Dean Moses

deanmdt@gmail.com

Abstract

This project demonstrates implementation of AWS Identity and Access Management (IAM) best practices for secure cloud access control. The documentation outlines the complete setup and configuration of IAM users, groups, policies, roles, and security measures following AWS recommended practices.

AWs IAM

Identity & Access Management

Contents

[AWS Identity & Access Management (IAM) Implementation Project 1](#_Toc191563565)

[Project Overview 1](#_Toc191563566)

[Business Context 1](#_Toc191563567)

[Implementation Details 1](#_Toc191563568)

[1. IAM Setup and User Management 1](#_Toc191563569)

[2. Policy Configuration and Testing 2](#_Toc191563570)

[3. Security Enhancements 2](#_Toc191563571)

[4. Programmatic Access Configuration 3](#_Toc191563572)

[5. Role-Based Access for Services 3](#_Toc191563573)

[6. Access Auditing and Monitoring 3](#_Toc191563574)

[Technical Skills Demonstrated 4](#_Toc191563575)

[Security Benefits Achieved 4](#_Toc191563576)

[Next Steps and Recommendations 4](#_Toc191563577)

# AWS Identity & Access Management (IAM) Implementation Project

## Project Overview

This project demonstrates implementation of AWS Identity and Access Management (IAM) best practices for secure cloud access control. The documentation outlines the complete setup and configuration of IAM users, groups, policies, roles, and security measures following AWS recommended practices.

## Business Context

Organizations migrating to AWS need secure and well-managed access controls to protect cloud resources. This project addresses this need by establishing a robust identity management framework that:

1. Follows the principle of least privilege
2. Implements multi-factor authentication
3. Separates root account from daily operations
4. Provides audit capabilities for access activities
5. Creates secure role-based access for services

## Implementation Details

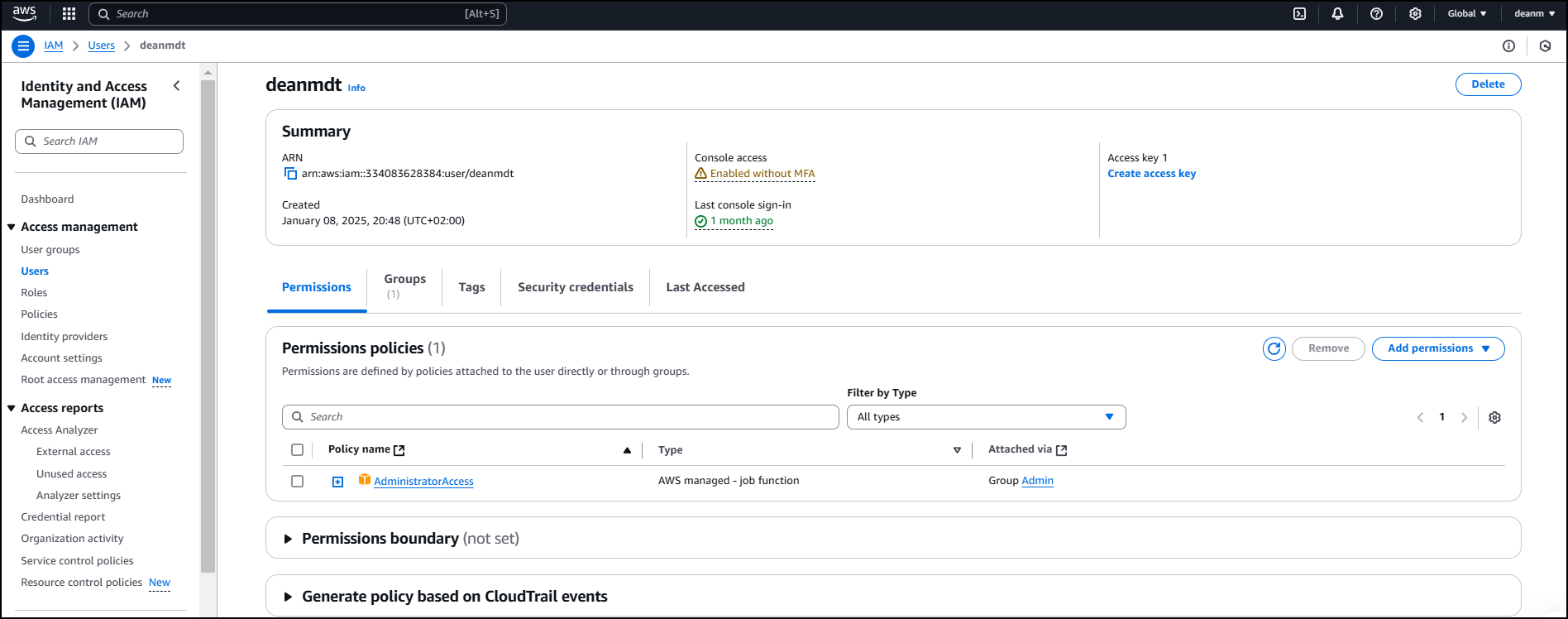
### 1. IAM Setup and User Management

I first accessed the IAM console by searching “IAM” in the AWS search bar. An important observation at this stage was confirming that IAM is a global service, indicated by the disabled region selector in the console’s top-right corner. This means IAM configurations apply across all AWS regions.

#### Creating an Administrative User

To follow security best practices, I created an administrative user instead of using the root account:

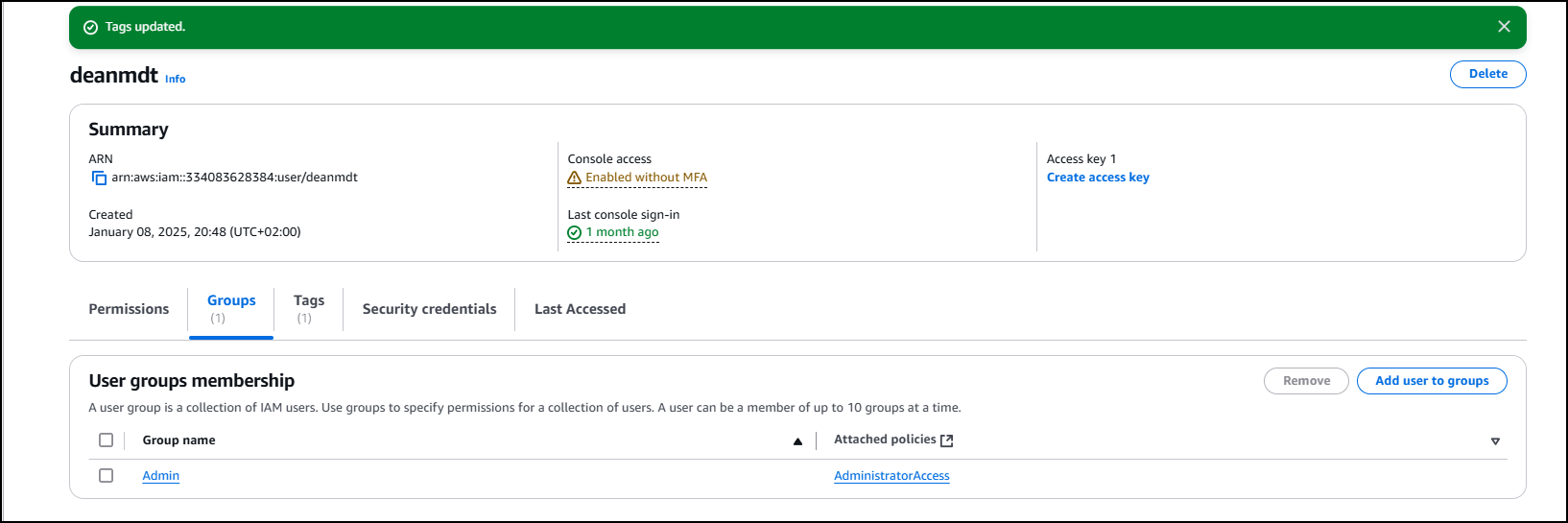
1. Navigated to Users → Create user
2. Named the user “deanmdt” and enabled management console access
3. Selected “Create IAM user” rather than Identity Center integration
4. Set a custom password and chose to not require password change at next sign-in

