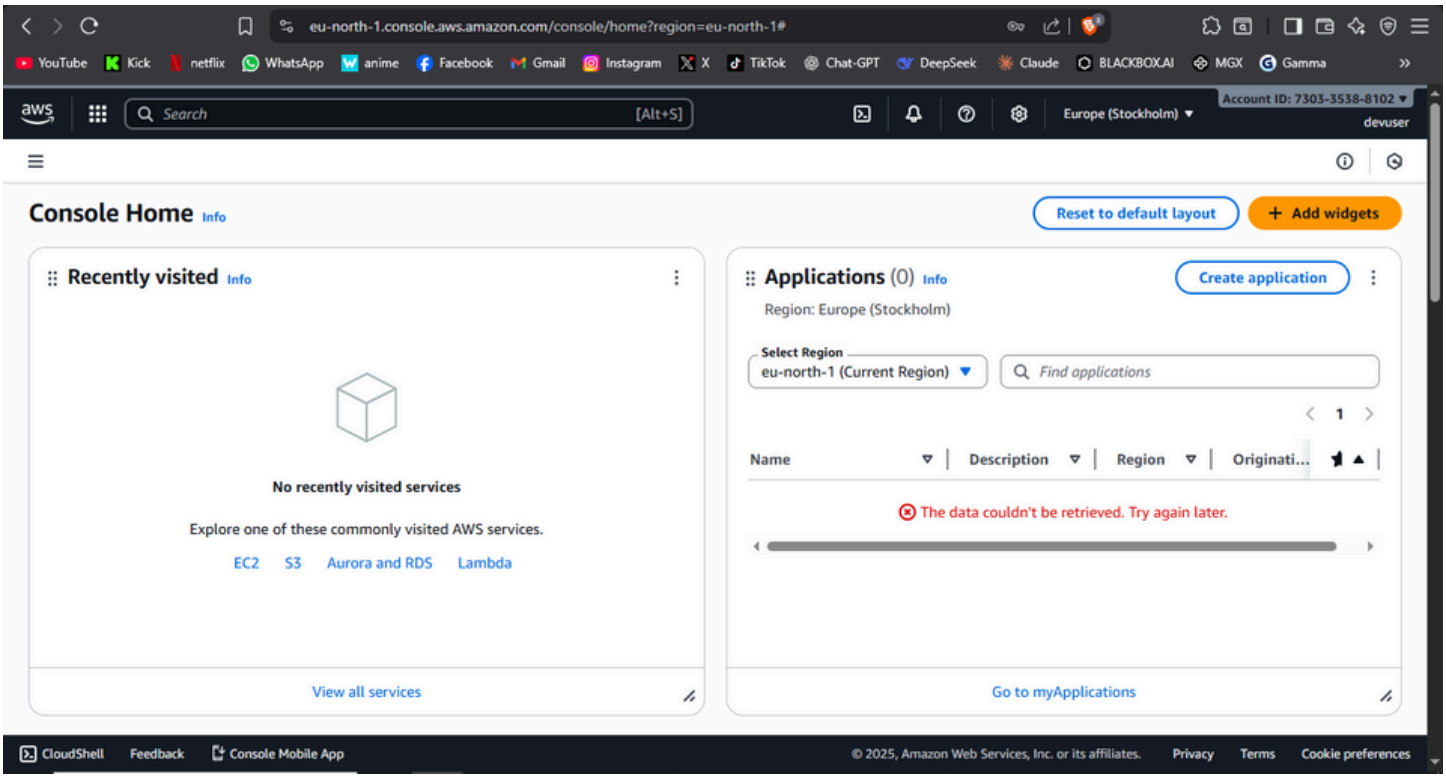
- **Action :** Connexion à la console de gestion AWS en utilisant les identifiants fournis pour *devuser*.





Observation :

La connexion s'est effectuée avec succès, nous amenant directement sur la page d'accueil de la console AWS.

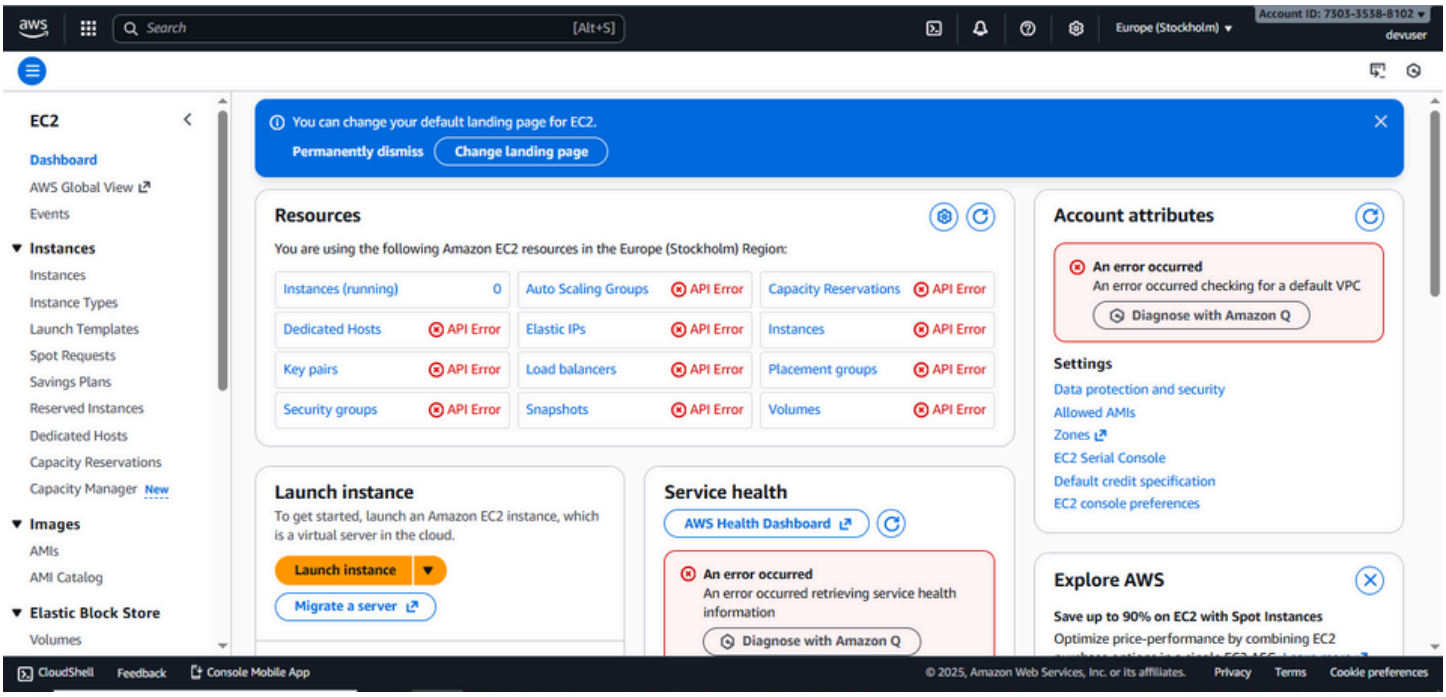


Tâche 2 : Tentative d'accès en lecture aux services AWS

Action 1 : Accéder à Amazon EC2

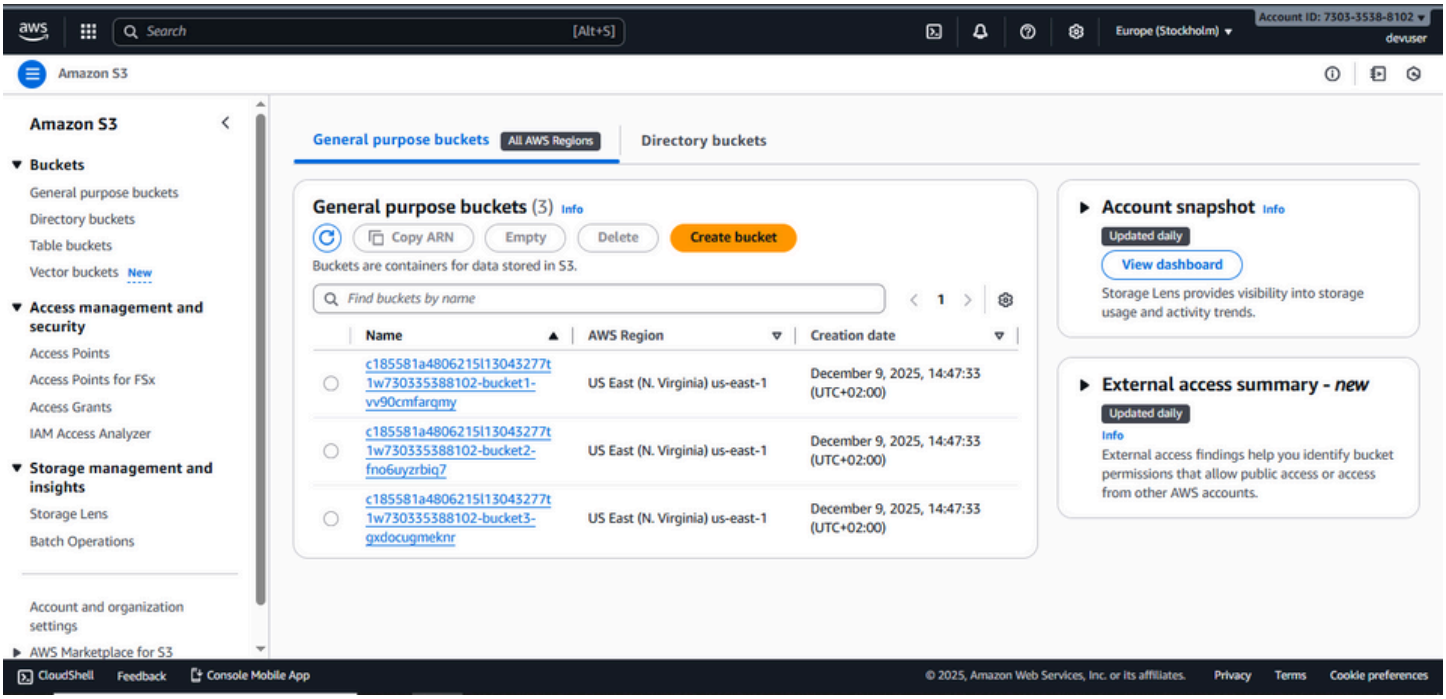
- **Étape :** Navigation vers le tableau de bord EC2 pour consulter les instances.

