Securing AWS IAM:

IMPLEMENTING LEAST PRIVILEGE & MFA ENFORCEMENT

Diego Acosta Cantu

Cybersecurity Portfolio

Table of Contents

Introduction

As organizations increasingly migrate to cloud and hybrid environments, implementing IAM (Identity and Access Management) correctly is crucial for maintaining a strong security posture. IAM provides a centralized approach to managing user access, enforcing least privilege, and reducing the attack surface against both internal and external threats. By restricting permissions, organizations can better protect the Confidentiality, Integrity, and Availability (CIA) of their data and assets.

I created this project to gain hands-on experience with cloud security, IAM, and least privilege architectures, as they are among the most effective defenses against today's ever-evolving cyber threats. This hands-on approach will deepen my understanding of securing cloud environments and applying real-world security best practices.

Prerequisites

This document assumes a fundamental understanding of cybersecurity, cloud computing and AWS Identity & Access Management (IAM). It focuses on implementing security best practices rather than explaining foundational cloud concepts in depth.

What is IAM and why does it matters?

First of all, we need to know what IAM is, and its importance. Identity and Access Management (IAM) is how networks, in this case a cloud platform like AWS control who gets access to what and what they're allowed to do. It helps keep systems secure by ensuring that only the right people and services have the right level of access and nothing more, nothing less. This is important because too much access can lead to accidental mistakes or even security breaches.

IAM also helps businesses stay compliant with security regulations. By using best practices like multi-factor authentication (MFA), least privilege access, and logging with CloudTrail, organizations can protect their cloud resources and reduce risks from unauthorized access, reducing the overall attack surface.

Why is Least Privilege important in Cybersecurity?

On the other hand, "Least Privilege" is a security principle that assigns users and systems only the minimum access they need to do their jobs and nothing more. In cybersecurity, this reduces the risk of accidental mistakes, insider threats, and cyberattacks by limiting what an attacker or a compromised account can do. For example, instead of giving every employee admin rights, least privilege ensures they only have access to the files or systems necessary for their role.

This approach helps protect sensitive data, prevent security breaches, and ensure compliance with security regulations. By enforcing least privilege through role-based access control (RBAC), just-in-time access, and regular permission reviews, organizations can reduce risk and strengthen their security posture.

Project Objectives

The objectives of this project are as follows:

1. Secure AWS IAM (Identity & Access Management): Implement security best practices in AWS IAM to minimize unauthorized access, reduce attack surfaces, and improve identity-based security controls within an AWS environment.

2. Implement Least Privilege Access for Users, Groups, and Roles: Create and configure IAM users, groups, and roles while applying the Principle of Least Privilege (PoLP) to ensure each identity has only the minimum permissions necessary for their tasks.

3. Enable Multi-Factor Authentication (MFA): Strengthen IAM user security by enforcing MFA, requiring an additional authentication factor beyond passwords to enhance account protection.

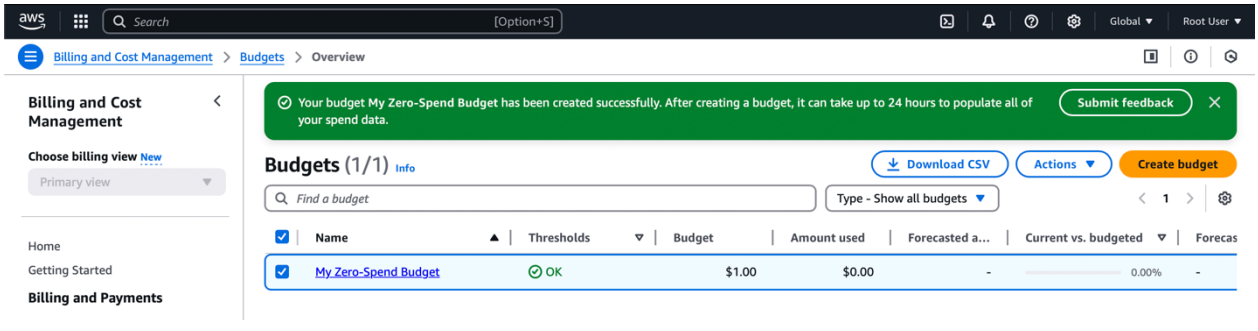
4. Set Up IAM Policies & Roles for Secure Access: Develop custom IAM policies to restrict permissions effectively, preventing excessive access. Configure IAM roles for secure access to AWS services, ensuring role-based access control (RBAC) principles are followed.

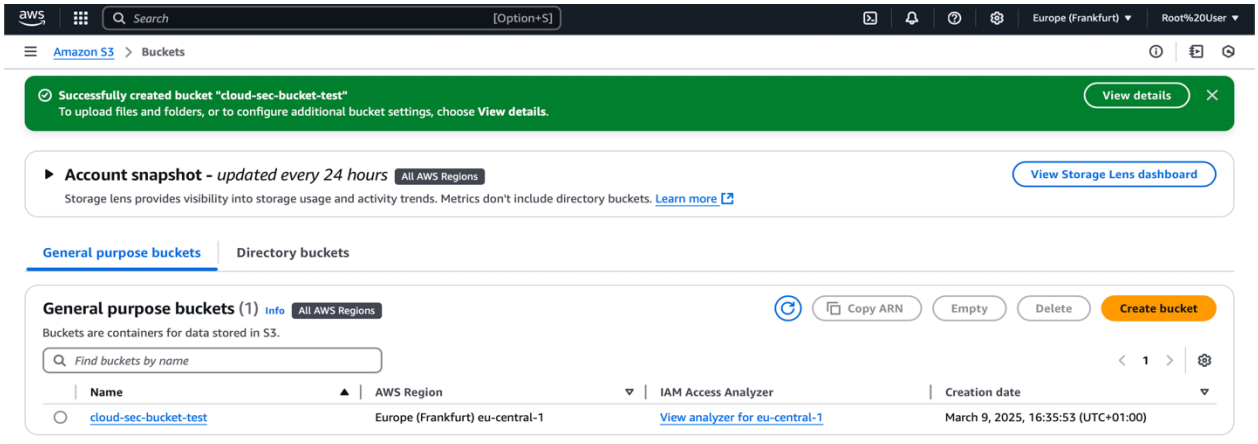
5. Test and Validate Security Settings: Perform security testing by attempting unauthorized actions as an IAM user with restricted permissions, verifying that access is properly denied. Use AWS CloudTrail logs to monitor IAM activity and validate security configurations.

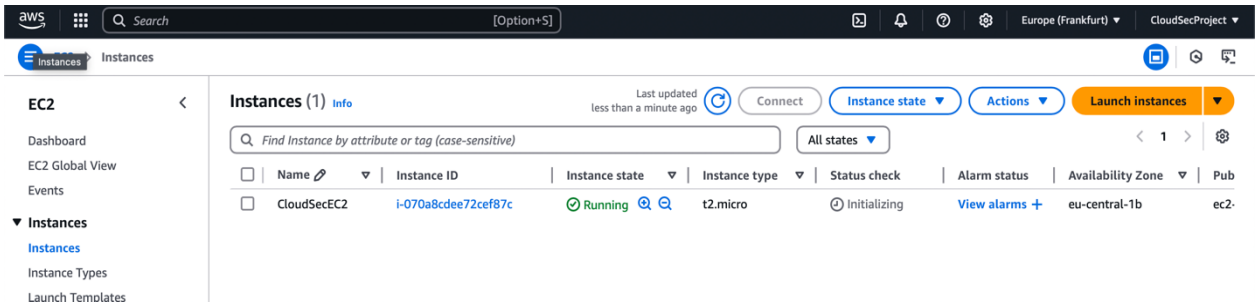
Setting Up the AWS Account

For this project, I set up a fictional AWS account with fictional personal information. I also created a test S3 bucket called cloud-sec-project-test-bucket to practice setting user policies and an EC2 instance named CloudSecEC2 to demonstrate user group management. The goal of this project was to demonstrate how AWS IAM, the Least Privilege Principle, and MFA implementation work in practice.

Since this is purely a demonstrative, student project, I made sure to stay within AWS's Free Tier to avoid spending any real money. To help with that, I used a Zero-Spend Budget template in AWS Budgets, which is part of AWS Billing and Cost Management. This let me set my budget to $0 and get notified if my spending went over $0.01 or if any suspicious or unauthorized transactions happened. It's actually a really useful feature that organizations can use to track and control their AWS expenses, ensuring they don't get surprised by unexpected costs.



