ATTACK DEFENSE

by PentesterAcademy

Name	AirIAM
URL	https://attackdefense.com/challengedetails?cid=2499
Type	AWS Cloud Security : Defense

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.

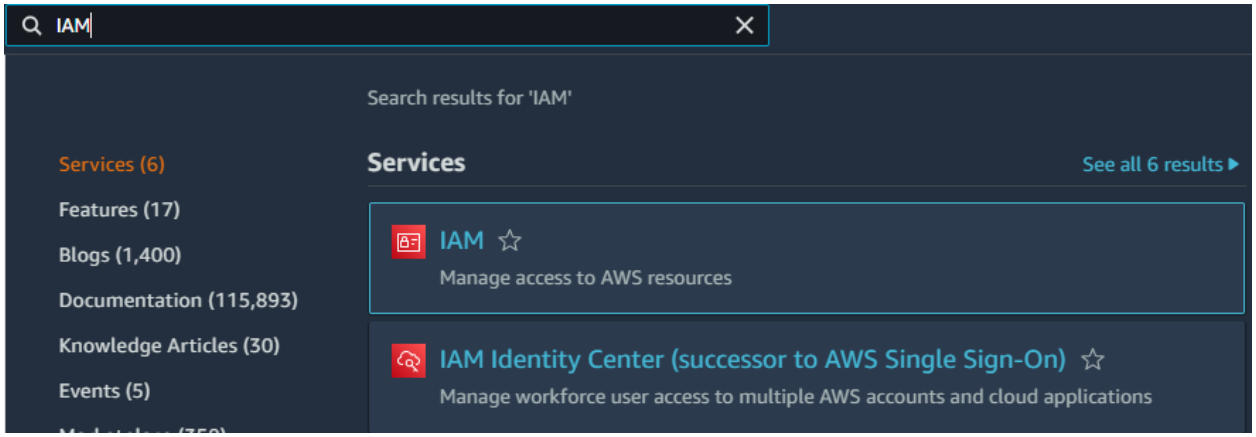
Solution:

Step 1: Click the lab link button to get access credentials.

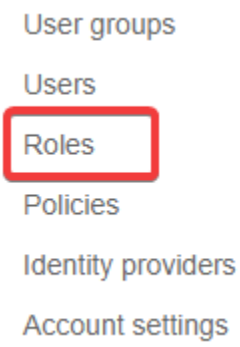
Access Credentials to your AWS lab Account

Login URL	https://413056973906.signin.aws.amazon.com/console
Region	US East (N. Virginia) us-east-1
Username	student
Password	Ad4tmyDl7mrjroDB
Access Key ID	AKIAWALA5HRJODZEO44W
Secret Access Key	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

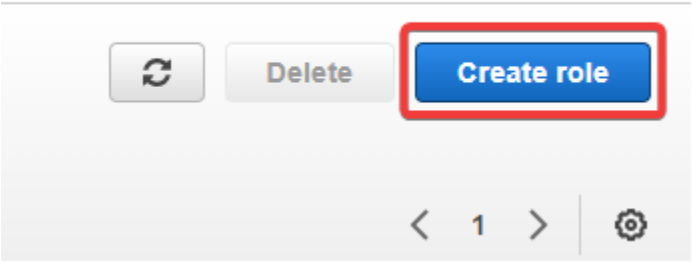
Step 2: Search for IAM in the search bar and navigate to the IAM dashboard.