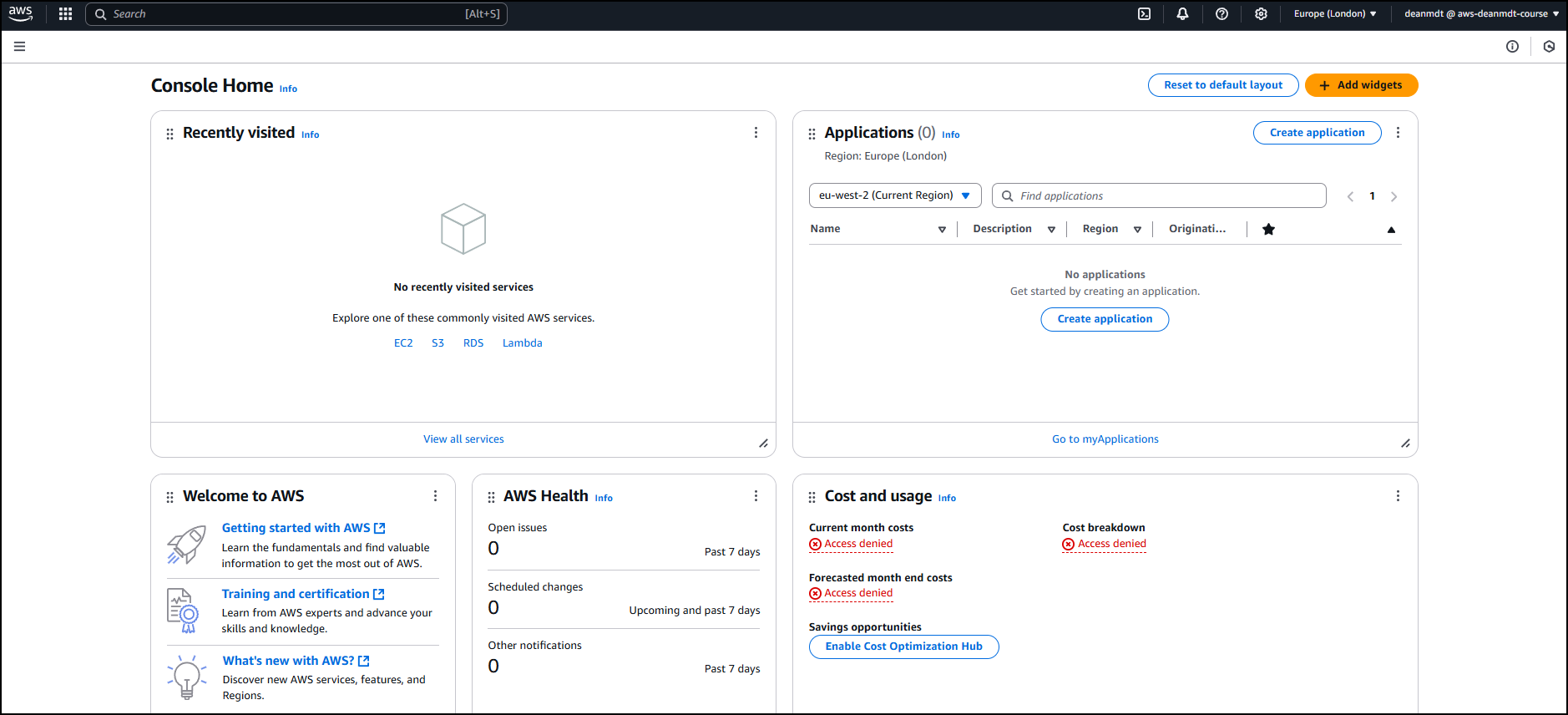


#### Group-Based Permission Management

To implement a scalable permission structure:

1. Created an “admin” group with AdministratorAccess policy
2. Added the “deanmdt” user to this group
3. Added a tag to the user (Department: Engineering) to demonstrate resource metadata tagging



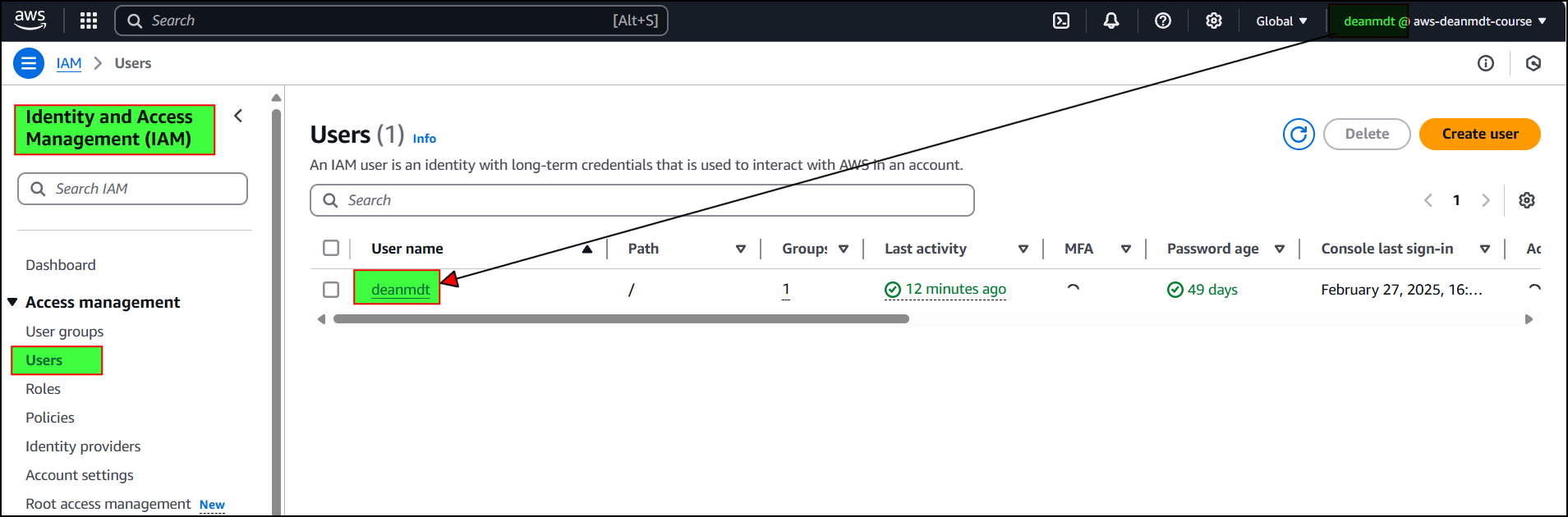


This approach demonstrates the preferred method of permission inheritance through groups rather than direct policy attachment, which simplifies access management as the organization grows.

