Terraform AWS Sample: IAM

By Ronald Stewart Lim

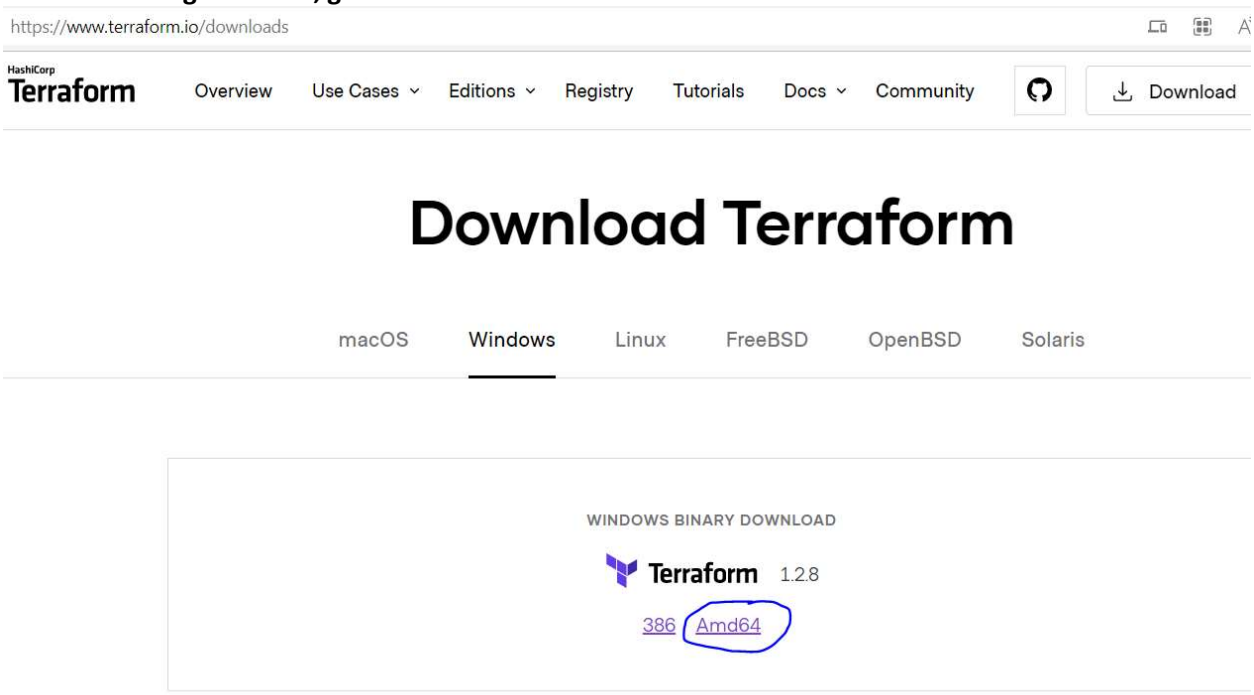
Terraform AWS pre-requisites:

- Install Terraform and define in Windows environment path if it is in Windows OS
- Install AWS CLI
- Generate Access Key

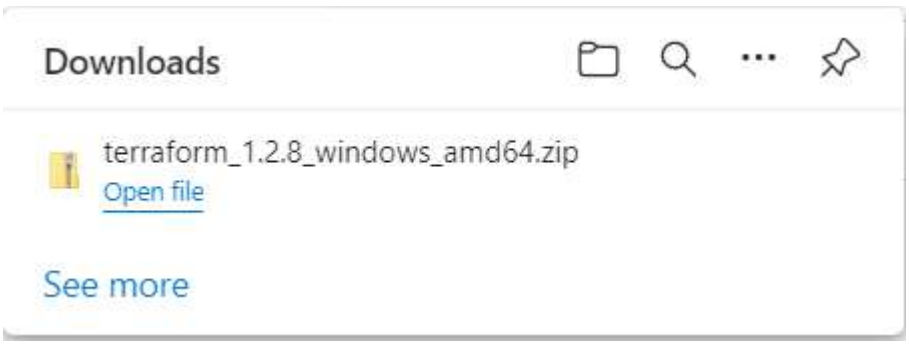
Pre-requisite: Install Terraform

Download terraform exe file in terraform site:
https://www.terraform.io/downloads

