

## Introduction

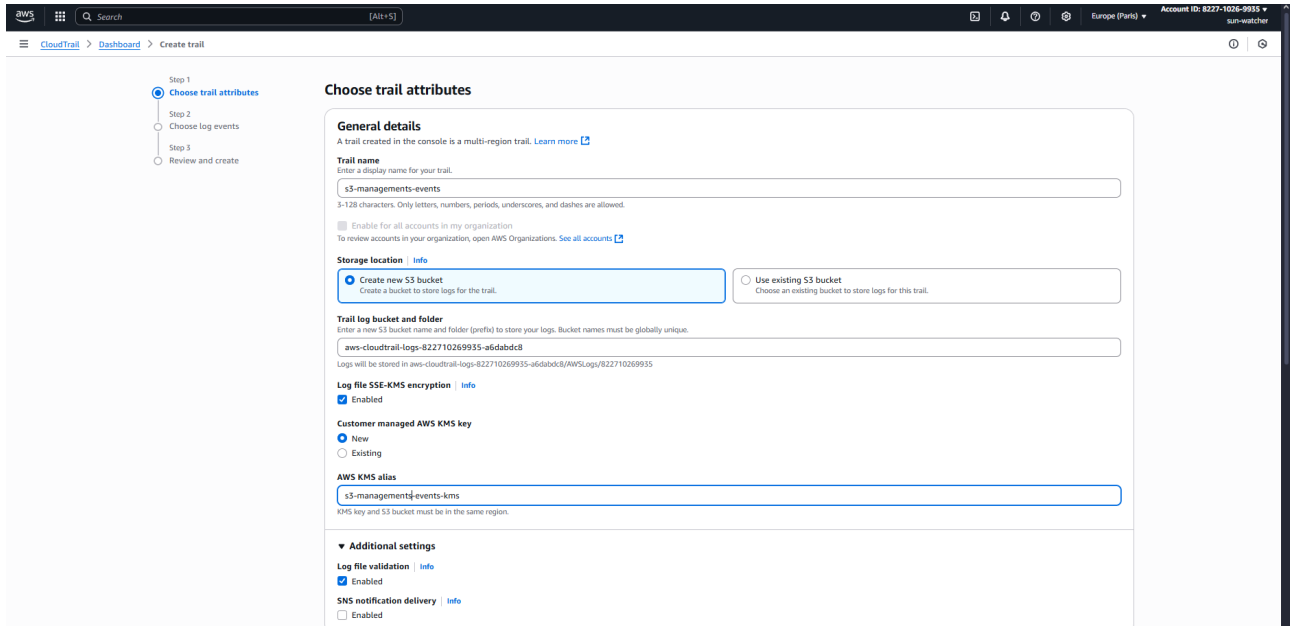
Le but de ce projet est de créer une alarme qui détecte quand la visibilité d'un bucket S3 change. Pour ça, on va utiliser CloudWatch qui va remonter l'alerte et l'afficher dans un dashboard. Quand une modification est détectée, une fonction Lambda sera déclenchée pour envoyer un mail de notification. En plus, on active les logs CloudTrail pour garder une trace des événements.

# Contents

1	Configuration de CloudTrail	3
2	Configuration de SNS Topic	4
3	Configuration de Lambda	5
4	Création de dashboard et alarme	10
5	Déclenchement de l'alarme	16
6	Log Cloudtrail	18

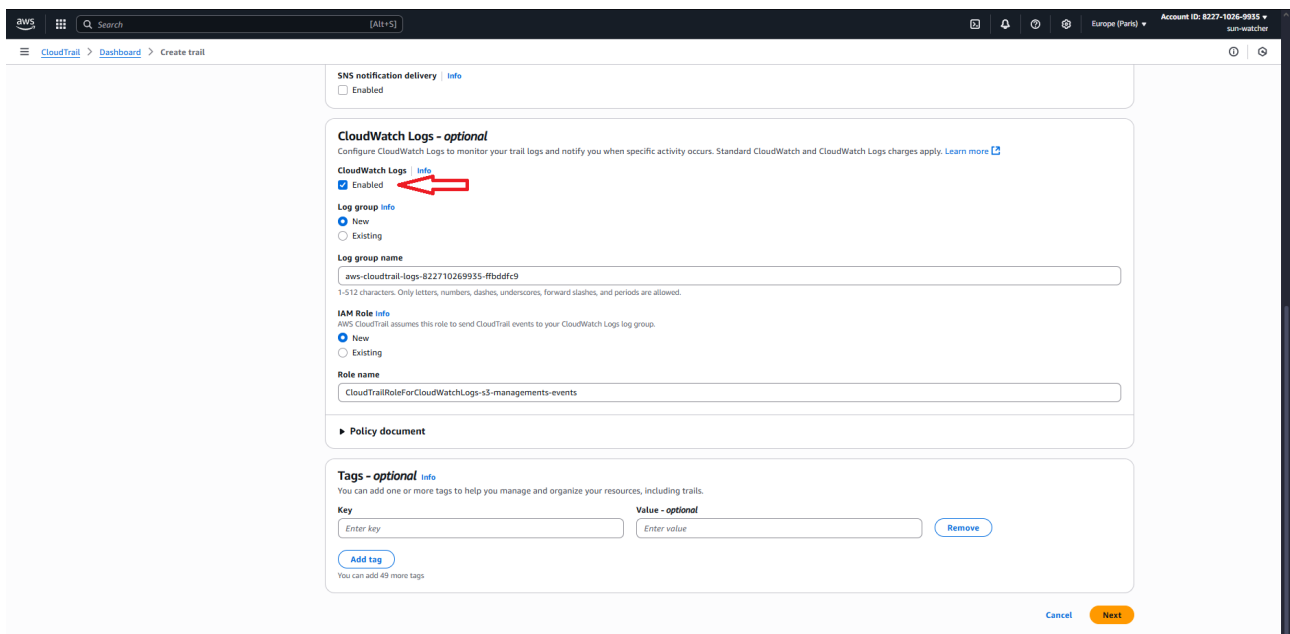
# 1 Configuration de CloudTrail

Configurons tout d'abord le CloudTrail, en activant **CloudWatch Logs**, ce qui nous permettra de garder toute trace pour chaque changement effectué (voir [Figure 1](#)). Etant donné que nous n'avons pas encore de **bucket S3** pour contenir les logs, nous allons directement en créer un. L'utilisation de **CloudWatch Logs** va créer un **Log group** qui permettra de nous rediriger vers les logs de ce **Trail**.



The screenshot shows the 'Choose trail attributes' step in the AWS CloudTrail console. The trail name is 's3-managements-events'. The storage location is 'Create new S3 bucket'. The trail log bucket and folder is 'aws-cloudtrail-logs-822710269935-a6dabdc8'. Log file SSE-KMS encryption is 'Enabled'. Customer managed AWS KMS key is 'New'. AWS KMS alias is 's3-managements-events-kms'. Additional settings: Log file validation is 'Enabled', and SNS notification delivery is 'Enabled'.

Figure 1: Configuration du Trail 1/3



The screenshot shows the 'CloudWatch Logs - optional' step in the AWS CloudTrail console. The 'CloudWatch Logs' checkbox is checked and highlighted with a red arrow. The log group name is 'aws-cloudtrail-logs-822710269935-ftbdfc9'. The IAM role is 'New' with the role name 'CloudTrailRoleForCloudWatchLogs-s3-managements-events'. The policy document is 'Policy document'. The tags section is empty.

Figure 2: Configuration du Trail 2/3

Etant donné que nous voulons uniquement garder une trace de toute modification de gestion, nous allons simplement cocher la case **Management events** comme nous pouvons le voir sur [Figure 3](#).

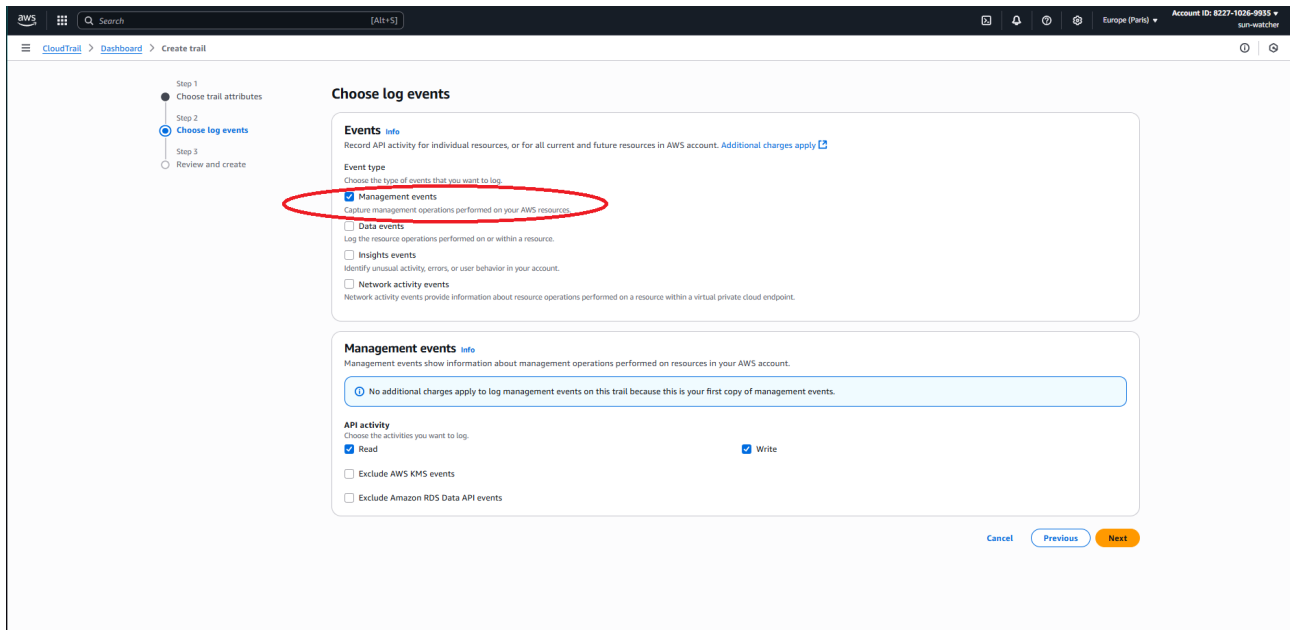


Figure 3: Configuration du Trail 3/3

## 2 Configuration de SNS Topic

Nous allons utiliser **SNS** pour envoyer des notifications par mail. Il faudra tout d'abord créer un **topic** (voir Figure 4) puis s'abonner à ce topic (voir Figure 5 et Figure 6).

