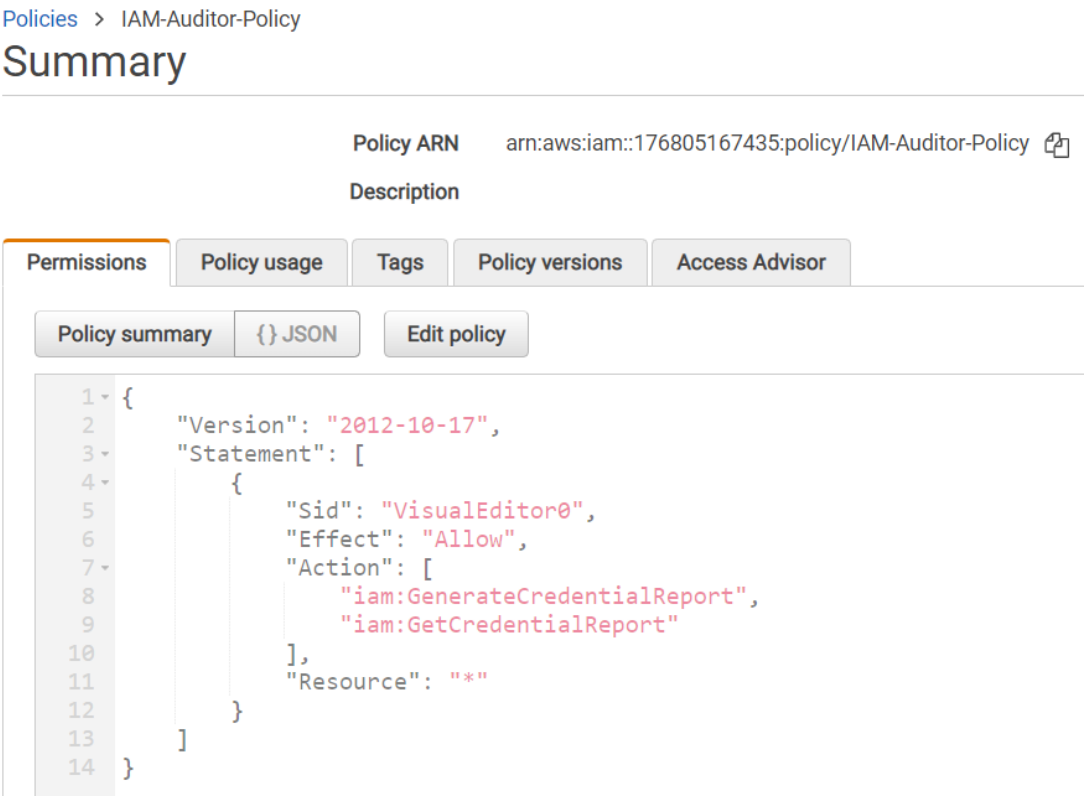
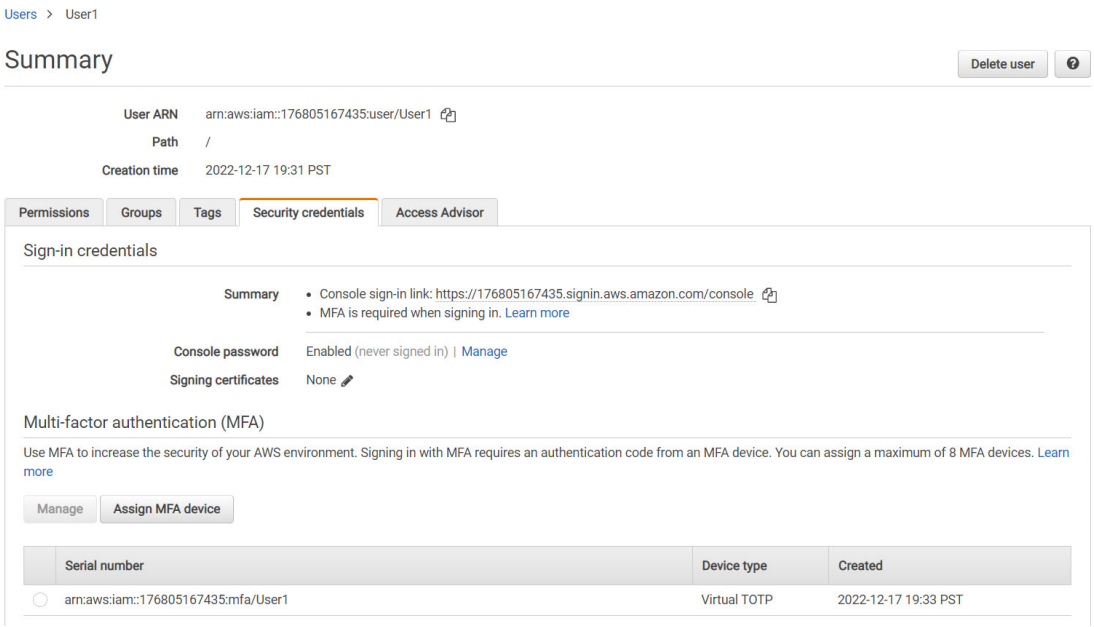
Task 1 (5%): Open the IAM console-> choose Policies, and then choose to Create policy then you can use Visual editor or JSON file to create a custom policy. Create a policy with the least privilege strategy which can get the IAM credential report (name it IAM-Auditor-Policy).