Since I am using windows, go to windows tab and click Amd64



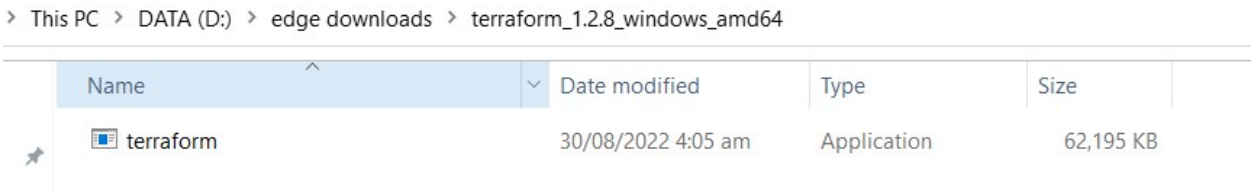
Then it will download



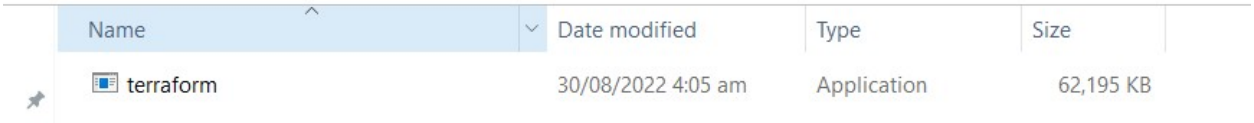
Terraform AWS Sample: IAM

By Ronald Stewart Lim

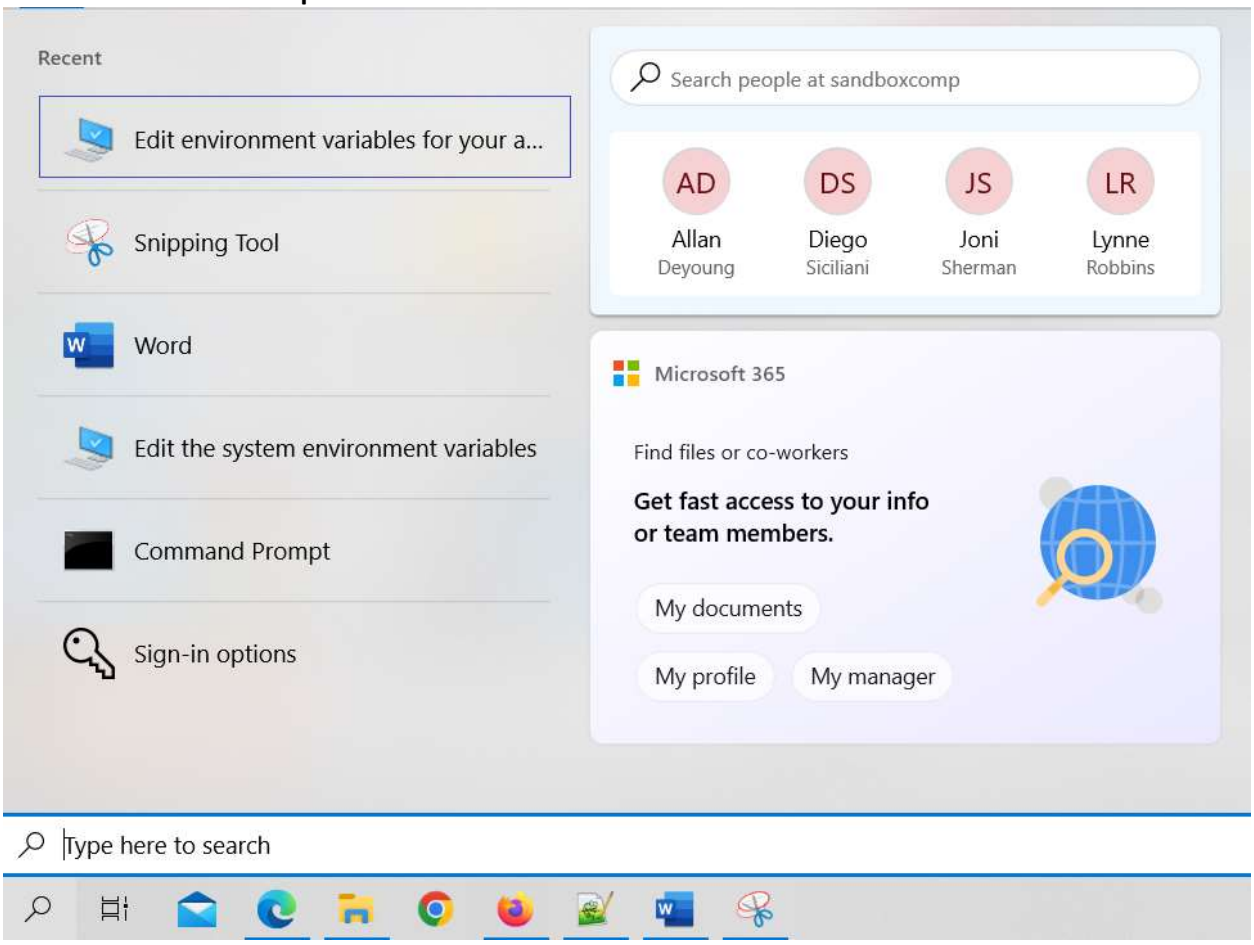
Then extract file you can see terraform executable file



Then create a directory and copy the file



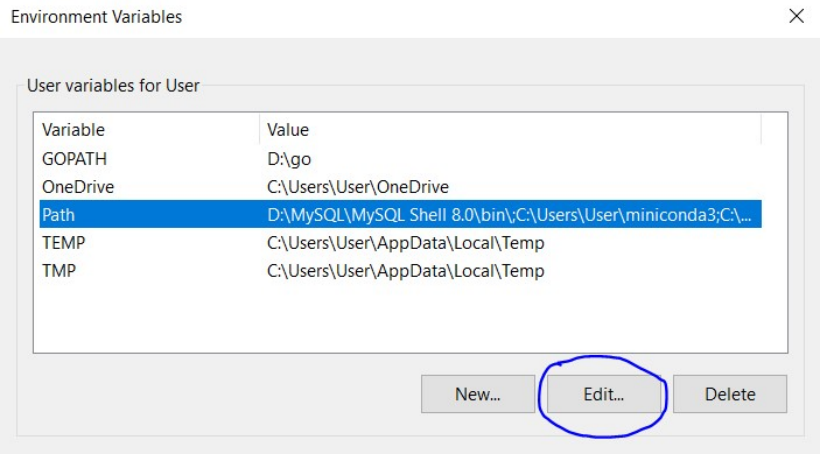
Then edit windows user path variable



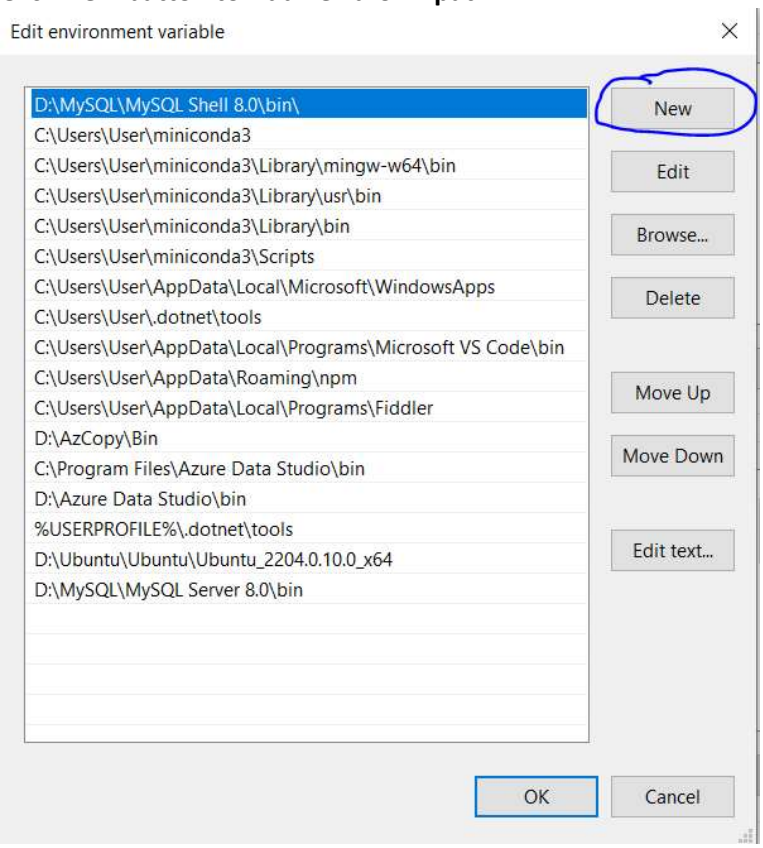
Go to Path variable and click Edit… button