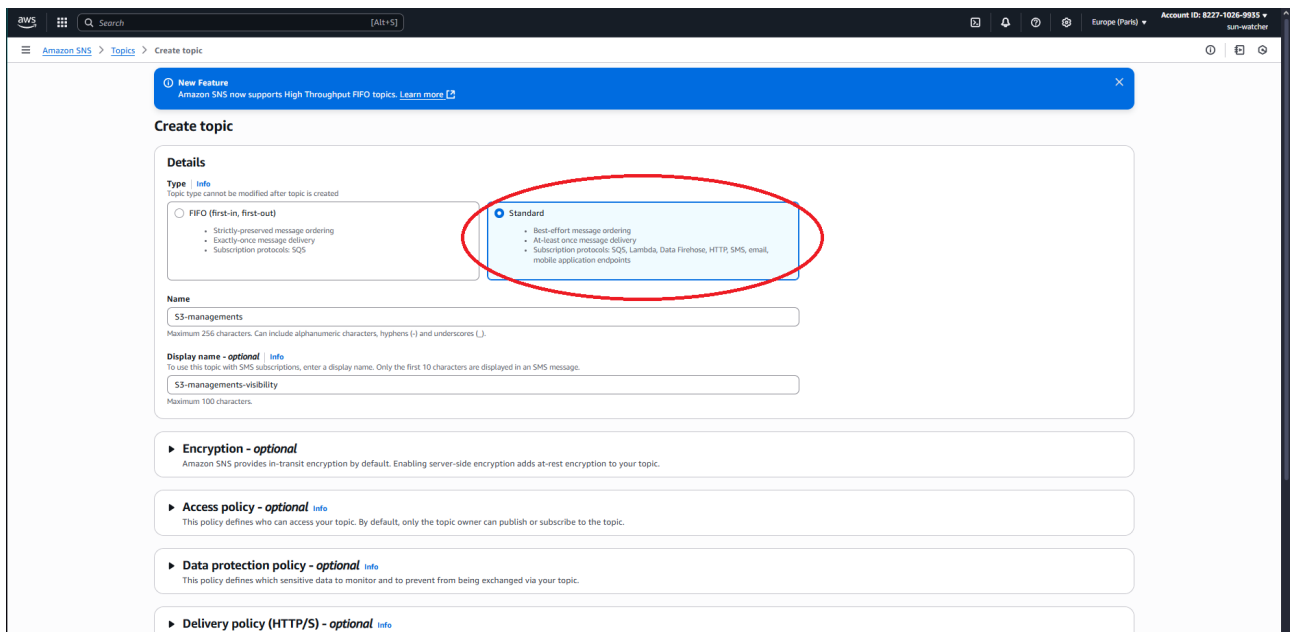


Figure 4: Création du topic

Amazon SNS > Subscriptions > Create subscription

**Create subscription**

**Details**

Topic ARN  
`arn:aws:sns:eu-west-3:822710269935:S3-managements`

Protocol  
Email

Endpoint  
An email address that can receive notifications from Amazon SNS.  
`test@example.com`

After your subscription is created, you must confirm it. [Info](#)

**Subscription filter policy - optional** [Info](#)  
This policy filters the messages that a subscriber receives.

**Redrive policy (dead-letter queue) - optional** [Info](#)  
Send undeliverable messages to a dead-letter queue.

[Cancel](#) [Create subscription](#)

Figure 5: Abonnement

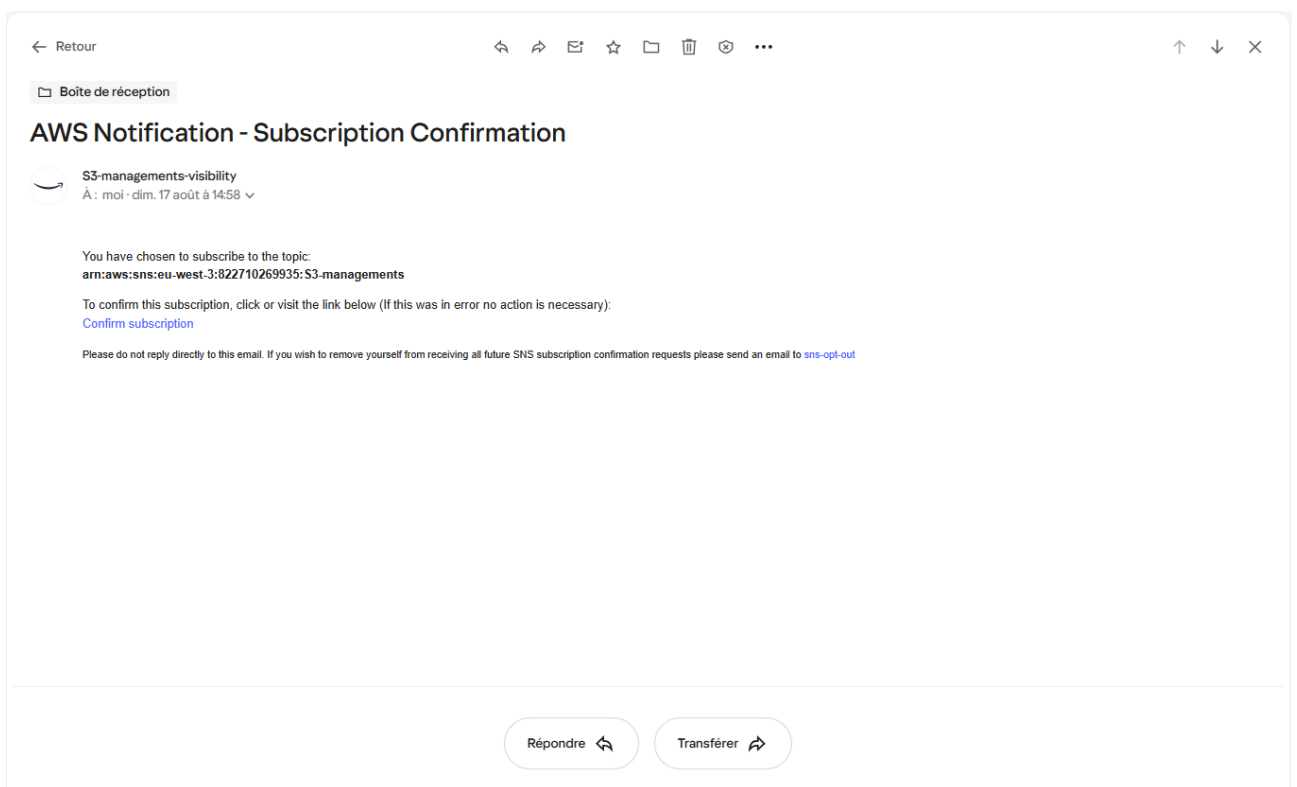


Figure 6: Confirmation de l'abonnement

### 3 Configuration de Lambda

Nous allons maintenant écrire une fonction permettant d'envoyer une notification lorsque l'alarme est déclenchée. Pour cela, nous allons utiliser **Lambda**. Nous testerons d'abord d'envoyer un simple "Hello World" au topic pour tester le code.

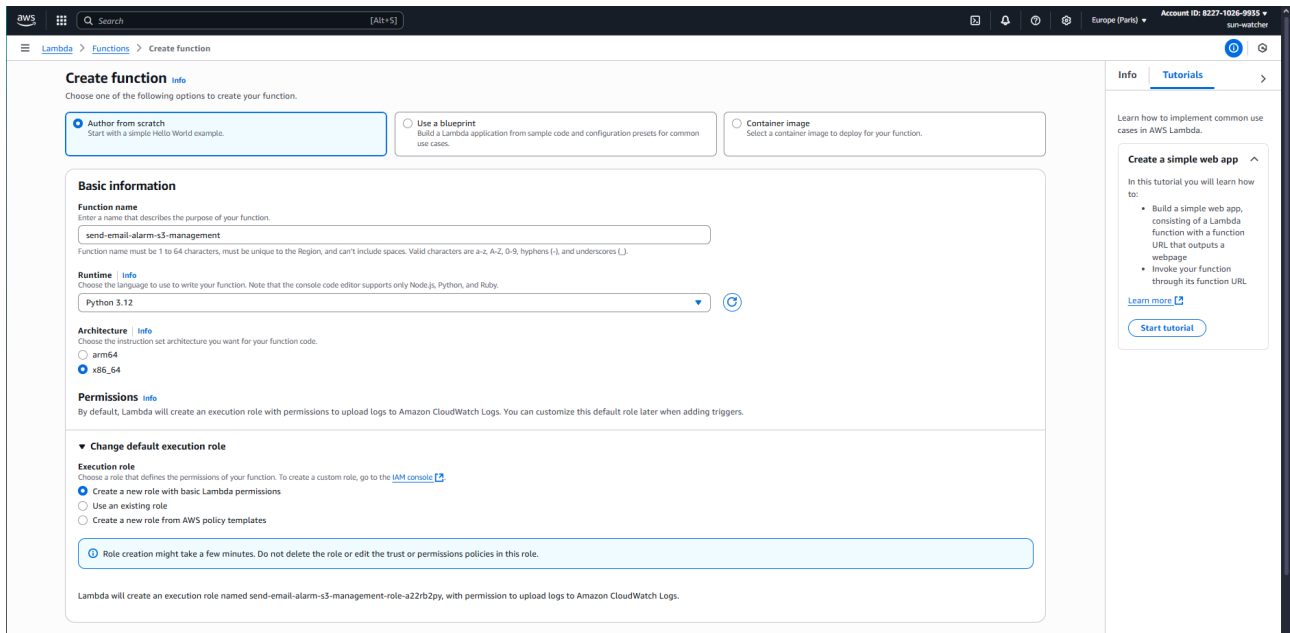


Figure 7: Création d'une fonction Lambda

Lors de ce test, nous pouvons voir sur la Figure 8 qu'une autorisation est requise pour pouvoir publier sur le topic.