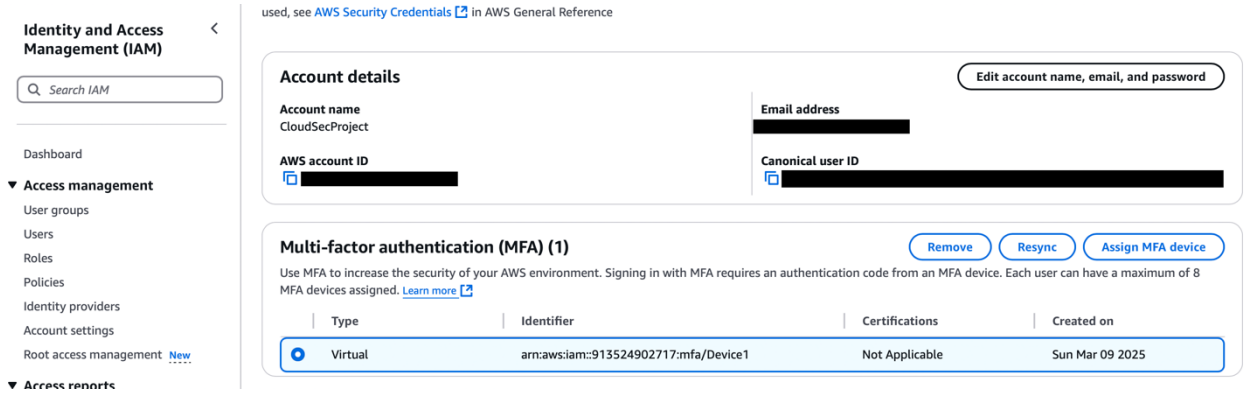




Securing the Root User

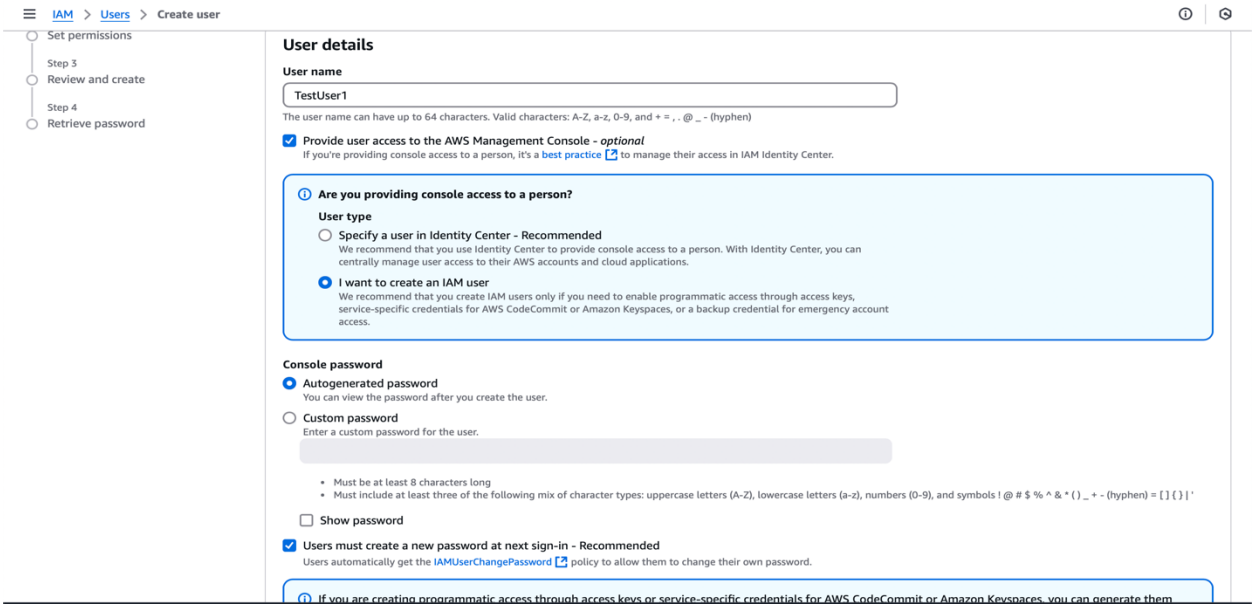
The AWS root account is the most powerful account in AWS, with unrestricted access to every service and resource. That's why using it for daily tasks is a huge security risk—if it gets compromised, it's game over. That's why the first step in hardening AWS security is locking down the root account and using IAM users for everyday tasks.

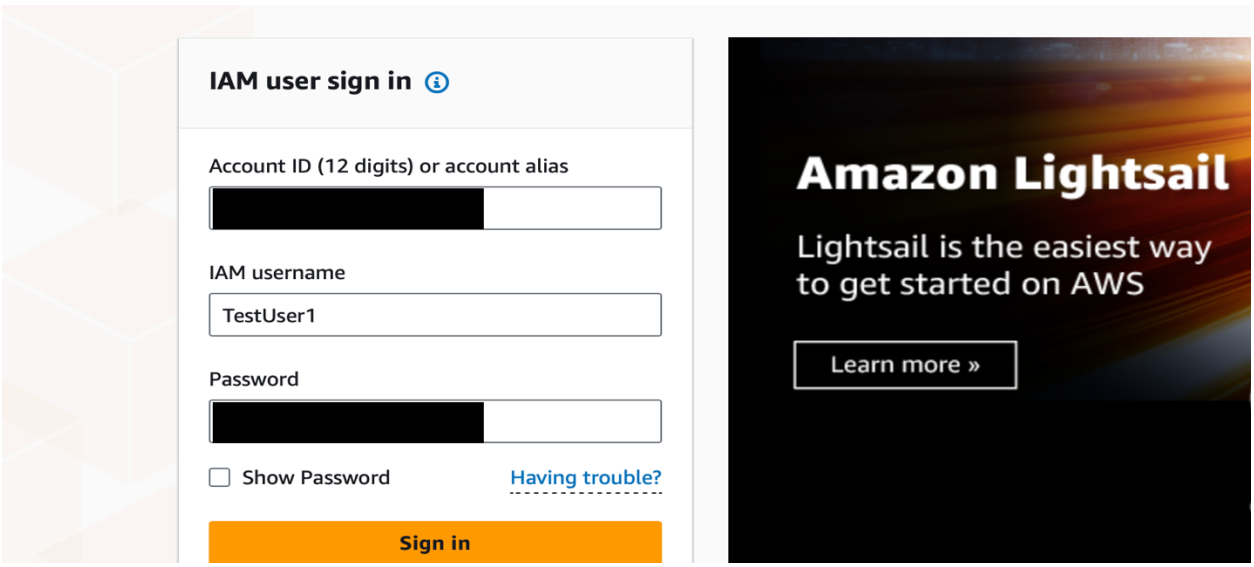
As part of this project, I enabled MFA (Multi-Factor Authentication) for the root user, adding an extra layer of protection in case someone ever got hold of the root password. It's also crucial to store root account details, especially keys and credentials, in a secure, encrypted vault. This ensures that the information stays protected and doesn't fall into the wrong hands.



Setting IAM Users

I created three test users to be able to explore different IAM scenarios: TestUser1, TestUser2, and TestUser3. I gave each user AWS Management Console access, so they could log in via the GUI. I also chose autogenerated passwords and enforced a password reset on first login, which is a small but important security step to ensure that every user sets a unique, strong password from the start.

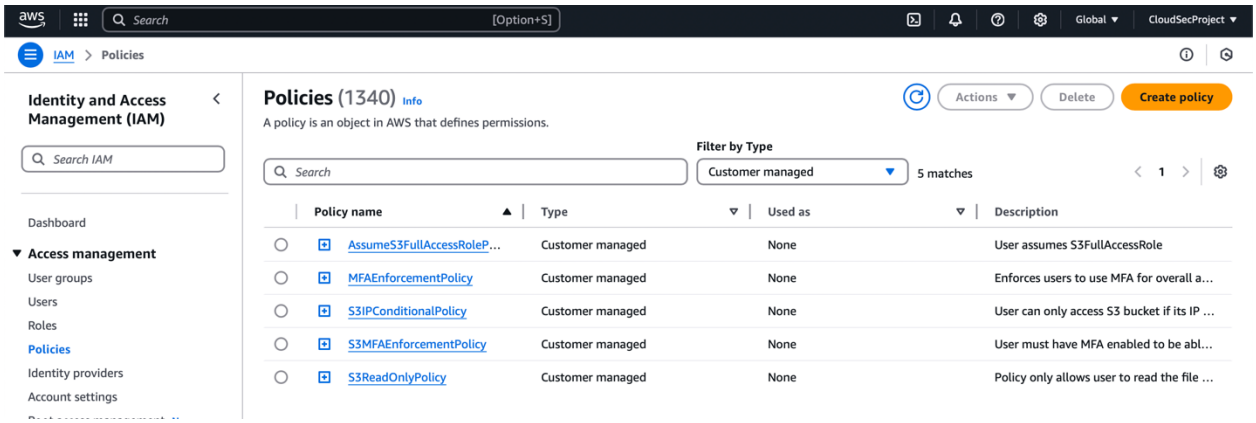




This process reflects a common security approach in real organizations and helps reduce the risk of using default or weak passwords.

