Maria Valencia

Csc 154

Lab 13

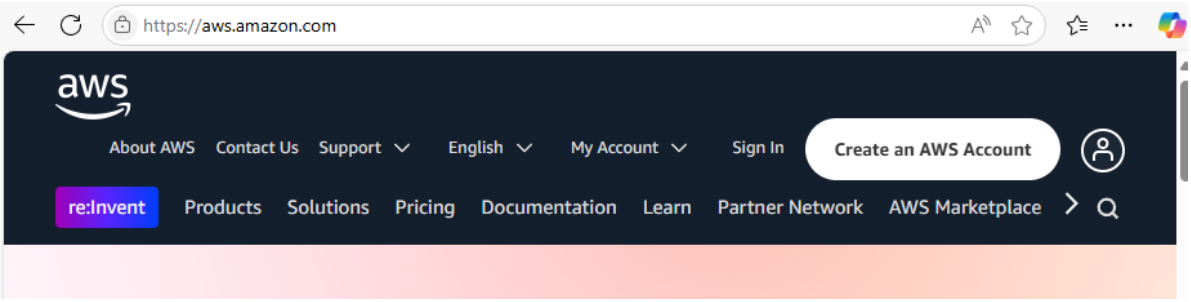
Cloud Security

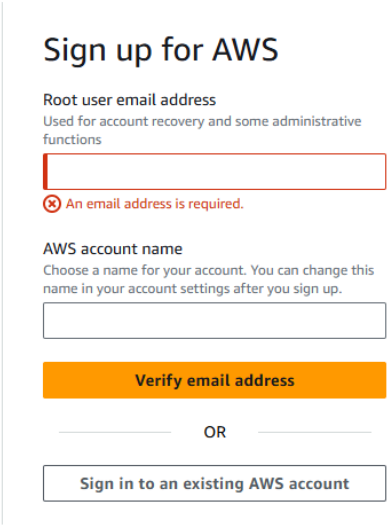
Exercise 13.1 - Create and Setup AWS Account

In this task I will create an AWS account, secure the root user, and create an IAM administrator.

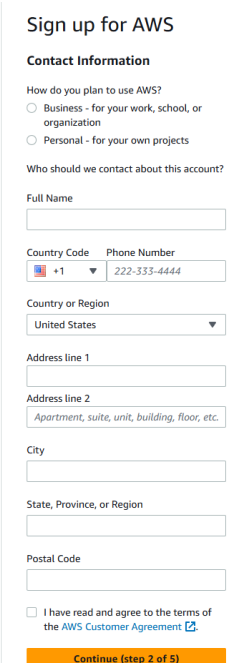
Step 1: Create AWS Account

From my host computer, I opened a browser and navigated to https://aws.amazon.com/. I pressed the "Create an AWS Account" button in the top right corner, entered my email address for the "Root user email address" and entered my name under the "AWS account name". Lastly, pressed "Verify email address".

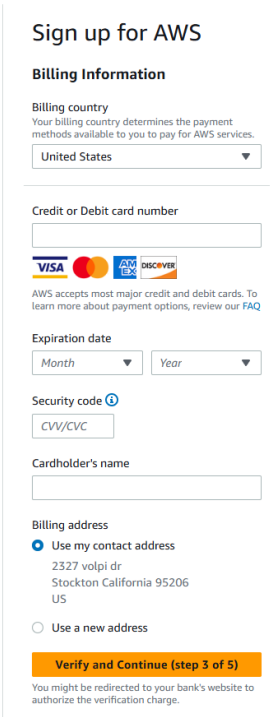




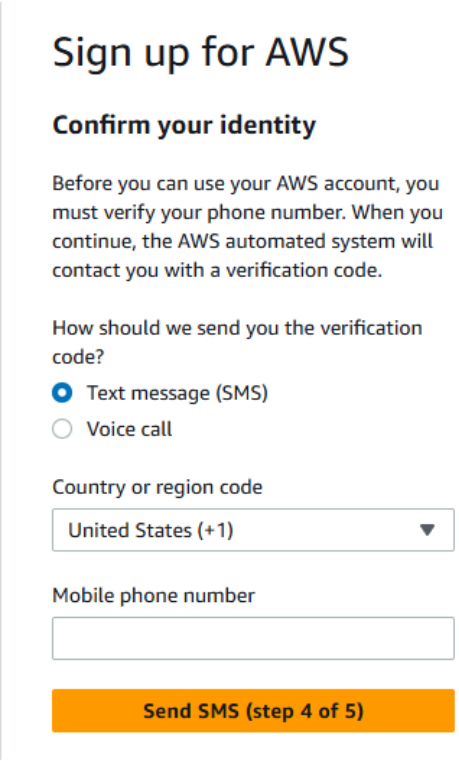
I entered the verification code that was sent to my email and pressed "Verify". Enter a "Root user password" and confirm the value then continue to the next step. The next step required my name, contact, address and use information. I select “personal” for the type of account. The next step required me to enter my billing information.

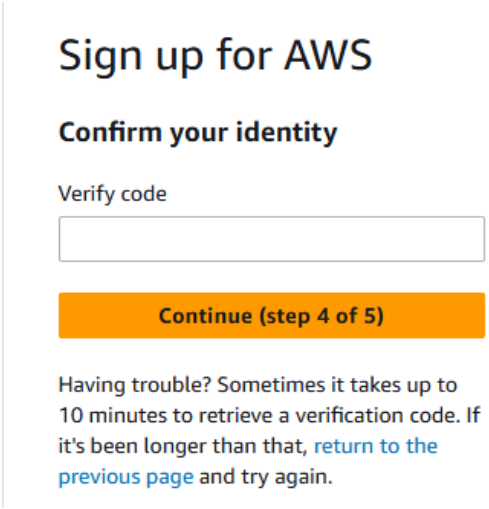


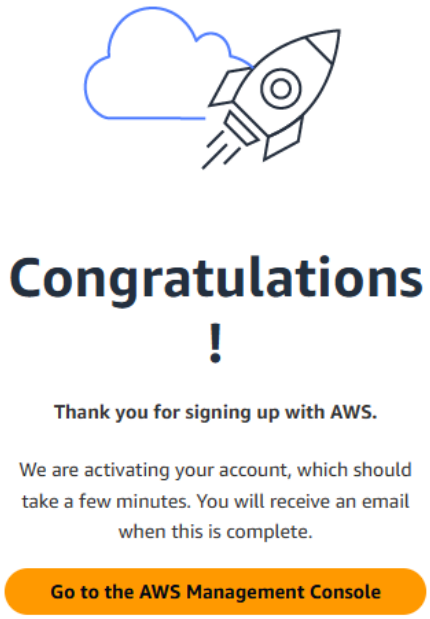
I then entered my payment information and press "Verify and Continue"



The next step required me to confirm my identity. So, I entered my phone number and completed the CAPTCHA. Upon submission I received a text message with a numeric code and used that code to verify my identity. In the final step, I select "Basic support -Free" and "Complete sign up".