- **Observation :** Plusieurs messages d'« API Error » sont apparus. Il était impossible d'afficher la liste des instances, de consulter les types d'instances ou d'accéder à toute ressource EC2
- **Raison :**
  L'utilisateur *devuser* ne possède pas les permissions nécessaires — comme *ec2:DescribeInstances* — dans sa politique IAM, ce qui empêche l'accès aux ressources EC2.



Action 2 : Accès à Amazon S3

- **Étape :** Navigation vers la console Amazon S3.

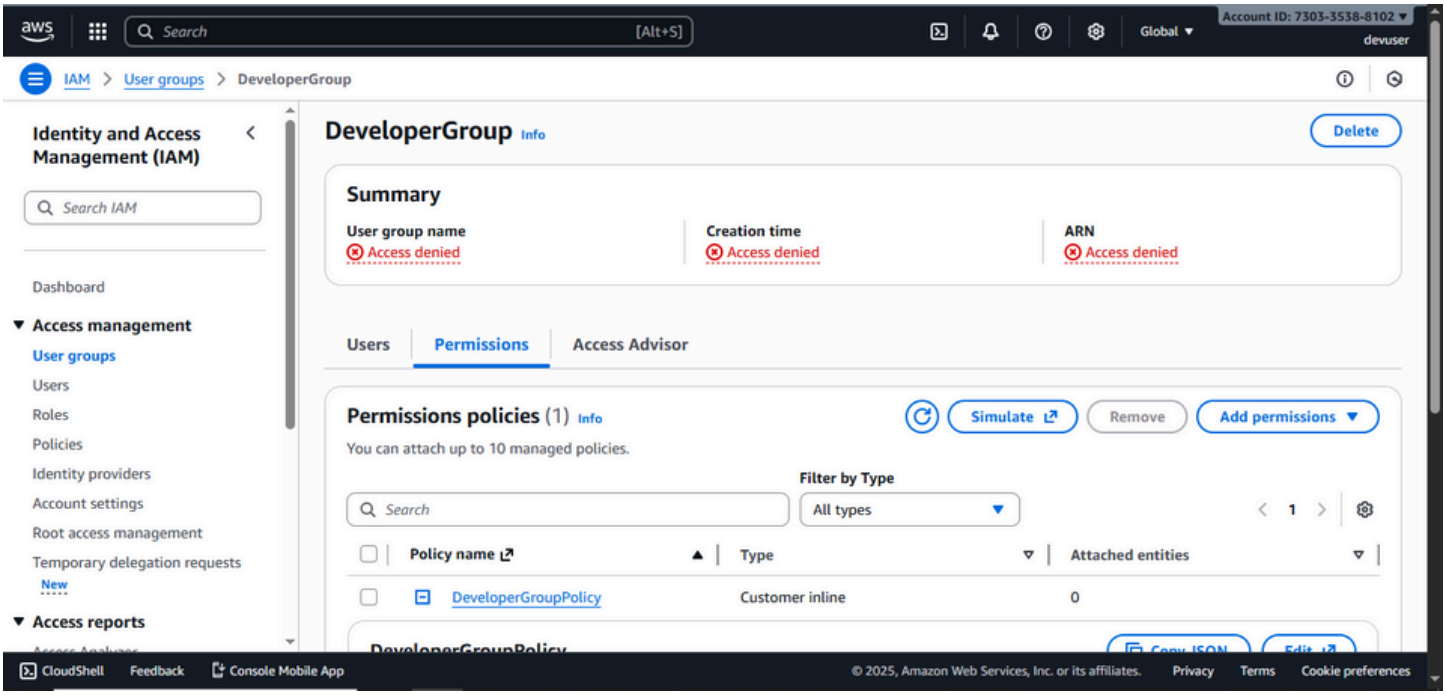


- **Observation :** La liste des buckets (*bucket1*, *bucket2*, *bucket3*) était visible, mais la colonne « Access » affichait « Insufficient permissions » ou « Error ».
- **Raison :** L'utilisateur *devuser* dispose de la permission *ListAllMyBuckets*, ce qui lui permet de voir les noms des buckets. Cependant, il ne possède pas les autorisations nécessaires pour consulter la configuration ou les paramètres d'accès public de chaque bucket.

Tâche 3 : Analyse de la politique basée sur l'identité

Action :

- Accès à la console IAM → *Users* → *devuser* → *Groups* → *DeveloperGroup* → *Permissions*.

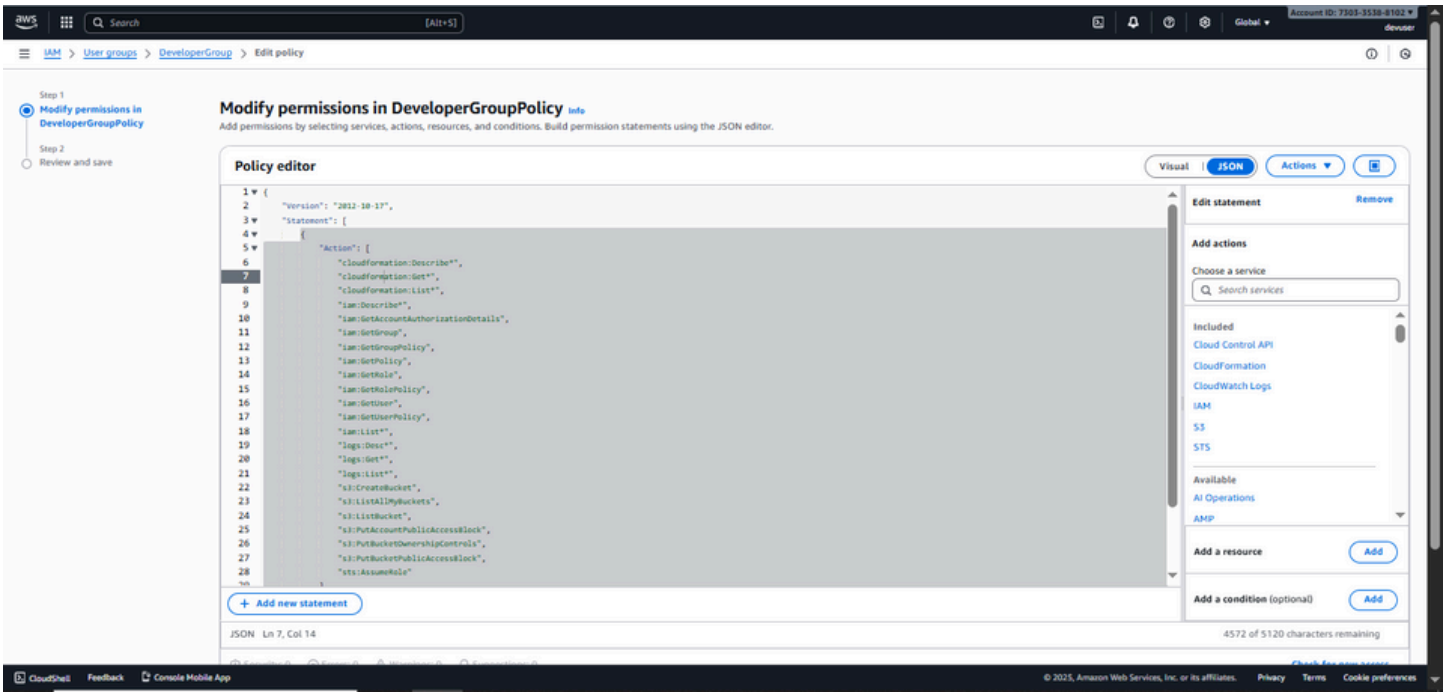


- Ouverture de la politique *DeveloperGroupPolicy* afin d'examiner le fichier JSON.

Analyse de la politique :
La politique associée au groupe est une **Identity-Based Policy** (politique basée sur l'identité).

- **Actions autorisées :** *s3:ListAllMyBuckets*, *s3:CreateBucket*, *s3:ListBucket.*
- **Actions manquantes ou non autorisées :** Absence d'autorisations *ec2:* (ce qui explique les erreurs lors de l'accès à EC2) et absence de permission *s3:PutObject* au niveau global.