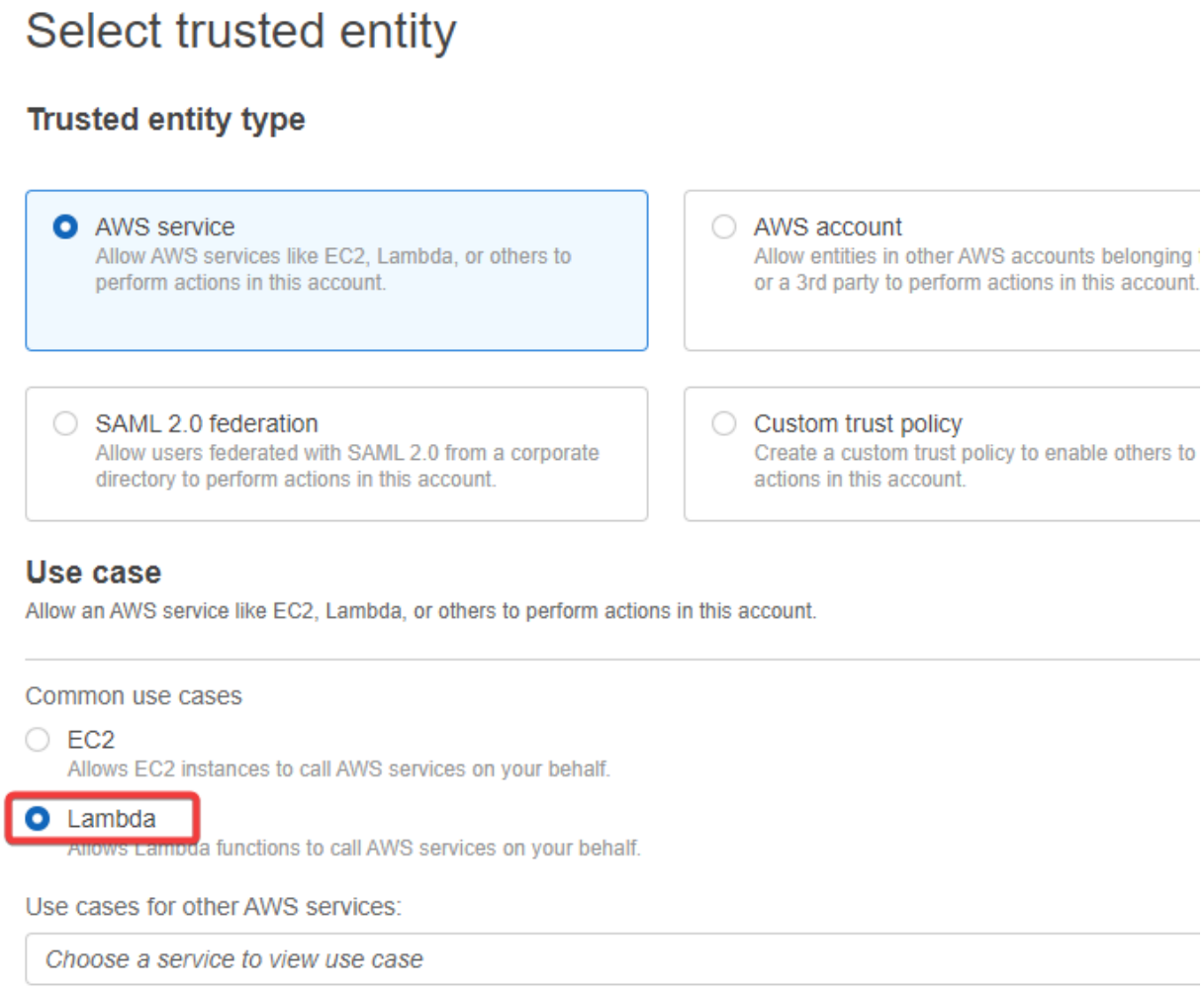
Step 3: Click on “Roles” from the left navigation pane.



Step 4: Click on the “Create role” button.

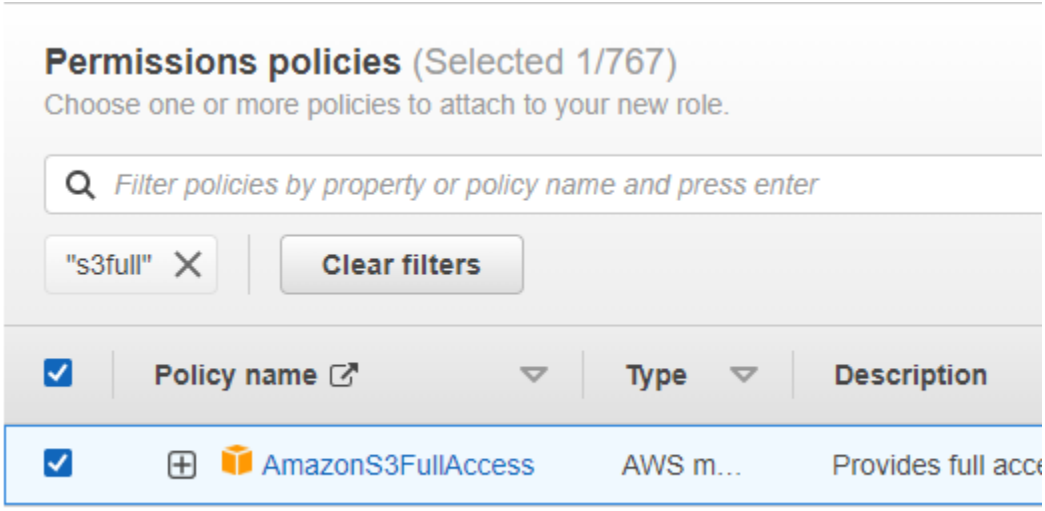


Step 5: Select trusted entity type as AWS service and use case as Lambda.

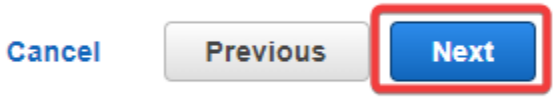


Click on the “Next” button.