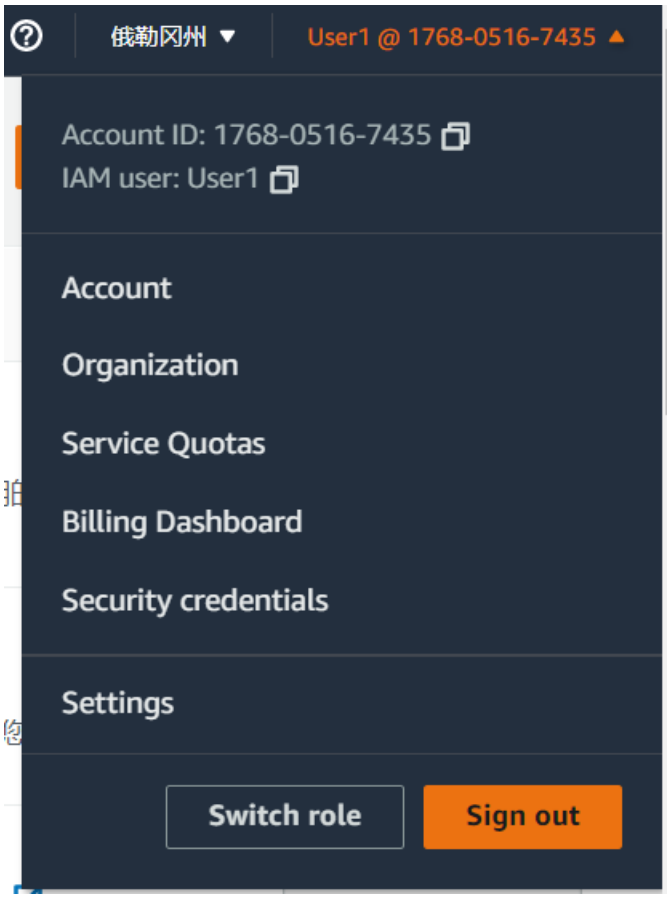


Task 2 (5%): Create the following users:

- User1 without any policy or permission with console access
    - Enable MFA for User1 by your admin user and login with user1 while MFA is enabled.
- User2 without any policy or permission with console access
    - Generate one access key for User2.
- User3 without any policy or permission with only console access

User1:


Users > User1
Summary
Delete user
User ARN arn:aws:iam::176805167435:user/User1
Path /
Creation time 2022-12-17 19:31 PST
Permissions
Groups
Tags
Security credentials
Access Advisor
Sign-in credentials
Summary
Console sign-in link: https://176805167435.signin.aws.amazon.com/console
MFA is required when signing in. Learn more
Console password Enabled (never signed in) | Manage
Signing certificates None
Multi-factor authentication (MFA)
Use MFA to increase the security of your AWS environment. Signing in with MFA requires an authentication code from an MFA device. You can assign a maximum of 8 MFA devices. Learn more
Manage
Assign MFA device
Serial number
Device type
Created
arn:aws:iam::176805167435:mfa/User1
Virtual TOTP
2022-12-17 19:33 PST


俄勒冈州
User1 @ 1768-0516-7435
Account ID: 1768-0516-7435
IAM user: User1
Account
Organization
Service Quotas
Billing Dashboard
Security credentials
Settings
Switch role
Sign out

User2:

Add user

1 2 3 4 5

Success

You successfully created the users shown below. You can view and download user security credentials. You can also email users instructions for signing in to the AWS Management Console. This is the last time these credentials will be available to download. However, you can create new credentials at any time.

Users with AWS Management Console access can sign-in at: https://176805167435.signin.aws.amazon.com/console

Download .csv

	User	Access key ID	Secret access key
	User2	AKIASSKTIYFFWLBLGXWD	qRboQaaEVwvAamf+jh9mVw+jwYOS3uERiTfRWRba Hide

Users > User2

Summary

User ARN	arn:aws:iam::176805167435:user/User2
Path	/
Creation time	2022-12-17 19:39 PST

Permissions | Groups | Tags | Security credentials | Access Advisor

Sign-in credentials

Summary	• User does not have console management access
Console password	Disabled | Manage
Signing certificates	None

Access keys

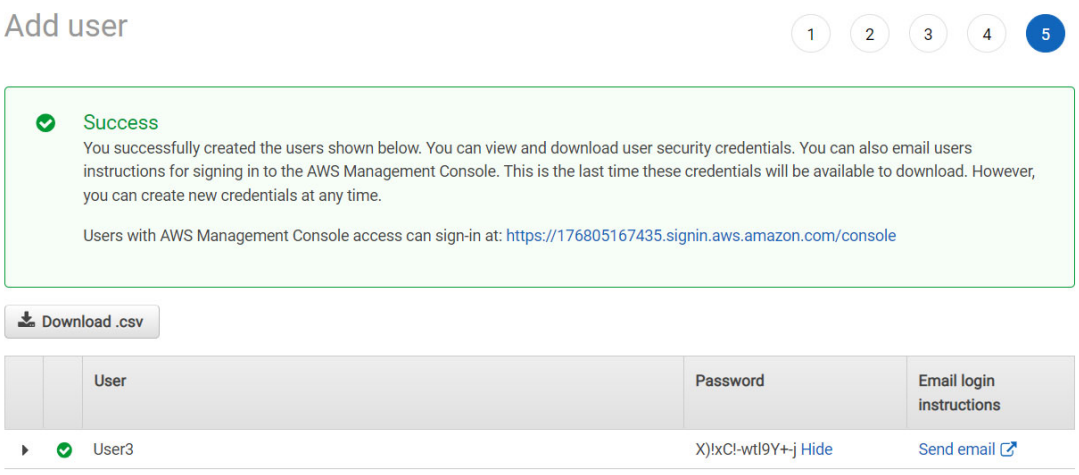
Use access keys to make programmatic calls to AWS from the AWS CLI, Tools for PowerShell, AWS SDKs, or direct AWS API calls. You can have a maximum of two access keys (active or inactive) at a time.