### 2. Policy Configuration and Testing

To demonstrate policy implementation and validation:

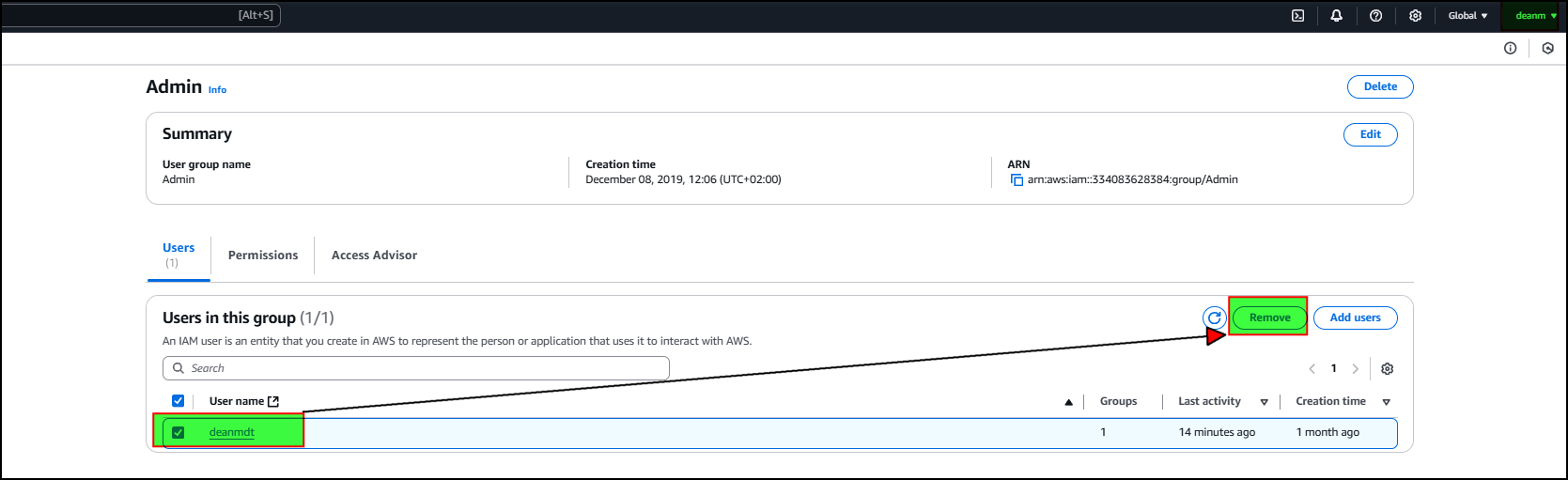
1. Removed the user from the admin group temporarily:

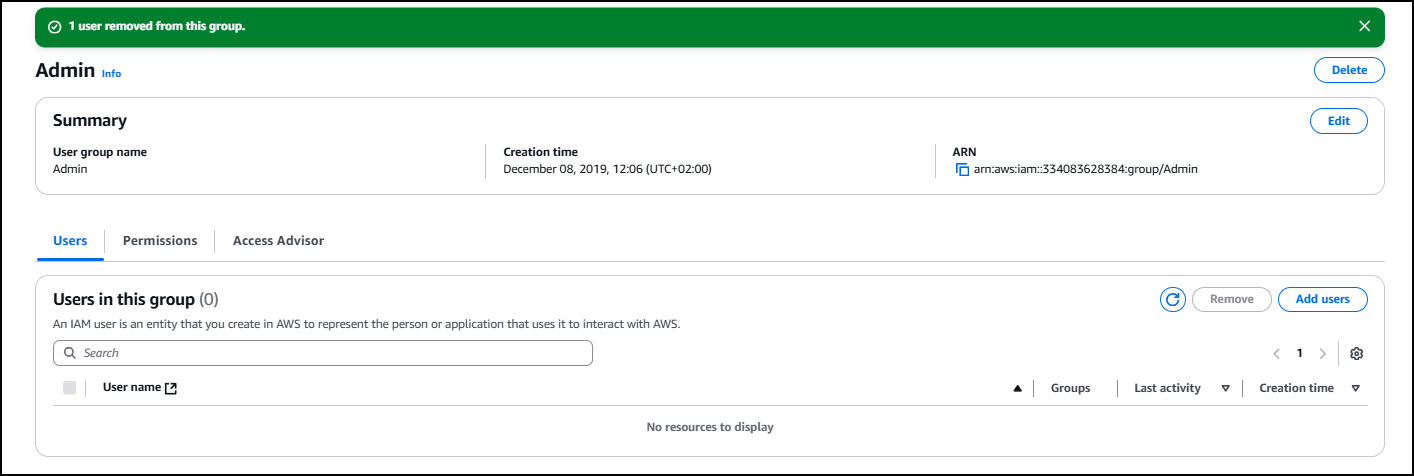


||

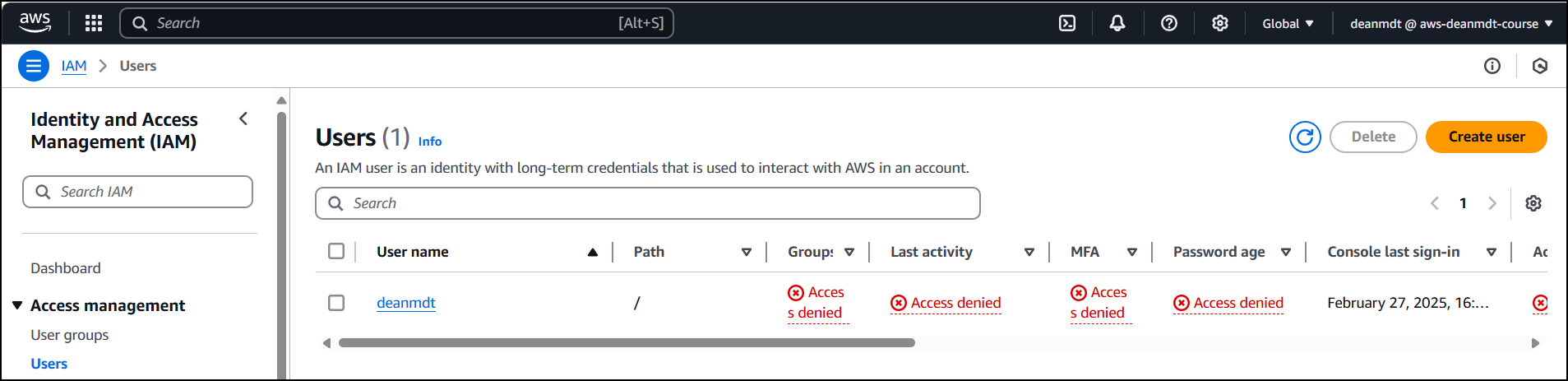
||

V

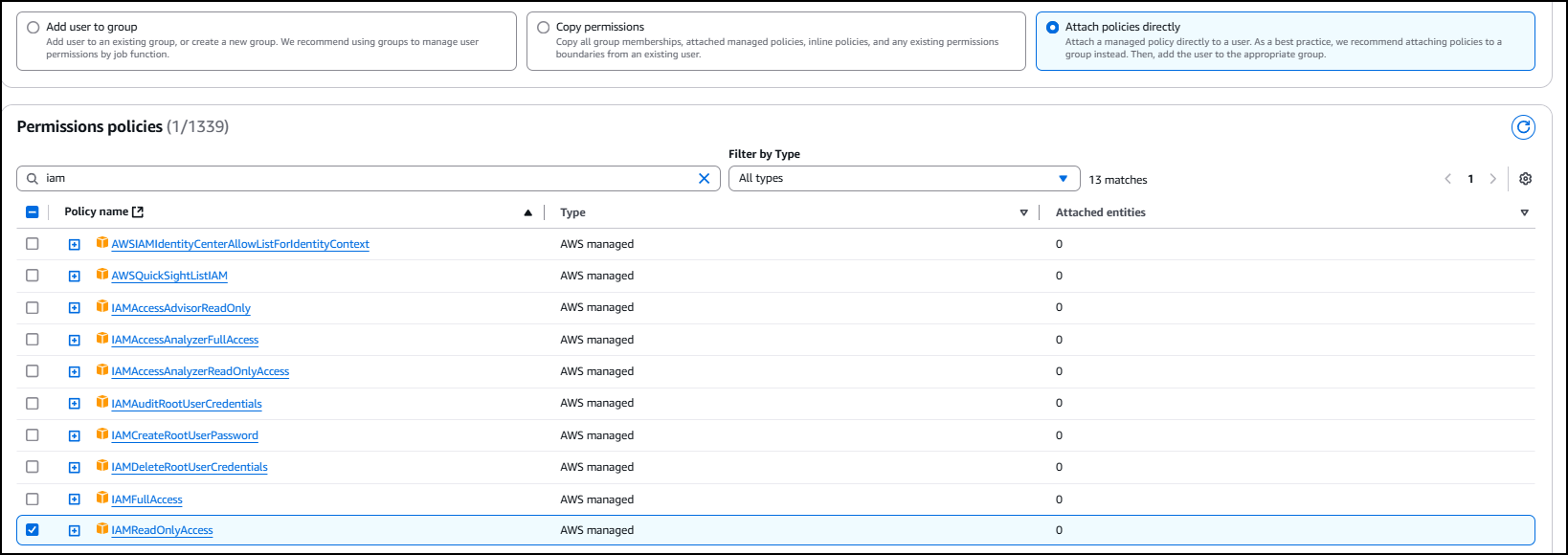




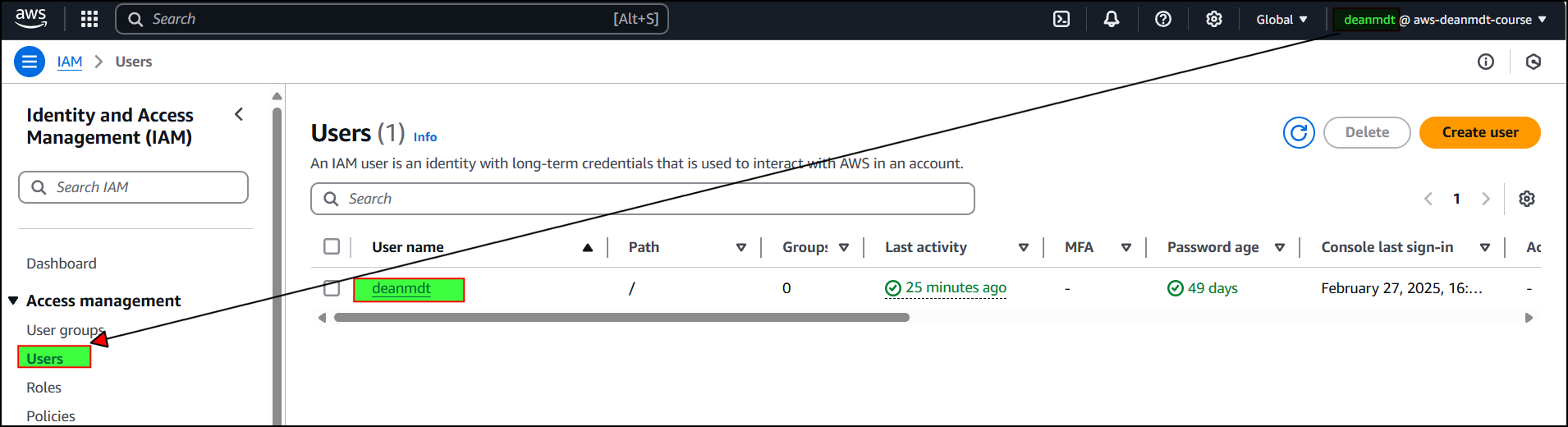
1. Verified that access was denied when attempting to list users