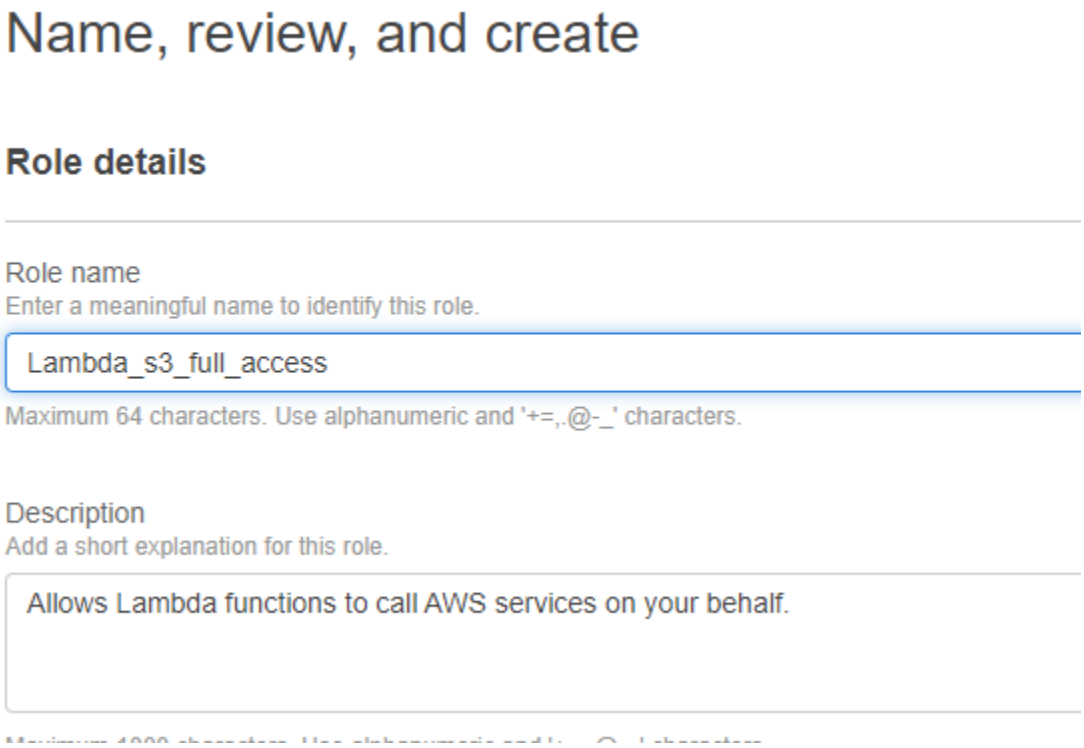
Step 6: Search for “S3full” in the policies and select “AmazonS3FullAccess”.



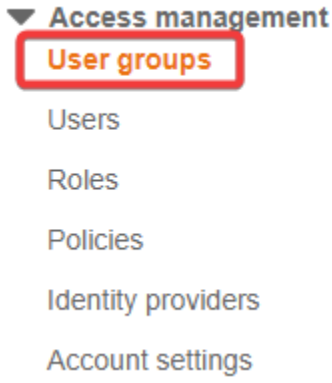
Step 7: Click on the “Next” button.



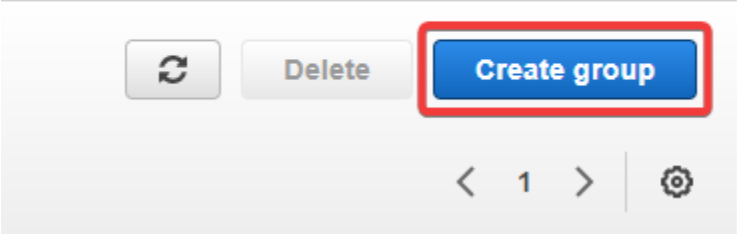
Step 8: Set the role name as “Lambda_s3_full_access” and click on the “Create” button.



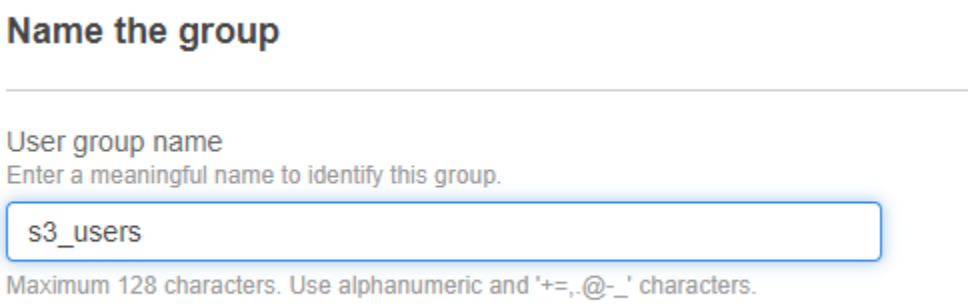
Step 9: Click on “User groups” from the left navigation pane.