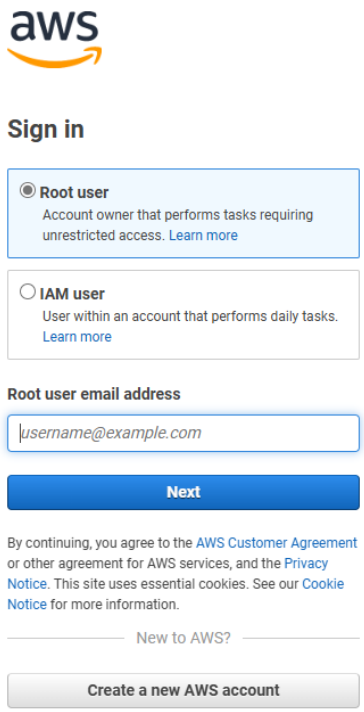




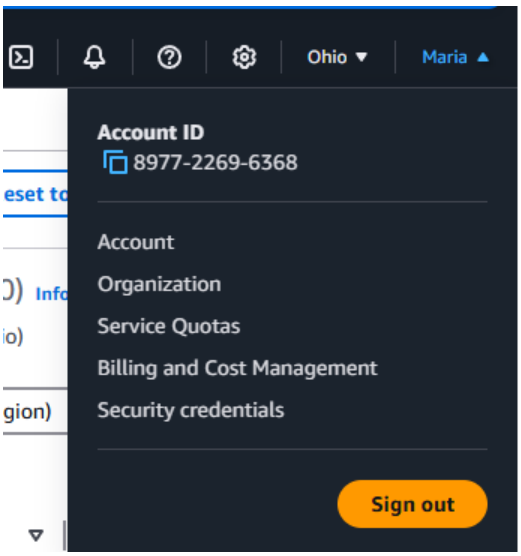


Step 2: Setup Root User MFA

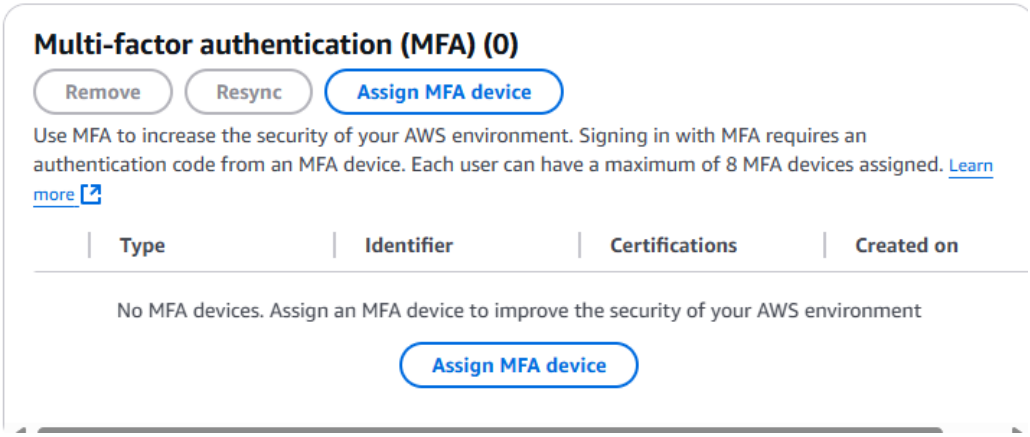
With my AWS account setup, I sign in to the management console by pressing the "Sign In to the Console". I selected "Root user" and entered the email address I used to setup the account. Then I entered my password, and I was in as the root user.



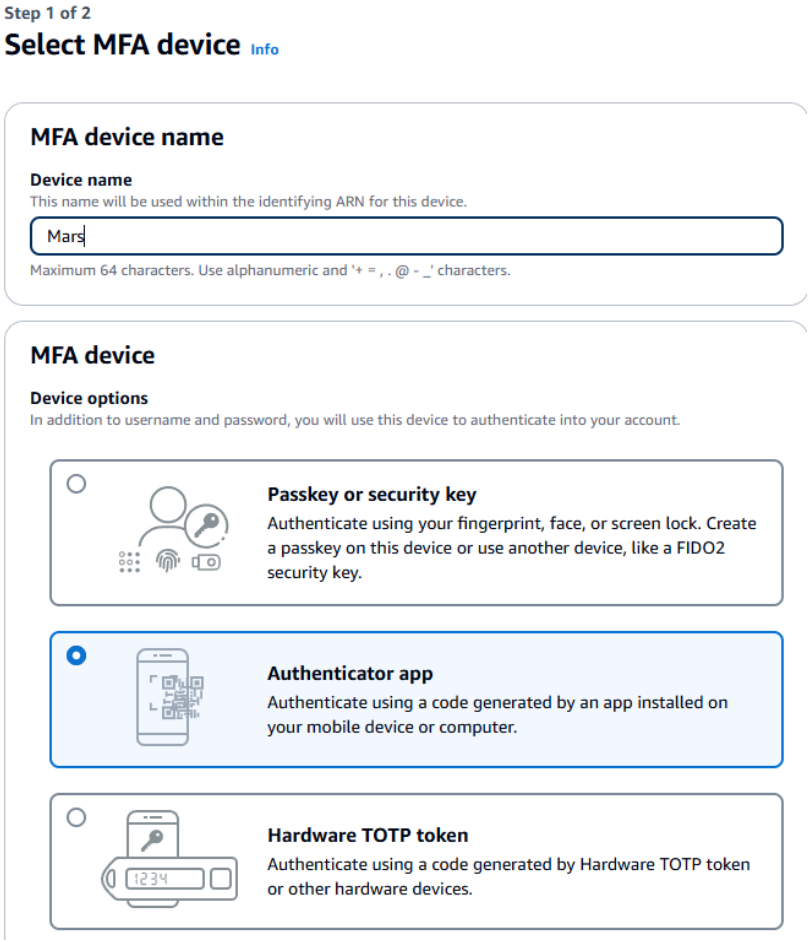
Once I logged in, I navigated to the user (root) settings selecting the account drop down menu in the upper right corner and pressing "Security credentials".



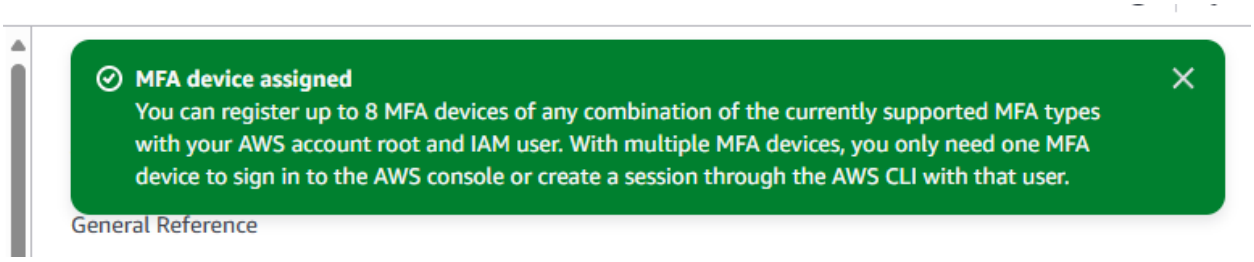
Then, I pressed "Assign MFA device" in the Multi-factor authentication table.



I select a "Device name" and choose an MFA device. I used duo mobile as that is an authenticator app I already have because of school.



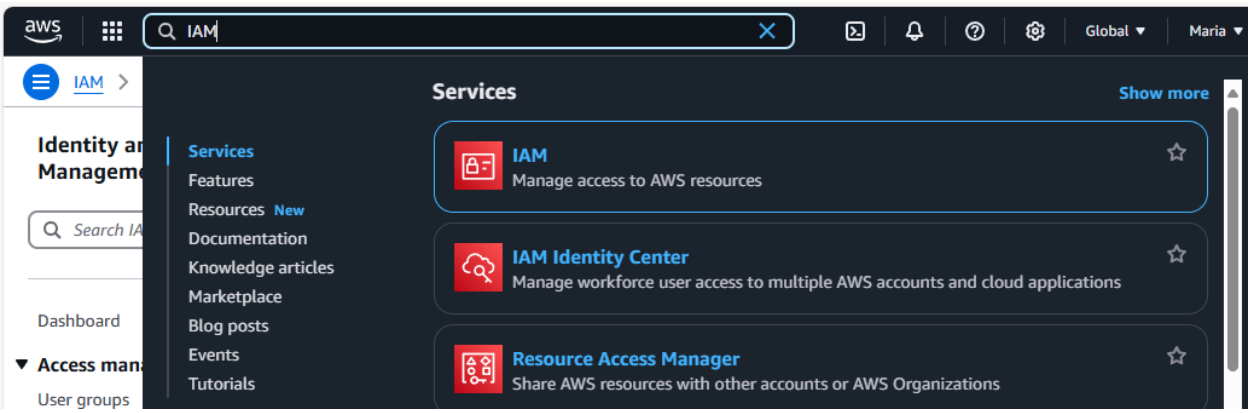
In duo mobile, I added AWS.



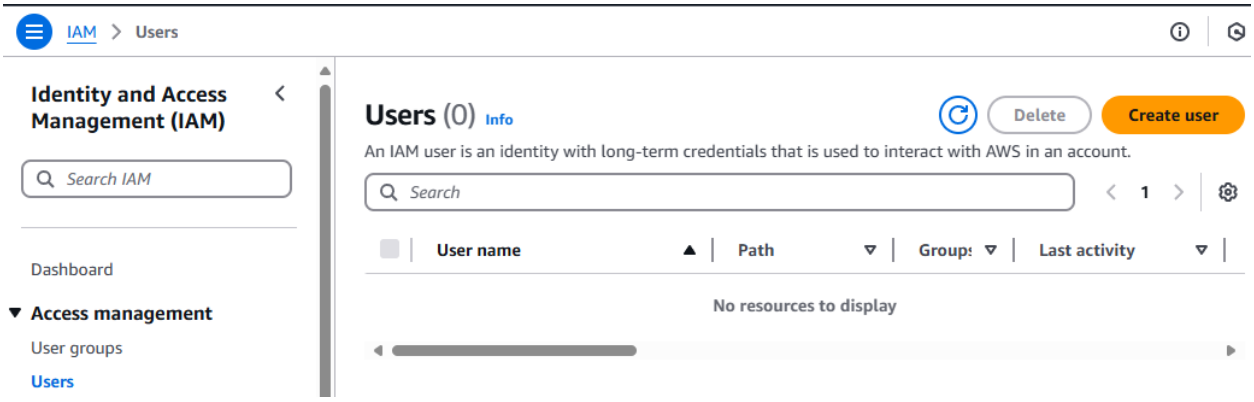
Step 3: Setup IAM User

Using the root user for administrative activities is considered a bad practice. So, I will create a non-root administrator IAM user for all activities in this AWS lab.