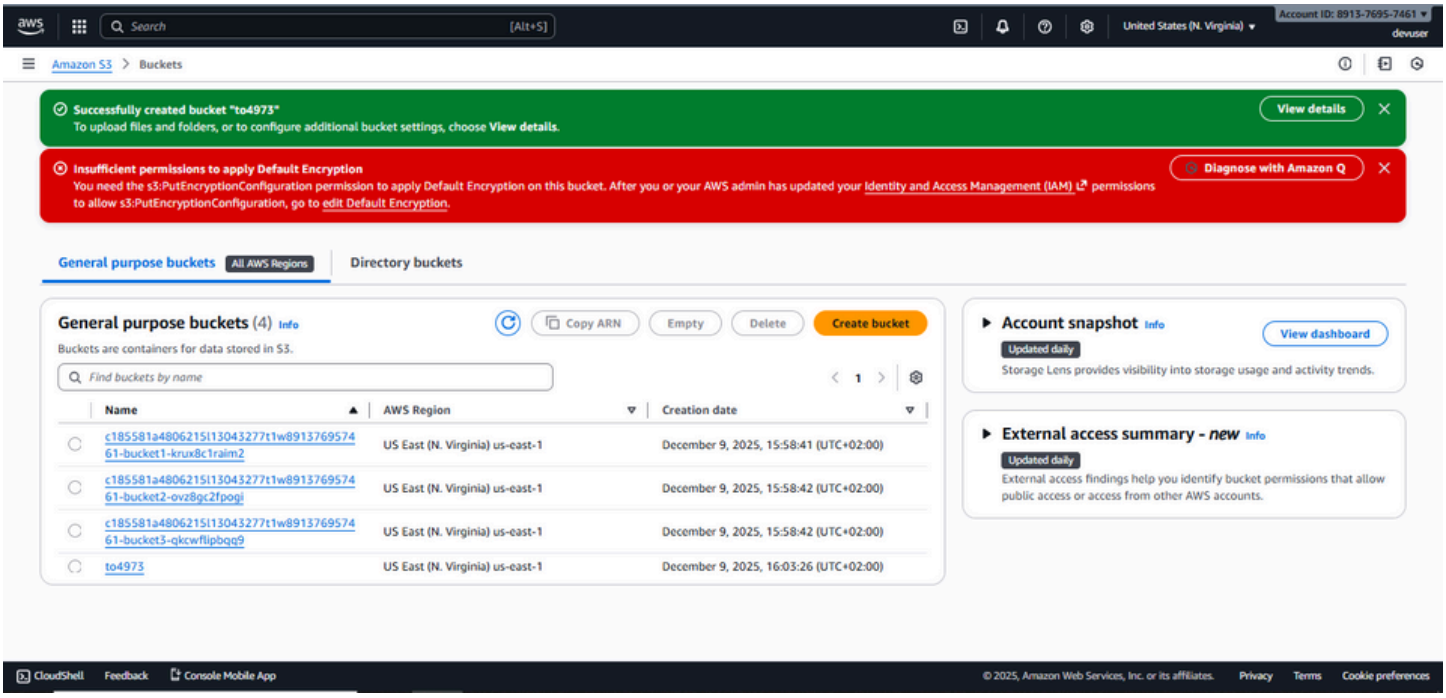
Cela explique pourquoi nous pouvions voir les noms des buckets (permission de liste), mais pas exécuter d'autres actions administratives.



Tâche 4 : Tentative d’accès en écriture aux services AWS

Action 1 : Création d’un bucket

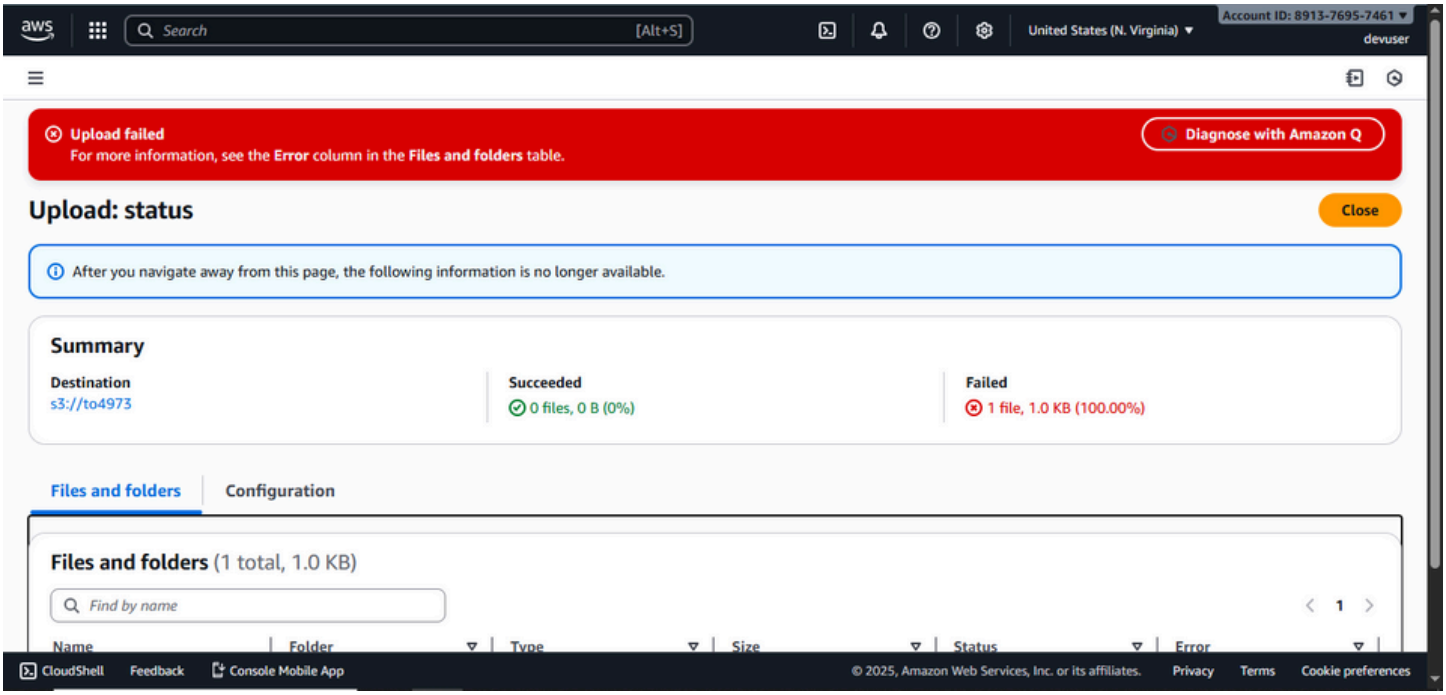
- **Étape :** Clic sur « Create bucket » puis création d’un bucket nommé *to4973*.



- **Résultat :** Succès.
- Raison : La politique *DeveloperGroupPolicy* autorise explicitement l’action *s3:CreateBucket*.

Action 2 : Téléversement d'un objet

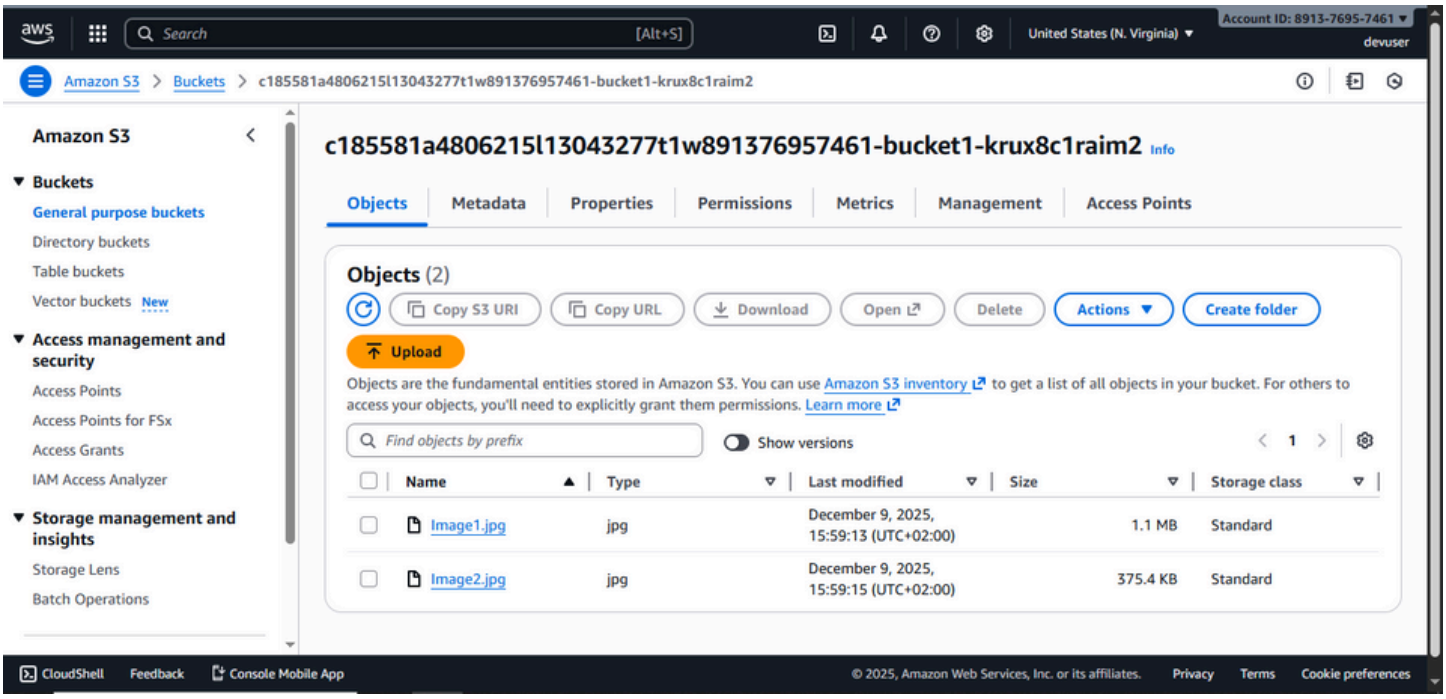
- **Étape :** Ouverture du bucket nouvellement créé, puis tentative de téléverser *Image1.jpg*.
- **Résultat :** Échec (message *Upload failed*).
- **Raison :** Bien que nous soyons propriétaires du bucket que nous venons de créer, la politique IAM associée à *devuser* ne contient pas l'autorisation *s3:PutObject*. Dans AWS, la permission de créer une ressource n'implique pas automatiquement la permission de modifier son contenu si la politique reste restrictive.