Terraform AWS Sample: IAM

By Ronald Stewart Lim

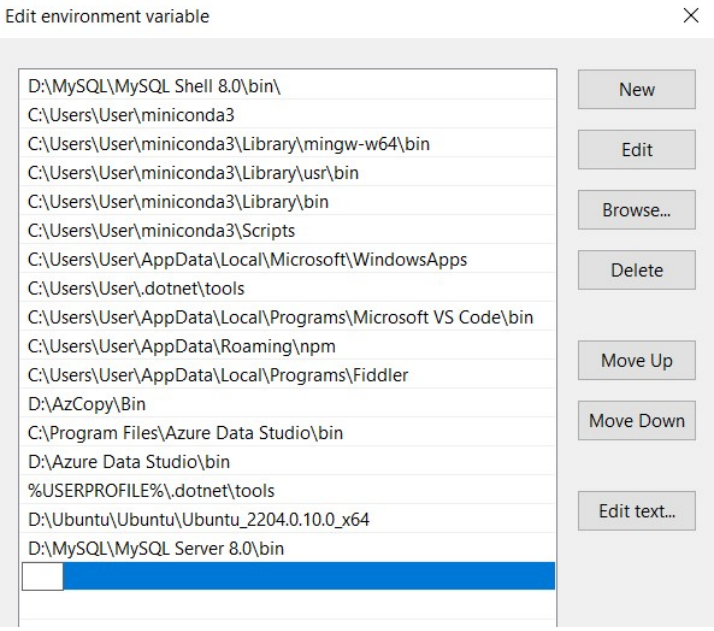


Click New button to Add Terraform path

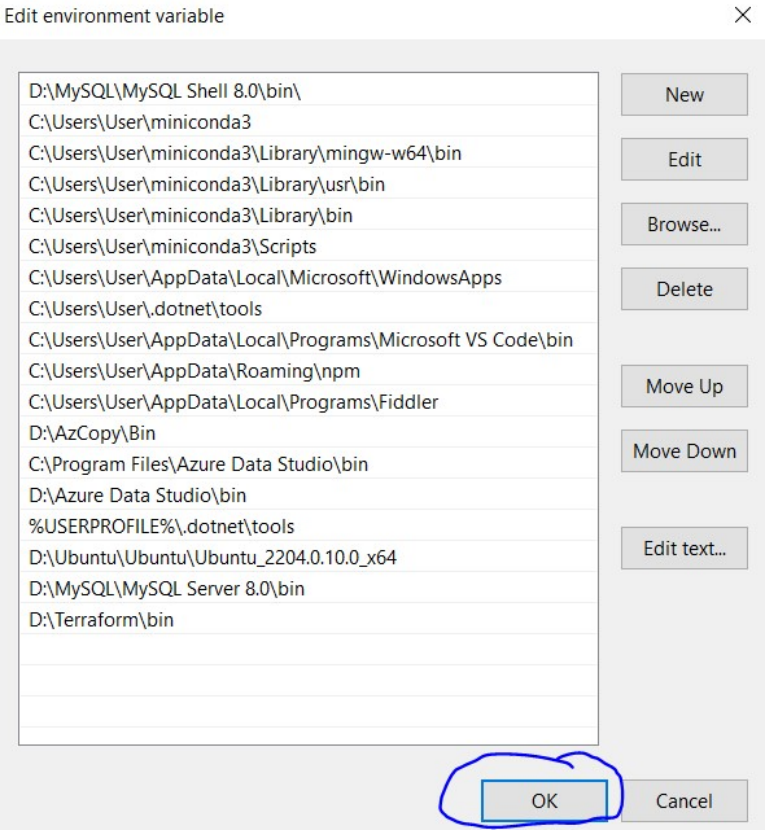


Terraform AWS Sample: IAM

By Ronald Stewart Lim



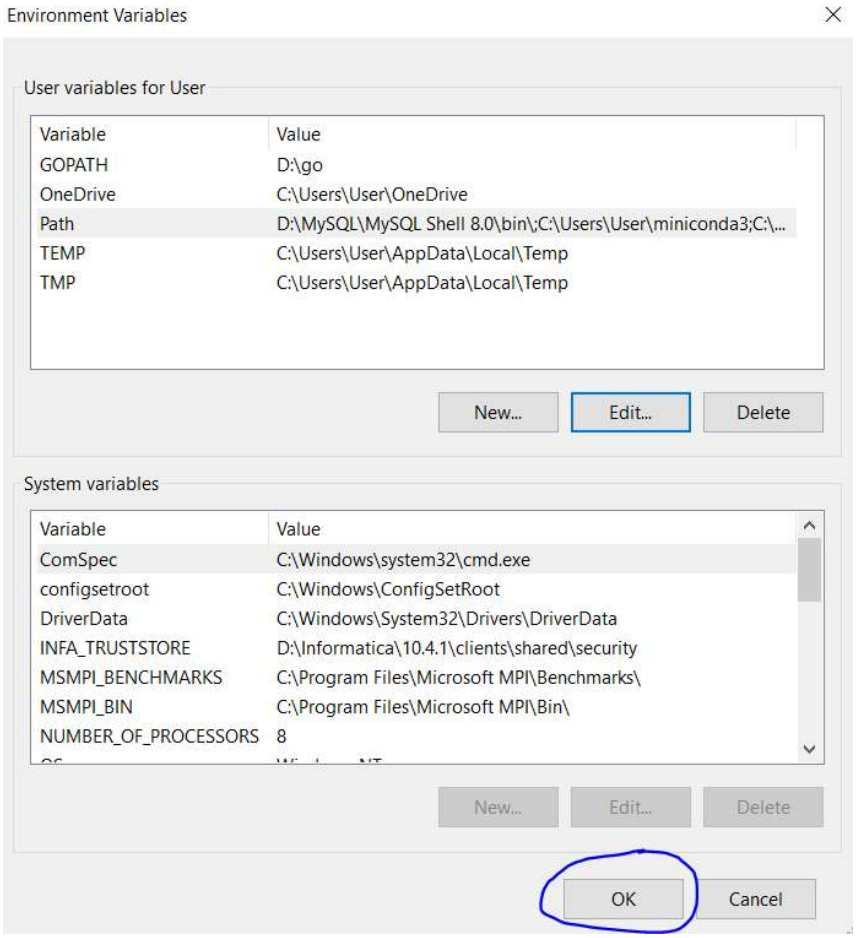
Click OK Button after specifying the path



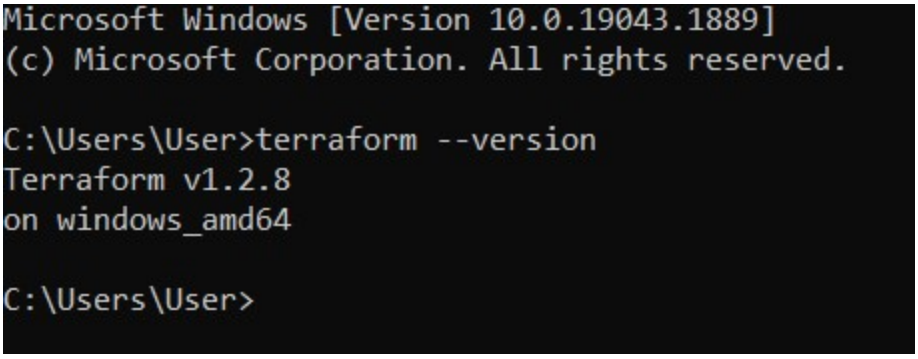
Click OK Button again

Terraform AWS Sample: IAM

By Ronald Stewart Lim



Open cmd and try to check if terraform is already accessible in any directory by typing: ***terraform --version***



Pre-requisite: Install AWS CLI

Terraform AWS Sample: IAM

By Ronald Stewart Lim

Now that Terraform environment is ready, please install aws cli and follow instructions by using the link below:
https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html

▼ **Windows**

Installation requirements

- We support the AWS CLI on Microsoft-supported versions of 64-bit Windows.
- Admin rights to install software