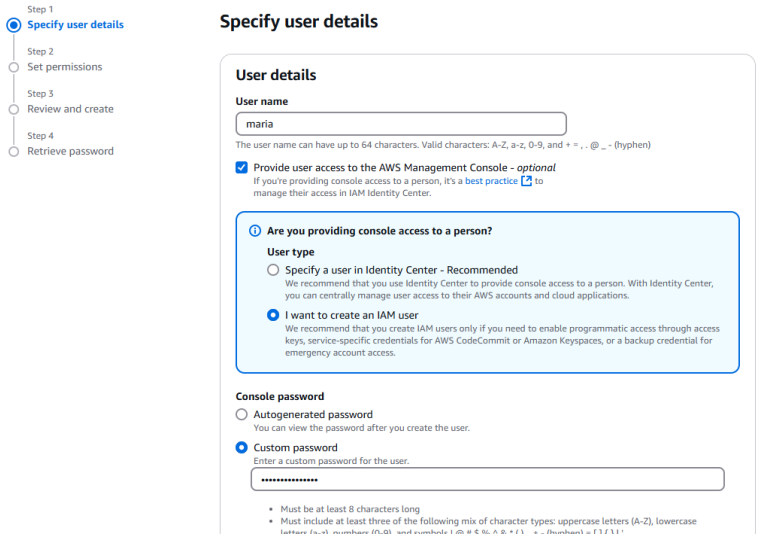
I searched for "IAM" in the services search bar at the top of the screen and followed the link for the IAM service.



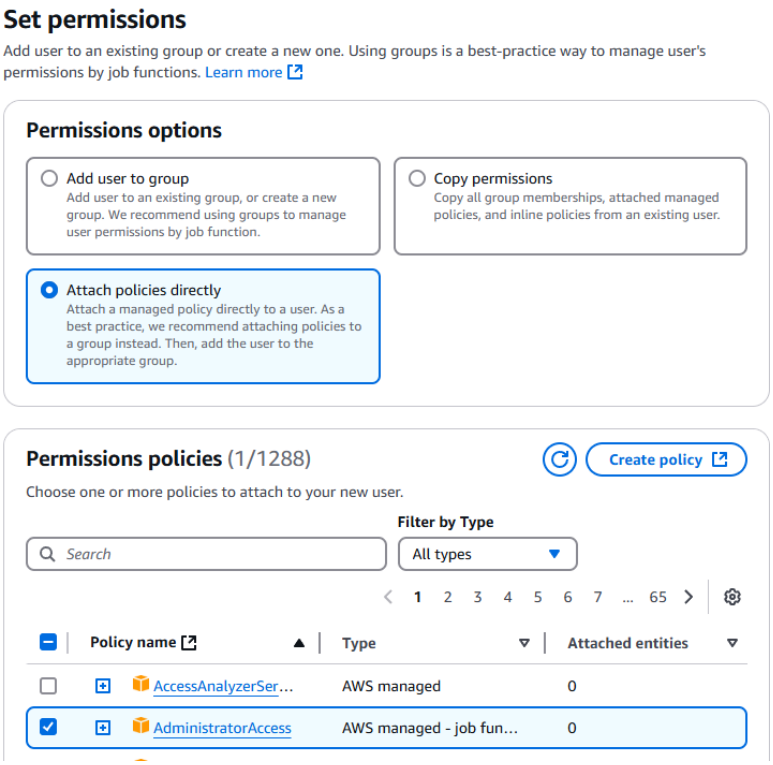
Then, I selected "Users" in the left navigation pane to begin the process of creating a user.



I pressed the "Create user" button on the Users page. Then, entered my name as the username, checked the box "Provide user access to the AWS Management Console" (so the user can log into the console). Then in the bluebox, I selected the "I want to create an IAM user" option. Entered a strong password and unselected the "Users must create a new password at next sign-in" checkbox.



After pressing next, I set the IAM user permissions to administrator by selecting the "Attach policies directly" option and marking the " AdministratorAccess " policy checkbox. Then pressed "Next".



I reviewed the IAM creation settings and pressed "Create user". To create a new user.

Review and create

Review your choices. After you create the user, you can view and download the autogenerated password, if enabled.

User details

User name	Console password type	Require password reset
maria	Custom password	No

Permissions summary

1

Name	Type	Used as
AdministratorAccess	AWS managed - job function	Permissions policy

Tags - *optional*

Tags are key-value pairs you can add to AWS resources to help identify, organize, or search for resources. Choose any tags you want to associate with this user.

No tags associated with the resource.

Add new tag

You can add up to 50 more tags.

Cancel Previous Create user

After that, I signed out and then signed in as the IAM user. To sign in, I used my AWS account number, IAM username and password.

IAM user sign in

Account ID (12 digits) or account alias

897722696368

IAM username

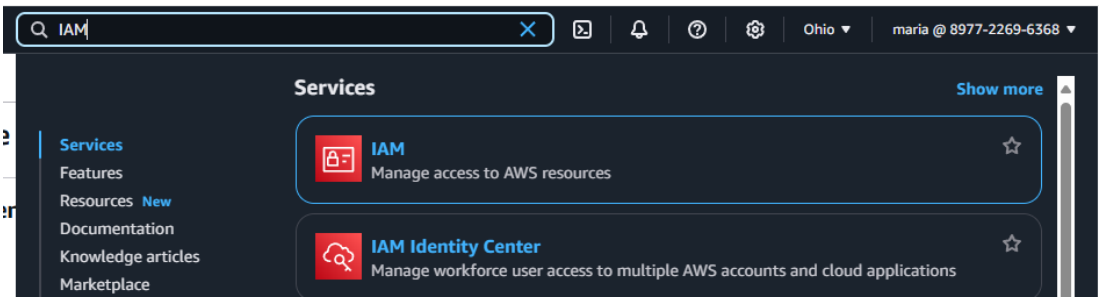
maria

Password

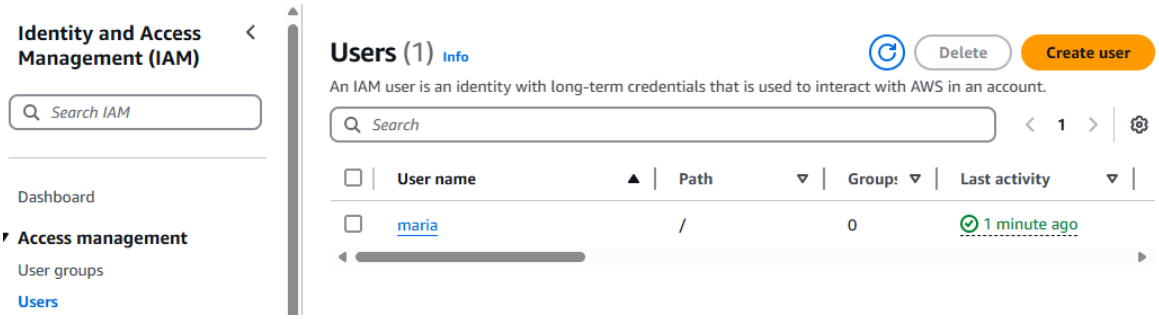
Show Password Having trouble?