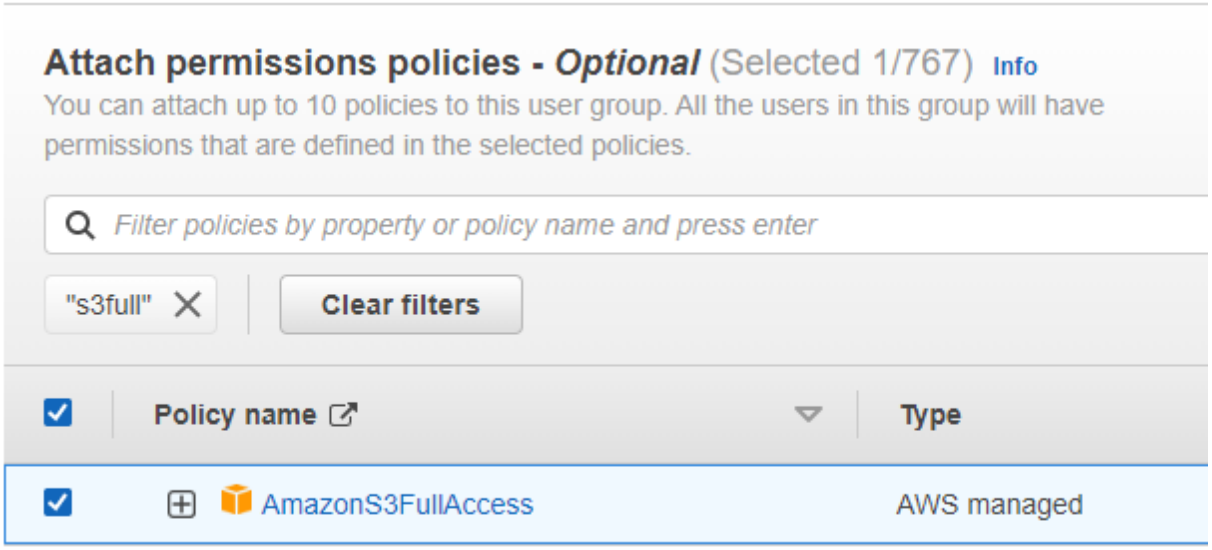
Step 10: Click on the “Create group” button.



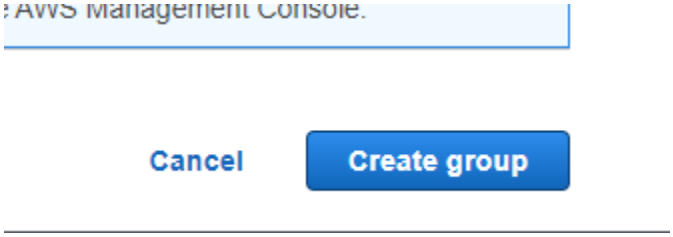
Step 11: Set the group name as “s3_users”.



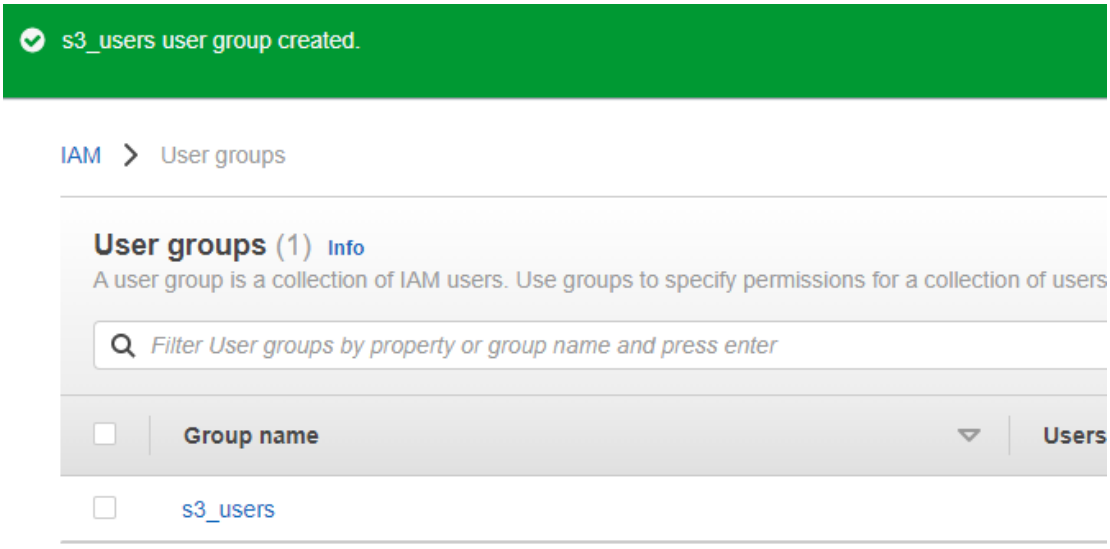
Step 12: Search “s3full” in policies and select “AmazonS3FullAccess”.



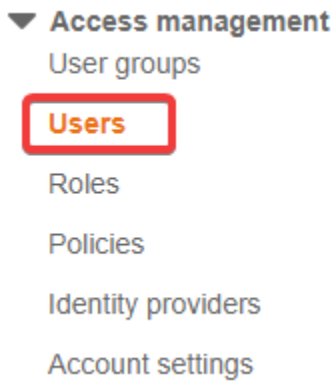
Step 13: Click on the “Create group” button.



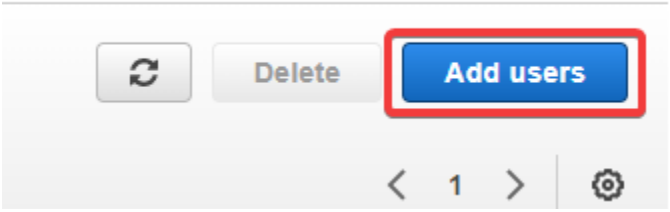
Successfully created the group.