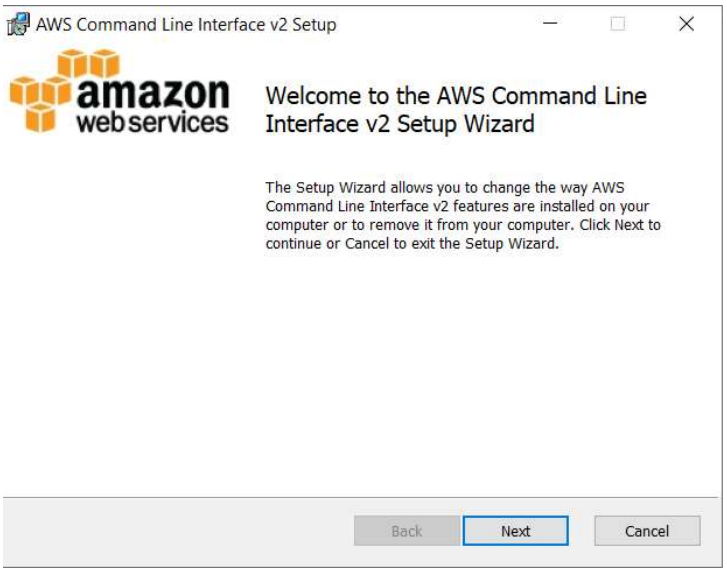
Install or update the AWS CLI

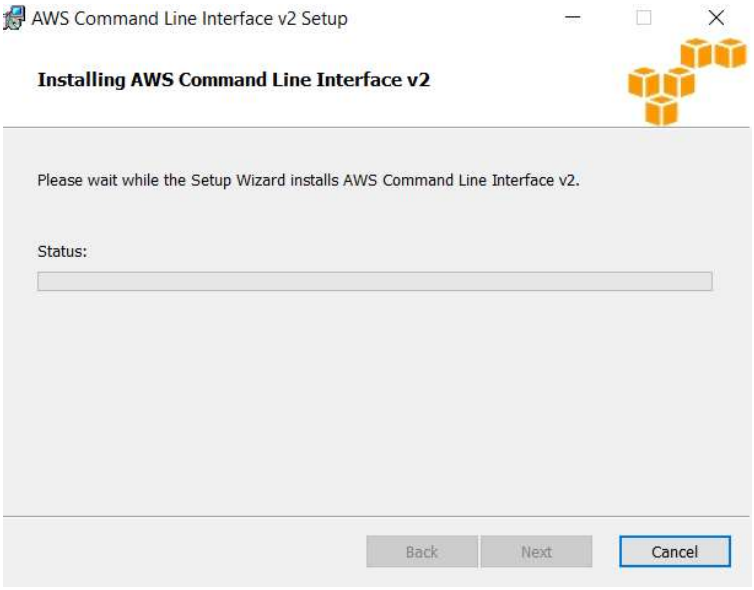
To update your current installation of AWS CLI on Windows, download a new installer each time you update to overwrite previous versions. AWS CLI is updated regularly. To see when the latest version was released, see the AWS CLI changelog on *GitHub*.

1. Download and run the AWS CLI MSI installer for Windows (64-bit):
   https://awscli.amazonaws.com/AWSCLIV2.msi
   Alternatively, you can run the `msiexec` command to run the MSI installer.

Terraform AWS Sample: IAM

By Ronald Stewart Lim





Terraform AWS Sample: IAM

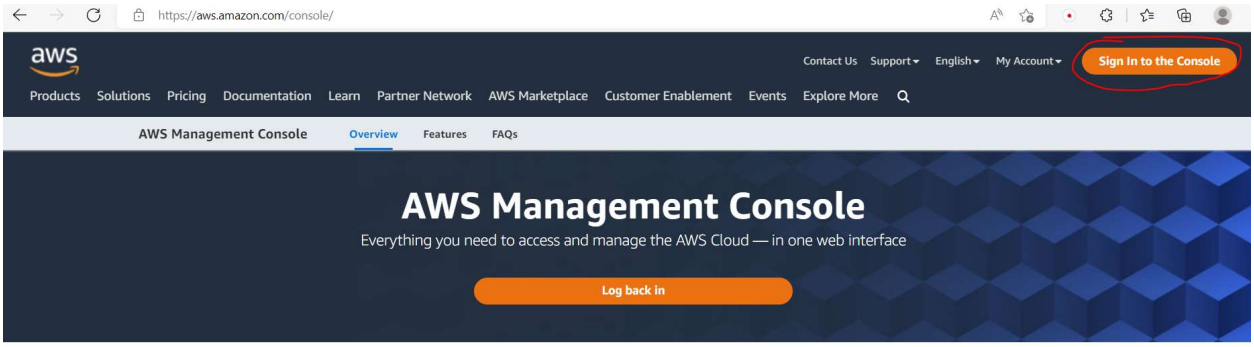
By Ronald Stewart Lim


```
Microsoft Windows [Version 10.0.19043.1889]
(c) Microsoft Corporation. All rights reserved.

C:\Users\User>aws --version
aws-cli/2.7.27 Python/3.9.11 Windows/10 exe/AMD64 prompt/off

C:\Users\User>
```

Pre-requisite: Generate Access Key

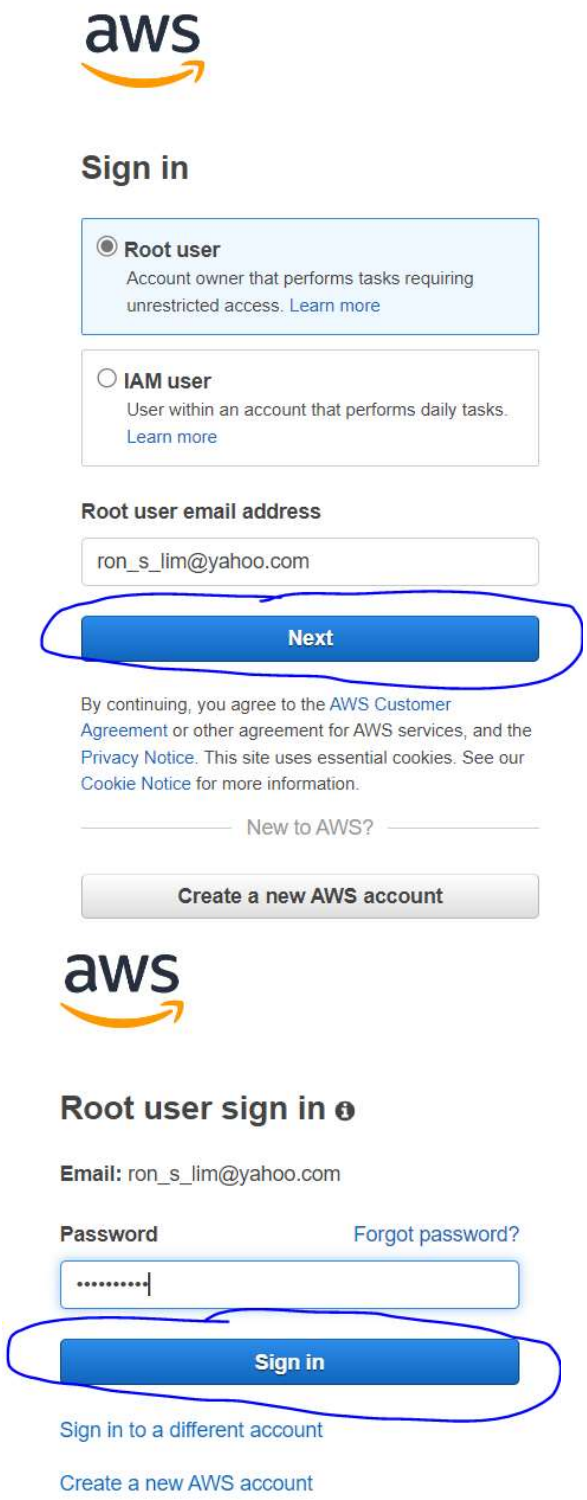
Now both Terraform and AWS CLI environment are ready. We will prepare the AWS accesses required for executing Terraform scripts