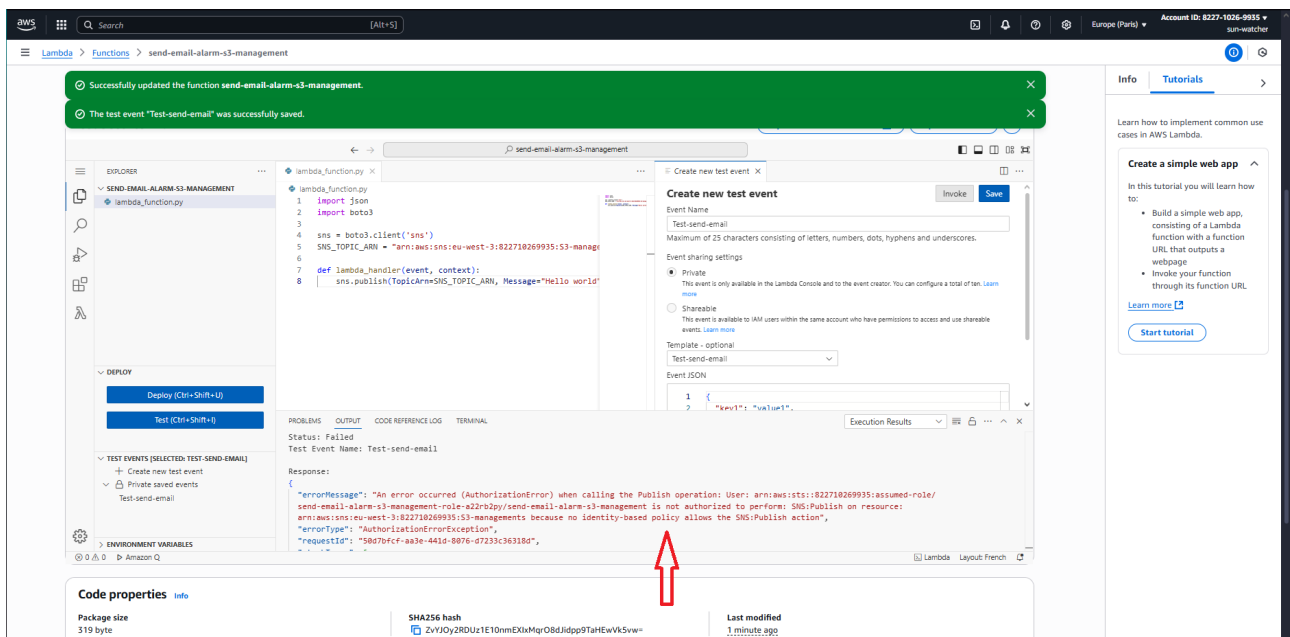


Figure 8: Test d'envoi de notification

Sur les Figure 9, Figure 10 et Figure 11, nous activons cette autorisation.