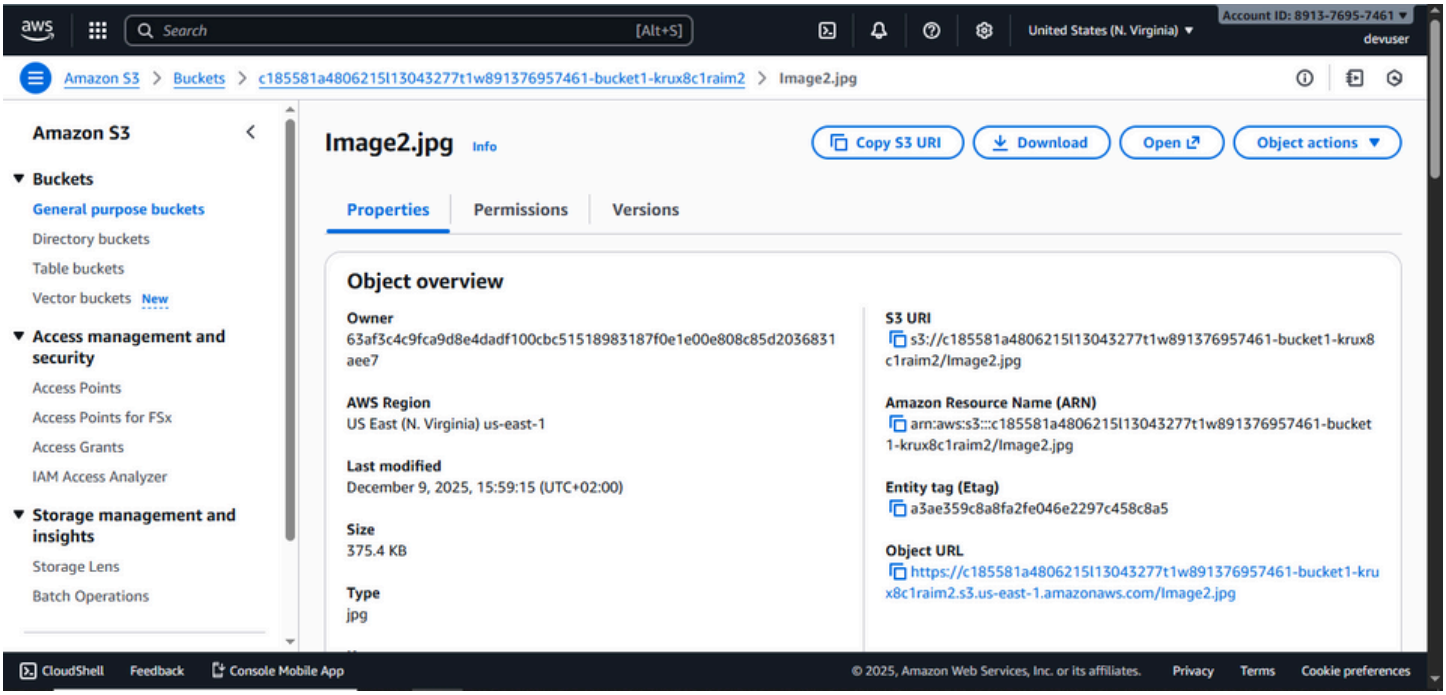


Tâche 5 : Assumer un rôle IAM

Test préalable : tentative de téléchargement avant d'assumer le rôle

- **Action :** Navigation vers *bucket1* puis tentative de téléchargement d'un fichier existant.
- **Résultat :** Échec (erreur *Access Denied*).



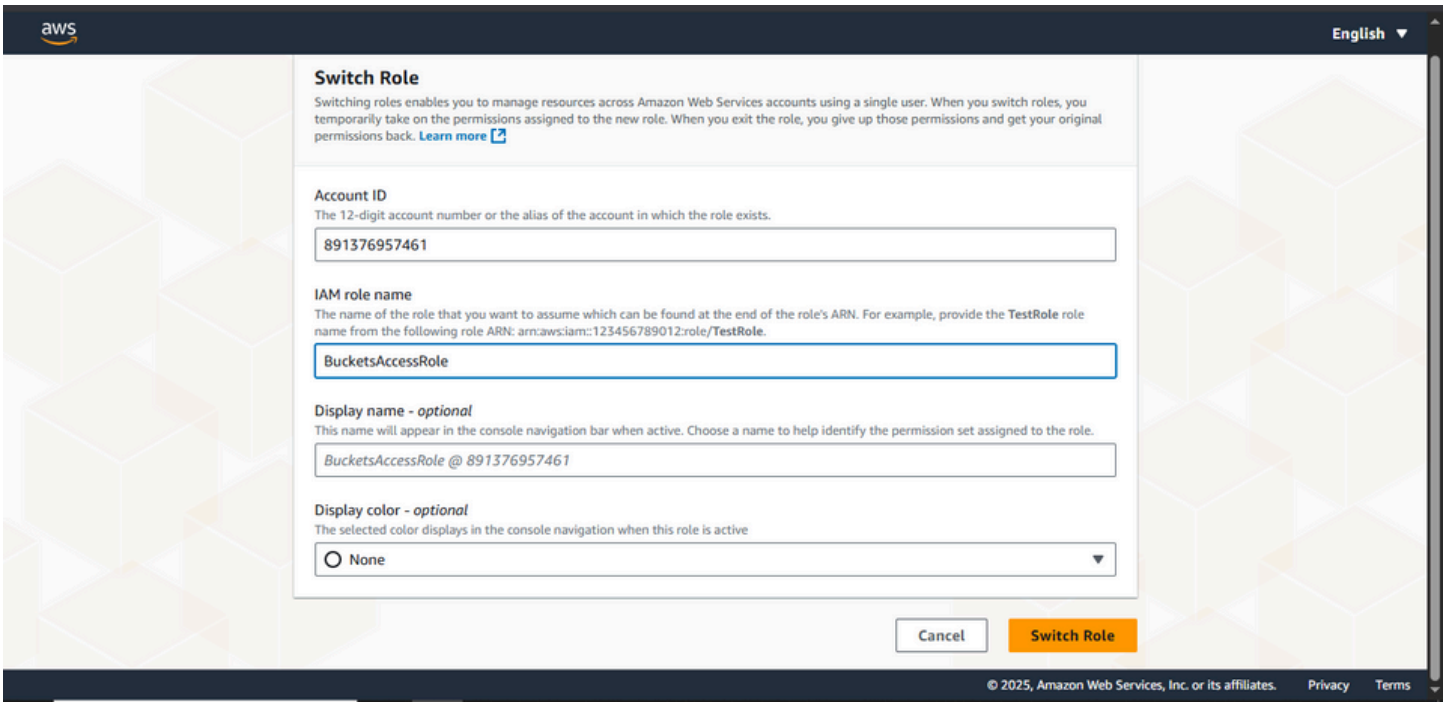


Raison :

La politique basée sur l'identité de *devuser* (*DeveloperGroupPolicy*) autorise *s3:ListAllMyBuckets* et *s3:ListBucket*, mais ne contient pas la permission *s3:GetObject*, indispensable pour télécharger un objet depuis un bucket. Cela confirme que les

permissions par défaut de l'utilisateur limitent l'accès en lecture au contenu des objets, même s'il peut voir la liste des buckets.

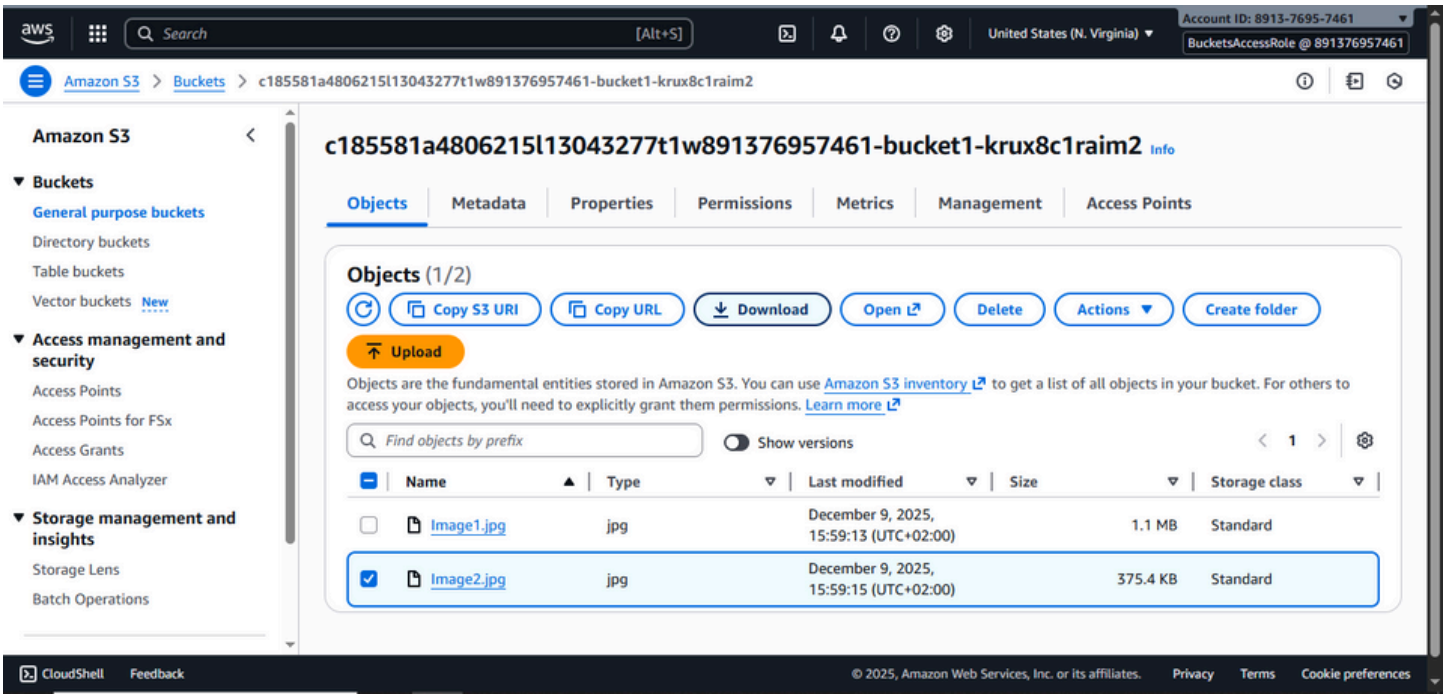
Action : (tu peux m'envoyer la suite pour la reformuler)