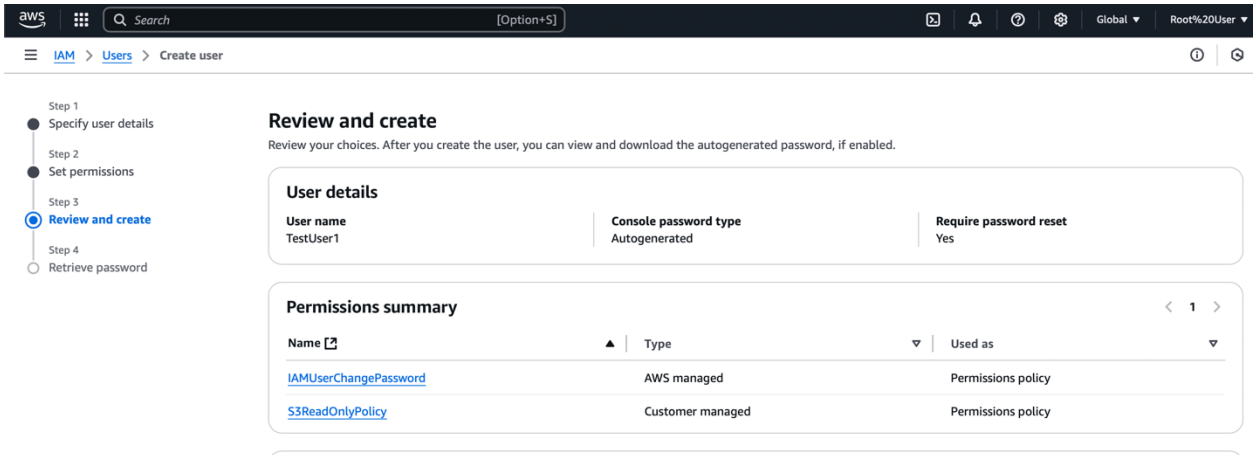
Implementing Least Privilege

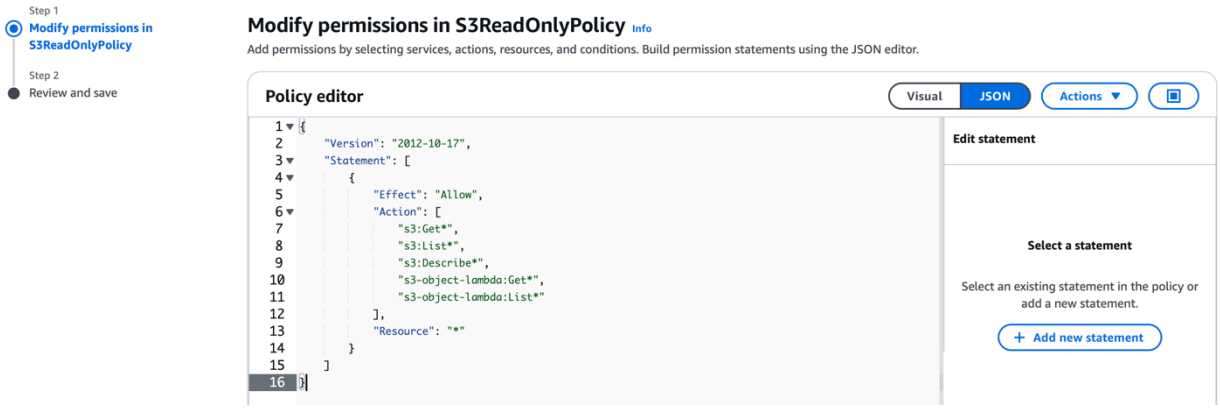
The Least Privilege Principle is all about making sure users only have the minimum permissions they need to do their jobs and nothing more, nothing less. It's one of the most basic but powerful security concepts since it limits the organization's attack surface.



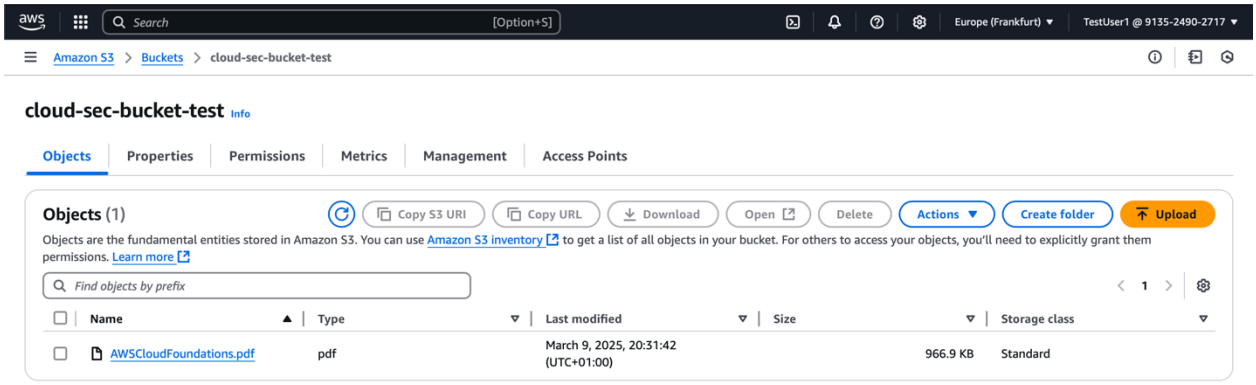
User 1 - ReadOnly

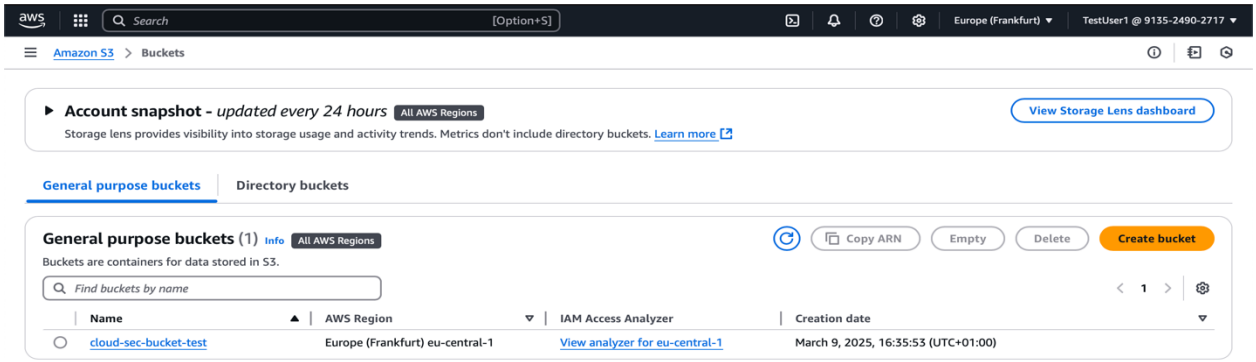
For *TestUser1*, I created a policy that allowed only read and list permissions on the *cloud-sec-project-test-bucket*. This meant they couldn't upload, delete, or modify objects in the bucket.

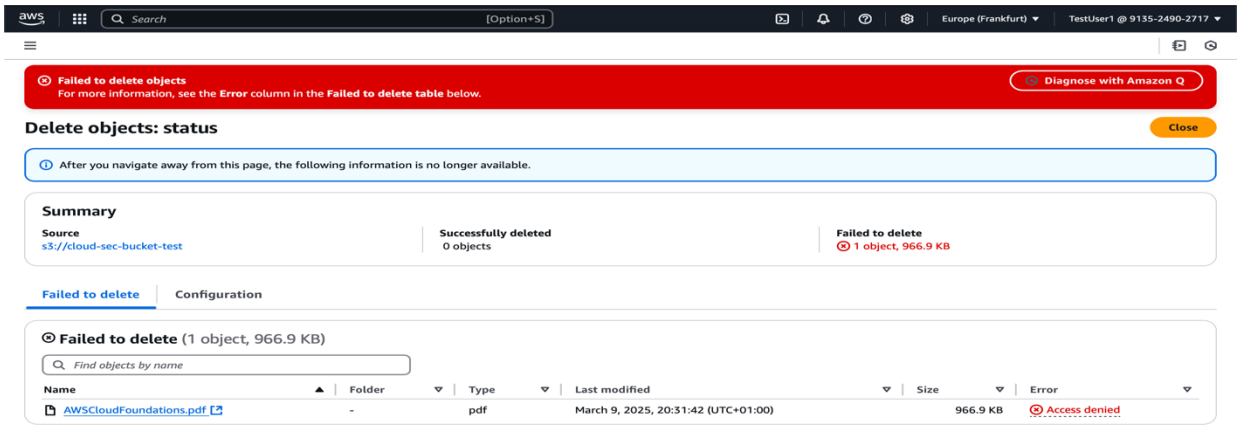


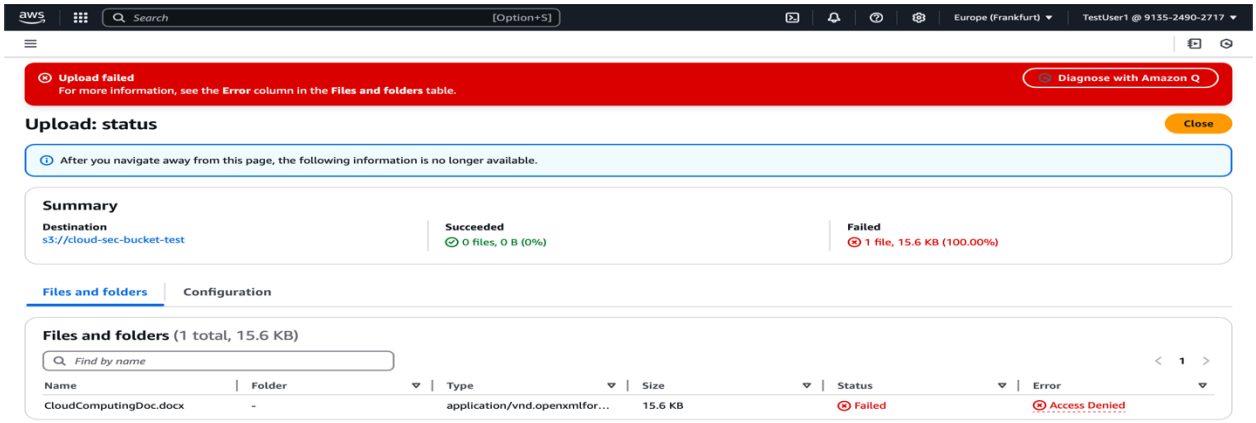


To test this, I tried deleting a file and uploading a new one. In the test case images below you can see how both actions were successfully blocked by the policy, showing that it was working as intended.