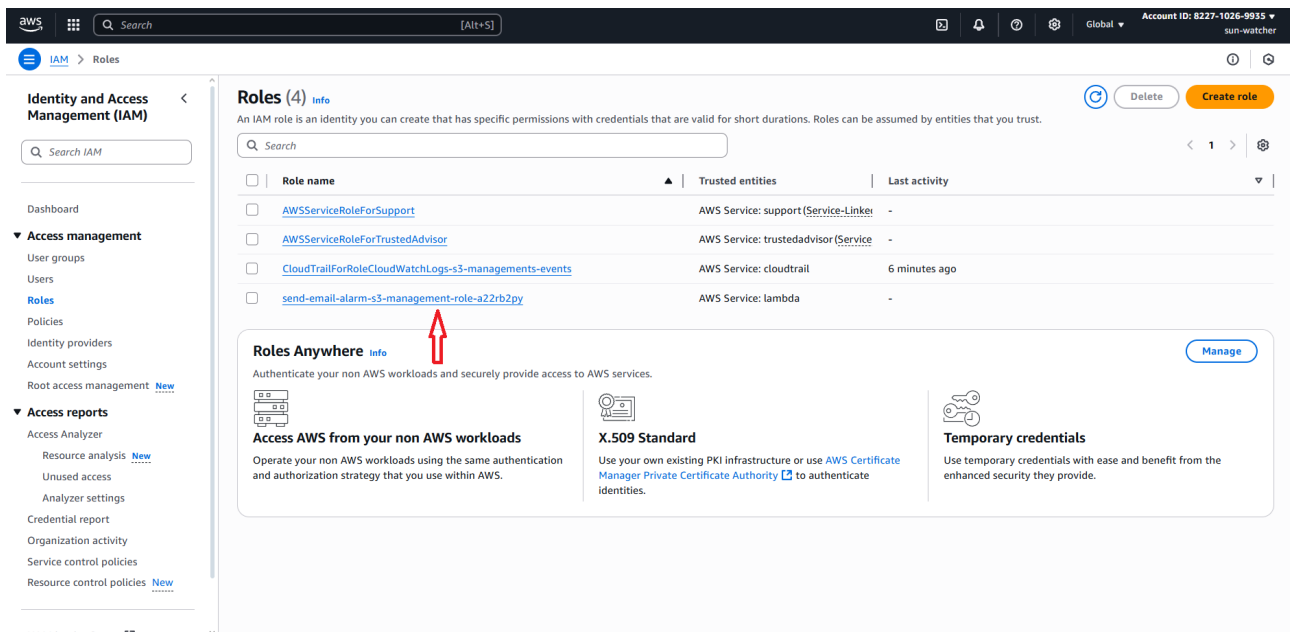


Figure 9: Autorisation de publication 1/3

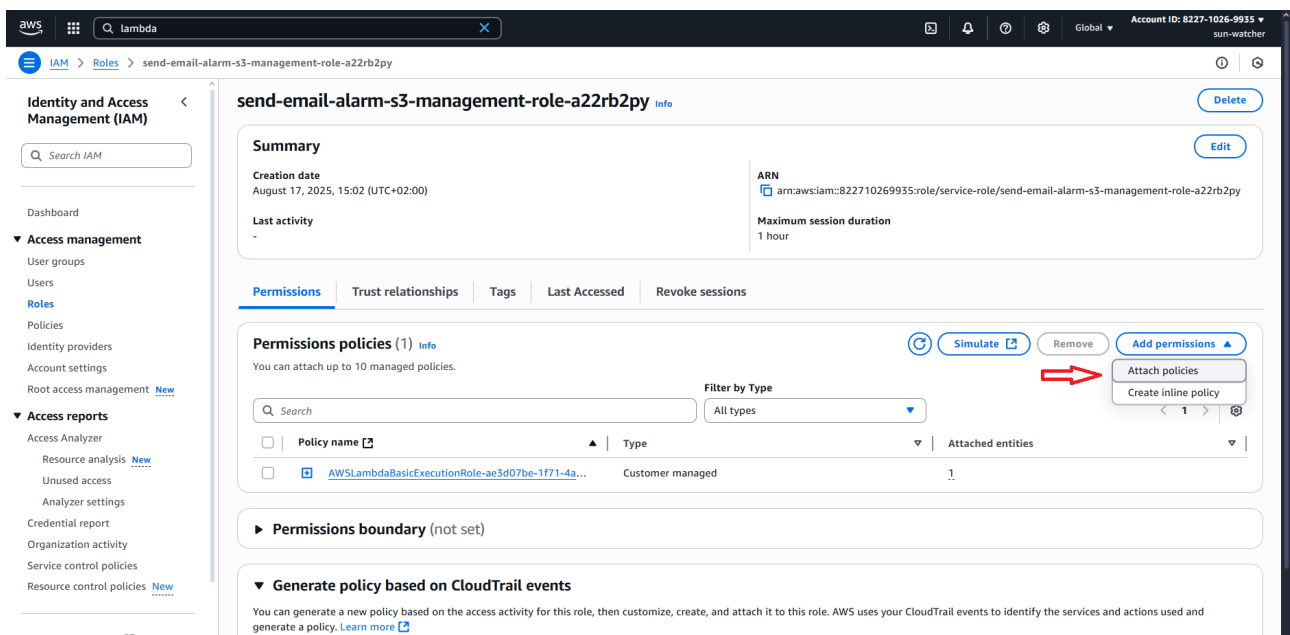


Figure 10: Autorisation de publication 2/3

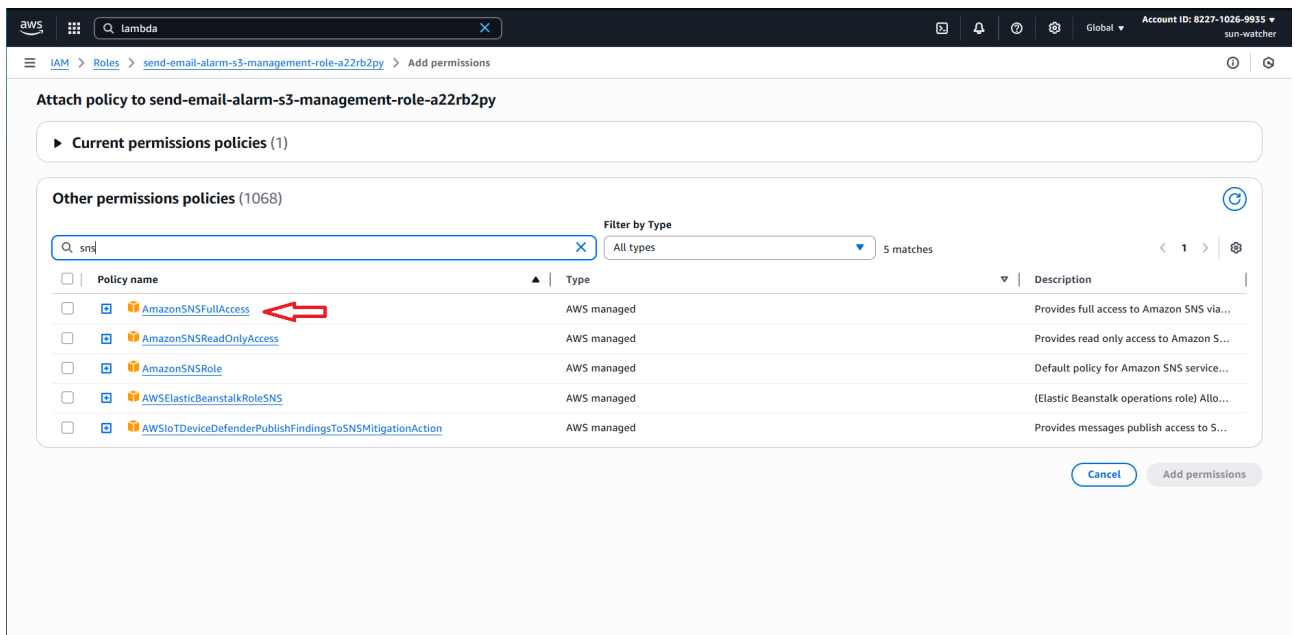


Figure 11: Autorisation de publication 3/3

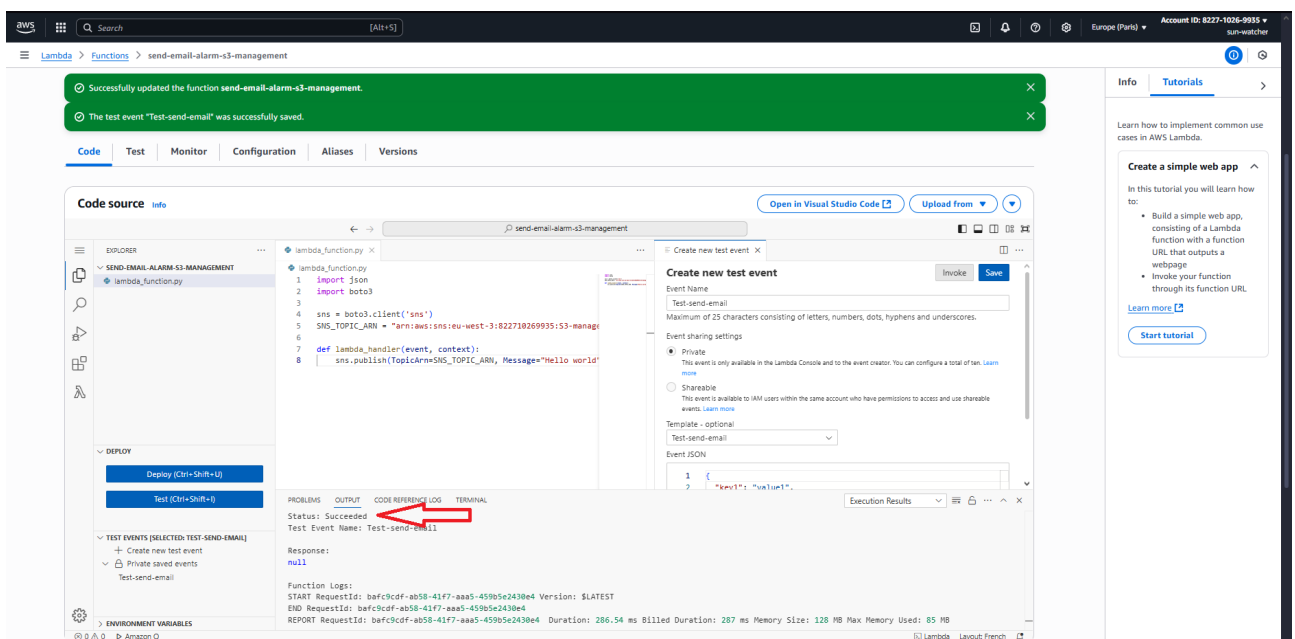


Figure 12: Test réussi



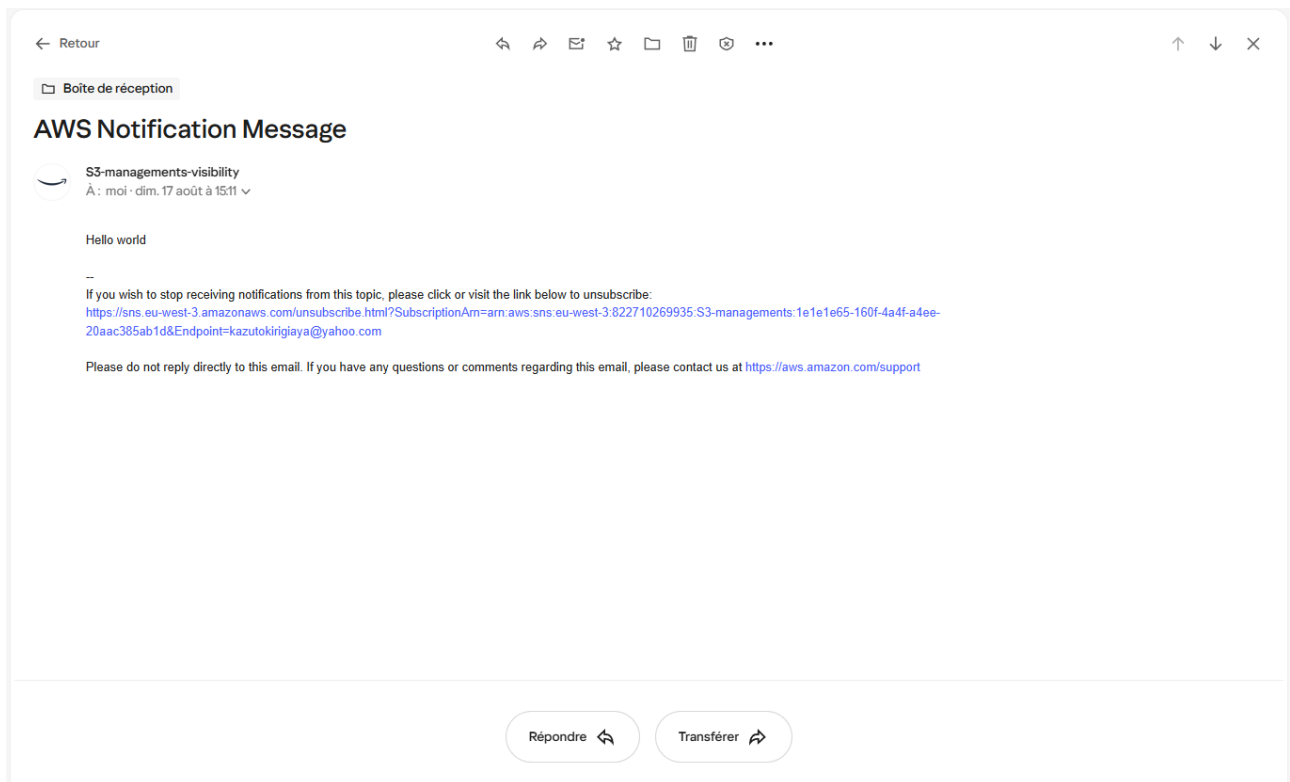


Figure 13: Mail reçu

Une fois le test réussi, nous pouvons ajouter le code suivant, qui permet d'envoyer un mail avec des informations importantes.

```

1  import json
2  import boto3
3
4  sns = boto3.client("sns")
5  SNS_TOPIC_ARN = "arn:aws:sns:eu-west-3:822710269935:Misconfiguration_S3_Visibility"
6
7  def lambda_handler(event, context):
8      alarm = event.get("alarmData", {})
9      alarm_name = alarm.get("alarmName")
10     state = alarm.get("state", {})
11     metric = None
12
13     # aller chercher le nom de la métrique
14     metrics = alarm.get("configuration", {}).get("metrics", [])
15     if metrics:
16         metric = metrics[0].get("metricStat", {}).get("metric", {}).get("name")
17
18     # construire un message résumé
19     message = {
20         "AlarmName": alarm_name,
21         "NewState": state.get("value"),
22         "Reason": state.get("reason"),
23         "Metric": metric,
24         "Region": event.get("region"),
25         "AccountId": event.get("accountId"),
26         "Time": event.get("time")
27     }
28
29     # publier
30     sns.publish(
31         TopicArn=SNS_TOPIC_ARN,
32         Subject=f"[ALARM] {alarm_name}",
33         Message=json.dumps(message, indent=2)

```

## 4 Création de dashboard et alarme

Nous allons maintenant créer le dashboard (voir Figure 14 et Figure 15), et ajouter une alarme à ce dashboard.