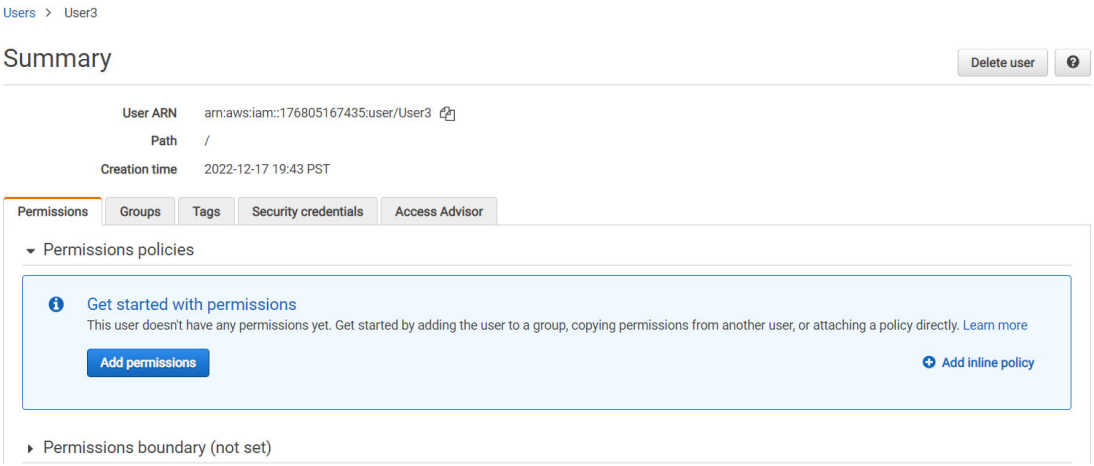
For your protection, you should never share your secret keys with anyone. As a best practice, we recommend frequent key rotation.
If you lose or forget your secret key, you cannot retrieve it. Instead, create a new access key and make the old key inactive. Learn more

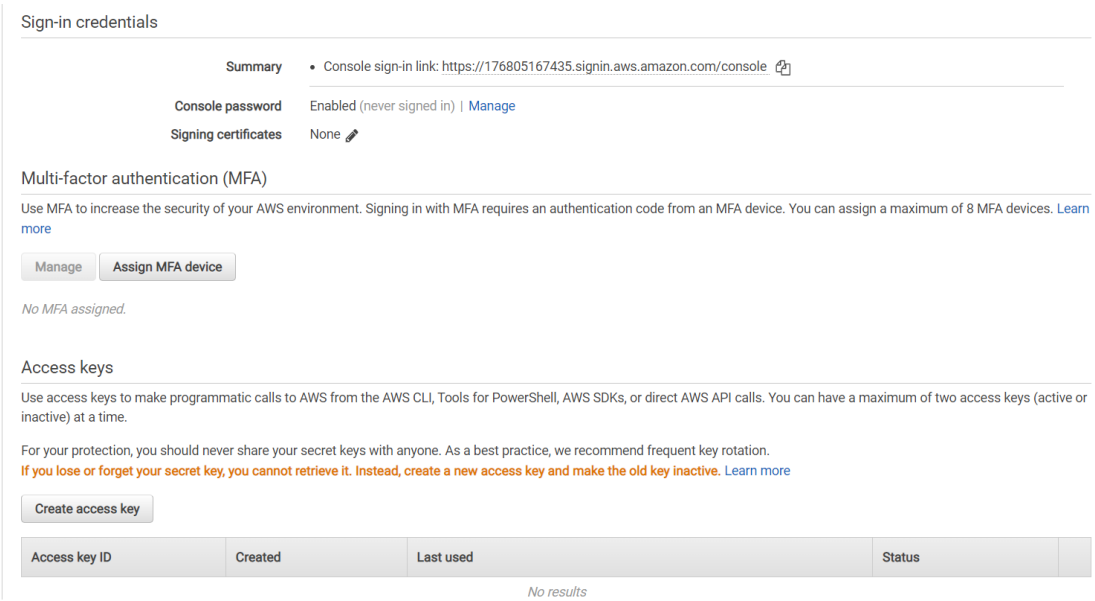
Create access key

Access key ID	Created	Last used	Status	
AKIASSKTIYFFWLBLGXWD	2022-12-17 19:39 PST	N/A	Active | Make inactive	✖

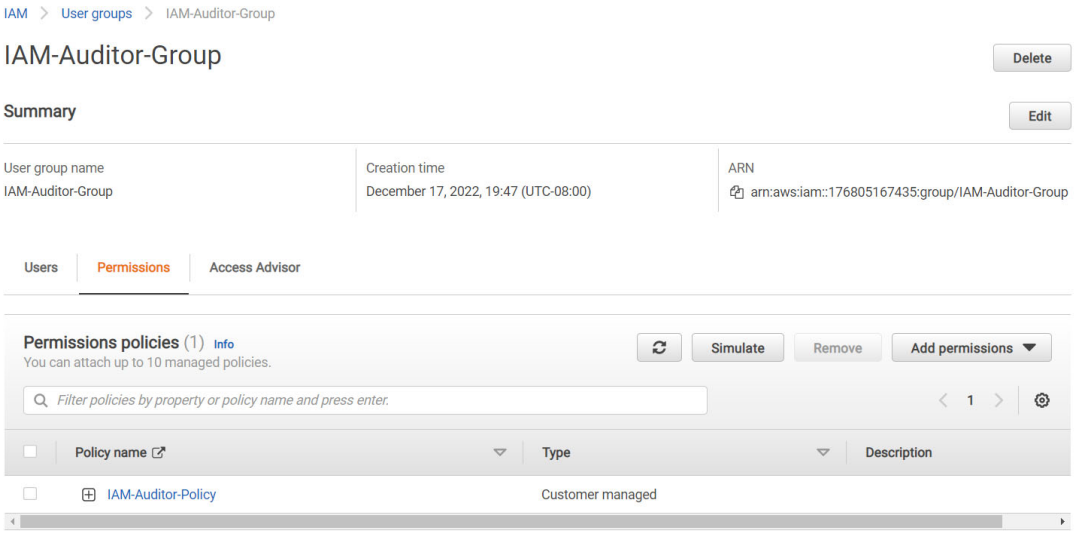
User3:



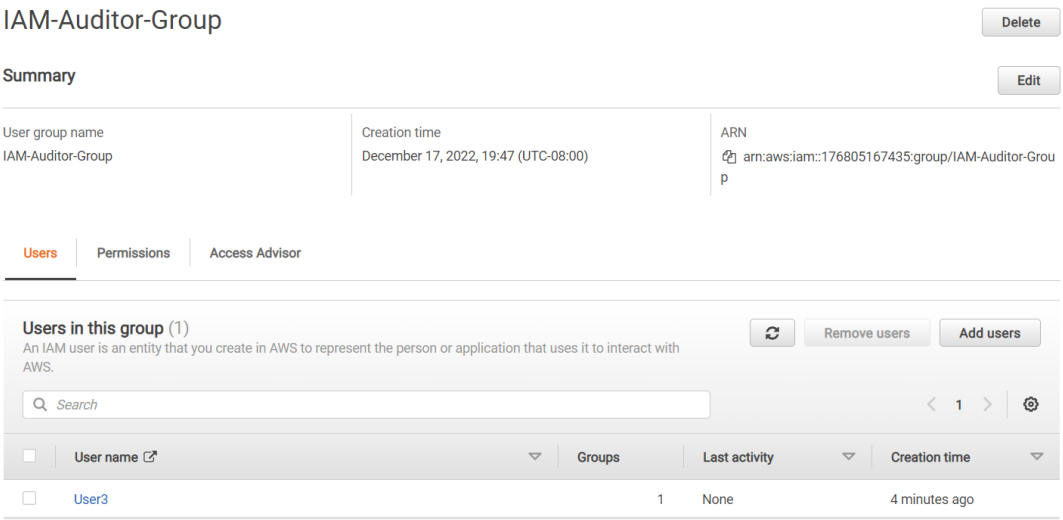




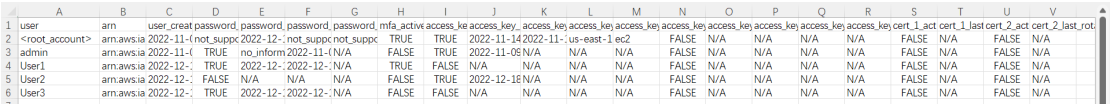
Task 3 (3%): Create a user group named IAM-Auditor-Group and attach the custom policy of task 1 to this user group.



Task 4 (2%): Add User3 to the IAM-Auditor group.



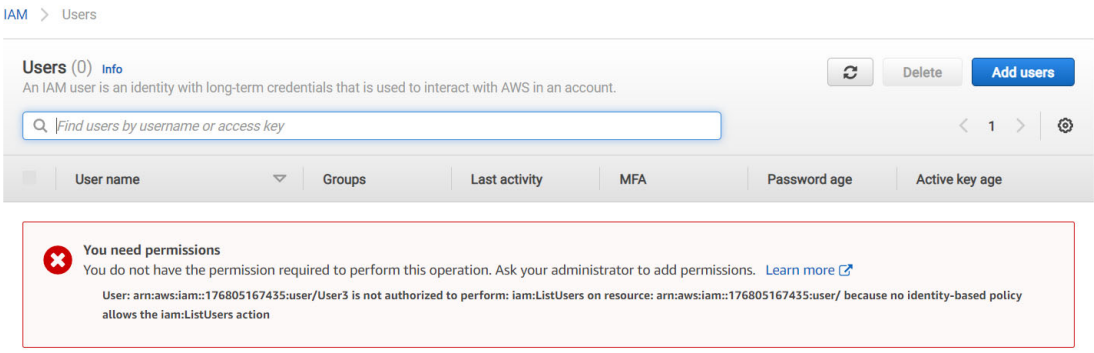
Task 5 (5%): Login with User3. Go to IAM service -> Setting and get the IAM credential report. Report your findings about each user from the generated credential report.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1	user	arn	user_creat	password_	password_	password_	password_	mfa_active	access_ke	access_key_	access_key	access_key	access_key	access_key	access_key	access_key	access_key	access_key	cert_1_act	cert_1_last	cert_2_act	cert_2_last_rot
2	<root_account>	arn:aws:ia	2022-11-(	not_suppc	2022-12-;	not_suppc	not_suppc	TRUE	TRUE	2022-11-14	2022-11-;	us-east-1	ec2	FALSE	N/A	N/A	N/A	N/A	FALSE	N/A	FALSE	N/A
3	admin	arn:aws:ia	2022-11-(	TRUE	no_inform	2022-11-(	N/A	FALSE	TRUE	2022-11-09	N/A	N/A	N/A	FALSE	N/A	N/A	N/A	N/A	FALSE	N/A	FALSE	N/A
4	User1	arn:aws:ia	2022-12-;	TRUE	2022-12-;	2022-12-;	N/A	TRUE	FALSE	N/A	N/A	N/A	N/A	FALSE	N/A	N/A	N/A	N/A	FALSE	N/A	FALSE	N/A
5	User2	arn:aws:ia	2022-12-;	FALSE	N/A	N/A	N/A	FALSE	TRUE	2022-12-1	N/A	N/A	N/A	FALSE	N/A	N/A	N/A	N/A	FALSE	N/A	FALSE	N/A
6	User3	arn:aws:ia	2022-12-;	TRUE	2022-12-;	2022-12-;	N/A	FALSE	FALSE	N/A	N/A	N/A	N/A	FALSE	N/A	N/A	N/A	N/A	FALSE	N/A	FALSE	N/A