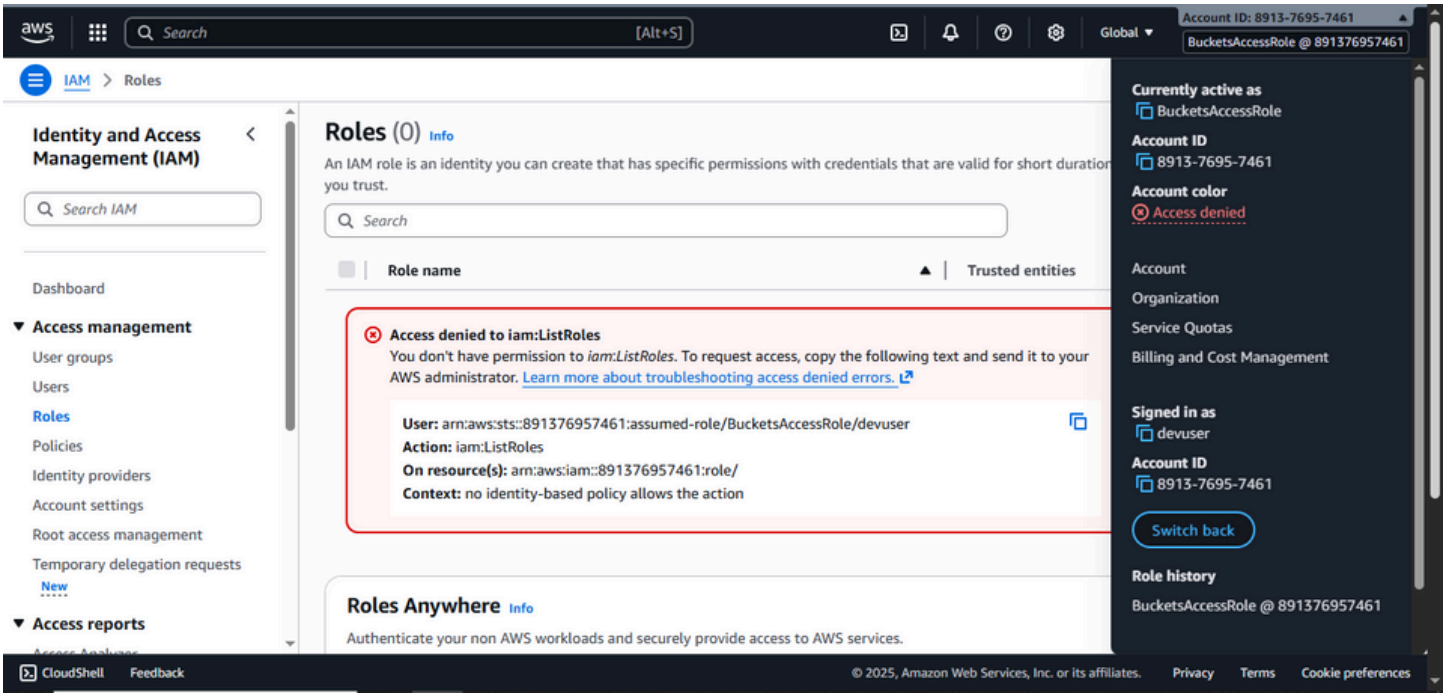


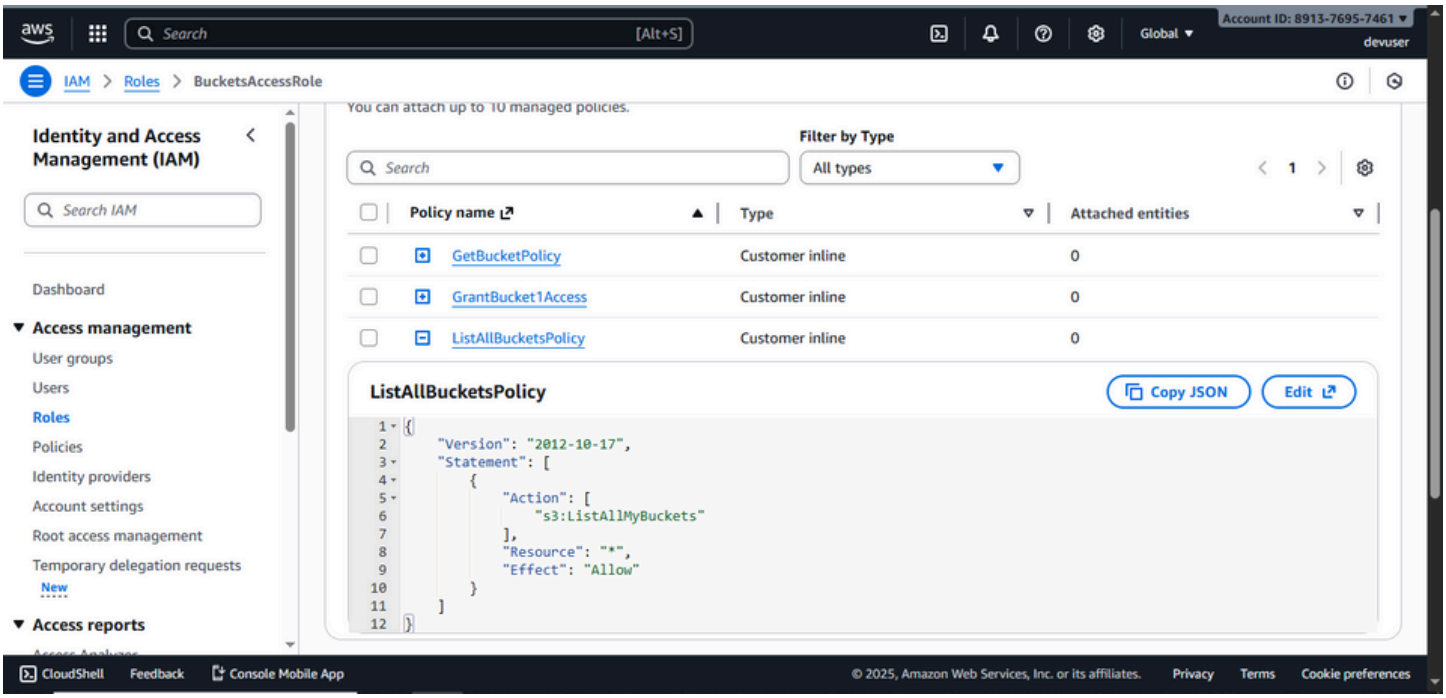
Action :

- Ouverture du menu utilisateur en haut à droite puis clic sur Switch Role.
- Saisie de l'ID du compte ainsi que du nom du rôle : BucketsAccessRole.
- Changement visible : L'identité affichée en haut à droite a été remplacée par *BucketsAccessRole*, confirmant que le rôle a bien été assumé.

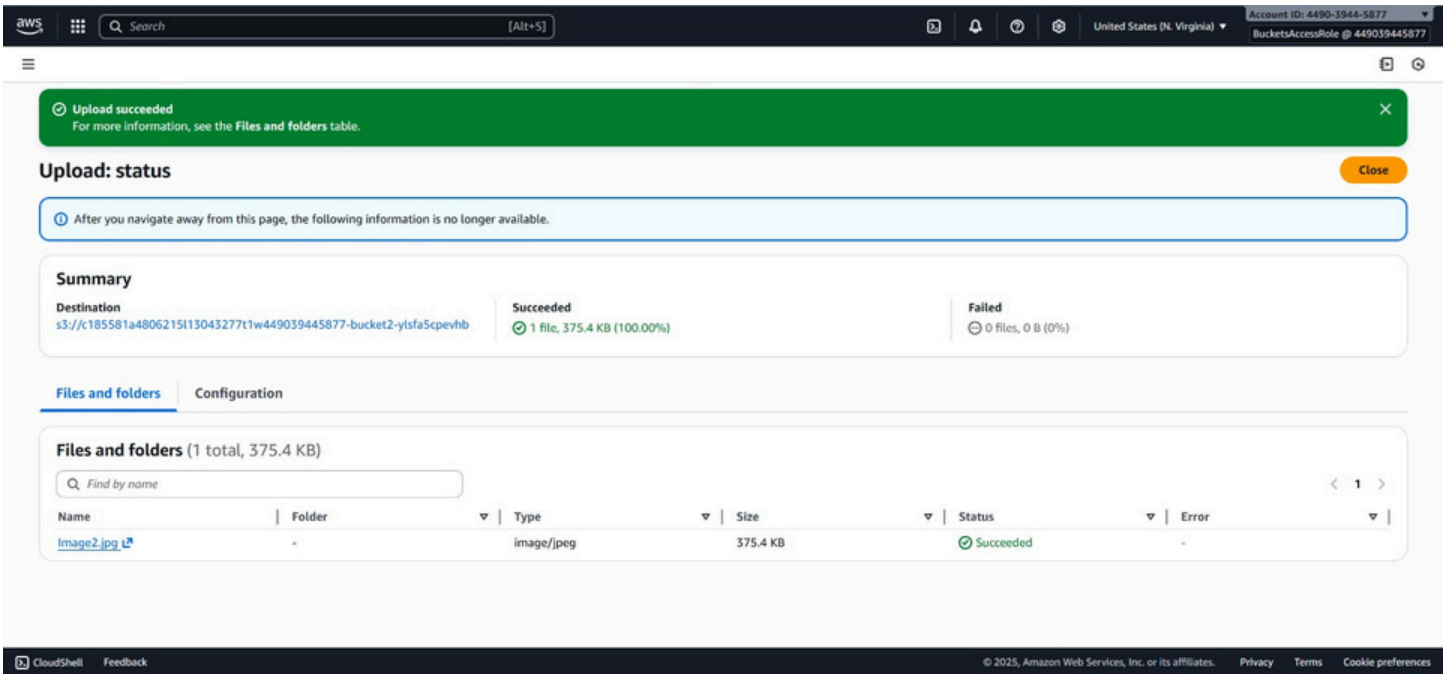
Test d'accès : (envoie-moi la suite et je te la reformule)