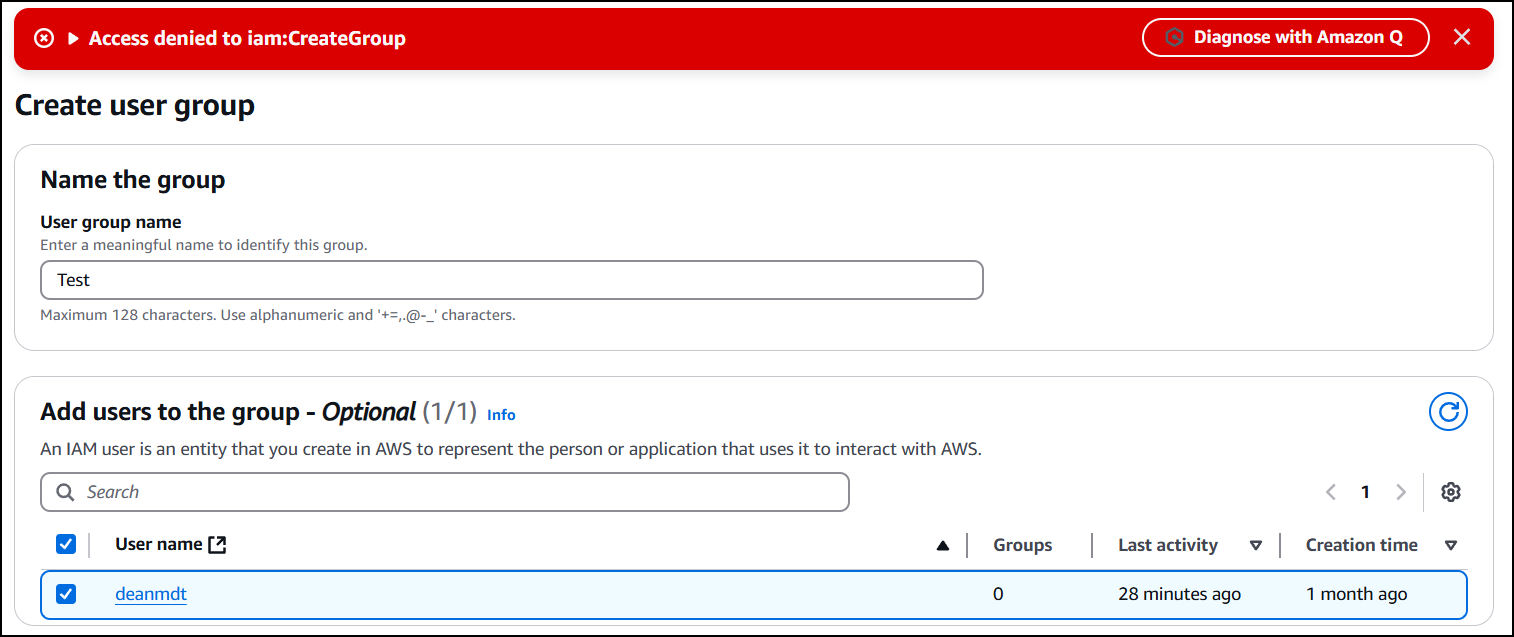


1. Attached the IAMReadOnlyAccess policy directly to the user



1. Verified that read operations succeeded but write operations (creating a group) failed





This test confirms that IAM policies properly restrict user actions based on assigned permissions.

#### Policy Structure Analysis

I examined the JSON structure of different policies:

1. **AdministratorAccess**: Contains "Action": "\*" and "Resource": "\*" to allow all actions on all resources
2. **IAMReadOnlyAccess**: Contains specific read actions like Get\* and List\* patterns to allow only viewing resources

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action": [

"iam:GenerateCredentialReport",

"iam:GenerateServiceLastAccessedDetails",

"iam:Get\*",

"iam:List\*",

"iam:SimulateCustomPolicy",

"iam:SimulatePrincipalPolicy"

],

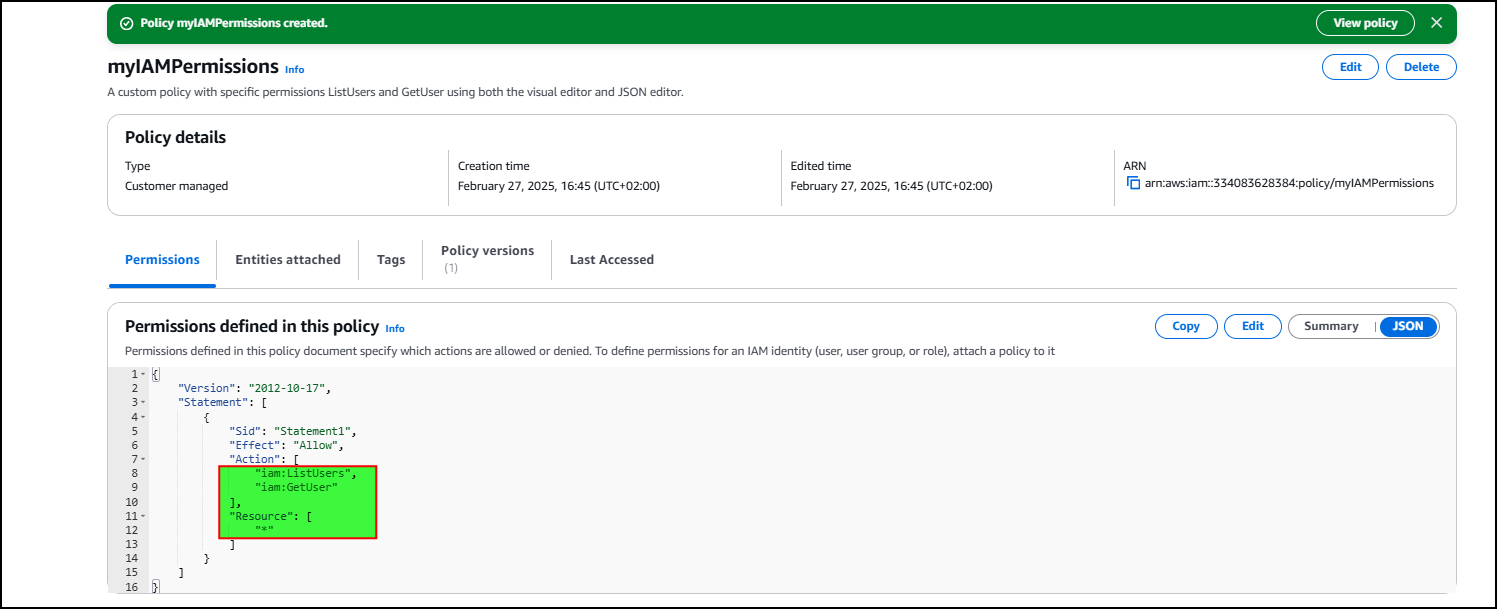
"Resource": "\*"

}

]

}

I also demonstrated creating a custom policy with specific permissions (ListUsers and GetUser) using both the visual editor and JSON editor.



### 3. Security Enhancements

#### Password Policy Implementation

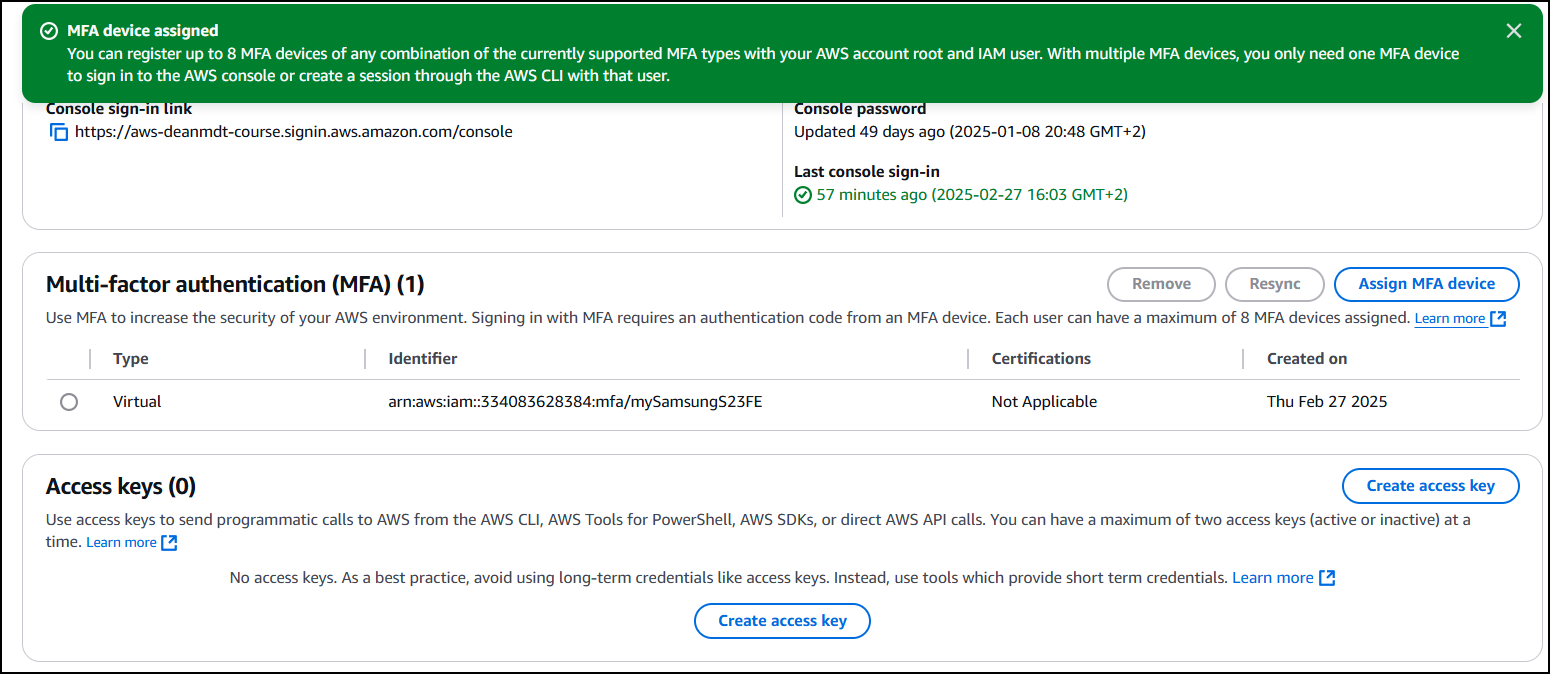
I configured a custom password policy with the following settings:

1. Minimum password length
2. Required character types (uppercase, lowercase, numbers, non-alphanumeric)
3. Password expiration period (90 days)
4. Prevention of password reuse

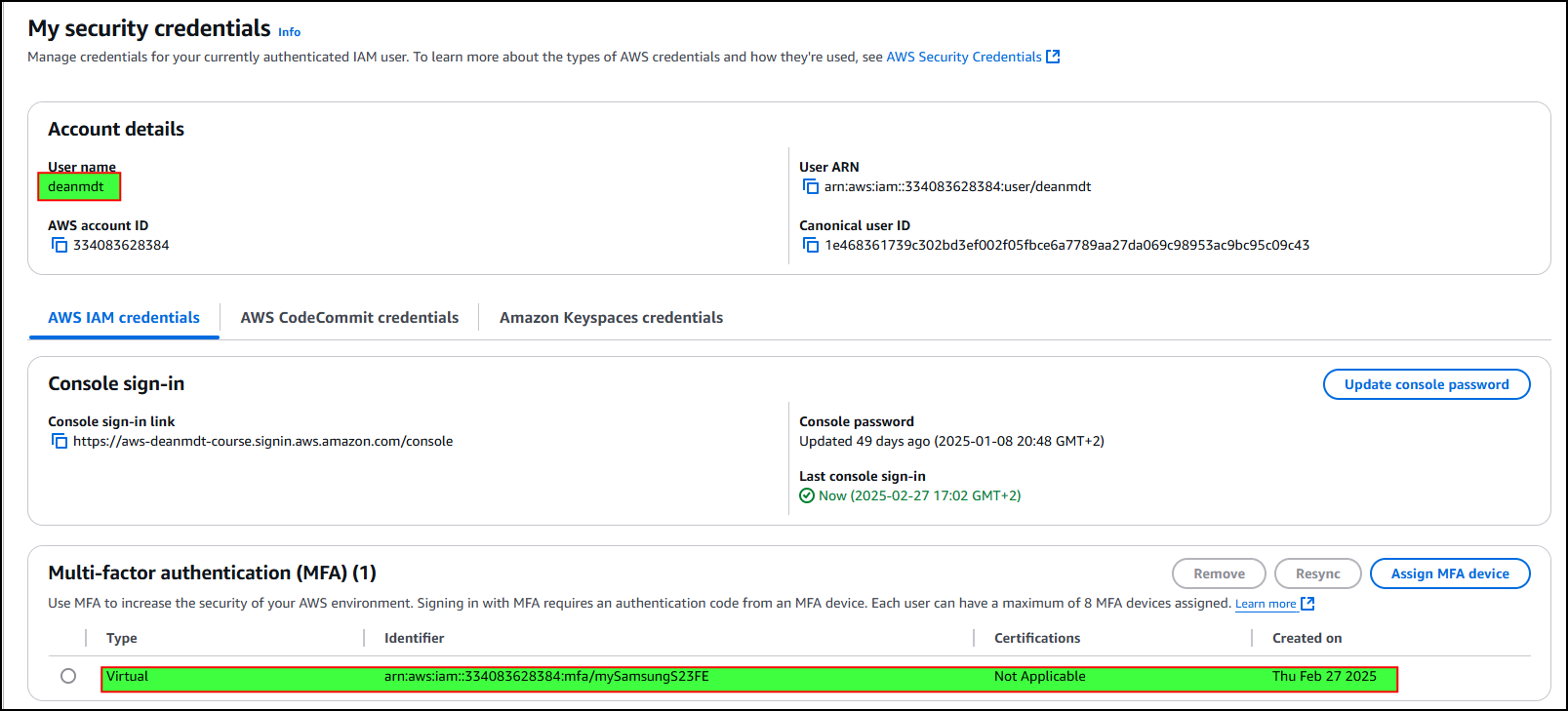
#### Multi-Factor Authentication (MFA)

I configured MFA for the root account by:

1. Accessing Security Credentials from the account menu
2. Setting up MFA using an authenticator app (Google Authenticator)
3. Scanning the QR code and providing two consecutive authentication codes
4. Verifying the MFA setup through the login process



Logged out and then back in using my admin user credentials and MFA:

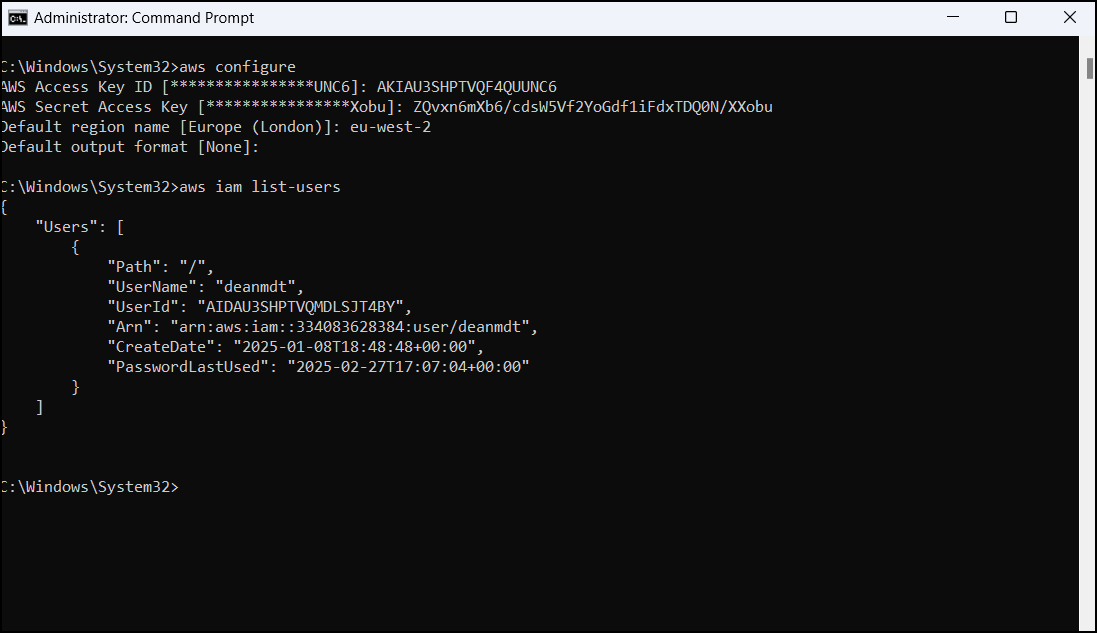


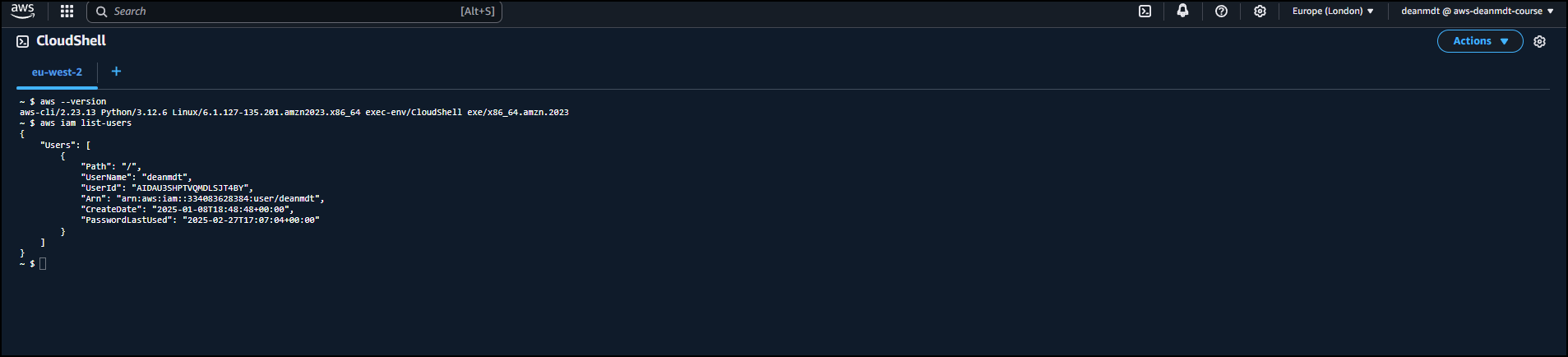
This demonstrates implementation of AWS’s strongest account protection recommendations.

### 4. Programmatic Access Configuration

To enable automation and CLI access:

1. Created access keys for the IAM user
2. Configured the AWS CLI using aws configure with the access key, secret key, and default region
3. Tested the configuration by running aws iam list-users



CloudShell: 

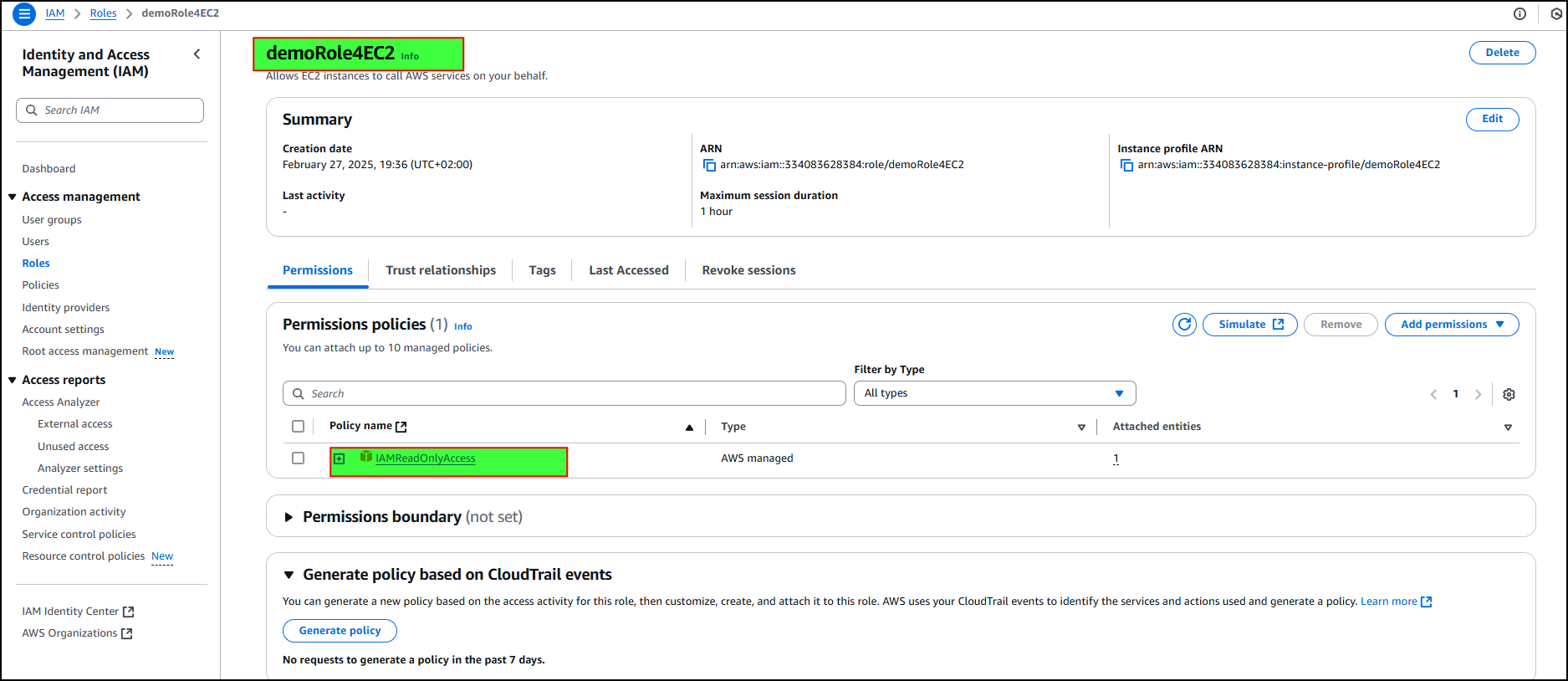
1. Verified that permission changes affected CLI access the same way they affected console access

This implementation allows for secure programmatic access to AWS resources through properly credentialed IAM identities.