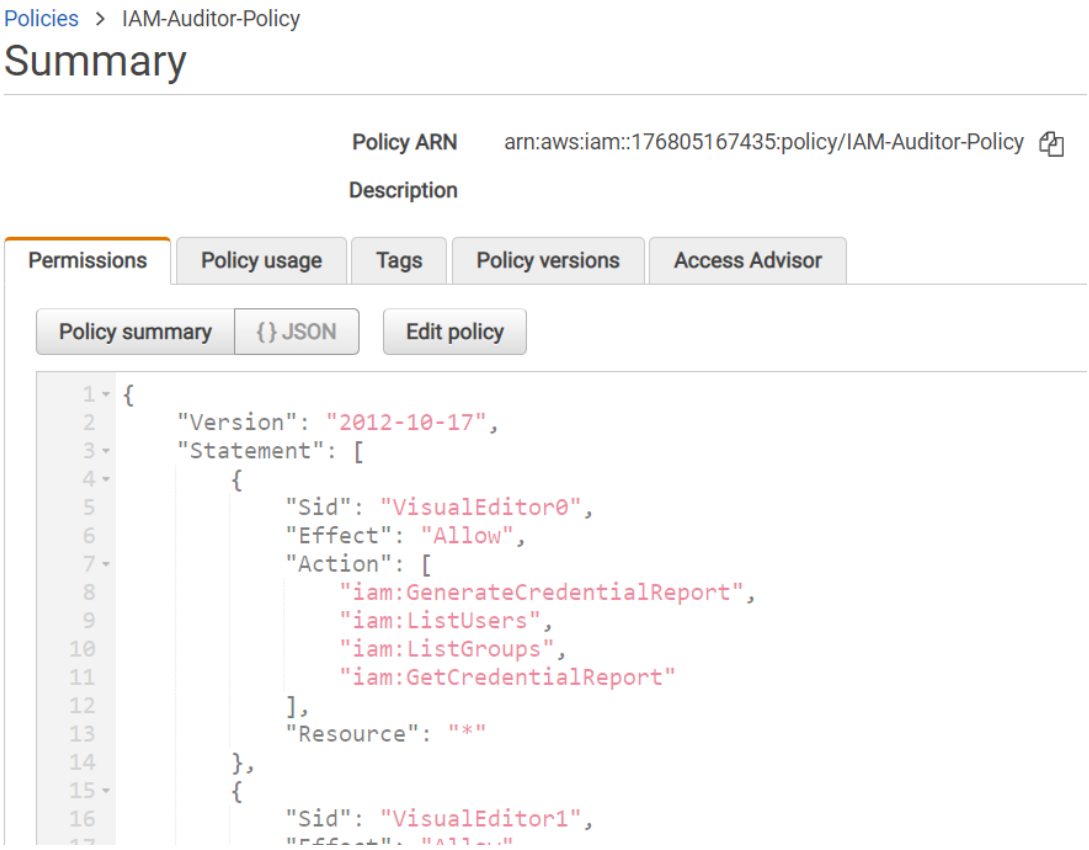
Task 6 (5%): When you are still logged in through User3, add User2 to the IAM-Auditor group. Explain your observation. In case of failure, describe the steps needed to be taken in order to User3 can add User2 to IAM Auditor.

User3 failed:



Steps:

1. Add ListUsers and ListGroups strategies to IAM-Auditor-Policy.
2. Add AddUserToGroup, GetGroup strategies to IAM-Auditor-Policy and specify group resource ARN as arn:aws:iam::176805167435:group/IAM-Auditor-Group to restrict access.
3. User3 can add User2 to IAM Auditor.



Policies > IAM-Auditor-Policy

Summary

Policy ARN	arn:aws:iam::176805167435:policy/IAM-Auditor-Policy
Description	

Permissions | Policy usage | Tags | Policy versions | Access Advisor

Policy summary | {} JSON | Edit policy

```
                "iam:ListUsers",
                "iam:ListGroups",
                "iam:GetCredentialReport"
            ],
            "Resource": "*"
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": [
                "iam:AddUserToGroup",
                "iam:GetGroup"
            ],
            "Resource": "arn:aws:iam::176805167435:group/IAM-Auditor-Group"
        }
    ]
}
```

Users added to this group.

IAM > User groups > IAM-Auditor-Group

IAM-Auditor-Group

Delete

Summary

Edit

User group name	Creation time	ARN
IAM-Auditor-Group	December 17, 2022, 19:47 (UTC-08:00)	arn:aws:iam::176805167435:group/IAM-Auditor-Group

Users | Permissions | Access Advisor

Users in this group (2)

An IAM user is an entity that you create in AWS to represent the person or application that uses it to interact with AWS.