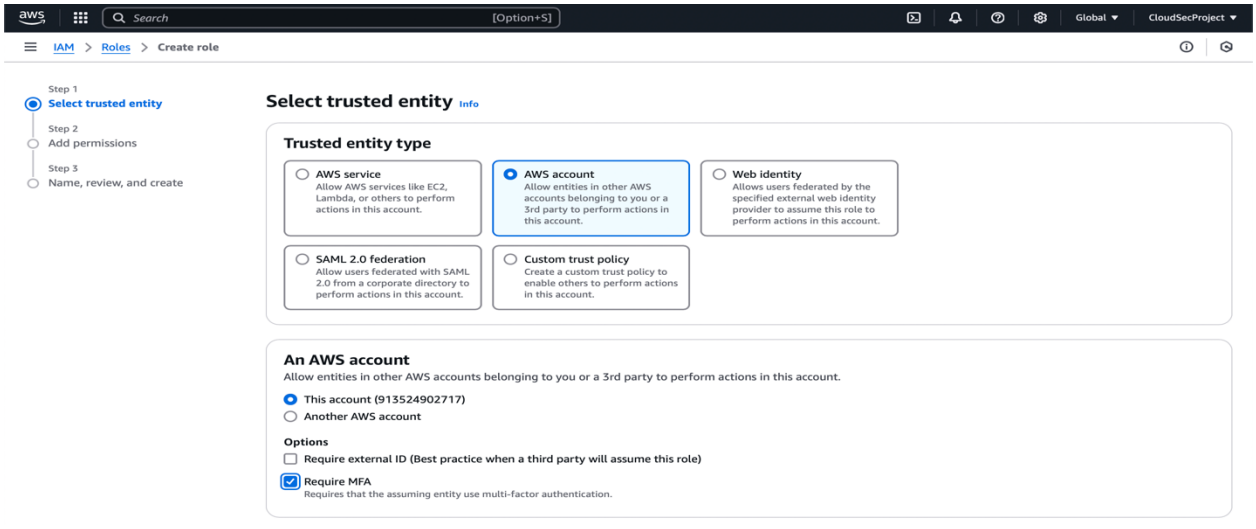


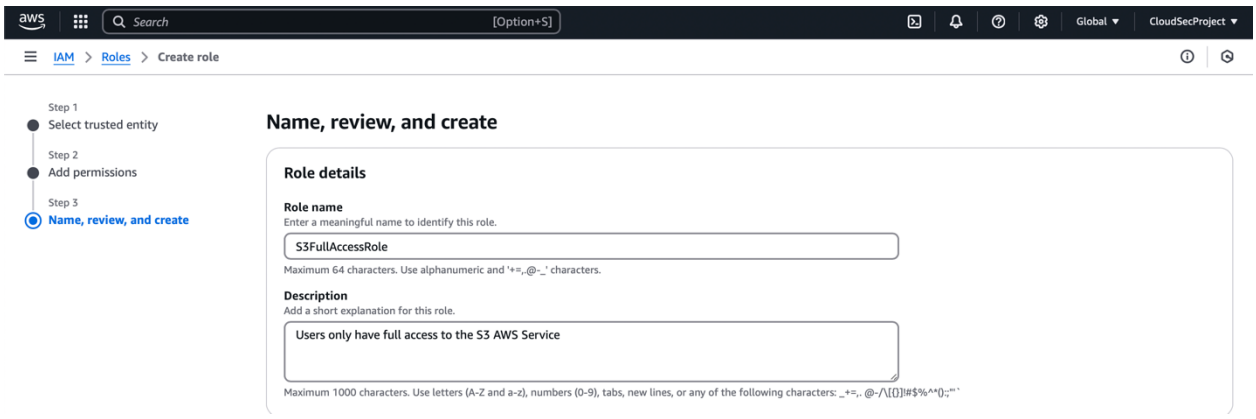


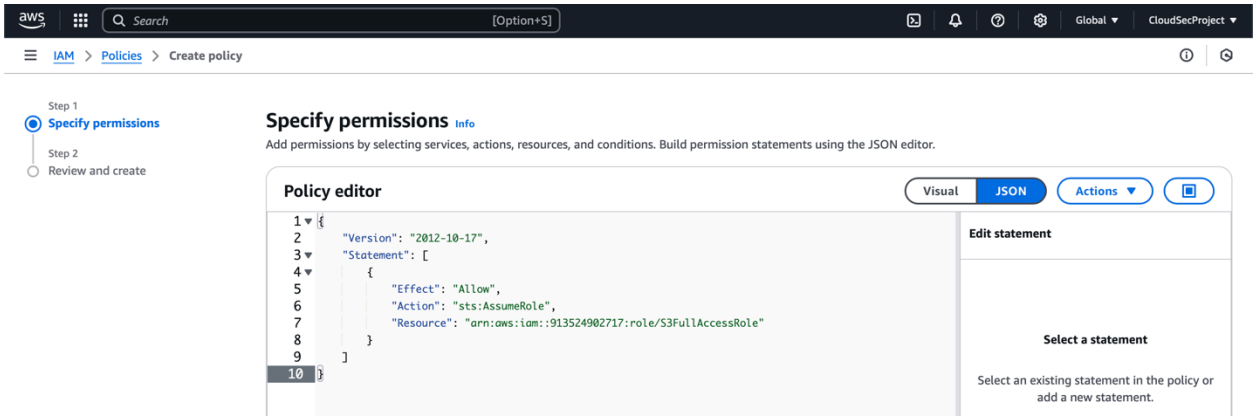


User 2 – Role-Based Access

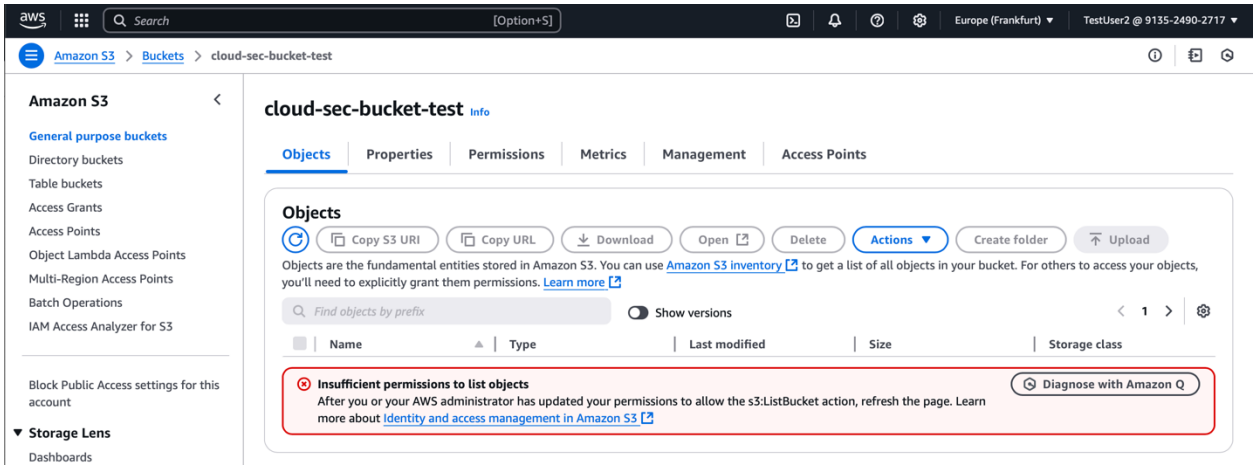
For *TestUser2*, I took a role-based approach. First, I created a new IAM role and attached the AmazonS3FullAccess policy to it, granting full access to the S3 bucket. However, by default, users can't assume roles unless explicitly allowed. So, I also created a custom inline policy for *TestUser2*, written in JSON, that specifically allowed them to assume the newly created role.

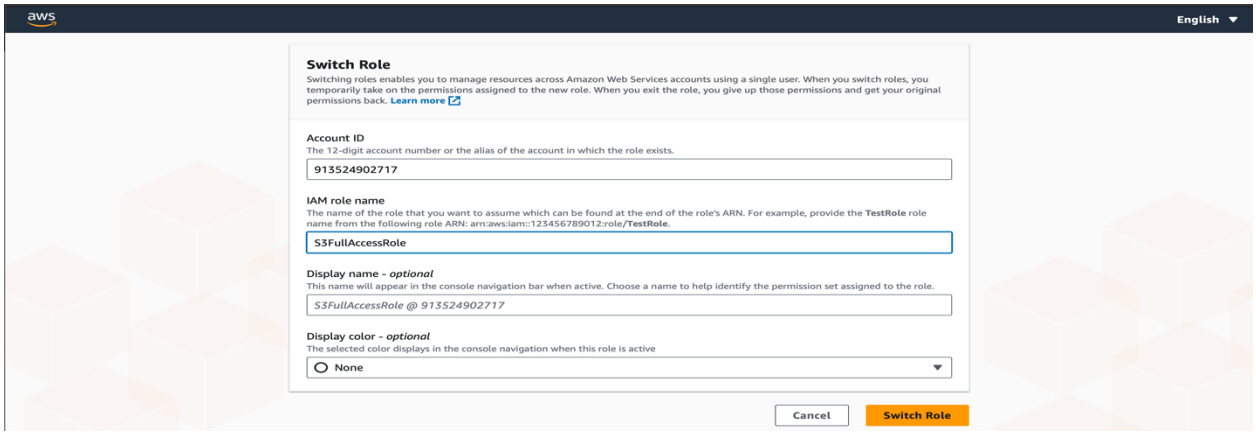


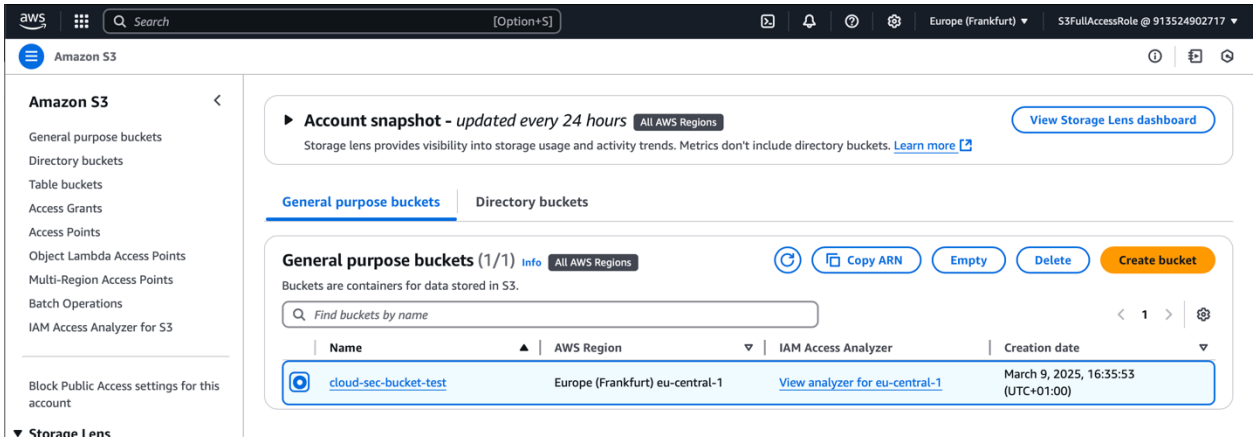




When testing, I logged in as *TestUser2* and noticed that I couldn't access the S3 bucket until I switched roles in the AWS Console. This is done by selecting the "Switch Role" option in the top-right corner of the AWS Management Console and entering the role's details. Once the role was assumed, TestUser2 gained full access to the S3 bucket and could upload, modify, and delete files as expected.