Search [Alt+S]
United States (N. Virginia)
Account ID: 8913-7695-7461
BucketsAccessRole @ 891376957461
Amazon S3 > Buckets > c185581a4806215l13043277t1w891376957461-bucket1-krux8c1raim2
Amazon S3
Buckets
General purpose buckets
Directory buckets
Table buckets
Vector buckets New
Access management and security
Access Points
Access Points for FSx
Access Grants
IAM Access Analyzer
Storage management and insights
Storage Lens
Batch Operations
c185581a4806215l13043277t1w891376957461-bucket1-krux8c1raim2 Info
Objects
Metadata
Properties
Permissions
Metrics
Management
Access Points
Objects (1/2)
Copy S3 URI
Copy URL
Download
Open
Delete
Actions
Create folder
Upload
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more
Find objects by prefix
Show versions
Name
Type
Last modified
Size
Storage class
Image1.jpg
jpg
December 9, 2025, 15:59:13 (UTC+02:00)
1.1 MB
Standard
Image2.jpg
jpg
December 9, 2025, 15:59:15 (UTC+02:00)
375.4 KB
Standard
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Search [Alt+S]
Global
Account ID: 8913-7695-7461
BucketsAccessRole @ 891376957461
IAM > Roles
Identity and Access Management (IAM)
Search IAM
Dashboard
Access management
User groups
Users
Roles
Policies
Identity providers
Account settings
Root access management
Temporary delegation requests New
Access reports
Roles (0) Info
An IAM role is an identity you can create that has specific permissions with credentials that are valid for short duration you trust.
Search
Role name
Trusted entities
Access denied to iam:ListRoles
You don't have permission to iam:ListRoles. To request access, copy the following text and send it to your AWS administrator. Learn more about troubleshooting access denied errors.
User: arn:aws:sts::891376957461:assumed-role/BucketsAccessRole/devuser
Action: iam:ListRoles
On resource(s): arn:aws:iam::891376957461:role/
Context: no identity-based policy allows the action
Roles Anywhere Info
Authenticate your non AWS workloads and securely provide access to AWS services.
Currently active as
BucketsAccessRole
Account ID
8913-7695-7461
Account color
Access denied
Account
Organization
Service Quotas
Billing and Cost Management
Signed in as
devuser
Account ID
8913-7695-7461
Switch back
Role history
BucketsAccessRole @ 891376957461
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences



1. **Bucket 1 :** Le téléchargement des objets a fonctionné sans problème.
2. **Bucket 2 :** L'upload d'objets était autorisé, car le rôle disposait de la permission *s3:PutObject* spécifiquement sur *bucket2*.



Analyse :

En assumant le rôle, nous avons temporairement abandonné les permissions de *devuser* pour adopter celles de BucketsAccessRole. Ce rôle possédait une politique dédiée autorisant la lecture et/ou l'écriture sur certains buckets, des permissions que *devuser* ne détenait pas initialement.

Voici la reformulation :

Tâche 6 : Comprendre les politiques basées sur les ressources

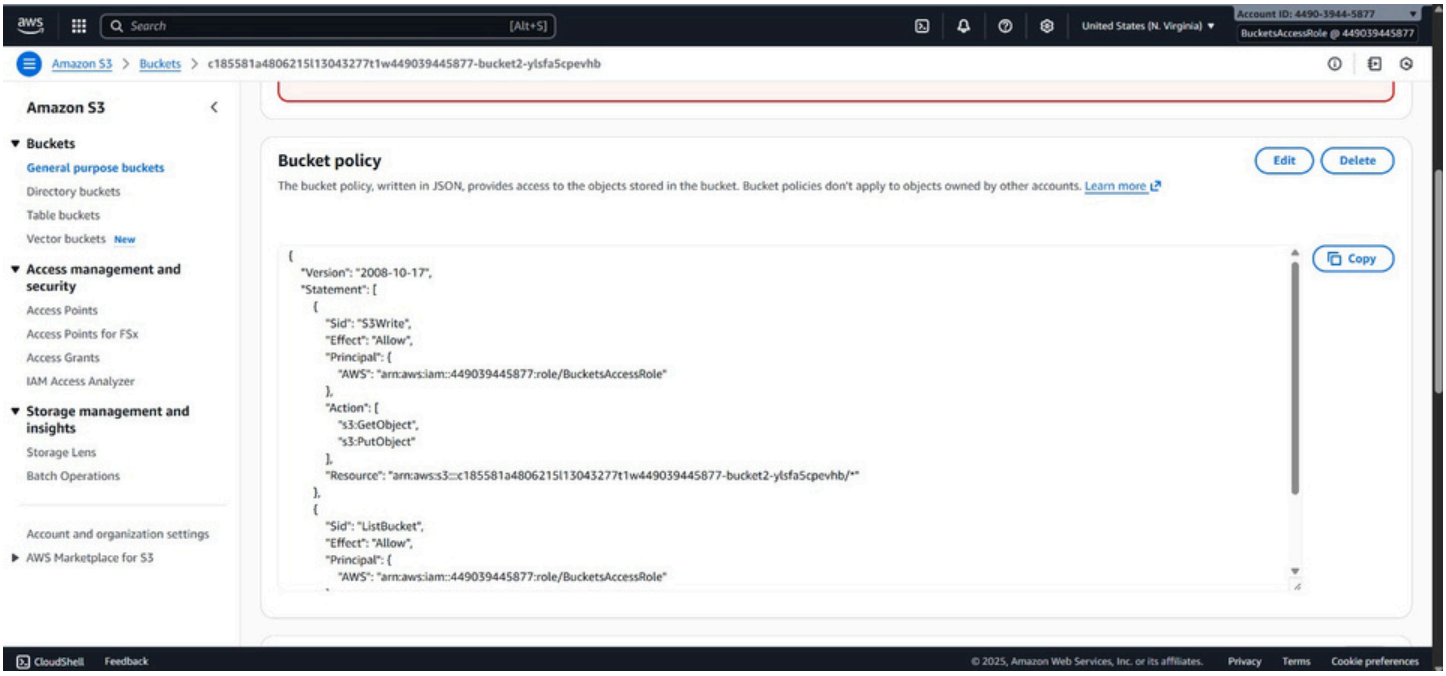
Action :

- En étant connecté avec le rôle *bucket-access-role*, nous avons navigué vers *Bucket 2* puis ouvert l'onglet **Permissions**.
- Nous avons consulté la **Bucket Policy**.

Analyse de la politique :
Il s'agit d'une **Resource-Based Policy**, c'est-à-dire une politique attachée directement à la ressource (le bucket S3), contrairement aux politiques basées sur l'identité qui sont liées à un utilisateur ou à un rôle IAM.

- La politique mentionnait explicitement *BucketsAccessRole* comme **Principal** autorisé.
- Cela confirme que l'accès à une ressource peut être accordé soit par la politique de l'utilisateur (Identity-Based Policy), soit par la politique de la ressource (Resource-Based Policy) lorsqu'il s'agit du même compte.
- Dans un contexte inter-comptes (cross-account), les deux types de politiques sont souvent nécessaires, mais dans ce laboratoire l'objectif principal était d'observer comment le rôle obtenait l'autorisation via la bucket policy.

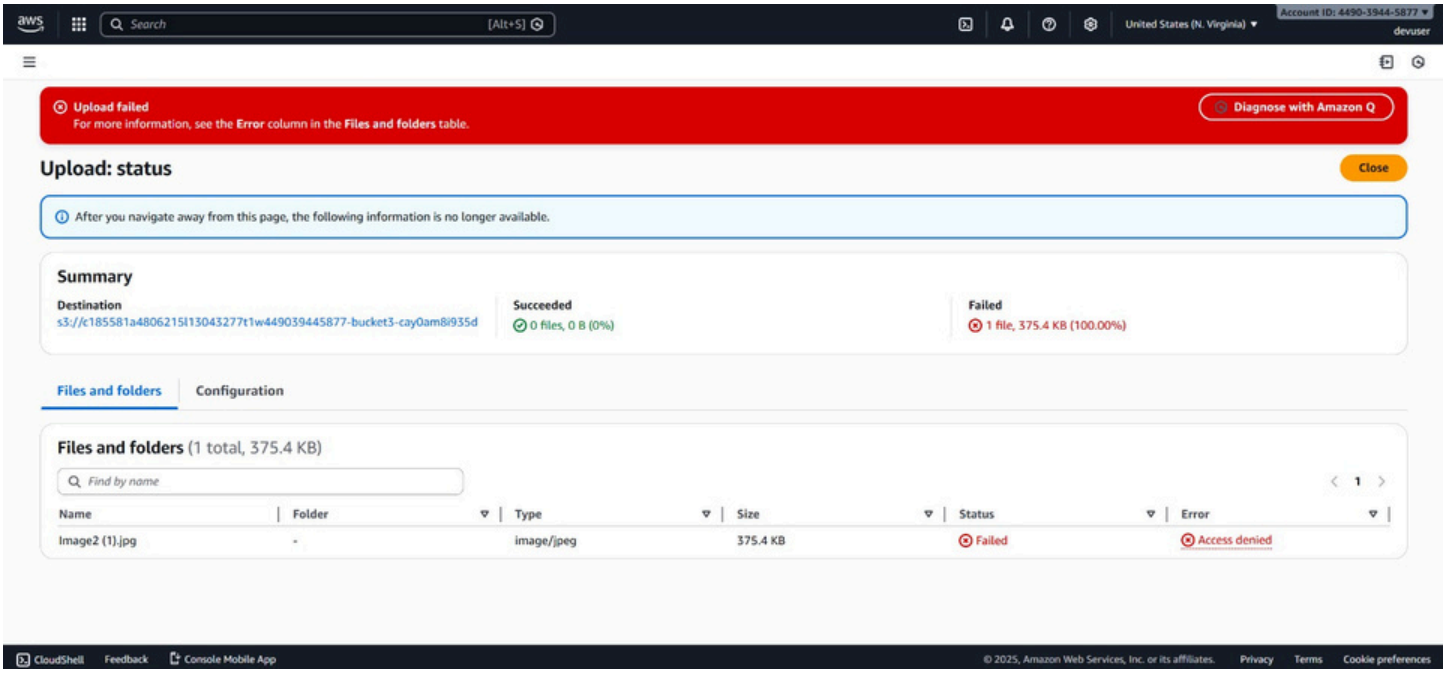
Si tu veux, envoie-moi la dernière partie (conclusion, résumé, etc.) et je la reformule aussi.