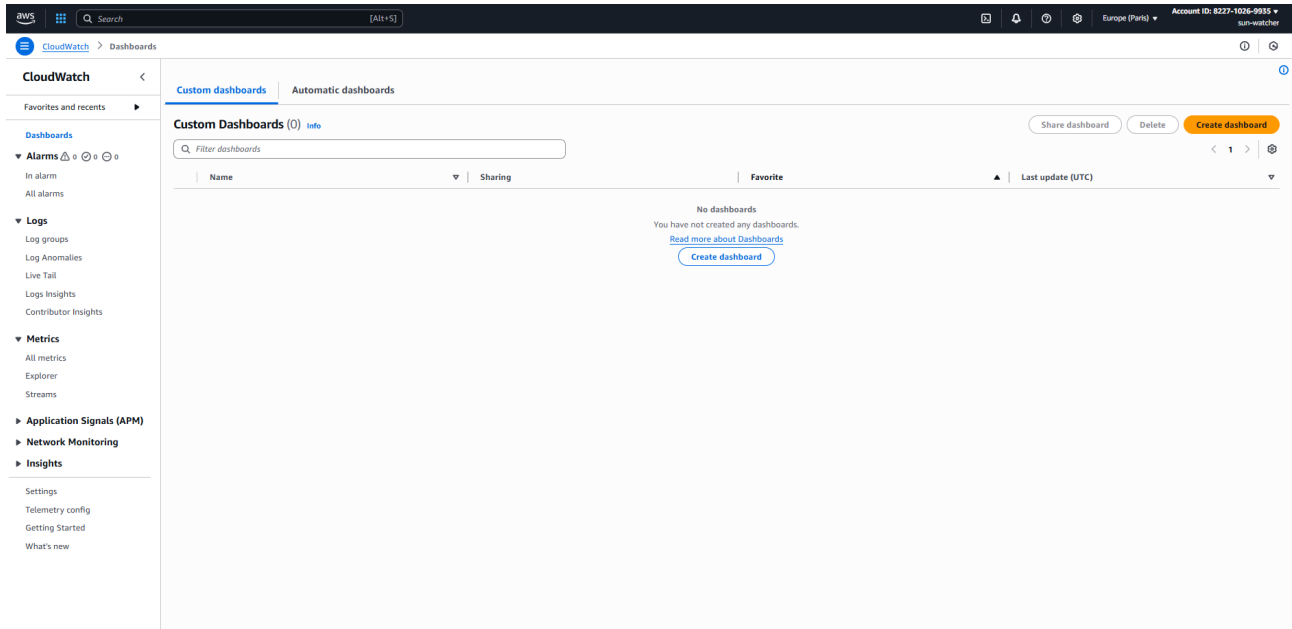


Figure 14: Création du dashboard 1/2

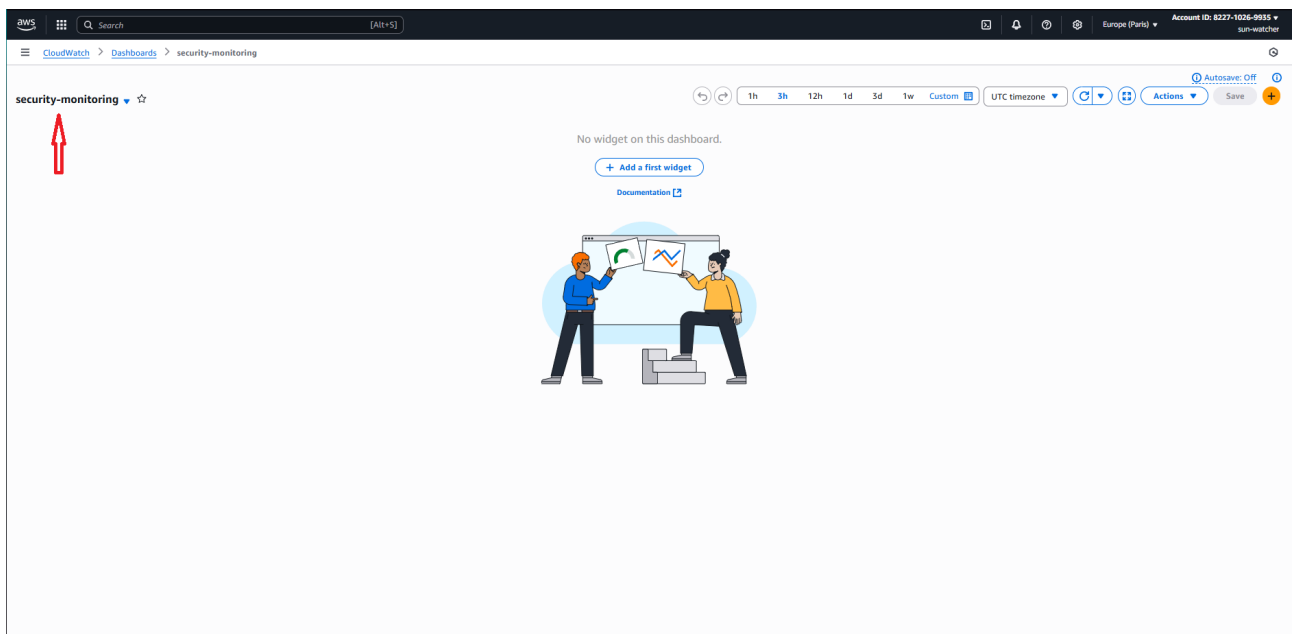


Figure 15: Création du dashboard 2/2

La métrique que nous utiliserons pour détecter un changement de visibilité sur un bucket est **PutBucketPublicAccessBlock**. Chaque fois que cette métrique a été détectée dans la dernière minute, l'alarme sera déclenchée et activera automatiquement la fonction lambda que nous avons écrite précédemment.

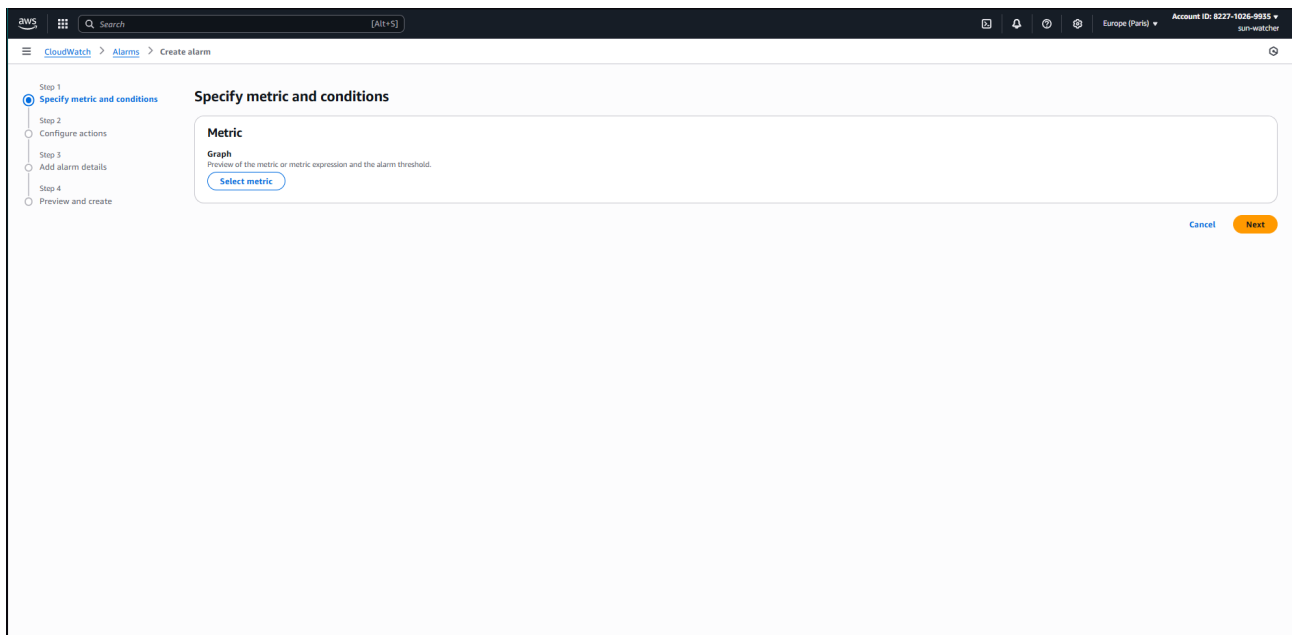


Figure 16: Première image

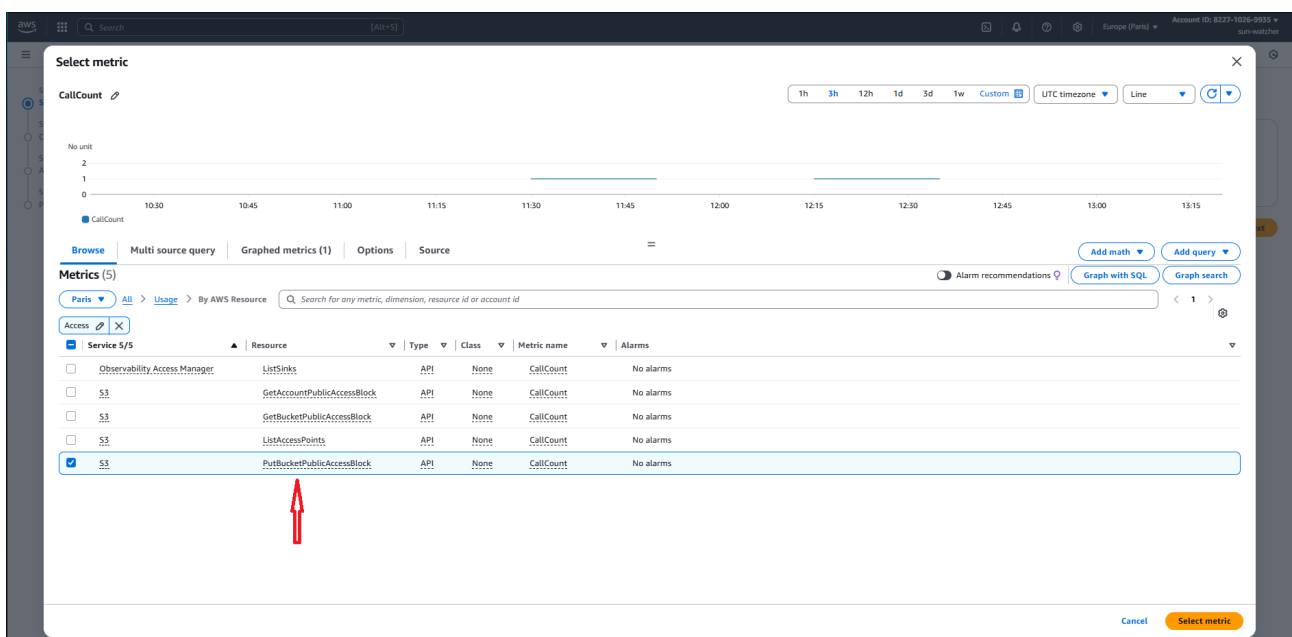


Figure 17: Création de l'alarme 1/5

**Specify metric and conditions**

**Metric**

Graph  
This alarm will trigger when the blue line goes above the red line for 1 datapoints within 1 minute.

No unit

2

1.5

1

10:30 11:00 11:30 12:00 12:30 13:00

CallCount

Namespace  
AWS/Usage

Metric name  
CallCount

Type  
API

Resource  
PutBucketPublicAccessBlock

Service  
S3

Class  
None

Statistic  
Sum

Period  
1 minute

**Conditions**

Threshold type

☒ Static  
Use a value as a threshold

☐ Anomaly detection  
Use a band as a threshold

Figure 18: Création de l'alarme 2/5

Sur la figure suivante, nous considérerons que à l'état de données manquantes, l'état de l'alarme sera "ok".

**Conditions**

Threshold type

☒ Static  
Use a value as a threshold

☐ Anomaly detection  
Use a band as a threshold

Whenever CallCount is...

Define the alarm condition.

☐ Greater  
> threshold

☒ Greater/Equal  
≥ threshold

☐ Lower/Equal  
≤ threshold

☐ Lower  
< threshold

than...

Define the threshold value.

1

Must be a number.

**Additional configuration**

**Datapoints to alarm**

Define the number of datapoints within the evaluation period that must be breaching to cause the alarm to go to ALARM state.

1 out of 1

**Missing data treatment**

How to treat missing data when evaluating the alarm.

Treat missing data as good (not breaching threshold)

Cancel Next

Figure 19: Création de l'alarme 3/5

**Configure actions**

**Notification**  
Add notification

**Lambda action** ←

**Alarm state trigger**  
Define the alarm states that will trigger this action. Remove

☒ **In alarm**  
The metric or expression is outside of the defined threshold. ←

☐ **OK**  
The metric or expression is within the defined threshold.

☐ **Insufficient data**  
The alarm has just started or not enough data is available.

**Function Type**  
Select the type of Lambda Function.

☒ **Select Lambda Function from the signed in account**

☐ Enter Function ARN for cross account function

**Choose a function**  
Q send-email-alarm-s3-management ← X

Existing Lambda Functions on the signed-in account, enter the function ARN for cross account functions

► **Configure version/alias**  
Add Lambda action

**Auto Scaling action**  
Add Auto Scaling action

**EC2 action**  
This action is only available for EC2 Per-Instance Metrics.  
Add EC2 action

Figure 20: Création de l'alarme 4/5

**Add alarm details**

**Name and description**

**Alarm name**  
Bucket visibility management

**Alarm description - optional** [View formatting guidelines](#)

[Edit](#) [Preview](#)

# This is an H1  
\*\*double asterisks will produce strong character\*\*  
This is [an example]https://example.com/ inline link.

Up to 1024 characters (0/1024)

☒ **Markdown formatting is only applied when viewing your alarm in the console. The description will remain in plain text in the alarm notifications.**

**Tags - optional** [Info](#)

No tags associated with the resource.