Sign in

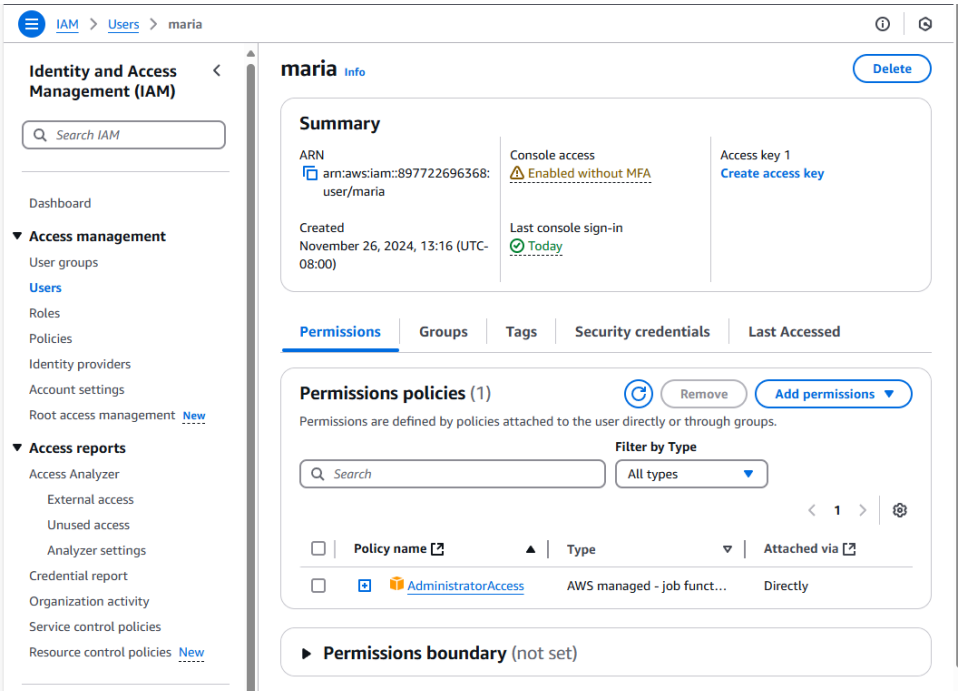
While logged in as my IAM administrator user (not root), I navigated to the IAM service by searching IAM in the search bar (top bar) and selecting the IAM service link.



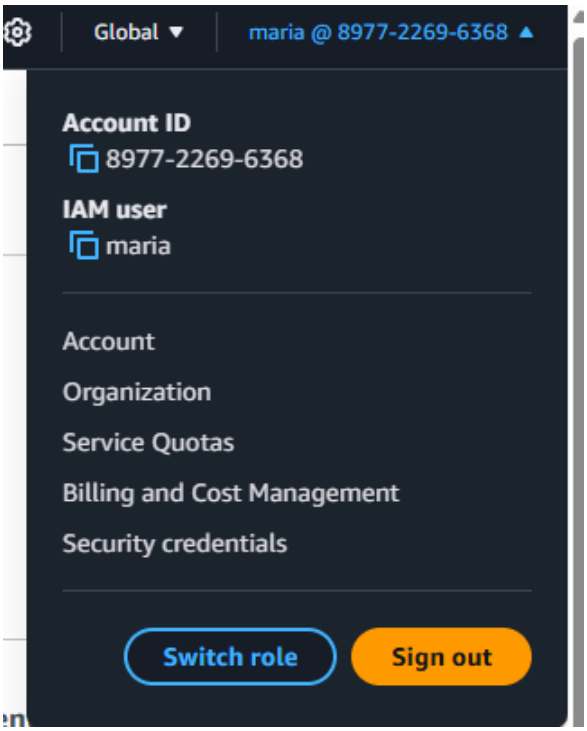
I selected "Users" in the left navigation menu to observe all IAM users.



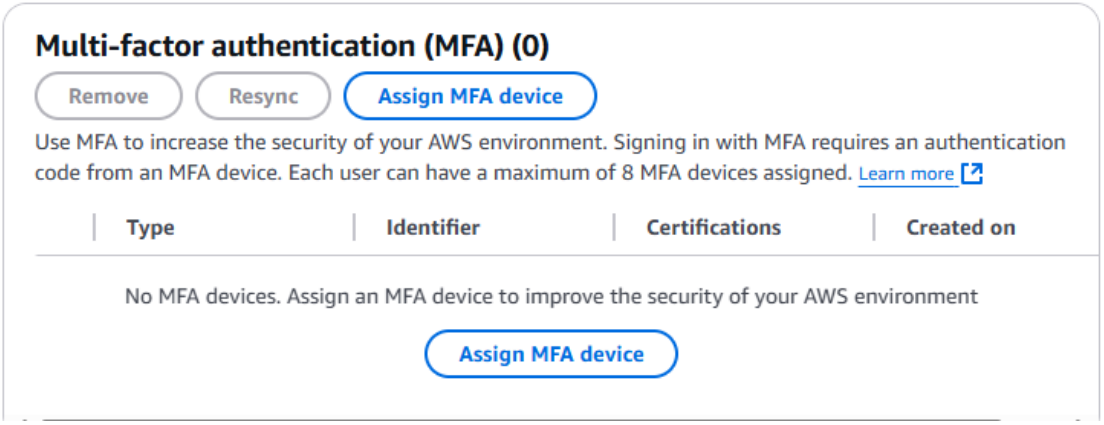
Then, I selected the IAM user I created (and are logged in as) to view its settings.

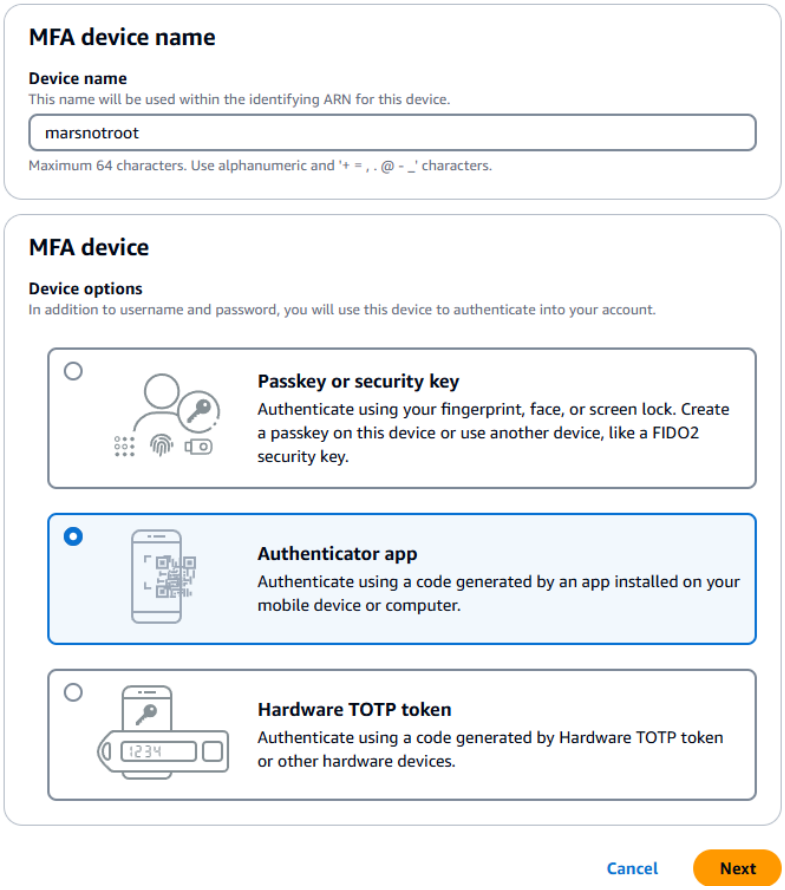


After viewing its settings, I selected the user "Security credentials" tab.

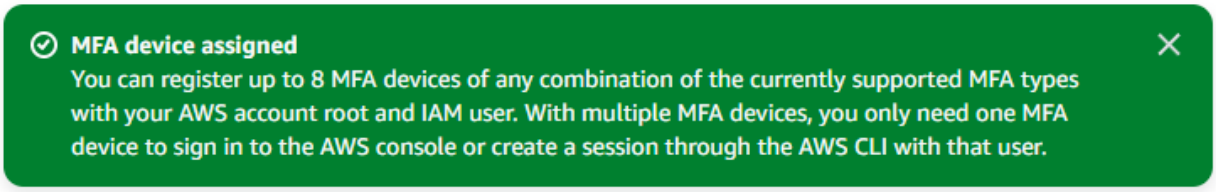


Then, I set up an MFA device following the same procedure as the root MFA device (using duo mobile). I named the device “marsnotroot” to distinguish it from the root account.





In duo mobile, I added AWS.