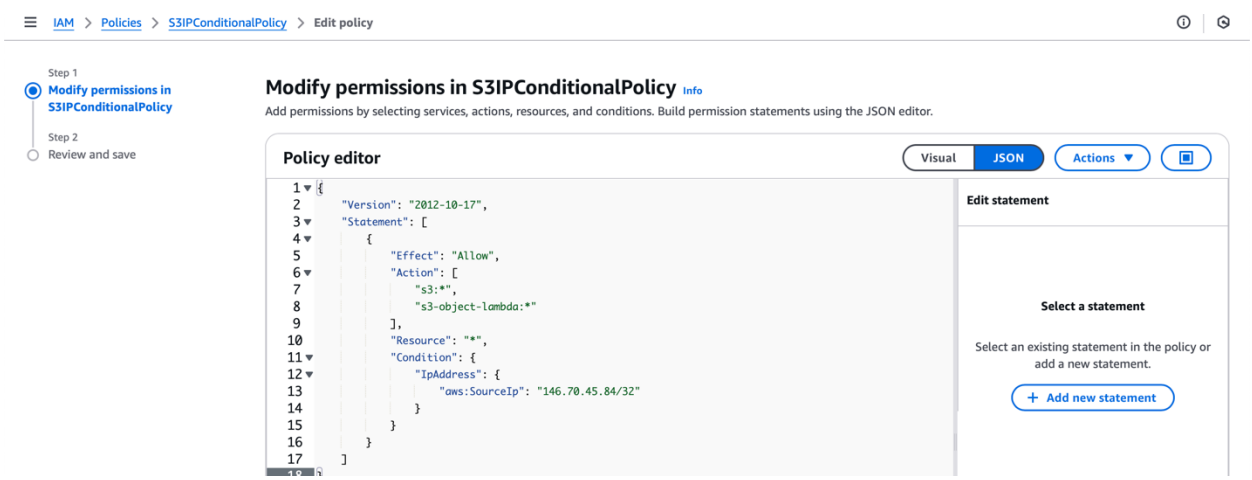




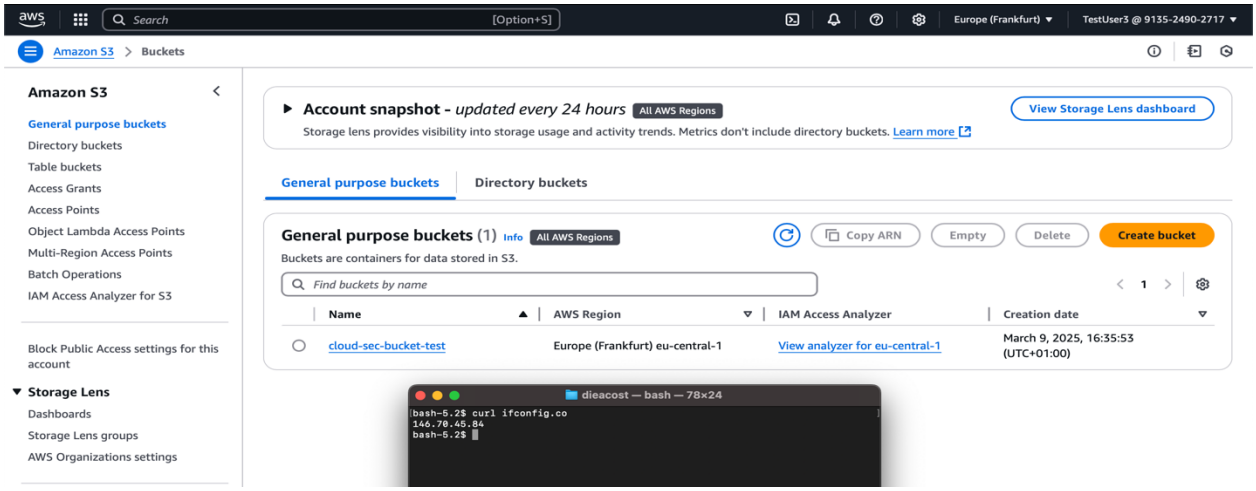
This process highlighted the security benefit of role-based access control (RBAC), where users only gain specific permissions when they actively assume a role, reducing the risk of unnecessary or accidental access.

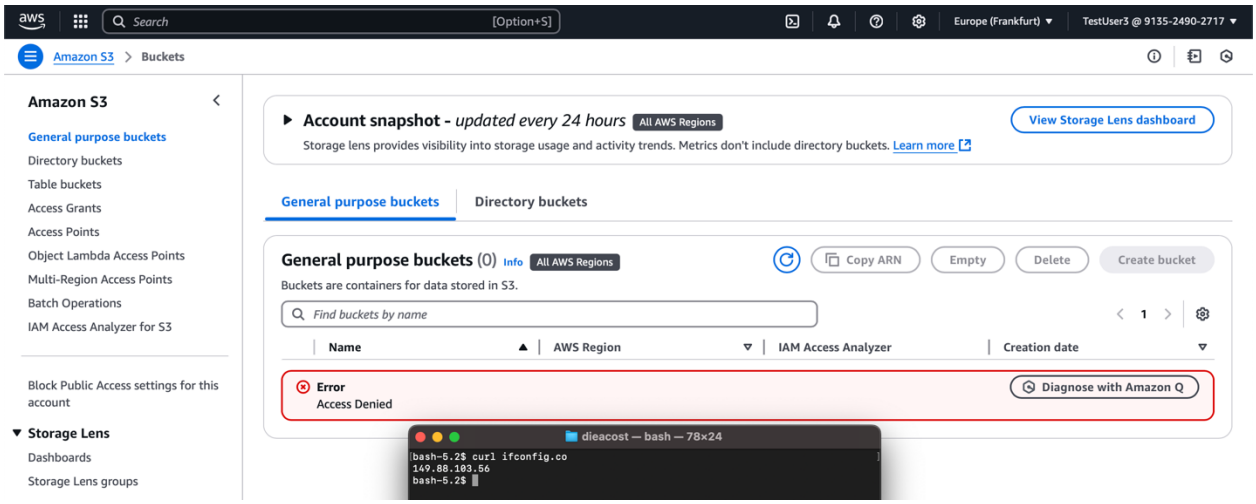
User 3 – Conditional Access

Finally, for *TestUser3* I created the most specific policy of them all. I created a policy only allowing a specific IP address to access the S3 bucket. With this approach I wanted to simulatet a scenario where a company only wants its employees to access the organization's resources from a specific location (a specific public IP address) in this case the office's public IP address.



For this demonstration I used an encrypted VPN so the address shown aren't related to my location at all. In the following images you can see how I tried accessing the S3 services with the allowed IP address and it worked just fine. However, when I tried accessing the S3 bucket with a different IP address, you can see how it appears as access denied or insufficient permissions).





This showed how powerful and precise IAM policies can be, especially when it comes to restricting access based on location. • Explanation of the JSON policy structure