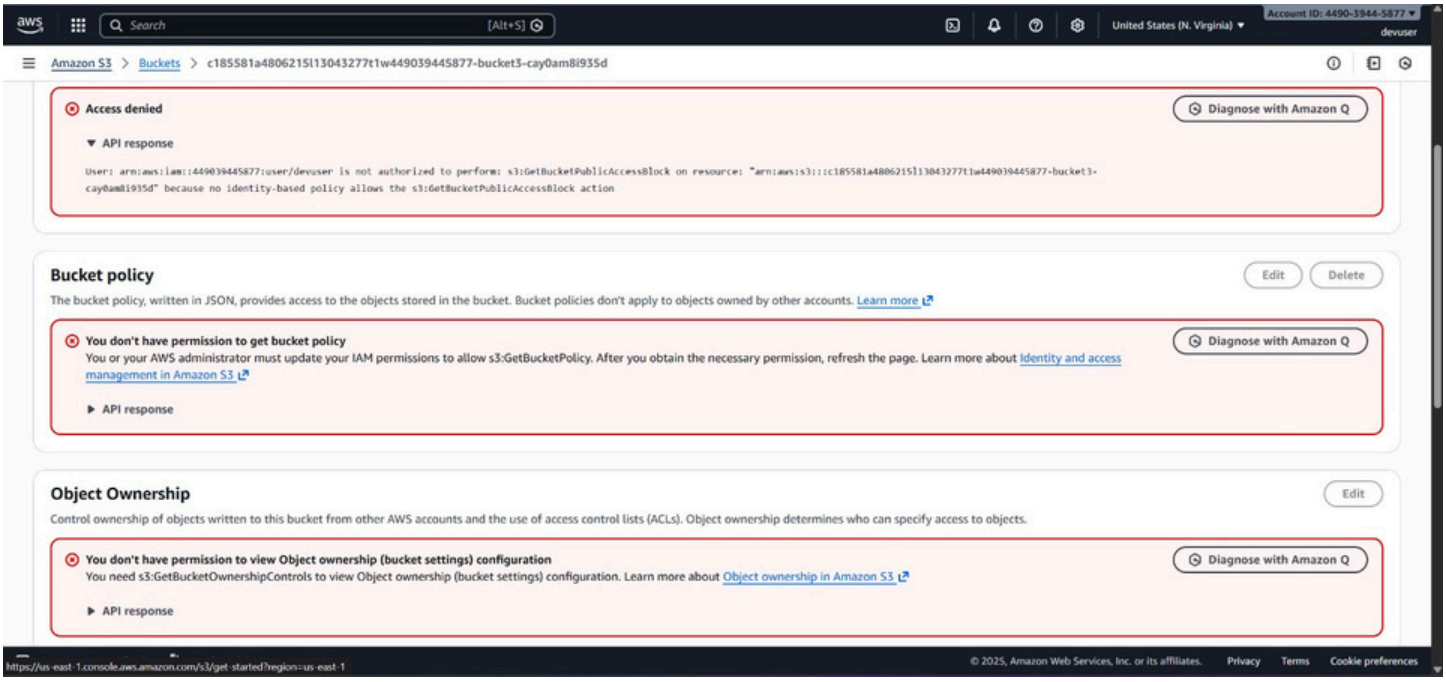


3. **Challenge Task:** : Téléversement dans Bucket 3

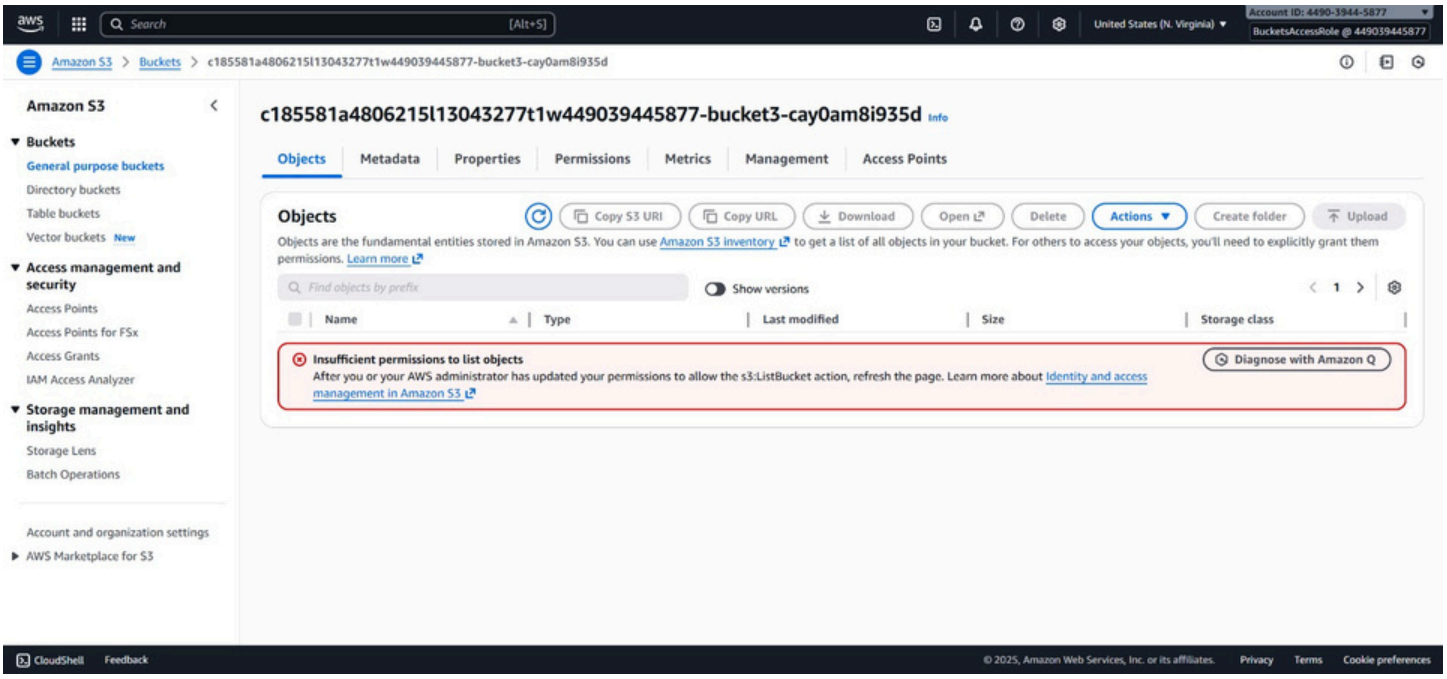
Objectif : Envoyer *Image2.jpg* dans *bucket3*.