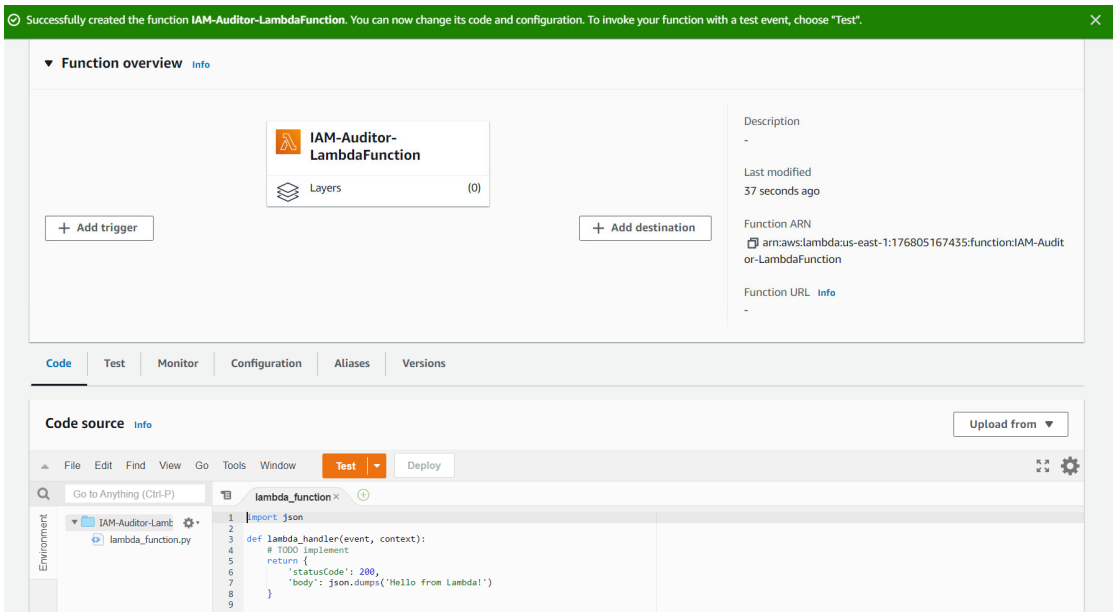
Remove users | Add users

Search

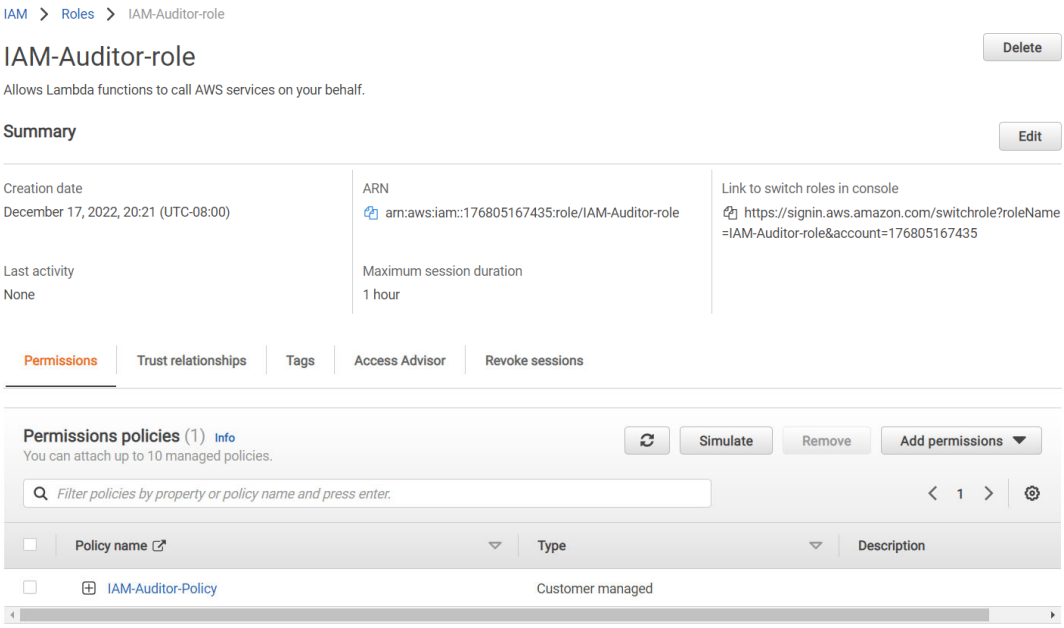
1

User name	Groups	Last activity	Creation time
User2	You need permissions	None	32 minutes ago
User3	0	22 minutes ago	29 minutes ago

Task 7 (5%): create a lambda function using python 3.9 as a runtime, and x86_64 as an Architecture.



Task 8 (10%): Create a role "IAM-Auditor-role" that can be assumed by a role of Lambda function, and attach the IAM-Auditor policy to this role.



Permissions | Trust relationships | Tags | Access Advisor | Revoke sessions

Trusted entities

Entities that can assume this role under specified conditions.

Edit trust policy

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::176805167435:role/service-role/IAM-Auditor-LambdaFunction-role-czuqucaj"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Task 9 (20%): In the Lambda function you created in Task 7:

- Define a function to assume the role that you created in task 8 ("IAM-Auditor-role").
- Define a function to generate and get IAM Credential Report.
- Provide a script that uses the first function to assume the IAM-Auditor role and the second function to generate and get the IAM credential report.

Define a function to assume the role that you created in task 8 ("IAM-Auditor-role"):

```
def assume_role():
    assumedRoleObiect=stsClient.assume_role(
    RoleArn=f"arn:aws:iam::176805167435:role/IAM-Auditor-role",
    RoleSessionName="IAMCredentialReport"
    )
    return assumedRoleObiect["Credentials"]
```

Define a function to generate and get IAM Credential Report:

```
def generate_IAM_credential_report(credentials):
    iam_client = boto3.client(
        'iam',
        aws_access_key_id=credentials["AccessKeyId"],
```

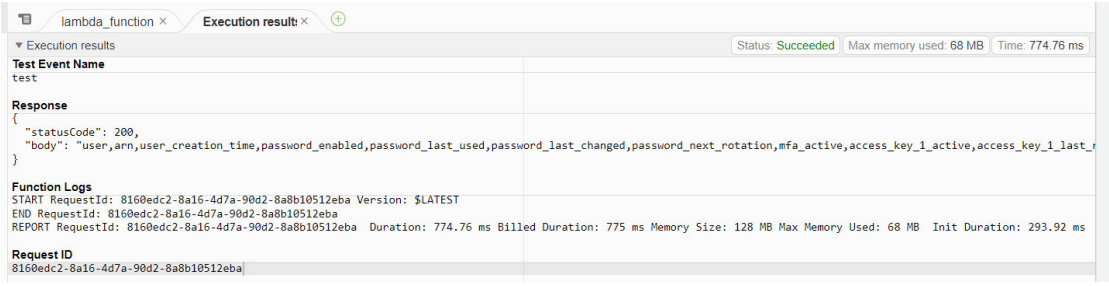
```
        aws_secret_access_key=credentials["SecretAccessKey"],
        aws_session_token=credentials["SessionToken"]
    )
    report_complete = False
    while not report_complete:
        gen_report = iam_client.generate_credential_report()
        report_complete = gen_report["State"] = "COMPLETE"
    credential_report = iam_client.get_credential_report()
    report = credential_report["Content"].decode("utf-8")
    return report
```

Provide a script that uses the first function to assume the IAM-Auditor role

and the second function to generate and get the IAM credential report:

lambda_function × Execution results ×

```
import json
import os
import boto3
import time

stsClient = boto3.client("sts")

def assume_role():
    assumedRoleObject=stsClient.assume_role(
    RoleArn=f"arn:aws:iam::176805167435:role/IAM-Auditor-role",
    RoleSessionName="IAMCredentialReport"
    )
    return assumedRoleObject["Credentials"]

def generate_IAM_credential_report(credentials):
    iam_client = boto3.client(
        'iam',
        aws_access_key_id=credentials["AccessKeyId"],
        aws_secret_access_key=credentials["SecretAccessKey"],
        aws_session_token=credentials["SessionToken"]
    )
    report_complete = False
    while not report_complete:
        gen_report = iam_client.generate_credential_report()
        report_complete = gen_report["State"] = "COMPLETE"

    credential_report = iam_client.get_credential_report()
    report = credential_report["Content"].decode("utf-8")
    return report

def lambda_handler(event, context):
    report = generate_IAM_credential_report(assume_role())
    return{
        'statusCode': 200,
        'body': report
    }
```



Task 10 (15%): In this task, you need to use the IAM credential report generated in the previous task to do a security audit. We are going to audit the AWS CIS controls 1.1 and 1.12.