[Add new tag](#)  
You can add up to 50 tags

[Cancel](#) [Previous](#) [Next](#)

Figure 21: Création de l'alarme 5/5

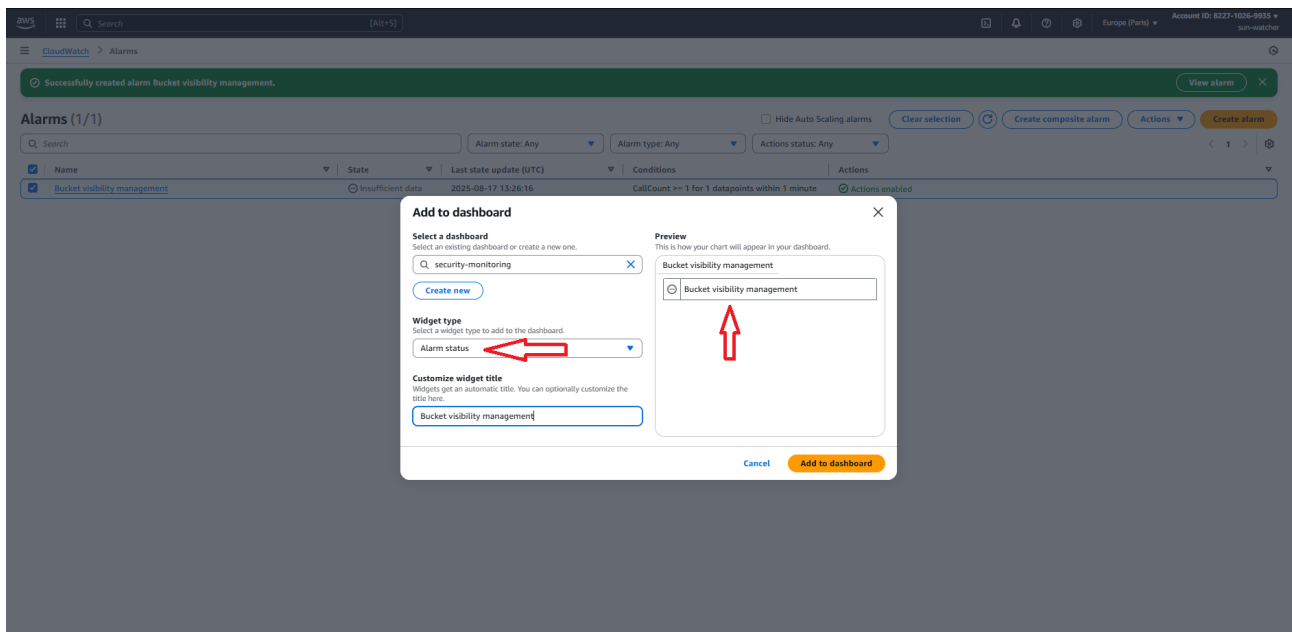


Figure 22: Ajout de l'alarme au dashboard

Nous devons créer une **policy** pour que lambda autorise CloudWatch à utiliser ses API. Nous aurons besoin de l'**ARN** de cette alarme pour créer ce policy (voir Figure 23).

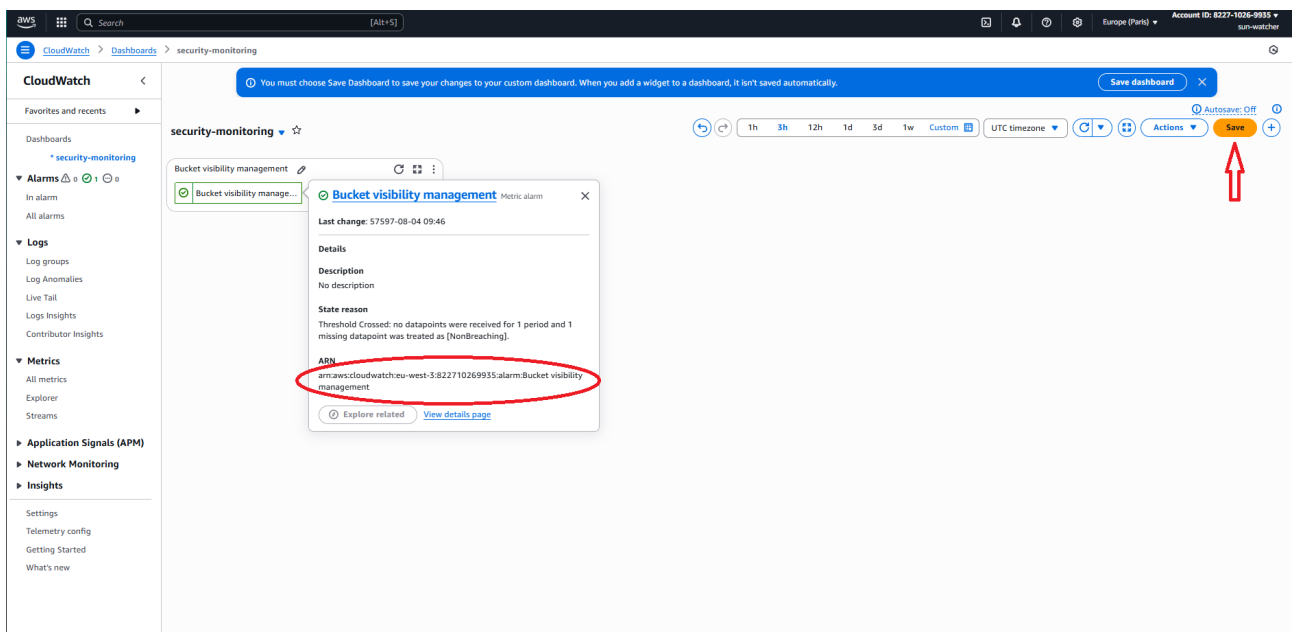


Figure 23: Alarme visible sur le dashboard

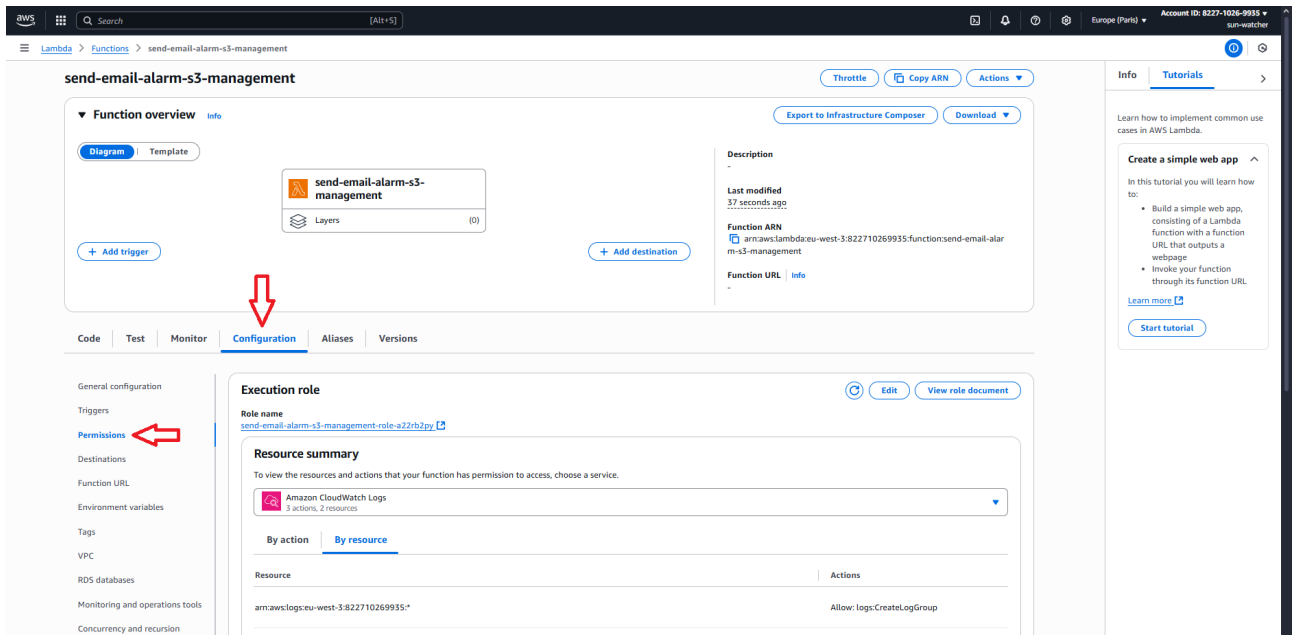


Figure 24: Création du policy 1/3

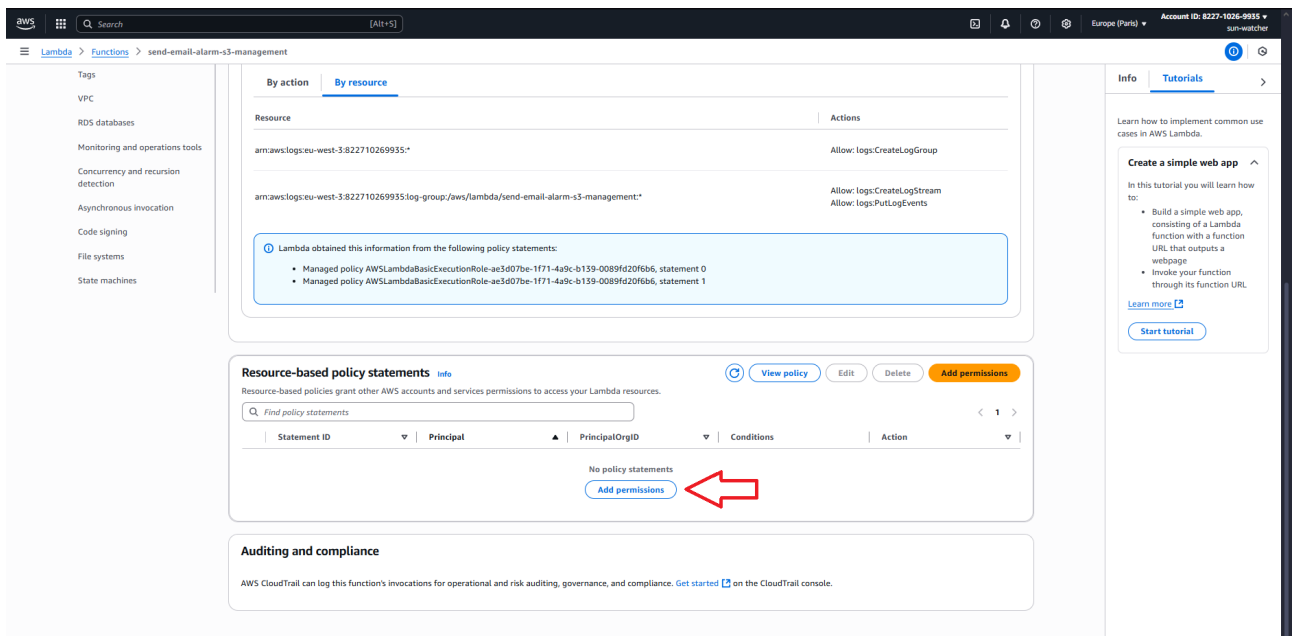


Figure 25: Création du policy 2/3

**Add permissions**

**Edit policy statement**

☐ AWS account  
Grant permissions to another AWS account, user, or role.