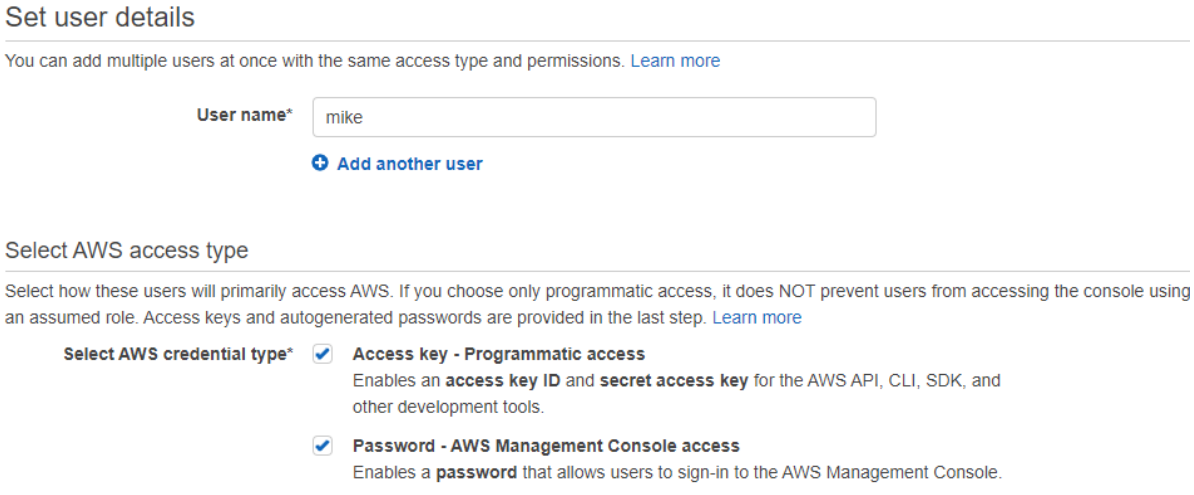
Step 14: Click on “Users”.



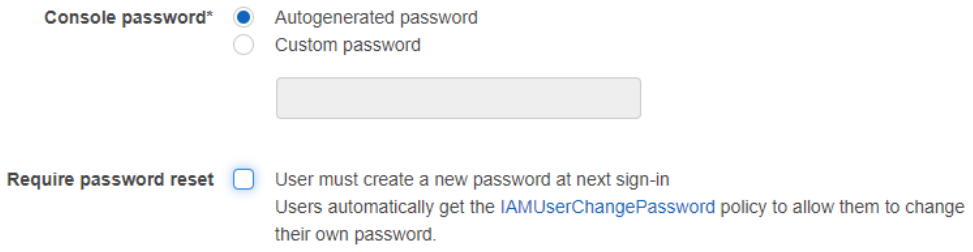
Step 15: Click on “Add users”.



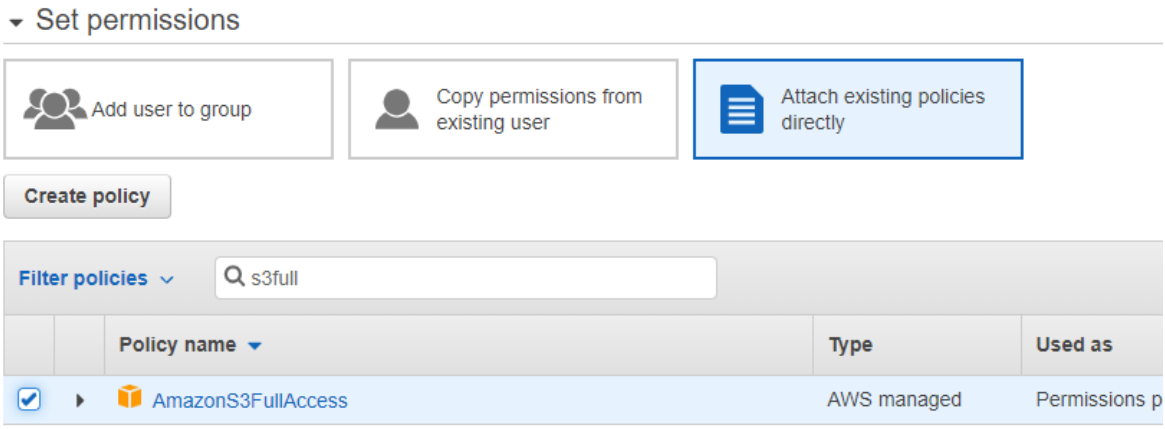
Step 16: Set the user name as mike enable access key and password.



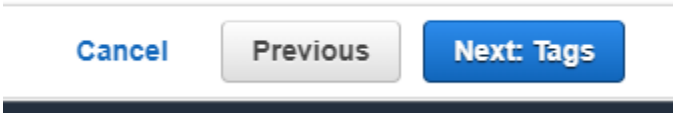
Step 17: Set the console password as auto generated.