Sign in to AWS Console as root user
Link: https://aws.amazon.com/console/



Terraform AWS Sample: IAM

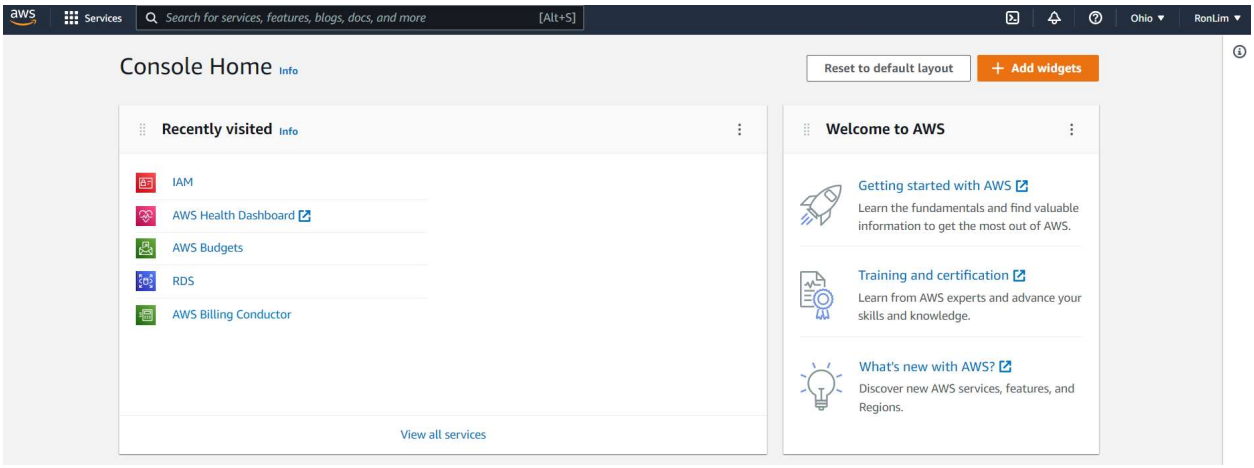
By Ronald Stewart Lim



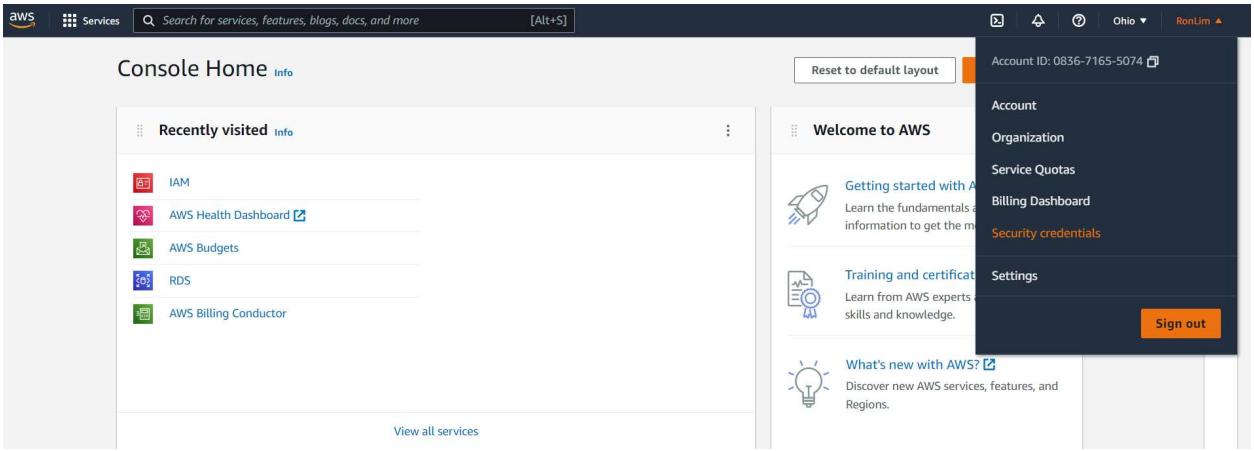
After signing in, you will go to your console home

Terraform AWS Sample: IAM

By Ronald Stewart Lim



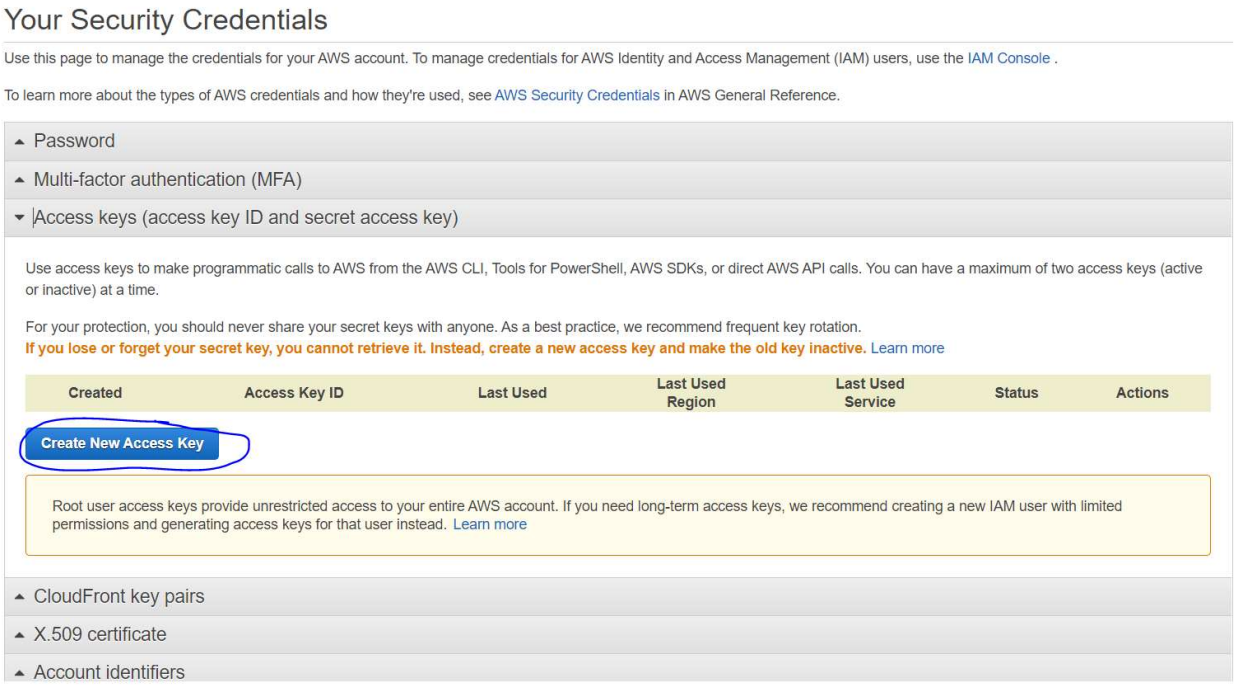
Click username or dropdown button to access Security credentials