- **CIS 1.1:** Based on the best practices the root user should not be used for daily activity. Create a function that reports the last time the root account has been used.
- **CIS 1.2:** Ensure multi-factor authentication (MFA) is enabled for all IAM users that have a console password. Now, create a function that reports all users with MFA disabled.
- **CIS 1.12:** We should ensure no access key is attached to the root account. Report if there is any key attached to the root account.

CIS 1.1: Based on the best practices the root user should not be used for daily activity. Create a function that reports the last time the root account has been used.

```
def cis1_1(report):
    reader = csv.DictReader(StringIO(report))
    for row in reader:
        if row["user"] == '<root_account>':
            return{
                "Last Used": row["password_last_used"]
```

```
            }
    return {"Last Used": -1}
```

CIS 1.2: Ensure multi-factor authentication (MFA) is enabled for all IAM users that have a console password. Now, create a function that reports all users with MFA disabled

```
def cis1_2(report):
    reader = csv.DictReader(StringIO(report))
    users = []
    for row in reader:
        if row["mfa_active"] == 'false' and row["password_enabled"] ==
'true':
            users.append(row["user"])
    return {"Mfa Disabled Users": users}
```

CIS 1.12: We should ensure no access key is attached to the root account. Report if there is any key attached to the root account.

```
def cis1_12(report):
    reader = csv.DictReader(StringIO(report))
    for row in reader:
        if row["user"] == '<root_account>':
            return{
                "Key Attached": (row["access_key_1_active"] == 'true' or
row["access_key_2_active"] == 'true')
            }
    return{
        "Key Attached": False
    }

def lambda_handler(event, context):
    report = generate_IAM_credential_report(assume_role())
    return{
        'statusCode': 200,
        'body': dict(cis1_1(report)| cis1_12(report)|cis1_2(report))
    }
```

Test Event Name

```
test
```

Response


```
{
  "statusCode": 200,
  "body": {
    "Last Used": "2022-12-18T03:19:01+00:00",
    "Key Attached": true,
    "Mfa Disabled Users": [
      "admin",
      "User3"
    ]
  }
}
```


Function Logs

```
START RequestId: 2b27890d-666c-4ae4-a6e4-2f0cb4204450 Version: $LATEST
END RequestId: 2b27890d-666c-4ae4-a6e4-2f0cb4204450
REPORT RequestId: 2b27890d-666c-4ae4-a6e4-2f0cb4204450  Duration: 801.54 ms Billed Duration: 802 ms Memory Size: 128 MB Max Memory Used: 68 MB  Init Duration: 330.86 ms
```

Request ID

```
2b27890d-666c-4ae4-a6e4-2f0cb4204450
```

Task 11 (10%): In this task, you need to create a simple text report from task 10 and use the SNS service to send the report to your own email address. To do so, go to Amazon Simple Notification Service (SNS) console -> create an SNS topic, and subscribe your email address to the topic. Then add SNS publish privilege to the Lambda function role. Finally, use Boto3 SNS publish function to send notifications.

Create an SNS topic:

Subscription: cbcbcc4c-8559-4aa5-8622-68bafb68afd6

Details

ARN
arn:aws:sns:us-east-1:176805167435:SendReport:cbcbcc4c-8559-4aa5-8622-68bafb68afd6