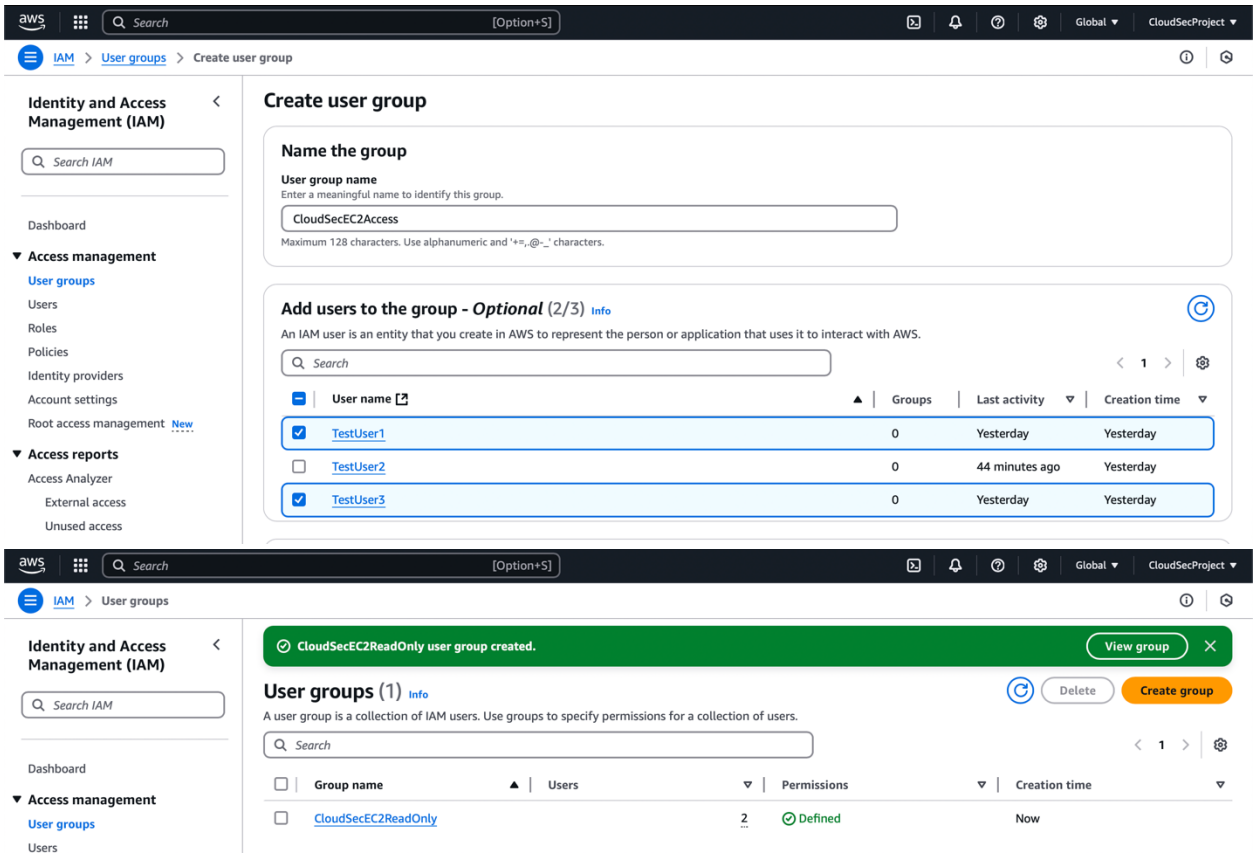
Setting Groups

Now, while assigning policies individually works fine with a few users, it quickly became clear how inefficient that approach would be in a larger organization. Managing permissions one by one is fine for a small setup, but if you're dealing with hundreds or thousands of endpoints, it's just not scalable.

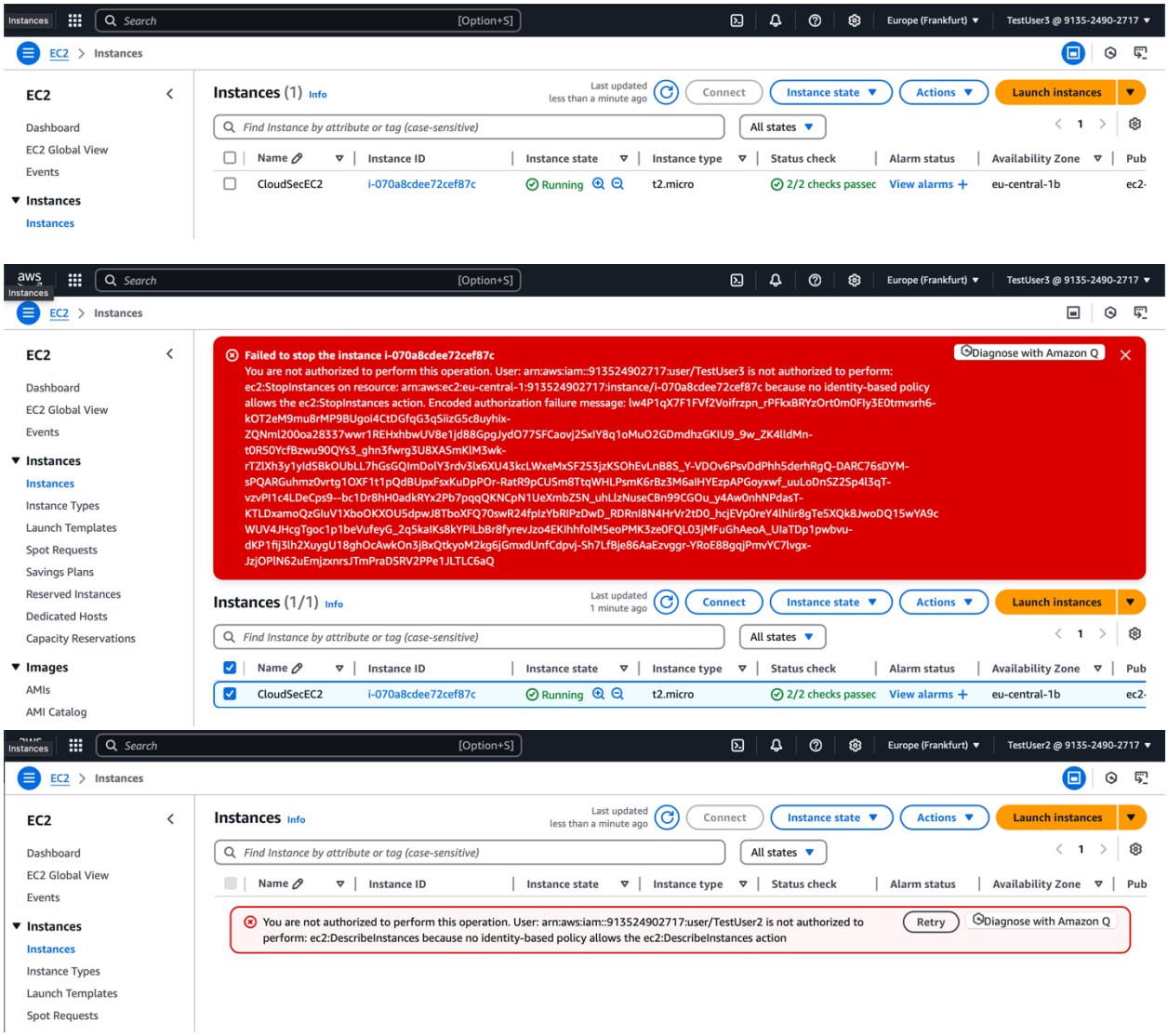
To demonstrate a better solution, I created an IAM user group called *CloudSecEC2ReadOnly*. I used the root account to set it up in the IAM Access Management section and attached a policy that only allowed users in the group to view

the *CloudSecEC2* instance. They couldn't stop, modify, or terminate it, but only view its details.

I added *TestUser1* and *TestUser3* to the group. During testing, *TestUser3* could successfully view the EC2 instance but was blocked from stopping it, exactly as the policy intended. When I switched to *TestUser2*, they couldn't even see the instance, since they weren't part of the group.



In the following images below you can see how I tried accessing the EC2 instance with TestUser3 and was successful. However, when I tried stopping the instance, the read only policy was working as intended by blocking the stopping attempt. On the other hand, when trying to access the AWS EC2 service with TestUser2 It wouldn't even let it find any instances given that TestUser isn't within the "CloudSecEC2ReadOnly" users group.



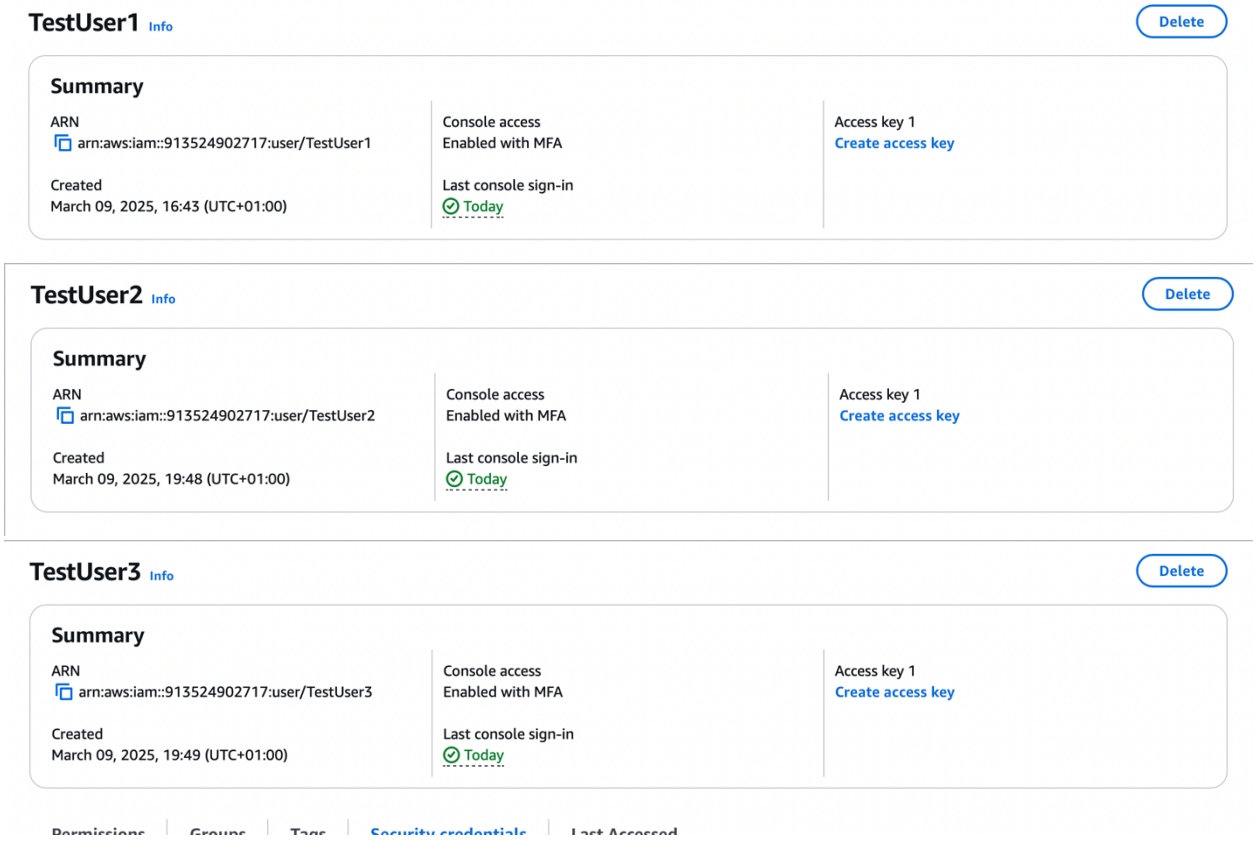
This test really highlighted the efficiency and power of group-based permissions. It's faster, simpler, and much easier to manage, especially when you're dealing with a large number of users.