☒ AWS service  
Grant permissions to another AWS service.

☐ Function URL  
Grant permissions to invoke your function through the function URL.

**Service**  
The AWS service to grant permissions to.  
**Other**

**Statement ID**  
Enter a unique statement ID to differentiate this statement within the policy.  
lambda-InvocationFunction-cloudwatch

**Principal**  
The service principal for this AWS service. [Learn more](#)  
lambda:alarms.cloudwatch.amazonaws.com

**Source ARN**  
The ARN for a resource. Find the ARN in the related service console.  
arn:aws:cloudwatch:eu-west-3:822710269935:alarm:Bucket visibility management

**Action**  
Choose an action to allow.  
lambda:InvokeFunction

[Cancel](#) [Save](#)

Figure 26: Création du policy 3/3

## 5 Déclenchement de l'alarme

Pour déclencher l'alarme, nous allons changer la visibilité d'un bucket.

**Permissions overview**

**Access finding**  
Access findings are provided by IAM external access analyzers. [Learn more about How IAM analyzer findings work](#)  
[View analyzer for eu-west-3](#)

**Block public access (bucket settings)** [Edit](#)  
Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to all your S3 buckets and objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to your buckets or objects within, you can customize the individual settings below to suit your specific storage use cases. [Learn more](#)

**Block all public access**  
☒ On [Individual block Public Access settings for this bucket](#)

**Bucket policy** [Edit](#) [Delete](#)  
The bucket policy, written in JSON, provides access to the objects stored in the bucket. Bucket policies don't apply to objects owned by other accounts. [Learn more](#)

**Public access is blocked because Block Public Access settings are turned on for this bucket**  
To determine which settings are turned on, check your Block Public Access settings for this bucket. [Learn more about using Amazon S3 Block Public Access](#)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AWSCloudTrailAclCheck20150319-9ada684-7769-4d86-9b79-416af217915c",
      "Effect": "Allow",
      "Principal": {
        "Service": "cloudtrail.amazonaws.com"
      },
      "Action": "s3:GetBucketAcl",
      "Resource": "arn:aws:s3::aws-cloudtrail-logs-822710269935-70f2ec89"
    }
  ]
}
```

[Copy](#)

Figure 27: Avant le changement de visibilité



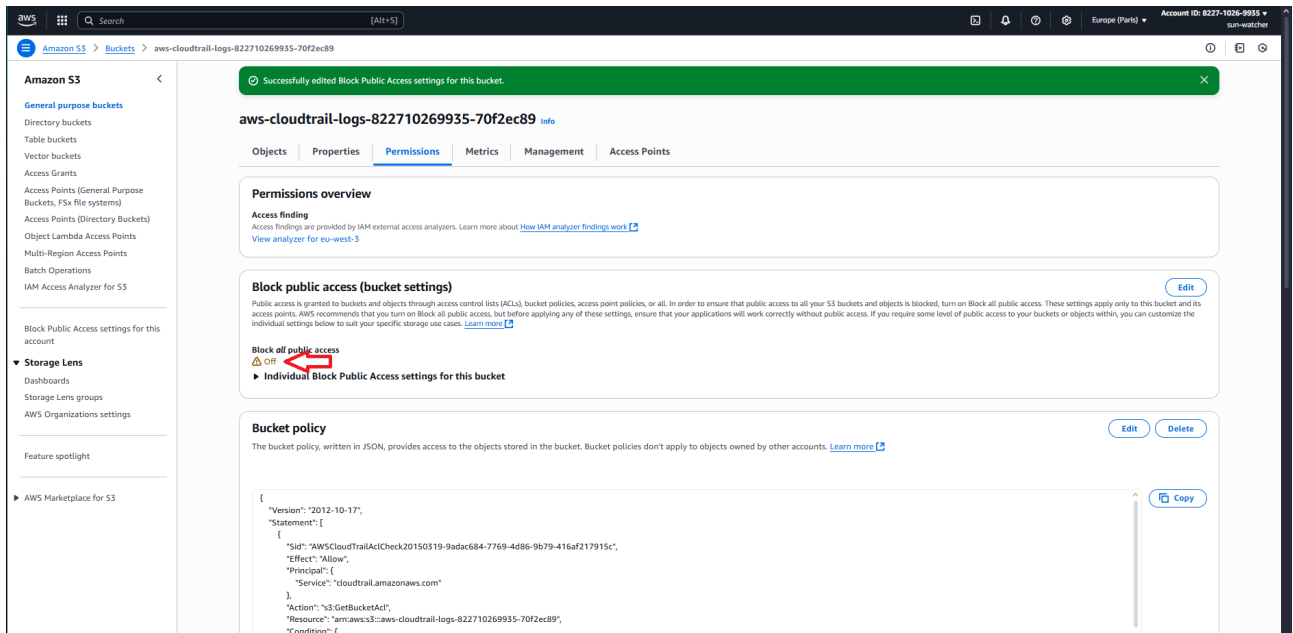


Figure 28: Après le changement de visibilité

Nous pouvons voir que l'alarme a bien été déclenchée (voir Figure 29), que la fonction lambda s'est exécutée avec succès (voir Figure 29) et que le mail a bien été envoyé (voir Figure 30).

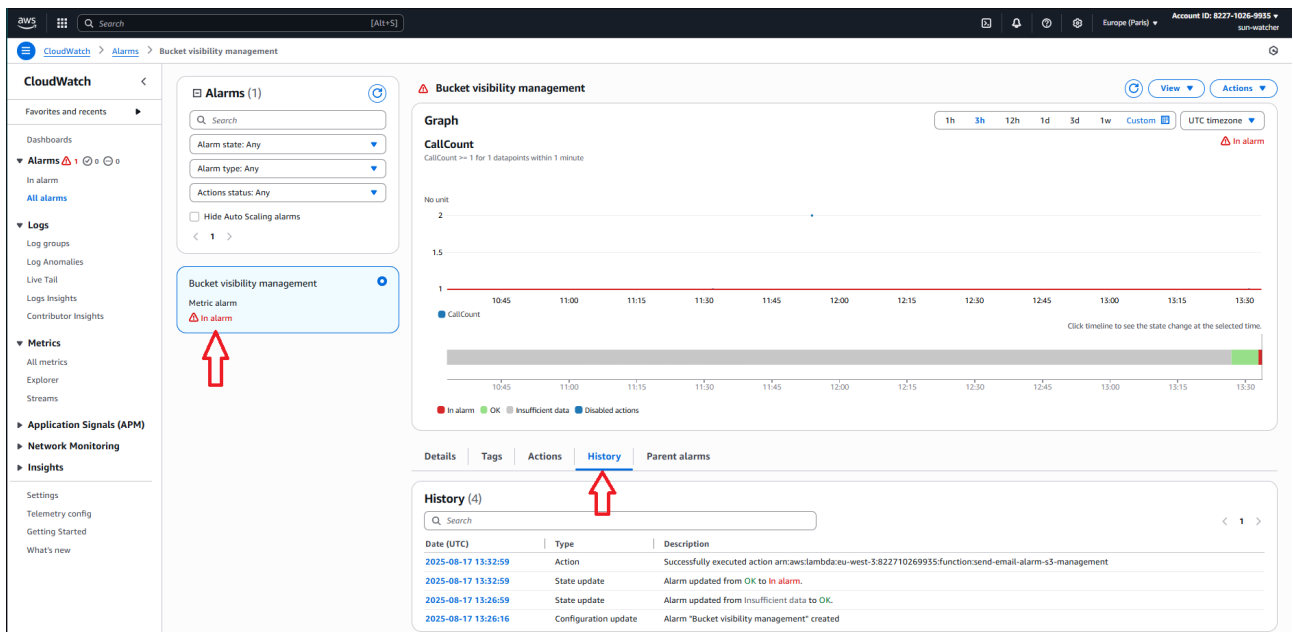


Figure 29: Alarme en état d'alerte

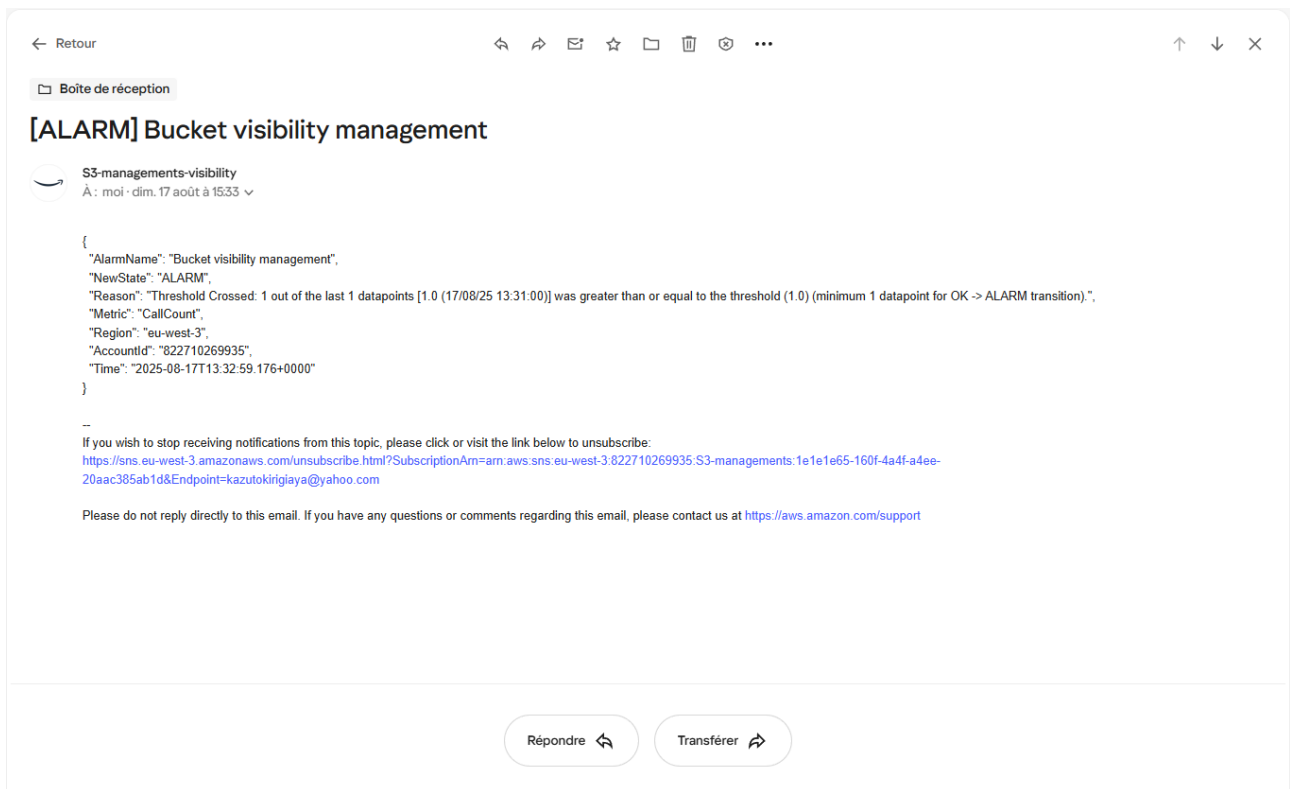


Figure 30: Mail d'alerte reçu

## 6 Log Cloudtrail

Nous pouvons voir que CloudTrail enregistre chaque évènement qui a été effectué, dont le "PutBucketPublicAccessBlock" (Sur la figure suivante, cet évènement ne correspond pas à celui que nous avons effectué plus tôt, mais d'un autre que nous avons lancé plus tard). Cette fonctionnalité offre un aperçu facile à lire mais ne trace les évènements ne datant que d'au moins 90 jours. C'est pour cela que la création d'un Trail peut s'avérer utile car cela permet de garder les logs dans un bucket pour une durée indéterminée.

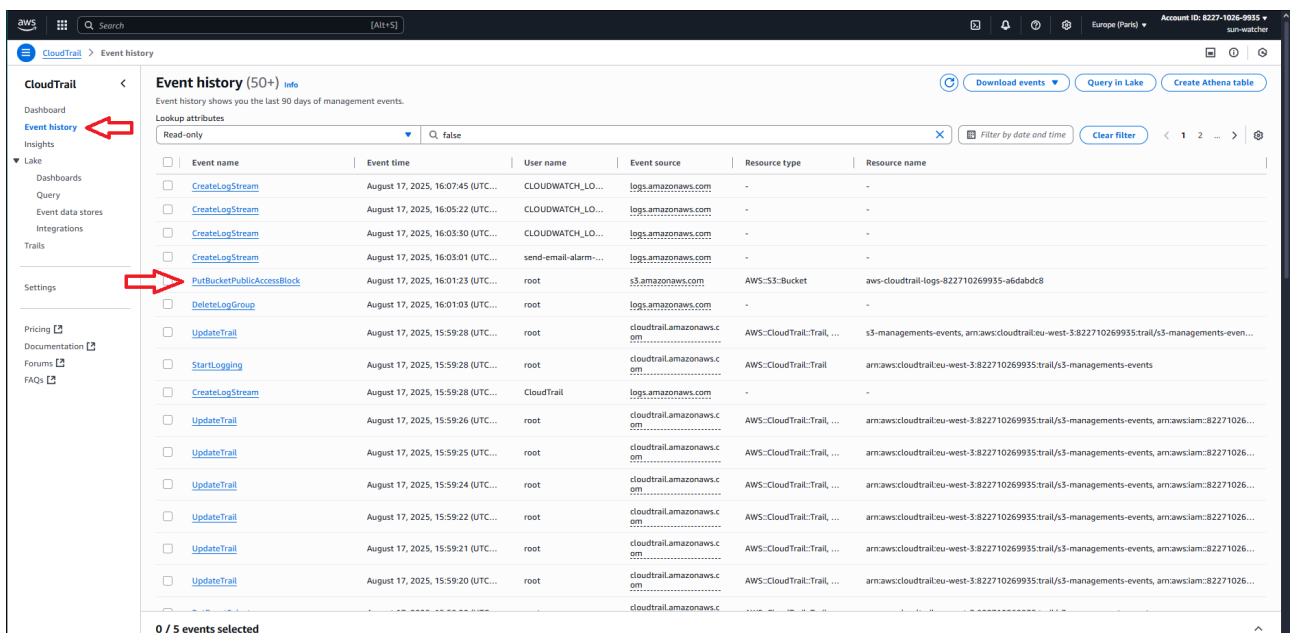


Figure 31: Cloudtrail Event History

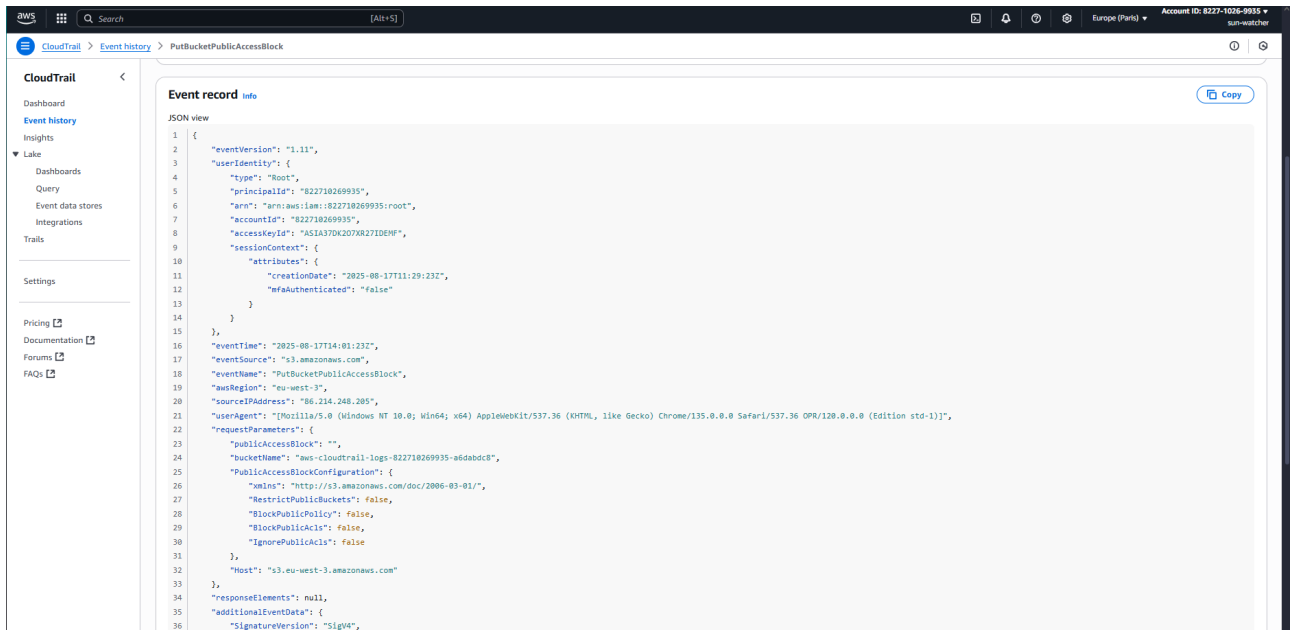


Figure 32: Détail de l'évènement

Sur la figure suivante, nous utilisons les logs qui ont été capturés par le Trail que nous avons créé afin de savoir quels sont ces logs qui ont capturé l'évènement "PutBucketPublicAccessBlock". Pour cela, nous pouvons utiliser la requête :