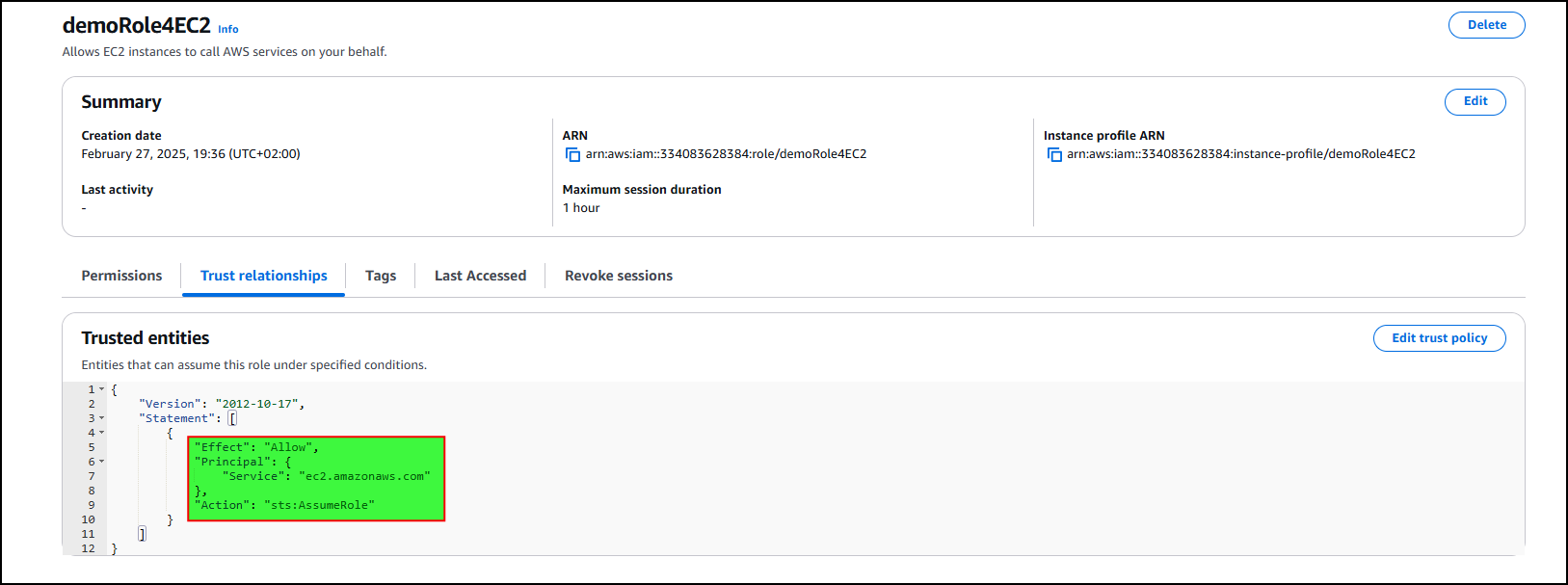
### 5. Role-Based Access for Services

I created a service role to demonstrate how AWS services can securely access other services:

1. Created a role for EC2 service
2. Attached the IAMReadOnlyAccess policy to this role



1. Verified the trust relationship configured to allow EC2 instances to assume this role



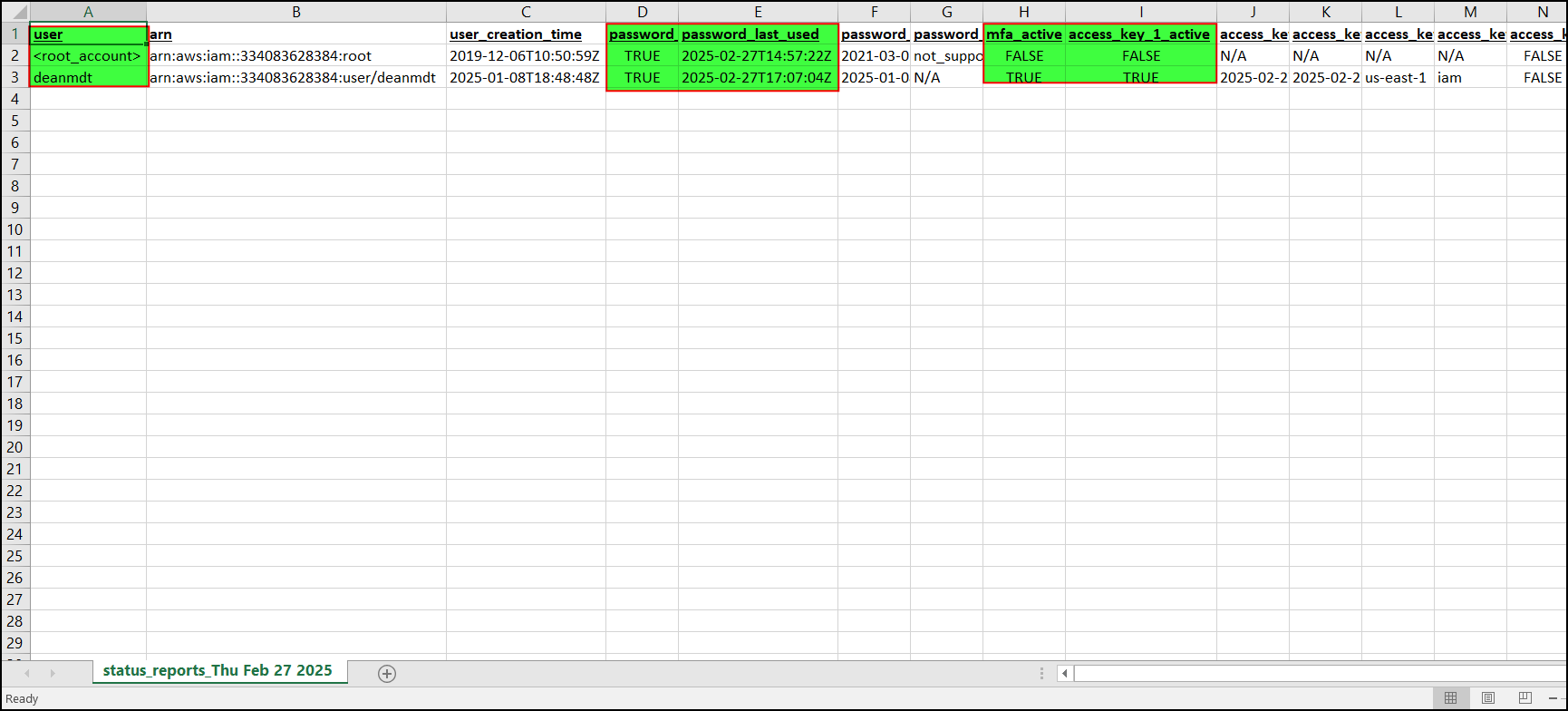
This implementation establishes a secure method for allowing AWS services to interact with each other without using long-term credentials.

### 6. Access Auditing and Monitoring

#### Credential Reports

I generated and analyzed a credential report that provided:

1. Password and access key usage details
2. MFA status for all users
3. Password last used and rotation information



#### Access Advisor

I used IAM Access Advisor to analyze service access patterns:

1. Reviewed which services the IAM user had accessed
2. Identified the policies that granted this access
3. Noted unused service permissions that could be removed to follow least privilege

This demonstrates implementation of AWS’s auditing tools to maintain security compliance and optimize permissions.

## Technical Skills Demonstrated

This project showcases competency in:

1. AWS IAM architecture and implementation
2. Security best practices for cloud environments
3. Policy configuration and testing
4. Authentication enhancement (MFA)
5. CLI configuration and programmatic access
6. Role-based access control
7. Security auditing and monitoring

## Security Benefits Achieved

The implementation provides:

1. Protection against root account compromise
2. Granular access control through custom policies
3. Streamlined permission management through groups
4. Enhanced authentication security with MFA
5. Secure service-to-service communication
6. Audit capabilities for compliance and security review

## Next Steps and Recommendations

For a production environment, I would recommend:

1. Implementing a regular security review process using credential reports
2. Setting up CloudTrail for comprehensive activity logging
3. Creating additional role separation based on job functions
4. Establishing a formal process for access key rotation
5. Integrating with a corporate identity provider using IAM Identity Center

This IAM implementation establishes a solid foundation for a secure AWS environment by ensuring proper identity management, access control, and security monitoring.