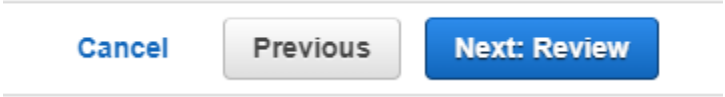
Step 18: Attach policy by choosing “Attach existing policies directly” and search for “s3full” and select “AmazonS3FullAccess” policy.



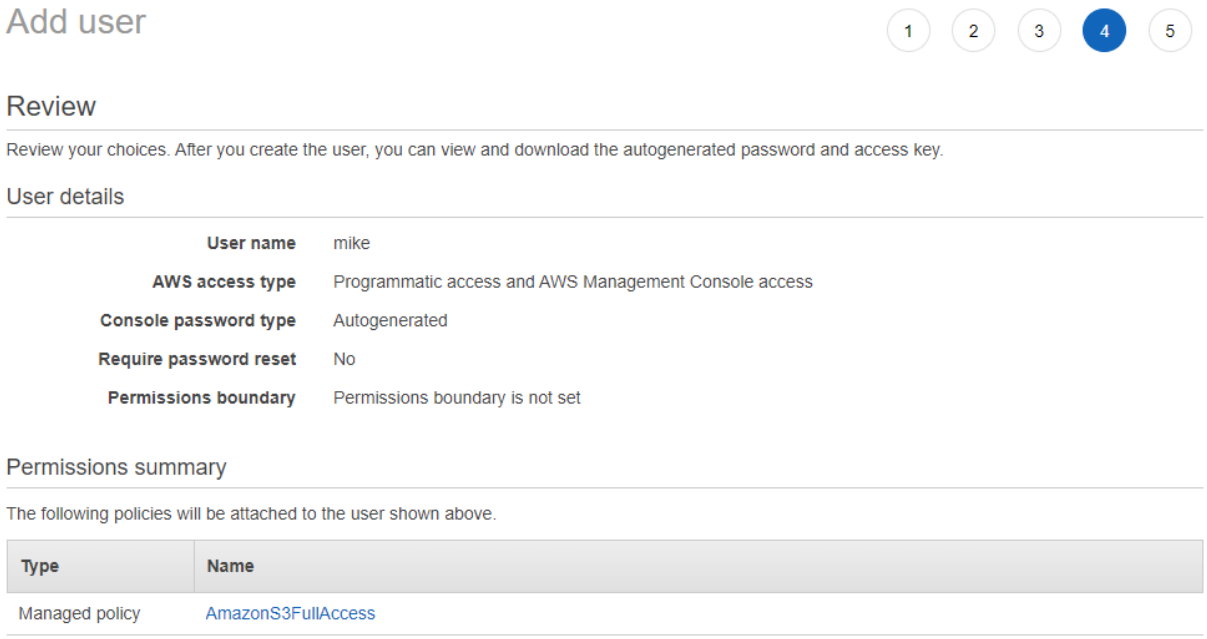
Click on the “Next button”.



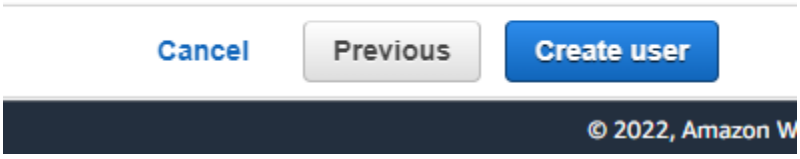
Click on the “Next button”.



Review the configurations for the created user.



Click on the “Create user” button.



Step 19: Switch to root user.

Command: sudo su

```
┌──(kali㉿kali)-[~/labs]
└─$ sudo su
```

Step 20: Configure the AWS CLI using the provided credentials.

It must have AWS credentials configured that can be used by the CLI to use AirIAM.

Command: aws configure

```
┌──(root㉿kali)-[/home/kali/labs]
└─# aws configure
AWS Access Key ID [****************4T4N]: AKIAWALA5HRJODZEO44W
AWS Secret Access Key [****************JitL]: 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
Default region name [us-east-1]:
Default output format [None]:
```

Step 21: Install AirIAM.

Command: pip3 install airiam

AirIAM is an AWS IAM to least privileged Terraform execution framework. It compiles AWS IAM usage and leverages that data to create a least-privileged IAM Terraform that replaces the existing IAM management method. AirIAM is written in Python and aims to simplify and increase the adoption of infrastructure code for IAM management.

```
┌──(root㉿kali)-[/home/kali/labs]
└─# pip3 install airiam
Collecting airiam
  Downloading airiam-0.1.83-py3-none-any.whl (38 kB)
Requirement already satisfied: colorama==0.4.3 in /usr/l
Requirement already satisfied: termcolor==1.1.0 in /usr/
Requirement already satisfied: boto3>=1.12.43 in /usr/lo
Requirement already satisfied: python-terraform==0.10.1
Requirement already satisfied: requests>=2.22.0 in /usr/
```

Step 22: Explore AirIAM commands.