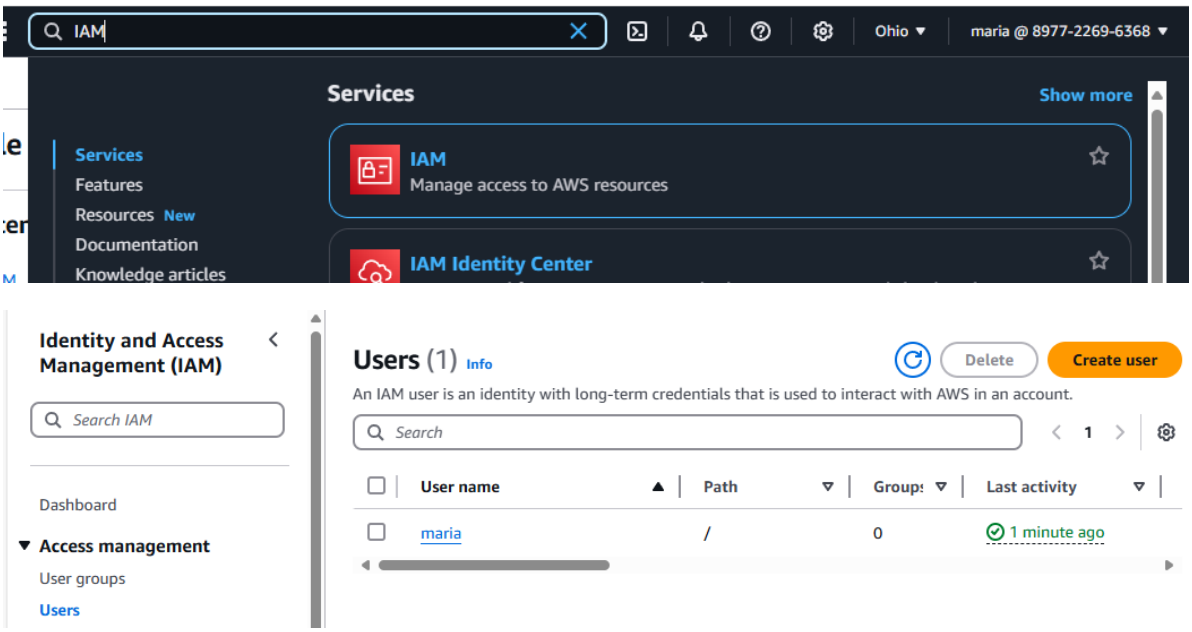


Exercise 13.2 - Scout Suite CSPM

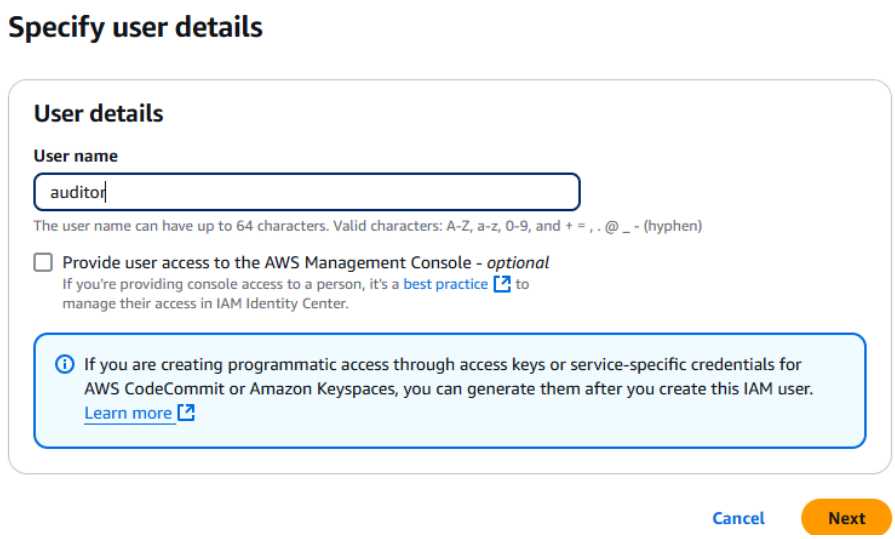
In this task I will create an IAM user with limited permissions and scan my AWS account to detect security misconfigurations. I will then identify and remediate security issues raised by the scanning tool.

Step 1: Create IAM User

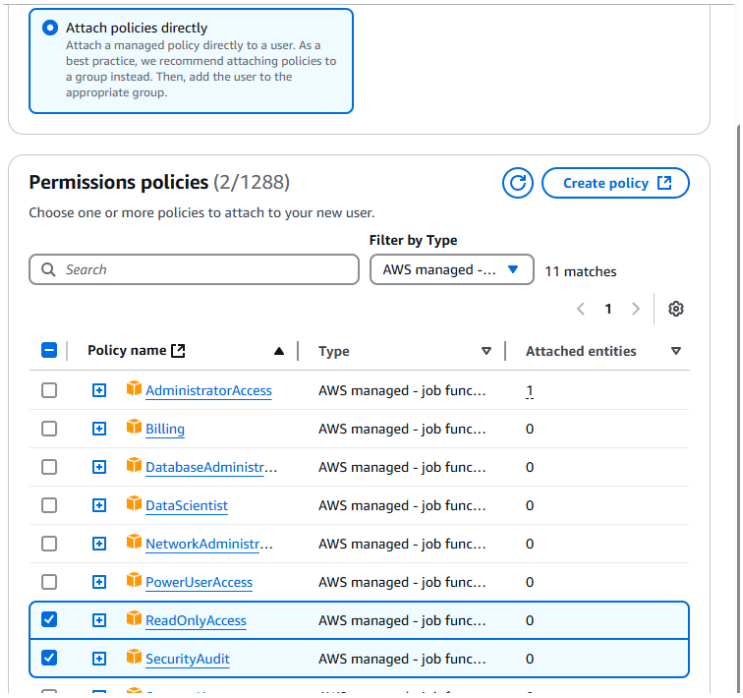
I logged into my AWS account using my administrator IAM user (not root). Then, navigated to the IAM service and the Users page.



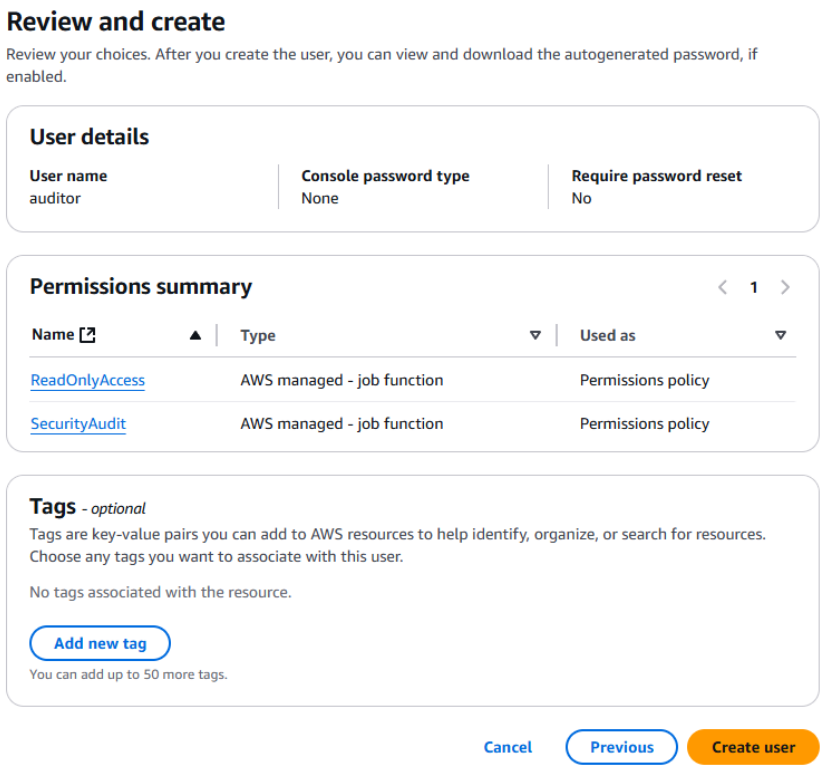
I pressed the "Create user" button to start the user creation process and entered "auditor" as the username. I left "Provide user access to the AWS Management Console" unchecked. Then pressed Next.