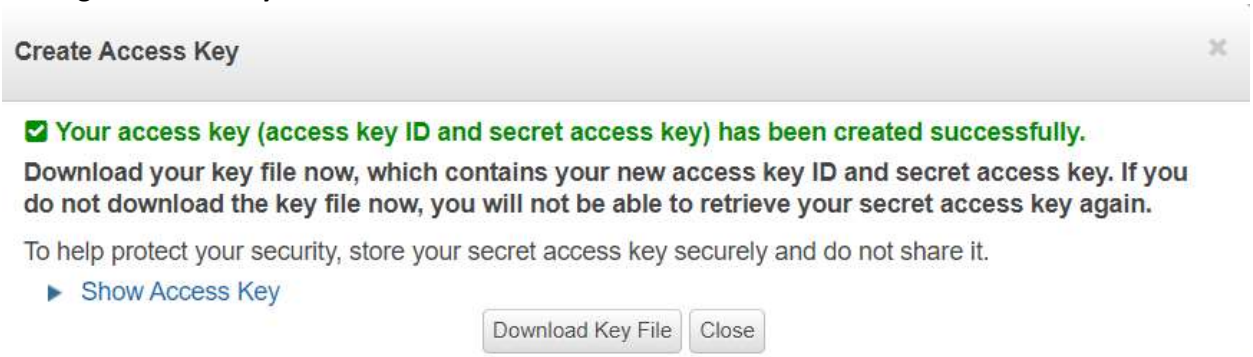
Expand Access keys and click Create New Access Key button

Terraform AWS Sample: IAM

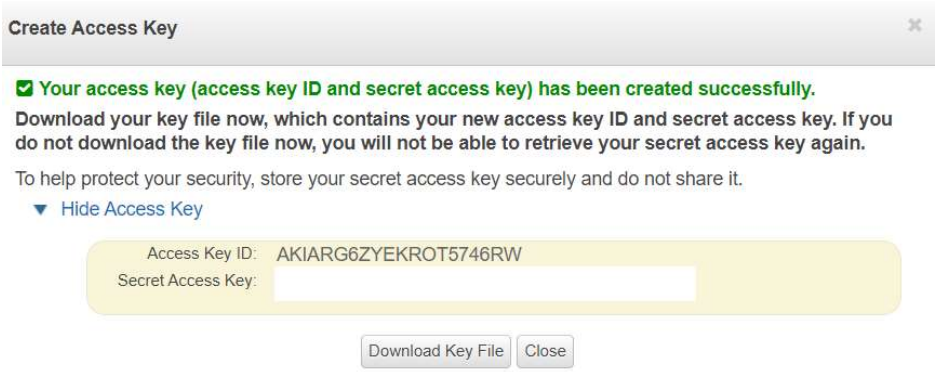
By Ronald Stewart Lim



After that, Access Key was created. You can either expand the Access Key or download the Key File by clicking Download Key File button.



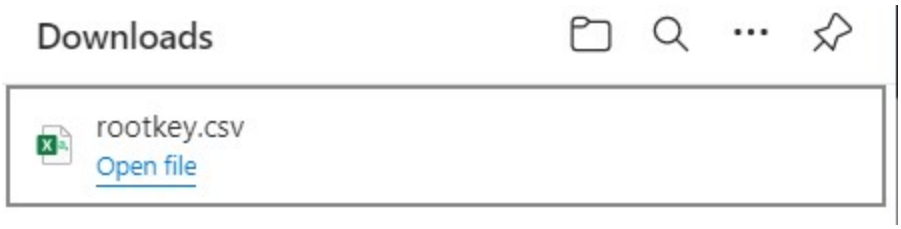
Using Show Access Key (Secret Access Key was not shown in screenshot):



Or By Downloading csv Access Key File by clicking Download Key File button:

Terraform AWS Sample: IAM

By Ronald Stewart Lim



Terraform Implementation

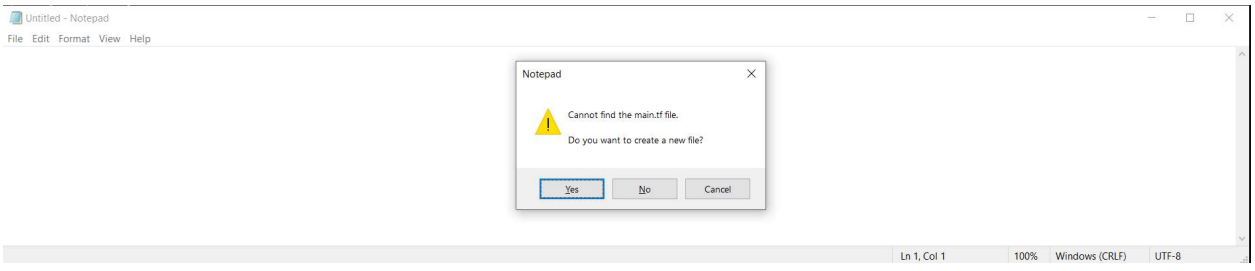
Now we are ready for implementing Terraform AWS. For the succeeding steps, we will setup the IAM Users and Group and associate the users to the group.

Open cmd and configure the region and AWS Access Keys using command below:
aws configure

```
C:\Users\User>aws configure
AWS Access Key ID [****************UJVT]: AKIARG6ZYEKROT5746RW
AWS Secret Access Key [****************HhVv]:
Default region name [us-east-1]:
Default output format [None]:

C:\Users\User>
```

Then go to the path where you will plan to execute Terraform and edit the main.tf. If file does not exists then create the file

```
D:\Terraform\terraform_aws\IAM\Basics>notepad main.tf

D:\Terraform\terraform_aws\IAM\Basics>
```

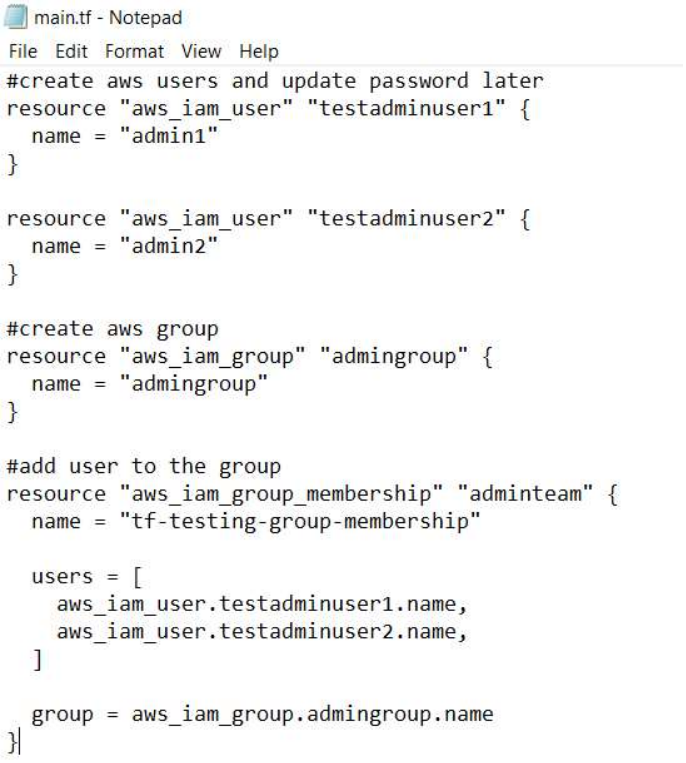


Terraform AWS Sample: IAM

By Ronald Stewart Lim

Then edit main.tf with the Terraform-based language or code and save changes. The main.tf file contains the terraform code that will create 2 admin users and create admin group and associate the admin users to admin group.