Implementing MFA

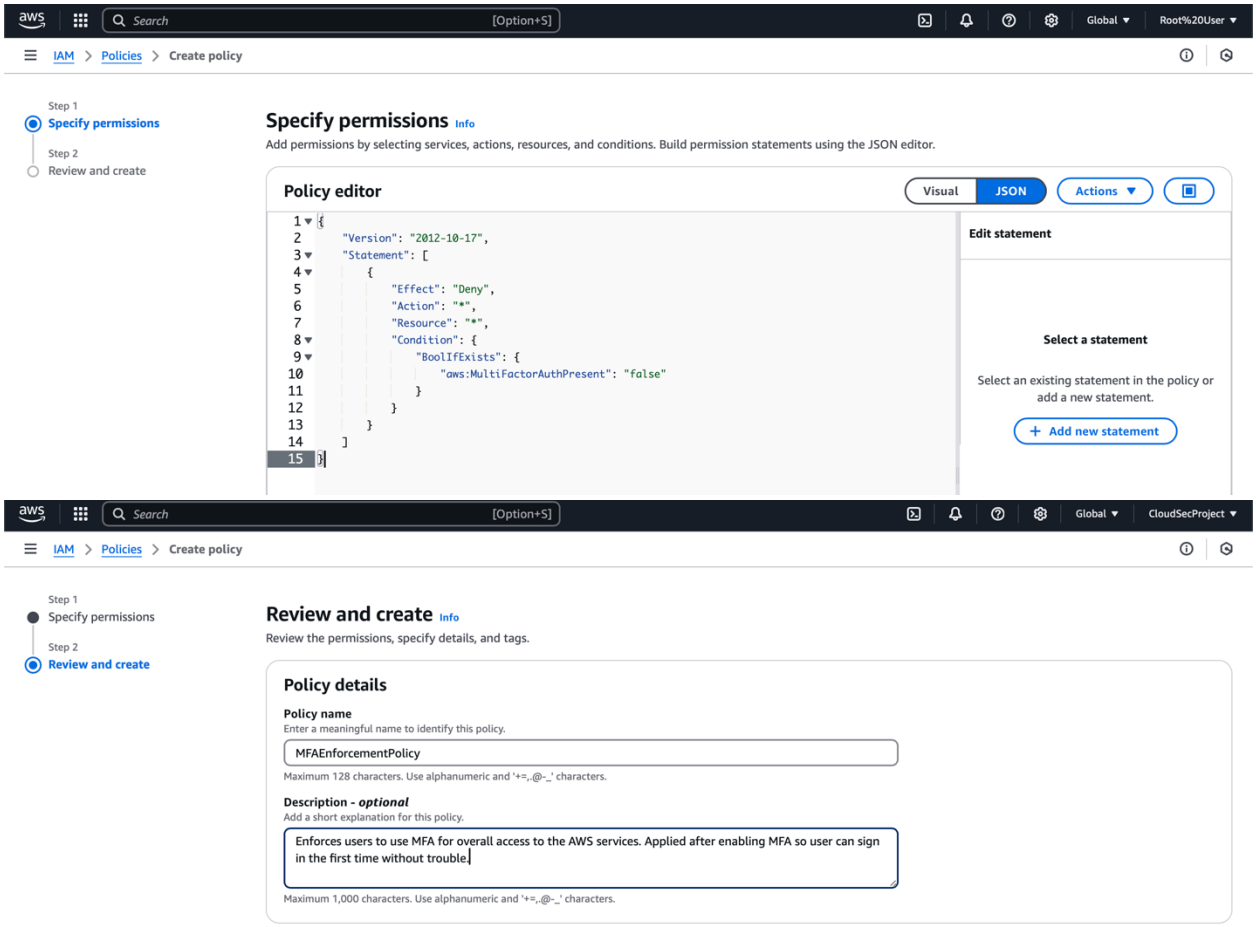
MFA (Multi-Factor Authentication) is one of the simplest and most effective ways to protect user accounts. It is about having multiple authentication methods (something you know, something you have, something you are, something you do, etc). Without it, if a user's

password is compromised, an attacker can easily get in. But with MFA, even if someone steals the password, they'd still need a second factor like a code from an authenticator app to access the account.

That's why I made sure to enable MFA for all IAM users. I used the authenticator app option, which in this case is a "something you have" authentication method because its a code, and registered all accounts to a central device for easier management during this demonstration. However, in real-world scenarios, each user should register their own MFA device to ensure individual responsibility and security. Organizations also tend to use corporate identity management systems to track, enforce, and manage MFA devices across teams.



After setting up MFA, I created a custom policy that required users to authenticate with MFA to access AWS services. Timing is crucial here because this policy must be attached after the first sign-in and MFA setup, or users could get locked out of their accounts in a sort of catch-22 situation.



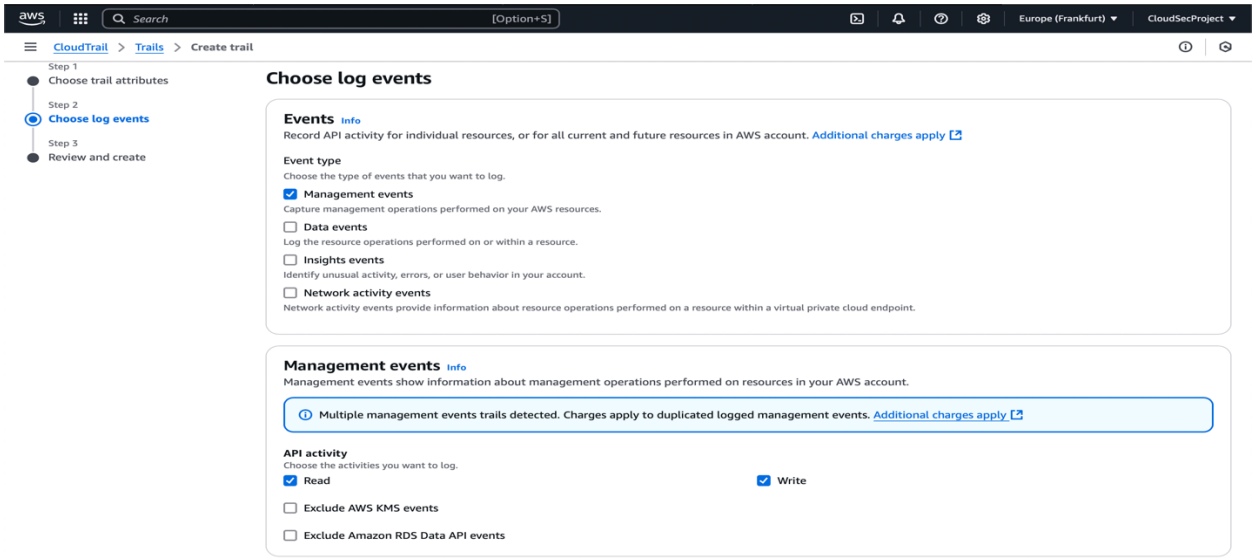
This enforcement ensures that MFA stays in place for the entire lifespan of the IAM user, minimizing the risk of it being disabled and keeping accounts secure by default.