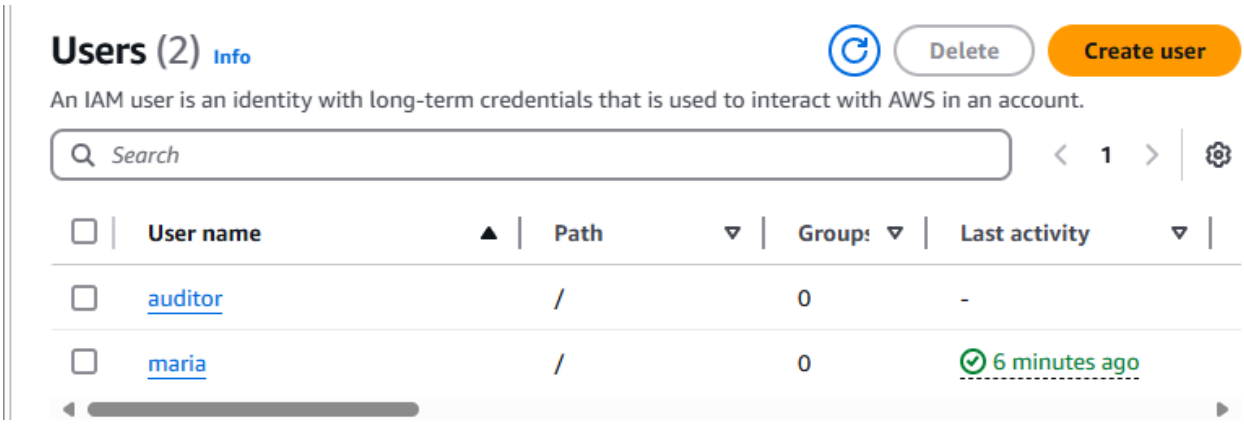
In the next step of the wizard, I chose "Attach policies directly". Then, I used the filter "AWS managed - job function" and selected " ReadOnlyAccess " and " SecurityAudit " permission policies.



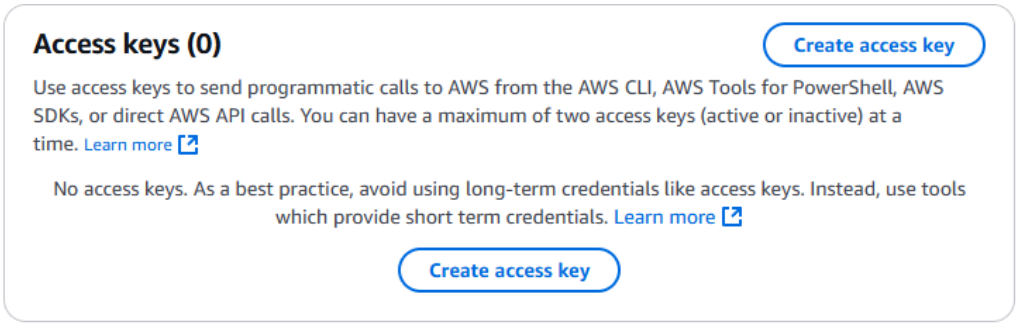
Then pressed Next and then "Create user".



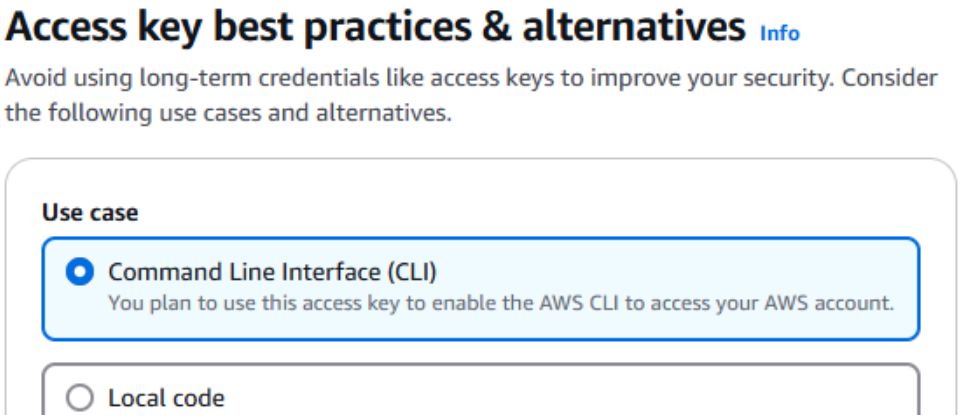
Now, I see it in my users. I selected the created user "auditor" from the Users page.



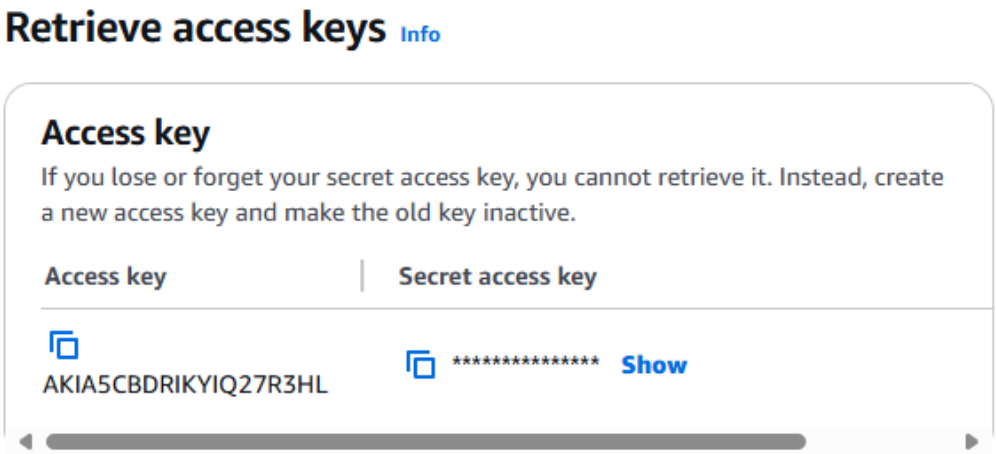
After, I navigated to the "Security credentials" tab and scroll down to the "Access keys" section.



Then, I pressed the "Create access key" button to create a user token that can be used within the command line interface.



With the create access key wizard launched, I chose the "Command Line Interface (CLI)" option and agreed to the confirmation. Then press Next and then "Create access key".



Step 2: Install and Configure AWS CLI

I launched my Ubuntu VM with Bridge Adapter network mode and opened a terminal. I made sure to update my system and then install the AWS CLI tool using the commands below.

```
maria@ubuntu:~/Desktop$ sudo apt install -y
[sudo] password for maria:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
0 upgraded, 0 newly installed, 0 to remove and 30 not upgraded.
maria@ubuntu:~/Desktop$ sudo apt install awscli -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
```

After a couple of minutes, the AWS CLI is installed and can be configured using the AWS version command.

```
maria@ubuntu:~/Desktop$ aws --version
aws-cli/1.22.34 Python/3.10.12 Linux/6.8.0-49-generic botocore/1.23.34
maria@ubuntu:~/Desktop$
```

Once the AWS CLI is installed, I configured the tool to use the "auditor" IAM credentials created in the previous step. After using the aws configure –profile auditor command, I entered my access key, secret key, region as "us-west-2", and output format as "json".

```
maria@ubuntu:~/Desktop$ aws configure --profile auditor
AWS Access Key ID [None]: AKIA5CBDRIKYIQ27R3HL
AWS Secret Access Key [None]:
```

```
Default region name [None]: us-west-2
Default output format [None]: json
maria@ubuntu:~/Desktop$
```

Step 3: Setup and Run Scout Suite

I updated sudo and installed Python virtual environment using the commands below.

```
maria@ubuntu:~/Desktop$ sudo apt update -y
Hit:1 http://us.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Get:3 http://us.archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
```

```
maria@ubuntu:~/Desktop$ sudo apt install python3-virtualenv -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
```

Here, I created a python virtual environment to run Scout Suite from to avoid any Python library conflicts. Using the commands below I:

- I created the virtual environment using the command below

```
maria@ubuntu:~/Desktop$ virtualenv -p python3 venv
created virtual environment CPython3.10.12.final.0-64 in 396ms
  creator CPython3Posix(dest=/home/maria/Desktop/venv, clear=False, no_vcs_ignor
e=False, global=False)
  seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle
, via=copy, app_data_dir=/home/maria/.local/share/virtualenv)
    added seed packages: pip==22.0.2, setuptools==59.6.0, wheel==0.37.1
  activators BashActivator,CShellActivator,FishActivator,NushellActivator,PowerS
hellActivator,PythonActivator
maria@ubuntu:~/Desktop$ source venv/bin/activate
```

- installed scout using the command below

```
(venv) maria@ubuntu:~/Desktop$ pip install scoutsuite
Collecting scoutsuite
  Downloading ScoutSuite-5.14.0-py3-none-any.whl (3.5 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.5/3.5 MB 16.0 MB/s eta 0:00:00
Collecting google-cloud-kms==1.3.0
  Downloading google_cloud_kms-1.3.0-py2.py3-none-any.whl (65 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 65.3/65.3 KB 3.7 MB/s eta 0:00:00
Collecting azure-mgmt-keyvault==8.0.0
  Downloading azure_mgmt_keyvault-8.0.0-py2.py3-none-any.whl (197 kB)
```