Tentative 1 : En tant que *devuser*

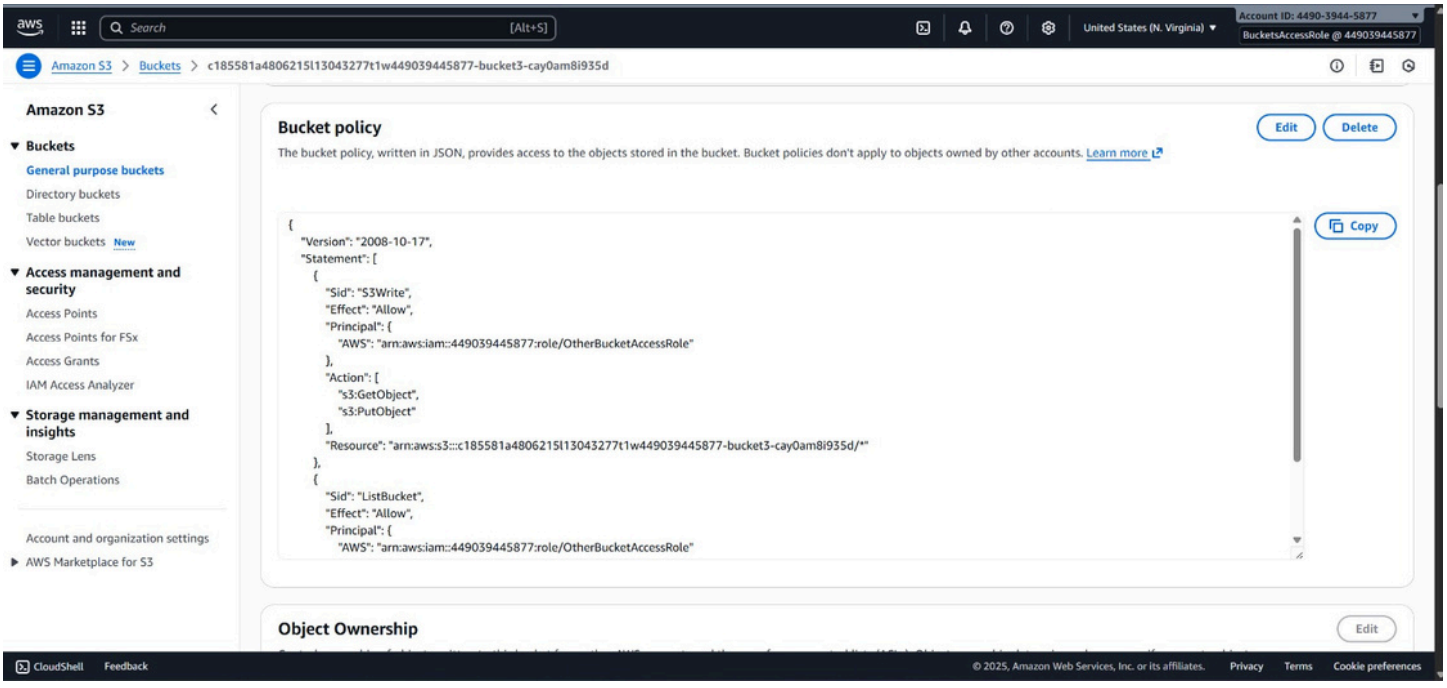




Tentative 2 : En utilisant le rôle *BucketAccessRole*

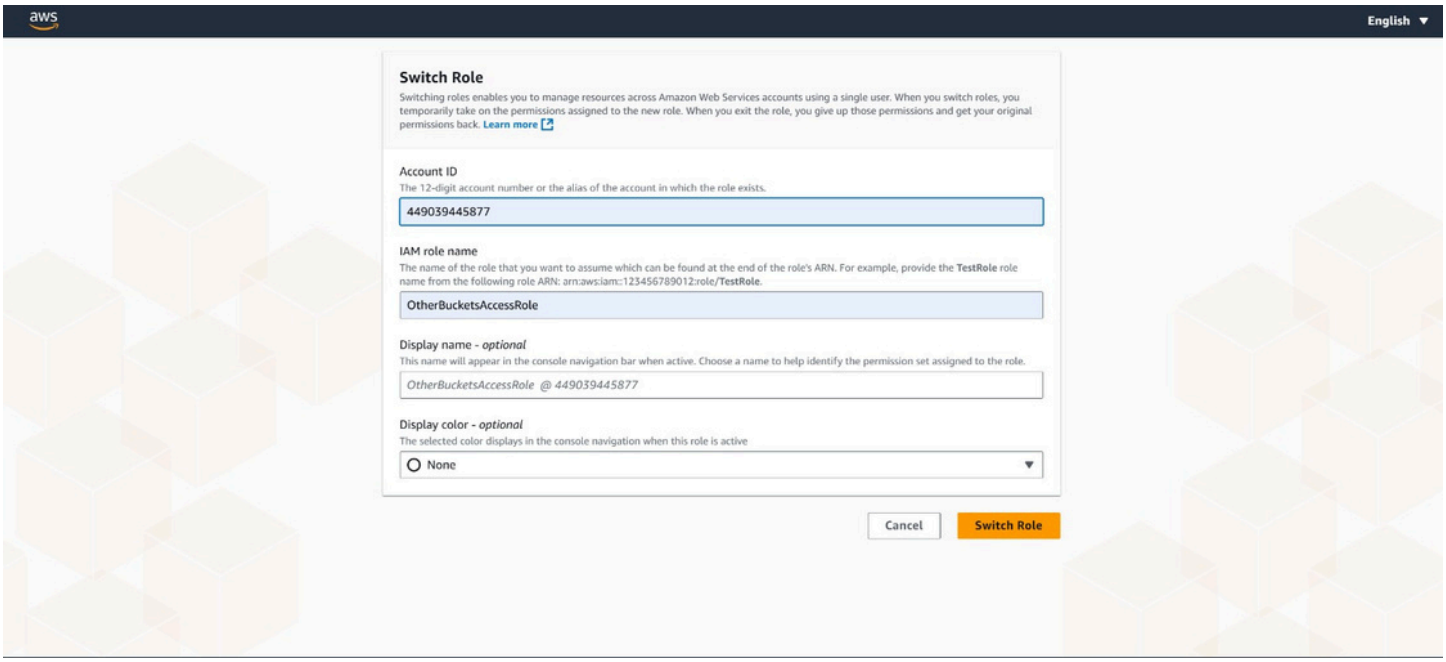


Résultat : Échec. La politique associée au rôle ne contenait aucune permission permettant d'interagir avec *bucket3*. Toutefois, nous pouvions consulter la bucket policy.



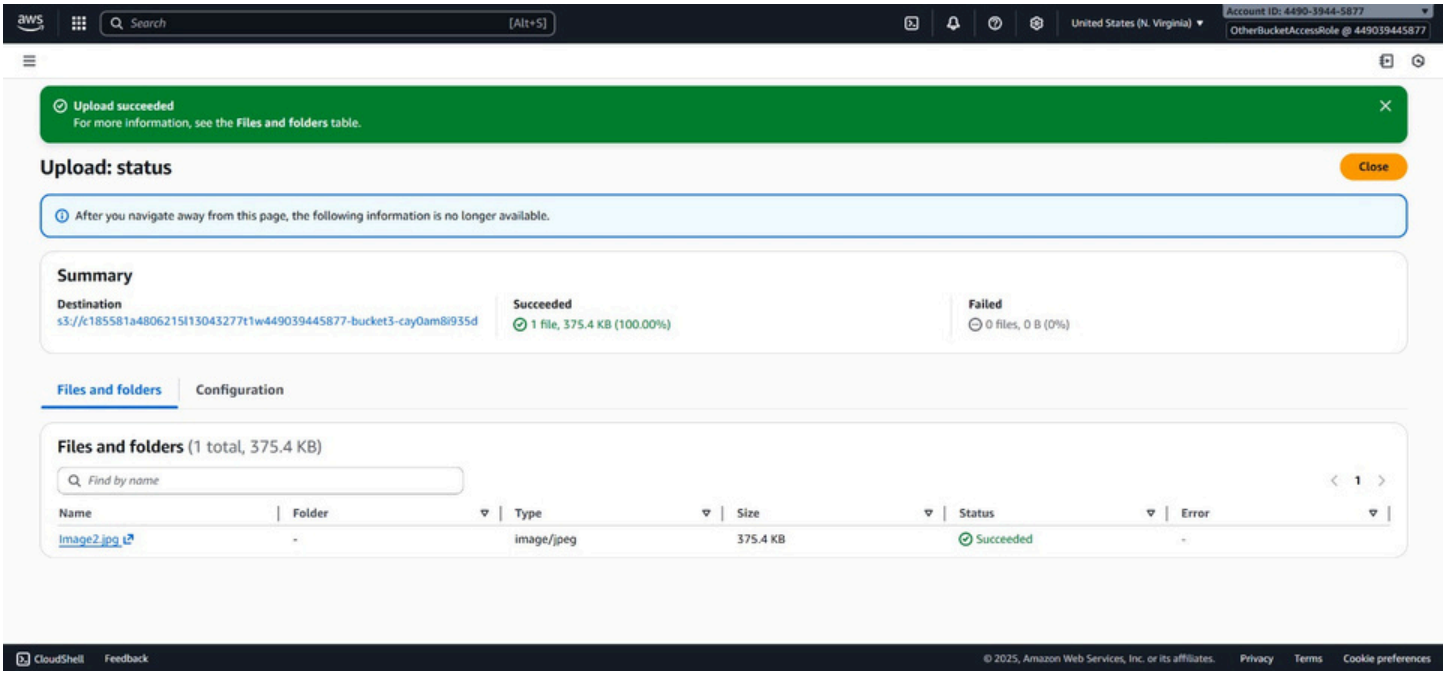
Solution :

1. Nous avons identifié un second rôle disponible dans le compte : **OtherBucketAccessRole**.
2. Nous avons de nouveau utilisé la fonctionnalité **Switch Role** pour passer du rôle *BucketsAccessRole* au rôle *OtherBucketAccessRole*.



1. Nous avons navigué vers *bucket3*.
2. Nous avons cliqué sur **Upload**, sélectionné *Image2.jpg*, puis validé l'envoi.
3. **Résultat :** Succès.

Explication :
Le rôle **OtherBucketAccessRole** possédait une politique autorisant explicitement l'action *s3:PutObject* sur la ressource *bucket3*. En assumant ce rôle spécifique, nous avons obtenu les permissions nécessaires pour réaliser la tâche.



Total score	15/15
TASK 4 - Create bucket	5/5
TASK 5 - Uploaded object	5/5
CHALLENGE TASK - Uploaded object	5/5

4. Conclusion

Ce laboratoire a clairement démontré la flexibilité et la granularité du contrôle d'accès dans AWS :

1. **Les politiques d'identité** associées à *devuser* offraient un accès de base, mais limitaient les actions sensibles, comme la création d'instances EC2 ou l'écriture dans des buckets non autorisés.
2. **Les rôles IAM** nous ont permis d'élever temporairement nos permissions afin d'accomplir des tâches spécifiques (gestion ciblée de certains buckets) sans avoir à attribuer ces privilèges de manière permanente à l'utilisateur.
3. **Les politiques basées sur les ressources**, appliquées directement aux buckets S3, ont ajouté une couche supplémentaire de contrôle, en définissant précisément quels *principals* (utilisateurs ou rôles) étaient autorisés à interagir avec les données du bucket.