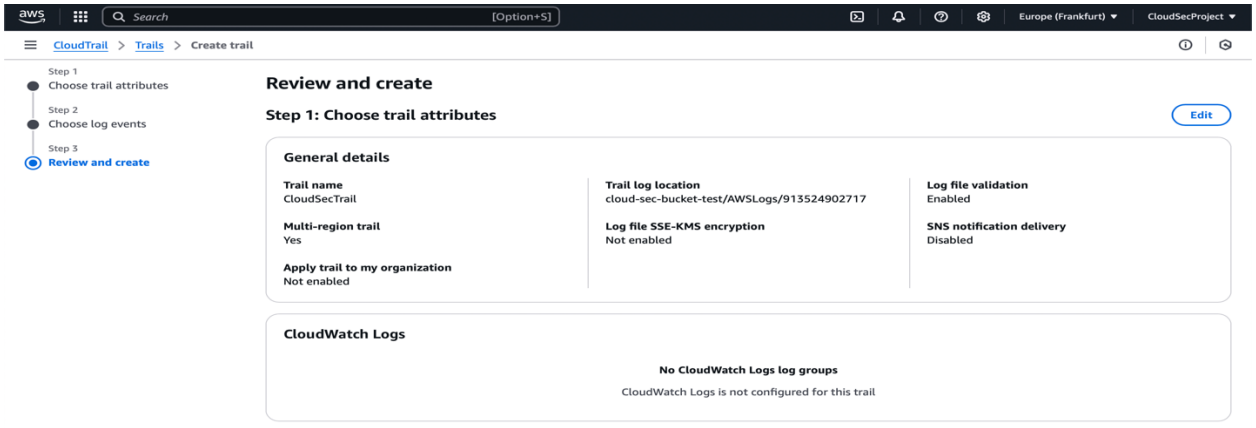
Setting Up CloudTrail Monitoring

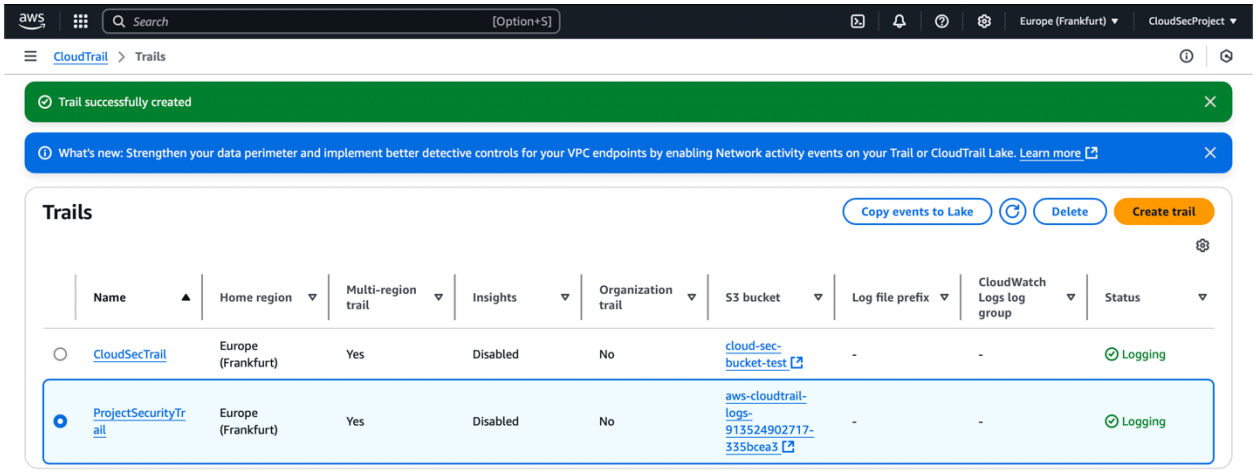
Now, we all know that accounting and log aggregation not only are part of cybersecurity's best practices but also in some cases necessary for compliance and auditing, as well making the work of the Incident Response andn Digital Forensics team easier. That's where AWS CloudTrail comes in. CloudTrail is a powerful tool for monitoring and logging activity across AWS services. It records who did what, when, and from where.

In this project, I set up a trail called *CloudSecProjectTrail.* The setup was straightforward, but I kept it as simple as possible to avoid additional costs. I enabled **SSE-S3** encryption (which is default and ensures logs are securely stored) and Log File Validation, which

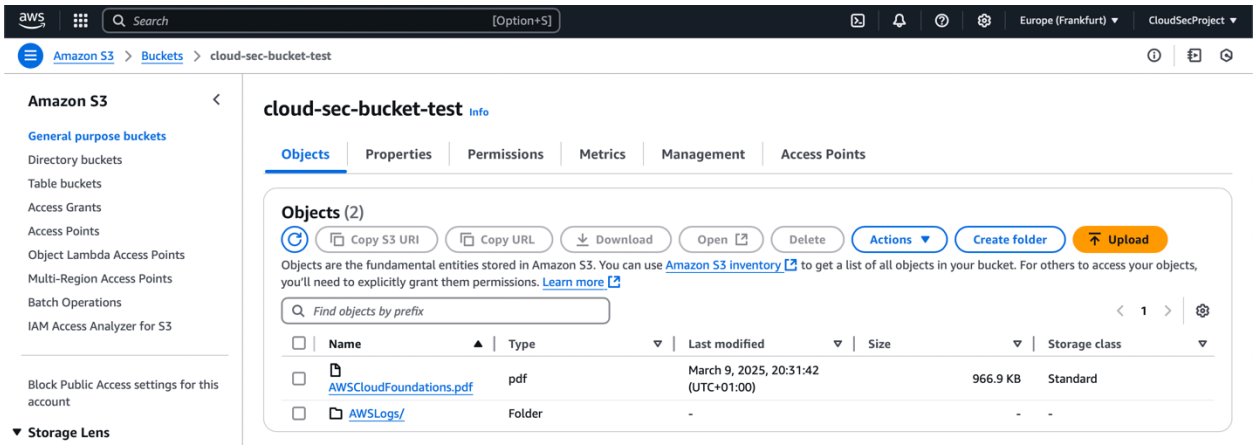
uses cryptographic hashes to ensure that logs haven't been tampered with. This is critical for security audits and forensic investigations.







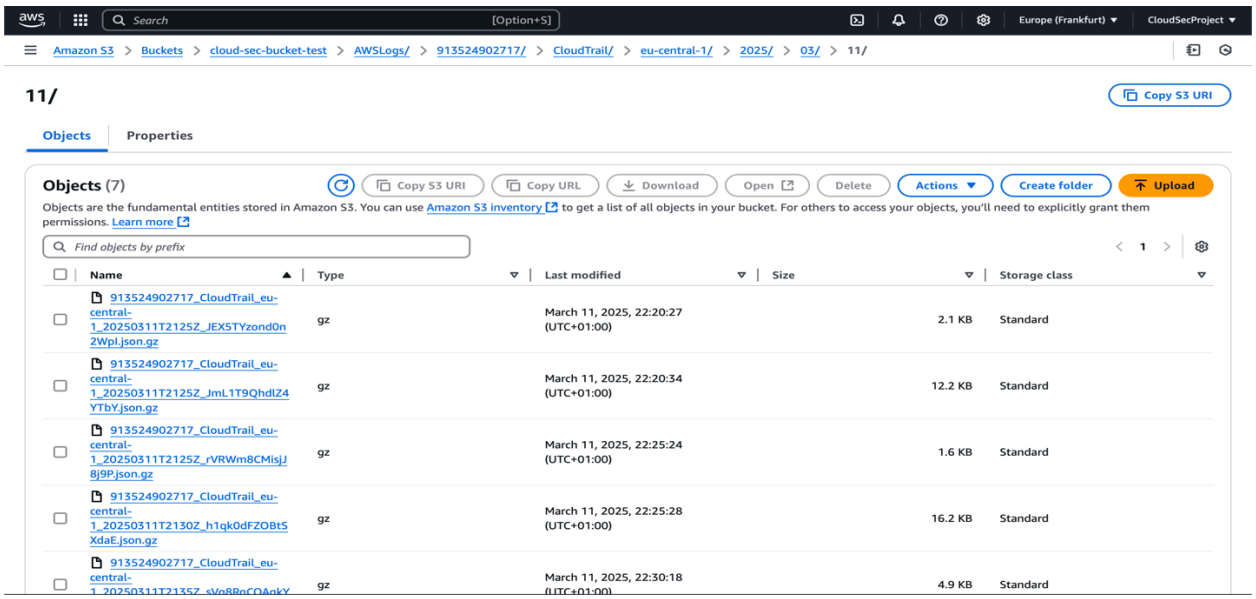
To finish the trail creation, I checked the log all management events, and linked the previously created S3 Bucket "*cloud-sec-project-test-bucket*" for the log files to be added once something has happened.



In the following images you can see how I carried out different test cases with the root account and test users by:

- Changing two IAM policies
- Creating a new EC2 instance
- Uploading new files to the S3 bucket
- Terminating the EC2 instance

CloudTrail captured all these activities and saved them as JSON files in the S3 bucket, which you can then download as .zip. These logs included details like who performed the action, when it happened, and from which IP address.



While AWS allows for setting up SNS alerts for real-time notifications, I didn't configure that due to budget constraints. However, in a real-world setup, this would be an essential step for real-time threat detection. Organizations can also set up personalized alarms to specific devices, emails, and people with SNS, however, this may imply additional costs which are beyond my budget as a student.

Lessons Learned

Many people think that just learning the theory of a subject is enough, but that's far from reality. There's a saying that goes, "You never stop learning because life never stops teaching." This project really proved that to me.

I learned a lot, especially about cloud security and IAM best practices. While I understood the concepts before, actually implementing them gave me a much deeper appreciation of how small misconfigurations can lead to major security risks. I also realized just how crucial the Least Privilege Principle is in real-world security.

I wouldn't say anything was extremely difficult, but I underestimated the time it would take. Setting up IAM users and switching between accounts slowed things down, but the hardest part was definitely creating and modifying IAM policies. Writing JSON-based policies requires absolute precision, and even a small mistake can result in unintended permissions. Since this was new to me, I had to research a lot and troubleshoot multiple times before getting everything to work, but once I did, it was incredibly rewarding.

Despite everything, I loved every second of this project. Cybersecurity is my passion, and projects like this remind me why. There's always something new to learn, and every challenge is an opportunity to grow. This was just another step in my journey, and I can't wait to take on even bigger challenges in the future.

Conclusion

This project reinforced the importance of IAM hardening, the Least Privilege Model, and MFA enforcement in securing cloud environments. Proper IAM management ensures that users and roles only have access to what they truly need, reducing the risk of accidental or malicious actions. Enforcing MFA adds an essential layer of security, making it much harder for attackers to gain access, even if credentials are compromised.

In the real world, these principles are critical. Large organizations like banks and healthcare providers must enforce strict IAM policies to protect sensitive financial and medical data from unauthorized access. Cloud service providers like AWS, Google Cloud, and Microsoft Azure implement Least Privilege to prevent users from making unintended system-wide changes.

Without proper IAM security, breaches can have devastating consequences. Incidents like the Capital One data breach (where misconfigured IAM roles led to a major security incident) highlight why enforcing Least Privilege and MFA isn't just best practice—it's essential for protecting critical data and infrastructure.

By applying these principles, organizations can significantly reduce security risks, prevent data breaches, and ensure compliance with industry regulations, making IAM a fundamental pillar of modern cybersecurity.