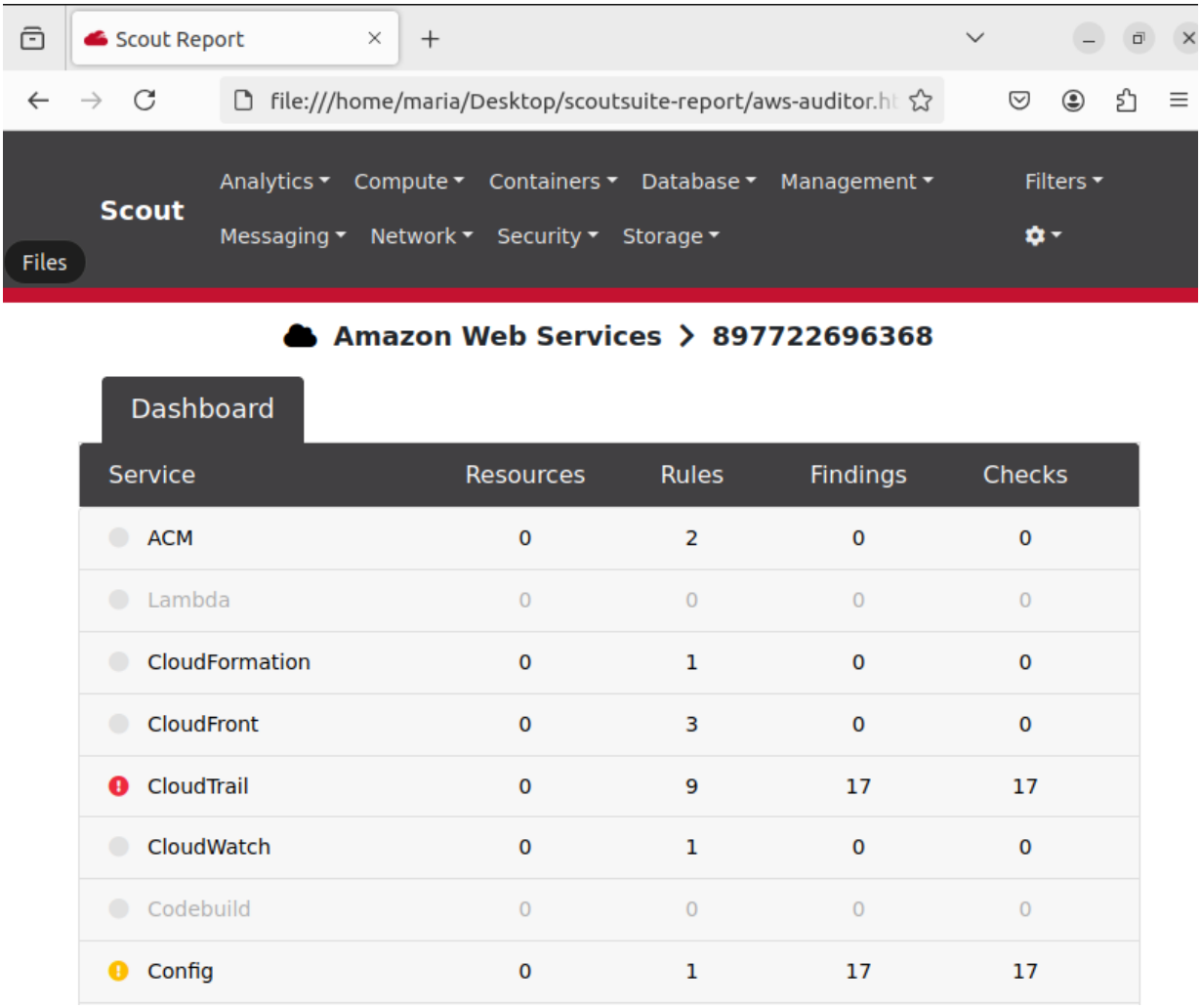
- verified its installation using the command below

```
(venv) maria@ubuntu:~/Desktop$ scout --help
usage: scout [-h] [-v] {aws,gcp,azure,aliyun,oci,kubernetes,do} ...
Files
  -h, --help            show this help message and exit
  -v, --version         show program's version number and exit

The provider you want to run scout against:
  {aws,gcp,azure,aliyun,oci,kubernetes,do}
    aws                 Run Scout against an Amazon Web Services account
    gcp                 Run Scout against a Google Cloud Platform account
    azure               Run Scout against a Microsoft Azure account
    aliyun              Run Scout against an Alibaba Cloud account
    oci                 Run Scout against an Oracle Cloud Infrastructure
                        account
    kubernetes          Run Scout against a Kubernetes cluster
    do                  Run Scout against an DigitalOcean account

To get addtional help on a specific provider run: scout.py {provider} -h
(venv) maria@ubuntu:~/Desktop$
```

After, I ran a posture scan to discover any potential security misconfigurations. Scout will make API calls to all AWS services using the auditor account and compare results to a rules engine that identifies any potential security flaws.

```
(venv) maria@ubuntu:~/Desktop$ scout aws --profile auditor
2024-11-26 14:44:24 ubuntu scout[7163] INFO Launching Scout
2024-11-26 14:44:24 ubuntu scout[7163] INFO Authenticating to cloud provider
2024-11-26 14:44:27 ubuntu scout[7163] INFO Gathering data from APIs
2024-11-26 14:44:27 ubuntu scout[7163] INFO Fetching resources for the ACM servi
ce
2024-11-26 14:44:27 ubuntu scout[7163] INFO Fetching resources for the Lambda se
rvice
2024-11-26 14:44:28 ubuntu scout[7163] INFO Fetching resources for the CloudForm
ation service
2024-11-26 14:44:28 ubuntu scout[7163] INFO Fetching resources for the CloudTrai
l service
2024-11-26 14:44:28 ubuntu scout[7163] INFO Fetching resources for the CloudWatc
h service
```

Once the scan was completed, an HTML report was generated and automatically opened in my VM's browser.



Service	Resources	Rules	Findings	Checks
ACM	0	2	0	0
Lambda	0	0	0	0
CloudFormation	0	1	0	0
CloudFront	0	3	0	0
CloudTrail	0	9	17	17
CloudWatch	0	1	0	0
Codebuild	0	0	0	0
Config	0	1	17	17

Step 4: Analyze and Fix

In this step, I will be selecting a vulnerability that was identified by scout suite, research its vulnerability and try to “cure” the vulnerability.

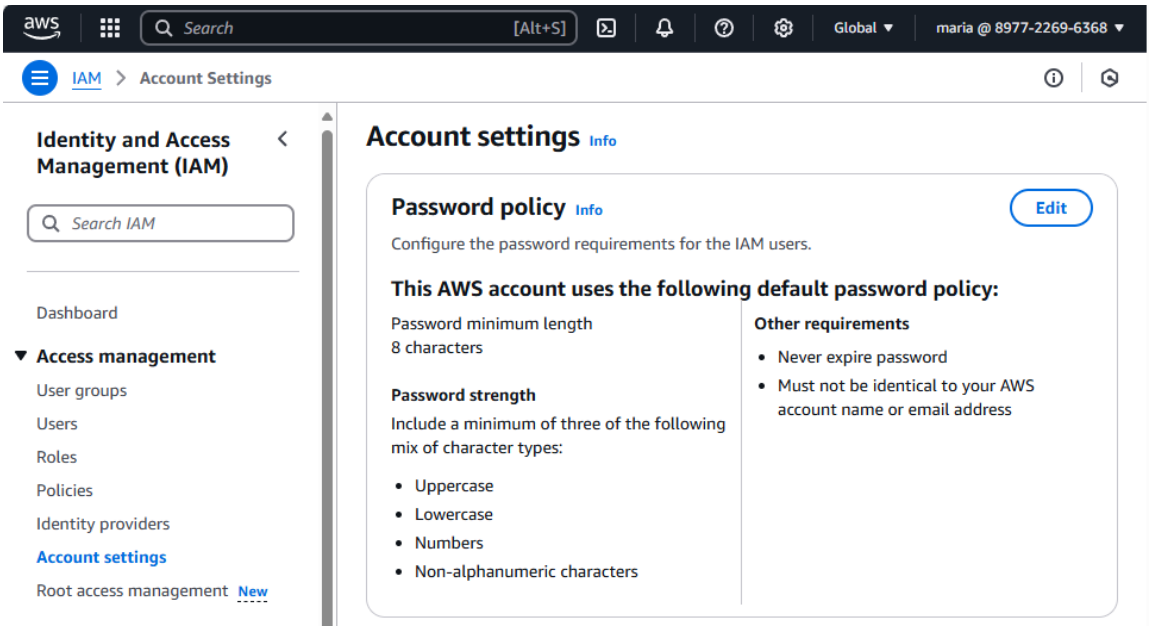
I chose the IAM service, specifically the password policy that allows the reuse of passwords.