Command: airiam -h

It will provide the command usage as the following.

1) find_unused - Scan your runtime IAM for unused entities
2) recommend_groups - Recommend IAM groups according to IAM users and their in-use privileges
3) terraform - Terraformize your runtime AWS IAM configurations

```
┌──(root㉿kali)-[/home/kali/labs]
└─# airiam -h

     ____      __          _____      ____     __        __
    / __ \    |__|  _  ____|___|_    _|    / __ \   |  \    /  |
   / /  \ \    __  | |/ ___| | |     / /  \ \  | |\ \  / /| |
  / /____\ \  |  | | |   /     | |    / /____\ \ | | \ V / | |
 /  ______  \_|  |_|  |     _| |_  /  ______  \  |  \  /  | |
/_/        \_\_____|__|    |_____|/_/        \_\_|   V   |_|
v0.1.83

AirIAM - Least privilege AWS IAM Terraformer

To continuously scan configurations, try the Bridgecrew free community plan.
https://www.bridgecrew.io

usage: airiam [-h] [-v] {find_unused,recommend_groups,terraform} ...

options:
  -h, --help            show this help message and exit
  -v, --version         Get AirIAM's version (default: False)

commands:
  {find_unused,recommend_groups,terraform}
    find_unused         Scan your runtime IAM for unused entities
    recommend_groups    Recommend IAM groups according to IAM users and their in-use privileges
    terraform           Terraformize your runtime AWS IAM configurations
```

Step 23: Find the unused IAM entities.

Command: airiam find_unused

```
┌──(root㉿kali)-[/home/kali/labs]
└─# airiam find_unused
```

It will check all the entities available on the AWS account.

```
AirIAM - Least privilege AWS IAM Terraformer

To continuously scan configurations, try the Bridgecrew free community plan.
https://www.bridgecrew.io

INFO:botocore.credentials:Found credentials in shared credentials file: ~/.aws/credentials
Getting all IAM configurations for account 413056973906
Getting IAM credential report
Generating usage reports for 5 principals
Generating report for arn:aws:iam::413056973906:user/identity
Generating report for arn:aws:iam::413056973906:user/mike
Generating report for arn:aws:iam::413056973906:user/student
Generating report for arn:aws:iam::413056973906:role/Lambda_s3_full_access
Generating report for arn:aws:iam::413056973906:role/TheOracle
Received reports for 5 principals
Collecting password configurations for all IAM users in the account
Completed data collection, writing to local file...
Identifying unused IAM entities in the account...
```

It will provide an overview similar to this.

```
The following 2 users were found to be unused:
Unused: identity: Never used!
Unused: mike: Never used!

No unused access keys were found in the account! Hurray!

No unused Console Login Profiles were found in the account! Hurray!

The following 1 roles are unused:
Unused: Lambda_s3_full_access: Never used!

The following 1 groups are redundant:
s3_users has no members

No unattached policies were found in the account! Hurray!

No unused policy attachments were found in the account! Hurray!

If you prefer to to change the current runtime and not move to IaC b
 at:
https://www.bridgecrew.io/
```

Here we have two users, one role and one group as unused.

Step 24: Check the AirIAM terraform options.

Command: airiam terraform -h

```
┌──(root㉿kali)-[/home/kali/labs]
└─# airiam terraform -h

v0.1.83

AirIAM - Least privilege AWS IAM Terraformer

To continuously scan configurations, try the Bridgecrew free community plan.
https://www.bridgecrew.io

usage: airiam terraform [-h] [-p PROFILE] [-d DIRECTORY] [--without-unused] [--without-groups] [-l LAST_U

options:
  -h, --help            show this help message and exit
  -p PROFILE, --profile PROFILE
                        AWS profile to be used (default: None)
  -d DIRECTORY, --directory DIRECTORY
                        Path where the output terraform code and state will be stored (default: results)
  --without-unused      Create terraform code without unused entities (default: False)
  --without-groups      Create terraform code without recommendation for user groups (default: False)
  -l LAST_USED_THRESHOLD, --last-used-threshold LAST_USED_THRESHOLD
                        "Last Used" threshold, in days, for an entity to be considered unused (default: 9
  --no-cache            Generate a fresh set of data from AWS IAM API calls (default: False)
  --without-import      Import the resulting entities to terraform's state file. Note - this might take a
```