Reference: https://registry.terraform.io/providers/hashicorp/aws/latest/docs



After that close the file and initialize the Terraform using command below:

terraform init

Terraform AWS Sample: IAM

By Ronald Stewart Lim

```
D:\Terraform\terraform_aws\IAM\Basics>terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v4.28.0...
- Installed hashicorp/aws v4.28.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.

D:\Terraform\terraform_aws\IAM\Basics>
```

(Optional) You can change the main.tf file into terraform-based format by command below:
terraform fmt

If no output means that the main.tf file is already in terraform format

```
D:\Terraform\terraform_aws\IAM\Basics>terraform fmt

D:\Terraform\terraform_aws\IAM\Basics>
```

If there is an output, the output mentions the file that was converted to terraform format

```
D:\Terraform\terraform_aws\IAM\Basics>terraform fmt
main.tf

D:\Terraform\terraform_aws\IAM\Basics>
```

To check and validate the terraform-based codes, execute the command below:
terraform validate

If the validation found no errors, then it will output the success message

```
D:\Terraform\terraform_aws\IAM\Basics>terraform validate
Success! The configuration is valid.


D:\Terraform\terraform_aws\IAM\Basics>
```

Terraform AWS Sample: IAM

By Ronald Stewart Lim

(Optional) Check the Terraform execution plan by using the command below:
terraform plan
Sample output below:

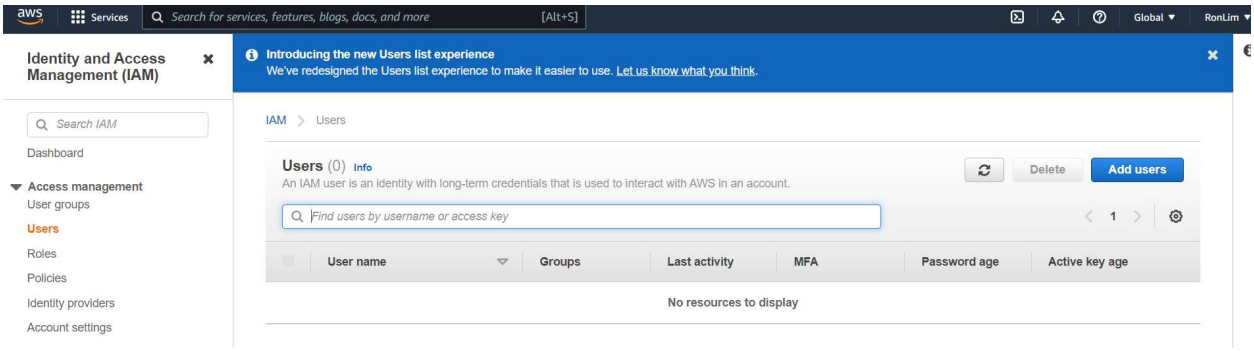
```
D:\Terraform\terraform_aws\IAM\Basics>terraform plan

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # aws_iam_group.admingroup will be created
  + resource "aws_iam_group" "admingroup" {
      + arn       = (known after apply)
      + id        = (known after apply)
      + name      = "admingroup"
      + path      = "/"
      + unique_id = (known after apply)
    }

  # aws_iam_group_membership.adminteam will be created
  + resource "aws_iam_group_membership" "adminteam" {
      + group = "admingroup"
      + id    = (known after apply)
      + name  = "tf-testing-group-membership"
      + users = [
          + "admin1",
          + "admin2",
        ]
    }

  # aws_iam_user.testadminuser1 will be created
  + resource "aws_iam_user" "testadminuser1" {
      + arn           = (known after apply)
      + force_destroy = false
      + id            = (known after apply)
      + name          = "admin1"
      + path          = "/"
      + tags_all      = (known after apply)
      + unique_id     = (known after apply)
    }
```

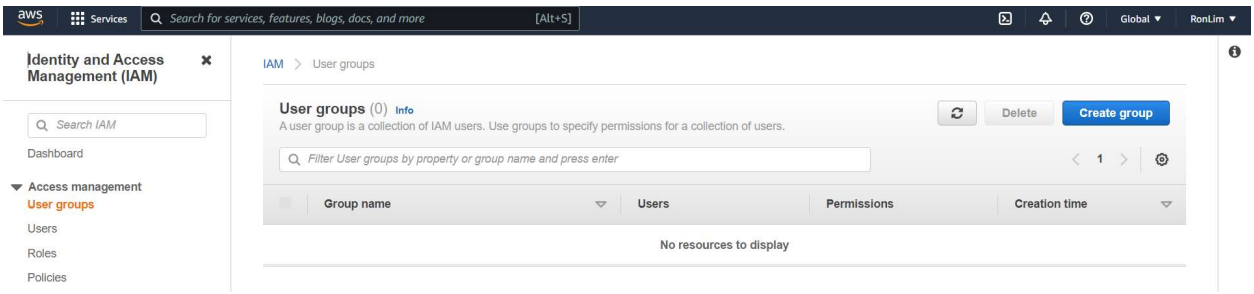
```
  # aws_iam_user.testadminuser2 will be created
  + resource "aws_iam_user" "testadminuser2" {
      + arn           = (known after apply)
      + force_destroy = false
      + id            = (known after apply)
      + name          = "admin2"
      + path          = "/"
      + tags_all      = (known after apply)
      + unique_id     = (known after apply)
    }

Plan: 4 to add, 0 to change, 0 to destroy.

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform apply" now.

D:\Terraform\terraform_aws\IAM\Basics>
```

Before applying changes, check the before state of the IAM user and group by going to AWS -> IAM -> user and AWS -> IAM -> group. Notice that the users and group do not exist yet

Terraform AWS Sample: IAM

By Ronald Stewart Lim





Execute the command below and confirm changes to create users and groups in AWS:
terraform apply

Note: terraform apply command already contains terraform plan so no need to execute terraform plan
terraform apply = terraform plan + confirmation to apply changes

sample output:

Terraform AWS Sample: IAM

By Ronald Stewart Lim

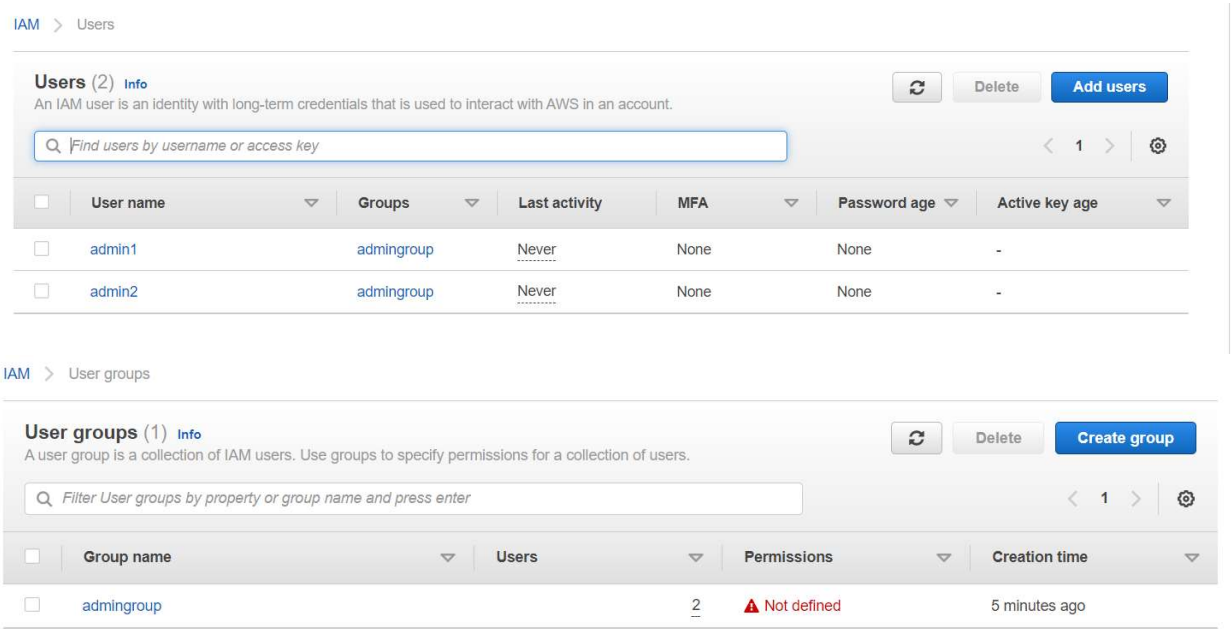
```
D:\Terraform\terraform_aws\IAM\Basics>terraform apply

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # aws_iam_group.admingroup will be created
  + resource "aws_iam_group" "admingroup" {
      + arn       = (known after apply)
      + id        = (known after apply)
      + name      = "admingroup"
      + path      = "/"
      + unique_id = (known after apply)
    }

  # aws_iam_group_membership.adminteam will be created
  + resource "aws_iam_group_membership" "adminteam" {
      + group = "admingroup"
      + id    = (known after apply)
      + name  = "tf-testing-group-membership"
      + users = [
          + "admin1",
          + "admin2",
        ]
    }

  # aws_iam_user.testadminuser1 will be created
  + resource "aws_iam_user" "testadminuser1" {
      + arn           = (known after apply)
      + force_destroy = false
      + id            = (known after apply)
      + name          = "admin1"
      + path          = "/"
      + tags_all      = (known after apply)
      + unique_id     = (known after apply)
    }
```

```
  # aws_iam_user.testadminuser2 will be created
  + resource "aws_iam_user" "testadminuser2" {
      + arn           = (known after apply)
      + force_destroy = false
      + id            = (known after apply)
      + name          = "admin2"
      + path          = "/"
      + tags_all      = (known after apply)
      + unique_id     = (known after apply)
    }

Plan: 4 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_iam_user.testadminuser2: Creating...
aws_iam_group.admingroup: Creating...
aws_iam_user.testadminuser1: Creating...
aws_iam_user.testadminuser1: Creation complete after 1s [id=admin1]
aws_iam_group.admingroup: Creation complete after 1s [id=admingroup]
aws_iam_user.testadminuser2: Creation complete after 1s [id=admin2]
aws_iam_group_membership.adminteam: Creating...
aws_iam_group_membership.adminteam: Creation complete after 1s [id=tf-testing-group-membership]

Apply complete! Resources: 4 added, 0 changed, 0 destroyed.

D:\Terraform\terraform_aws\IAM\Basics>
```

After applying changes, refresh the page and check again the state of the IAM user by going to AWS -> IAM -> user and AWS -> IAM -> group. Notice that the users and group now exist

Terraform AWS Sample: IAM

By Ronald Stewart Lim



Then let us try to destroy what we created for learning purposes using command below and confirm:
terraform destroy

sample output:

```
D:\Terraform\terraform_aws\IAM\Basics>terraform destroy
aws_iam_user.testadminuser1: Refreshing state... [id=admin1]
aws_iam_user.testadminuser2: Refreshing state... [id=admin2]
aws_iam_group.admingroup: Refreshing state... [id=admingroup]
aws_iam_group_membership.adminteam: Refreshing state... [id=tf-testing-group-membership]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  - destroy

Terraform will perform the following actions:

  # aws_iam_group.admingroup will be destroyed
  - resource "aws_iam_group" "admingroup" {
      - arn       = "arn:aws:iam::083671655074:group/admingroup" -> null
      - id        = "admingroup" -> null
      - name      = "admingroup" -> null
      - path      = "/" -> null
      - unique_id = "AGPARG6ZYEKRE72B46VRO" -> null
    }

  # aws_iam_group_membership.adminteam will be destroyed
  - resource "aws_iam_group_membership" "adminteam" {
      - group = "admingroup" -> null
      - id    = "tf-testing-group-membership" -> null
      - name  = "tf-testing-group-membership" -> null
      - users = [
          - "admin1",
          - "admin2",
        ] -> null
    }

  # aws_iam_user.testadminuser1 will be destroyed
  - resource "aws_iam_user" "testadminuser1" {
      - arn           = "arn:aws:iam::083671655074:user/admin1" -> null
      - force_destroy = false -> null
      - id            = "admin1" -> null
      - name          = "admin1" -> null
      - path          = "/" -> null
      - tags          = {} -> null
      - tags_all      = {} -> null
      - unique_id     = "AIDARG6ZYEKRD7PN6APMX" -> null
    }
```

Terraform AWS Sample: IAM

By Ronald Stewart Lim

```
  # aws_iam_user.testadminuser2 will be destroyed
  - resource "aws_iam_user" "testadminuser2" {
      - arn           = "arn:aws:iam::083671655074:user/admin2" -> null
      - force_destroy = false -> null
      - id            = "admin2" -> null
      - name          = "admin2" -> null
      - path          = "/" -> null
      - tags          = {} -> null
      - tags_all      = {} -> null
      - unique_id     = "AIDARG6ZYEKRCHDAMHVYE" -> null
    }

Plan: 0 to add, 0 to change, 4 to destroy.

Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value: yes

aws_iam_group_membership.adminteam: Destroying... [id=tf-testing-group-membership]
aws_iam_group_membership.adminteam: Destruction complete after 2s
aws_iam_group.admingroup: Destroying... [id=admingroup]
aws_iam_user.testadminuser2: Destroying... [id=admin2]
aws_iam_user.testadminuser1: Destroying... [id=admin1]
aws_iam_group.admingroup: Destruction complete after 0s
aws_iam_user.testadminuser2: Destruction complete after 1s
aws_iam_user.testadminuser1: Destruction complete after 1s

Destroy complete! Resources: 4 destroyed.

D:\Terraform\terraform_aws\IAM\Basics>
```

After destroying changes, refresh the page and check again the state of the IAM user by going to AWS -> IAM -> user and AWS -> IAM -> group. Notice that the users and group created earlier no longer exist