1. Description of the vulnerable AWS service:

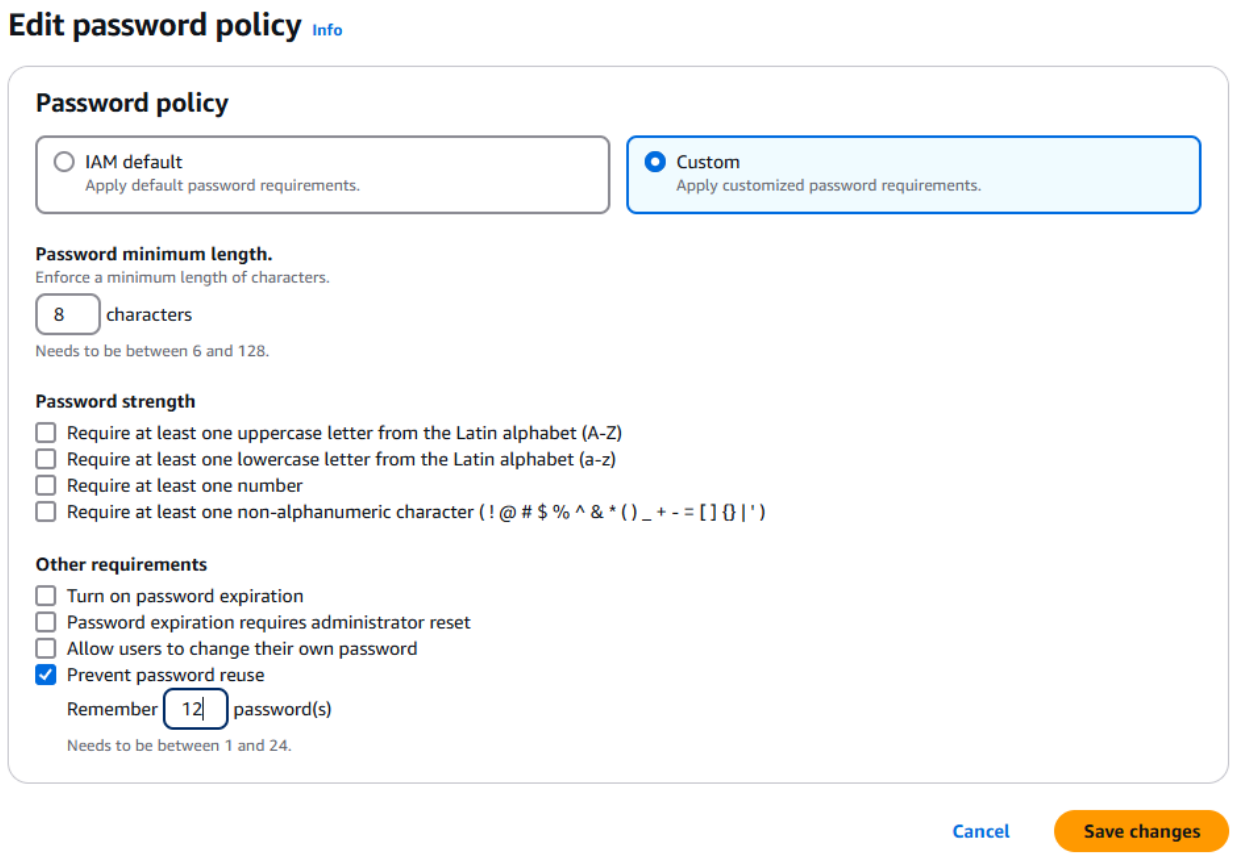
   a. The password policy allowed password reuse. As a result, password complexity requirements were not in line with security best practice.
2. How is security impacted by the vulnerability/misconfiguration:
   a. It undermines the principle of credential hygiene, increasing the risk of unauthorized access and attackers can exploit reused passwords from previous breaches.
3. How can the service be fixed (what steps are needed):
   a. Ensure the password policy is configured to prevent password reuse.
   b. In order to do that, I log into the AWS console, navigate to IAM > Account settings > password policy. Enable prevent password reuse. Save changes .

Next, cure the vulnerability/misconfiguration and re-run Scout to confirm the issue no longer exists in the report. Please keep in mind that some fixes may incur a cost and you should:

- I log into the AWS console, navigate to IAM > Account settings



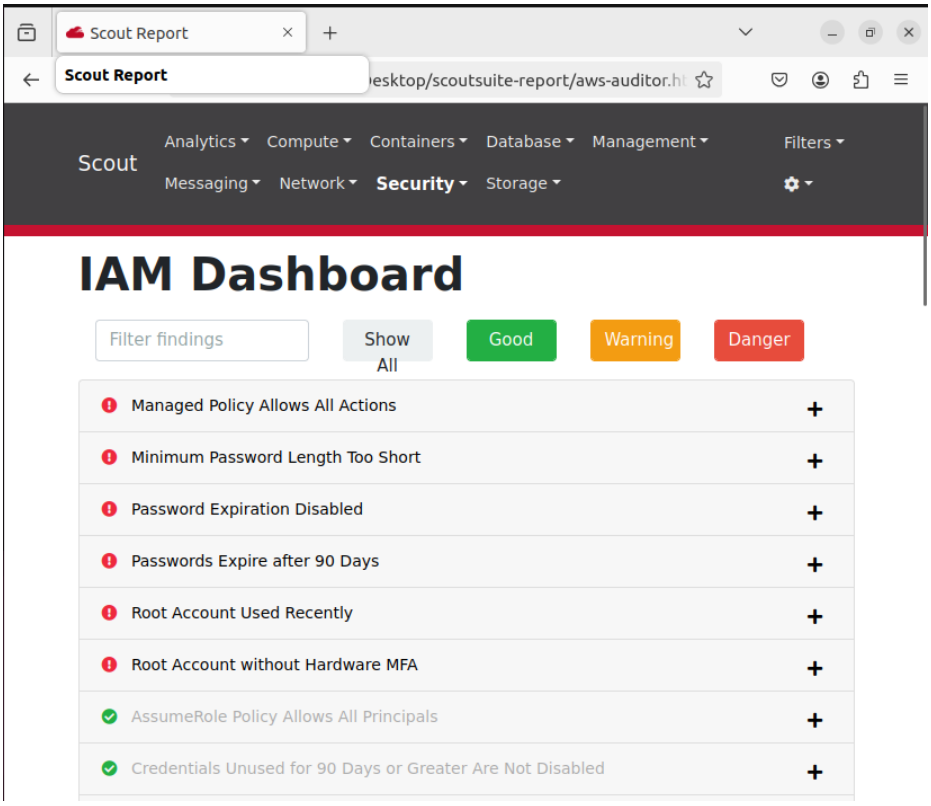
- In edit password policy, I check the “prevent password reuse” box and added “12” to the “remember x passwords” and clicked save changes.



- Here I ran scout again on the auditor profile like previously and waited for an HTML report to pop up on my VM.

```
To get addtional help on a specific provider run: scout.py {provider} -h
(venv) maria@ubuntu:~/Desktop$ scout aws --profile auditor
2024-11-26 15:02:04 ubuntu scout[9335] INFO Launching Scout
2024-11-26 15:02:04 ubuntu scout[9335] INFO Authenticating to cloud provider
2024-11-26 15:02:06 ubuntu scout[9335] INFO Gathering data from APIs
2024-11-26 15:02:06 ubuntu scout[9335] INFO Fetching resources for the ACM servi
ce
2024-11-26 15:02:06 ubuntu scout[9335] INFO Fetching resources for the Lambda se
rvice
2024-11-26 15:02:07 ubuntu scout[9335] INFO Fetching resources for the CloudForm
ation service
2024-11-26 15:02:07 ubuntu scout[9335] INFO Fetching resources for the CloudTrai
l service
```

- The HTML report popped up and the Password policy allows the reuse of passwords is fixed! It no longer pops up



For reference, this is what the IAM dashboard looked like before I cured the vulnerability.