Step 25: Create a terraform for current AWS account configuration without unused entities using AirIAM.

```
┌──(root㉿kali)-[/home/kali/labs]
└─# airiam terraform --without-unused

     ____      __          _____      ____     __        __
    / __ \    |__|  _  ____|_   _|    / __ \   |  \     /  |
   / /  \ \    __  | |/ ___| | |     / /  \ \  | |\ \   / /| |
  / /____\ \  |  | | |   /     | |    / /____\ \ | | \ \v / | |
 /  ______  \_|  |_| |  |     _| |_  /  ______  \ |  \  /  | |
/_/        \_\_____|__|    |_____|/_/        \_\_|   \/   |_|
v0.1.83

AirIAM - Least privilege AWS IAM Terraformer

To continuously scan configurations, try the Bridgecrew free community plan.
https://www.bridgecrew.io

INFO:botocore.credentials:Found credentials in shared credentials file: ~/.aws/credentials
Reusing local data
Filtered arn:aws:iam::413056973906:user/student from the analysis
INFO:root:Analyzing data for account 413056973906
INFO:root:Using the default UserOrganizer
WARNING:root:Migrating to the recommended groups isn't supported yet. Issue exists - https://github.com
WARNING:root:Will use the existing groups for terraform migration until it is implemented
WARNING:root:Filtering out 4 entities from terraform. These entities will have to be handled manually
Initializing terraform
```

Successfully migrated current IAM setup into terraform.

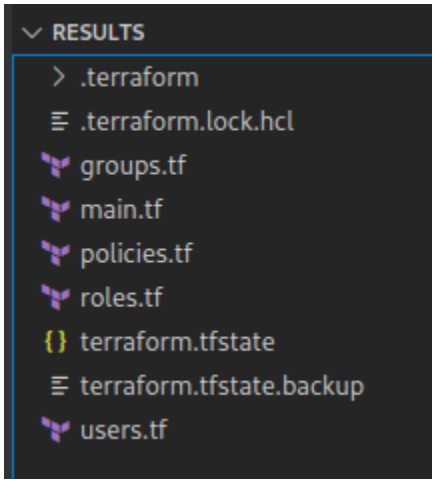
```
#5 of 5: Importing TheOracle/arn:aws:iam::aws:policy/AdministratorAccess to aws_iam_role_policy_attachment.T
existing entities to state
Successfully migrated your current IAM setup to terraform!
Migrated 1 users, 0 groups, 1 roles and 6 policies, as well as all connections between them, to terraform.
Your terraform files can now be found at the directory you specified: results
```

Step 26: Navigate to the results directory and list the files.

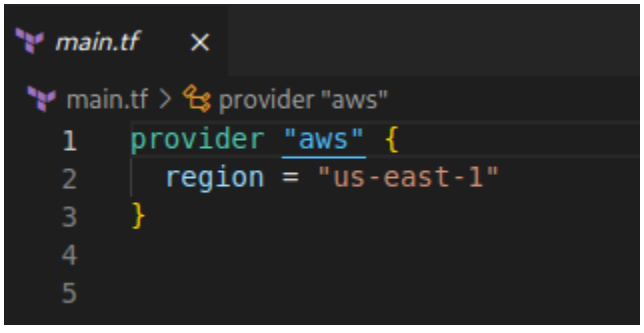
```
┌──(root㉿kali)-[/home/kali/labs]
└─# cd results

┌──(root㉿kali)-[/home/kali/labs/results]
└─# ls
groups.tf  main.tf  policies.tf  roles.tf  terraform.tfstate  terraform.tfstate.backup  users.tf
```

Step 27: Open results directory with a text editor.



Step 28: Check out the main.tf file. It will have a provider set with a region.



Step 29: All the roles configurations will be available at roles.tf file.

roles.tf

roles.tf > resource "aws_iam_role" "Lambda_s3_full_access"

```
resource "aws_iam_role" "Lambda_s3_full_access" {
  name        = "Lambda_s3_full_access"
  path        = "/"
  description = "Allows Lambda functions to call AWS services on your behalf."

  assume_role_policy = data.aws_iam_policy_document.Lambda_s3_full_access_assume_role_policy_document.json

  tags = {
    "Managed by"      = "AirIAM by Bridgecrew"
    "Managed through" = "Terraform"
  }
}

data "aws_iam_policy_document" "Lambda_s3_full_access_assume_role_policy_document" {
  version = "2012-10-17"
  statement {
    effect  = "Allow"
    actions = ["sts:AssumeRole"]
    principals {
      type        = "Service"
      identifiers = ["lambda.amazonaws.com"]
    }
  }
}

resource "aws_iam_role_policy_attachment" "Lambda_s3_full_access_AmazonS3FullAccess_managed" {
```

Step 30: All the users configurations will be available at users.tf file.

users.tf

users.tf > resource "aws_iam_user" "identity"

```
resource "aws_iam_user" "identity" {
  name          = "identity"
  path          = "/"
  force_destroy = true

  tags = {
    "pa_userid"       = "411eb8edf21678eef808d2c8b1b9a305"
    "Managed by"      = "AirIAM by Bridgecrew"
    "Managed through" = "Terraform"
  }
}
resource "aws_iam_user_group_membership" "identity_group_attachment" {
  user = aws_iam_user.identity.name

  groups = []
}
resource "aws_iam_user" "mike" {
  name          = "mike"
  path          = "/"
  force_destroy = true

  tags = {
    "Managed by"      = "AirIAM by Bridgecrew"
    "Managed through" = "Terraform"
  }
}
```

After generating the terraform form the existing AWS account without unused entities , it can be applied to a new account resulting in a configuration without unused entities.

References:

1. AirIAM (https://airiam.io/)