```
1 fields @timestamp, eventName, requestParameters.bucketName, userIdentity.arn
2 | filter eventName="PutBucketPublicAccessBlock"
3 | sort @timestamp desc
```

(Nous avons effectué plusieurs tests et il semblerait que lorsque nous modifions la visibilité sur le bucket contenant les logs, CloudTrail ne parvient pas à capturer cet évènement. En effectuant ce test sur un autre bucket, CloudTrail a pu tracer cette modification. C'est pour cette raison qu'il n'y a qu'un seul résultat après avoir effectuée la requête).

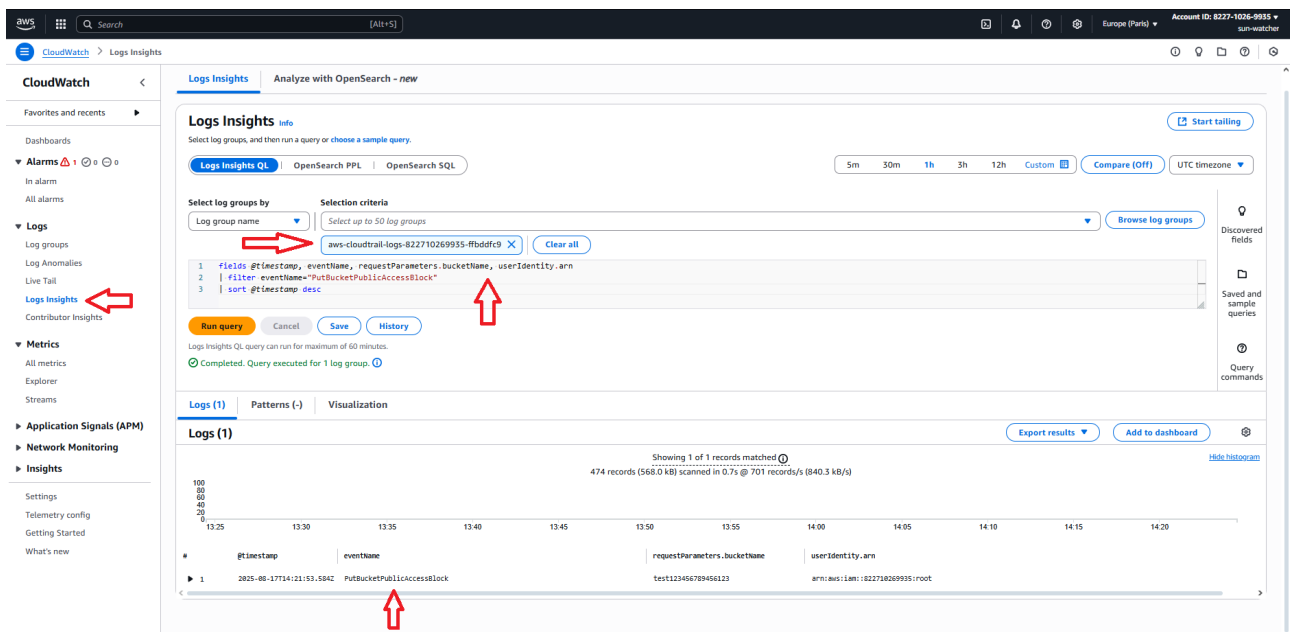


Figure 33: Utilisation et résultat de la requête