Endpoint
kda78@sfu.ca

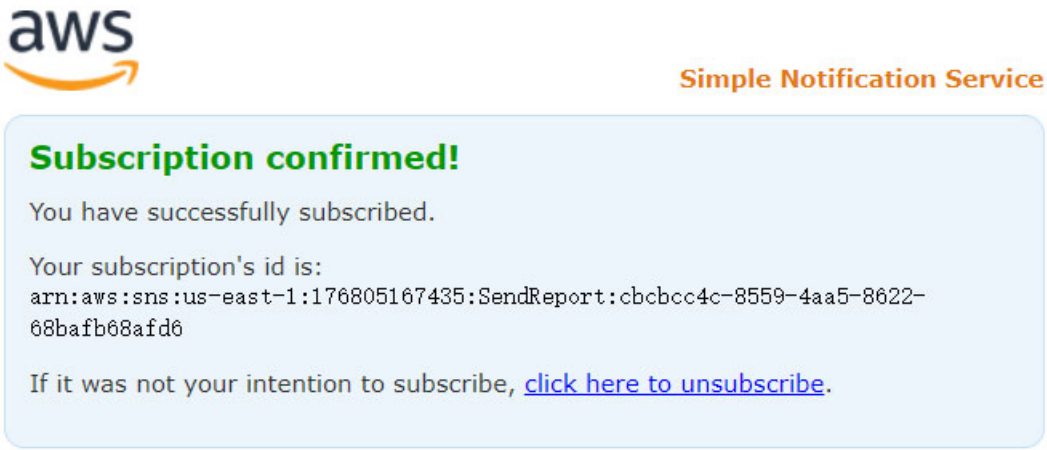
Topic
SendReport

Subscription Principal
arn:aws:iam::176805167435:root

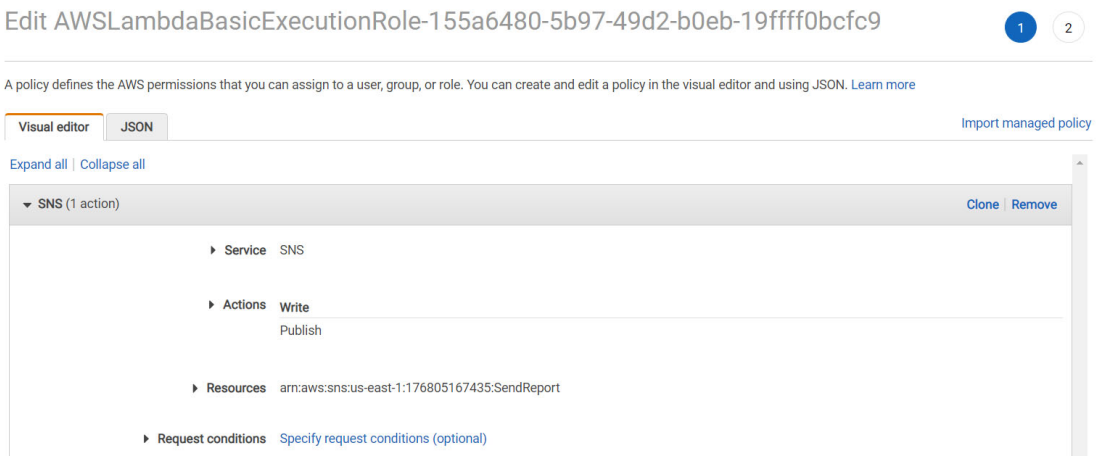
Status
Confirmed

Protocol
EMAIL

Subscribe:



Add Publish permission to lambda function:



Code:

```
def send_report(reports, credentials):
    sns_client = boto3.client('sns')
    sns_client.publish(
        TopicArn='arn:aws:sns:us-east-1:176805167435:SendReport',
        Message=json.dumps(reports),
    )
    return True


def lambda_handler(event, context):
    credentials=assume_role()
    report = generate_IAM_credential_report(credentials)
    result = dict(cis1_1(report)| cis1_12(report)|cis1_2(report))
    send_report(result,credentials)
```

```
    return{
        'statusCode': 200,
        'body': result
    }
```

lambda_function × Execution result: ×

Execution results — Status: Succeeded | Max memory used: 69 MB | Time: 1032.49 ms

Test Event Name

```
test
```

Response


```
{
  "statusCode": 200,
  "body": {
    "Last Used": "2022-12-18T03:19:01+00:00",
    "Key Attached": true,
    "Mfa Disabled Users": [
      "admin",
      "User3"
    ]
  }
}
```


Function Logs

```
START RequestId: b6dd4787-684c-4824-9645-d3fd3b9925f8 Version: $LATEST
END RequestId: b6dd4787-684c-4824-9645-d3fd3b9925f8
REPORT RequestId: b6dd4787-684c-4824-9645-d3fd3b9925f8  Duration: 1032.49 ms    Billed Duration: 1033 ms    Memory Size: 128 MB Max Memory Used: 69 MB  Init Duration: 322
```

Request ID

```
b6dd4787-684c-4824-9645-d3fd3b9925f8
```

Email:

AWS Notification Message

AN **AWS Notifications <no-reply@sns.amazonaws.com>**
23:35

收件人: kda78@sfu.ca

{"Last Used": "2022-12-18T03:19:01+00:00", "Key Attached": true, "Mfa Disabled Users": ["admin", "User3"]}

--
If you wish to stop receiving notifications from this topic, please click or visit the link below to unsubscribe:
https://sns.us-east-1.amazonaws.com/unsubscribe.html?SubscriptionArn=arn:aws:sns:us-east-1:176805167435:SendReport:cbcbcc4c-8559-4aa5-8622-68bafb68afd6&Endpoint=kda78@sfu.ca

Please do not reply directly to this email. If you have any questions or comments regarding this email, please contact us at https://aws.amazon.com/support

Task 12 (5%): AWS Event bridge enables scheduling events to trigger AWS services such as the Lambda function. Explain the steps that need to be taken to receive a daily report from a script that you created in Task 9.

Create a schedule:

Step 1: Schedule detail

Edit

Schedule detail

Schedule name	Description	Schedule group
SendDailyReport	-	default
Timezone	**Occurrence**	**Start date and time**
(UTC -08:00) America/Vancouver	Recurring	-
End date and time	**Flexible time window**	
-	Off	

Cron expression

0	12	*	*	?	*
Minutes	Hours	Day of month	Month	Day of week	Year

Next 10 trigger dates

Date and time are displayed in the selected time zone for which this schedule is set in UTC format, e.g. "Wed, Nov 9, 2022 09:00 (UTC -08:00)"

Sun, 18 Dec 2022 12:00:00 (UTC -08:00)
Mon, 19 Dec 2022 12:00:00 (UTC -08:00)
Tue, 20 Dec 2022 12:00:00 (UTC -08:00)
Wed, 21 Dec 2022 12:00:00 (UTC -08:00)
Thu, 22 Dec 2022 12:00:00 (UTC -08:00)
Fri, 23 Dec 2022 12:00:00 (UTC -08:00)
Sat, 24 Dec 2022 12:00:00 (UTC -08:00)
Sun, 25 Dec 2022 12:00:00 (UTC -08:00)
Mon, 26 Dec 2022 12:00:00 (UTC -08:00)
Tue, 27 Dec 2022 12:00:00 (UTC -08:00)

Set the target:

Step 2: Target

Edit

Target detail

Target	Target ARN
AWS Lambda IAM-Auditor-LambdaFunction	arn:aws:lambda:us-east-1:176805167435:function:IAM-Auditor-LambdaFunction

Payload

{}

Task 13 (10%): Use AWS Cloudtrial-> event history to query the last time the root user logged in.

the last time the root user logged in: December 17, 2022, 19:19:01 (UTC-08:00)

CloudTrail > Event history

Event history (15) Info

Download events | Create Athena table

Event history shows you the last 90 days of management events.

Lookup attributes

Event name | ConsoleLogin | 30m | 1h | 3h | 12h | Custom | 1

Event name	Event time	User name	Event source	Resource type
ConsoleLogin	December 17, 2022, 19:19:01 (UTC-08:00)	root	signin.amazonaws.com	-
ConsoleLogin	November 30, 2022, 13:24:26 (UTC-08:00)	root	signin.amazonaws.com	-
ConsoleLogin	November 29, 2022, 18:27:47 (UTC-08:00)	root	signin.amazonaws.com	-