

Deep Chavan

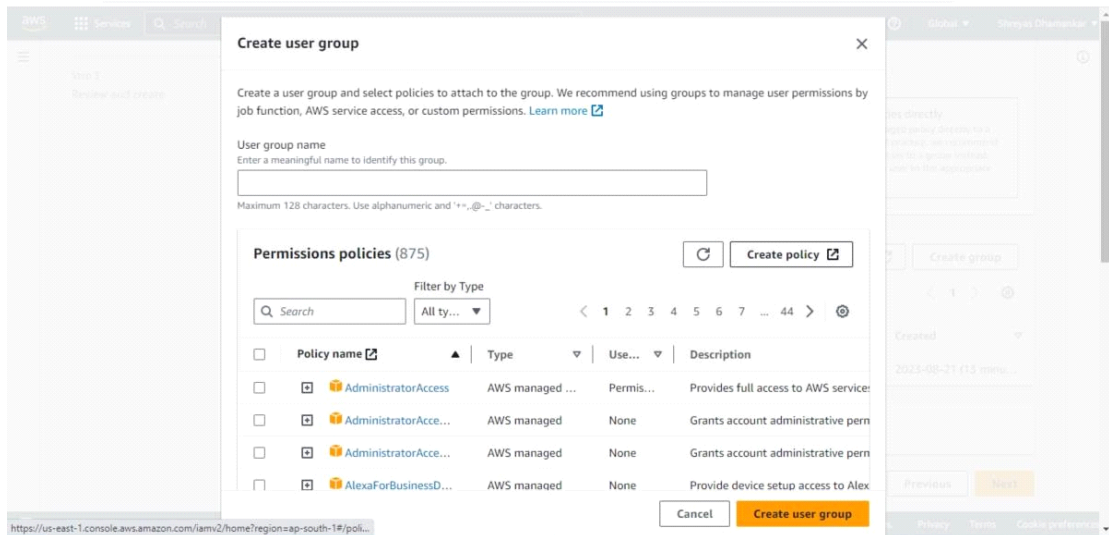
T11-15

Assignment No. 5

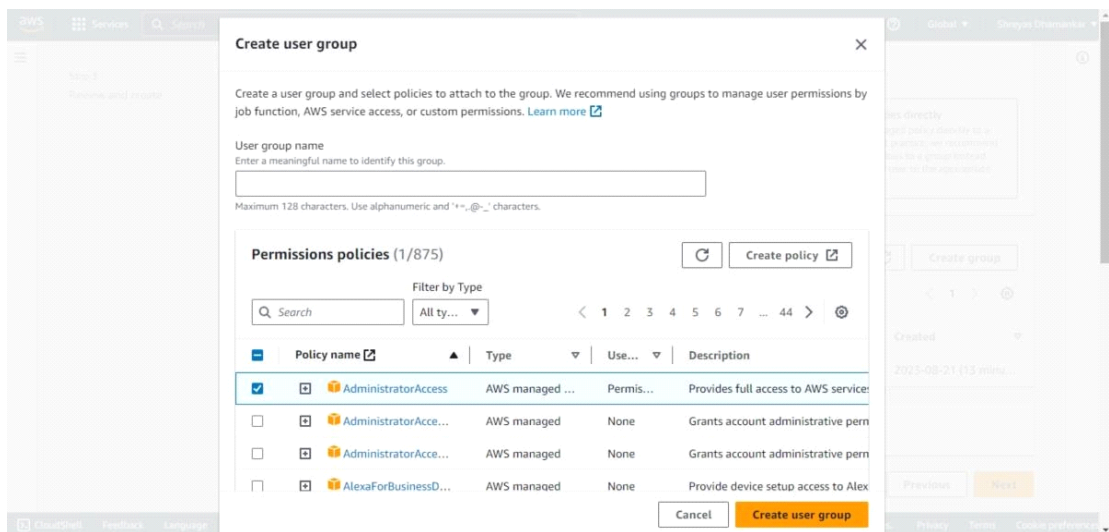
Aim: To understand the concept of terraform and how to use it to create an instance.

Theory:

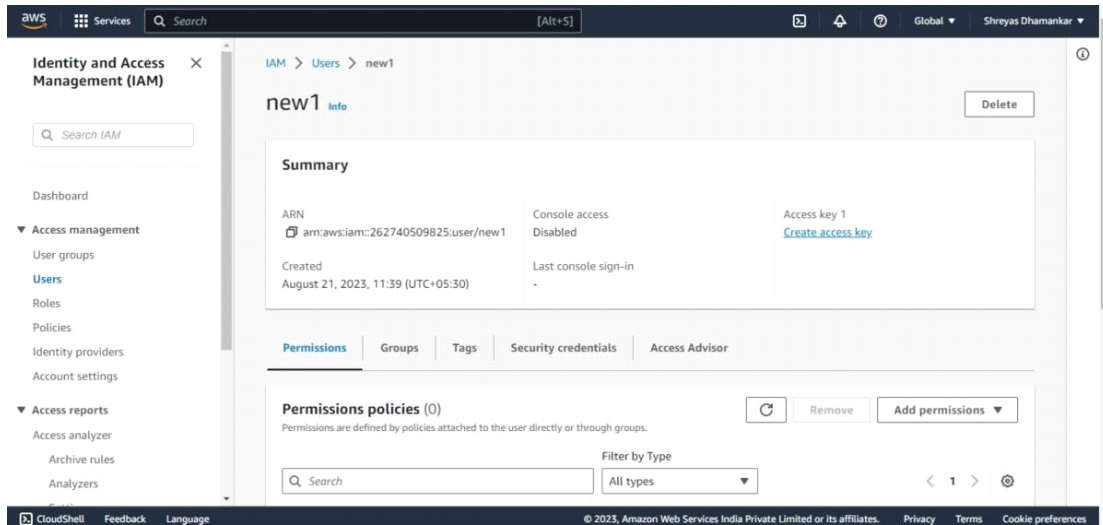
The screenshot shows the AWS IAM console interface for creating a new user. The breadcrumb navigation at the top reads 'IAM > Users > Create user'. On the left, a sidebar lists three steps: 'Step 1: Specify user details' (which is the current step), 'Step 2: Set permissions', and 'Step 3: Review and create'. The main content area is titled 'Specify user details' and contains a 'User details' section. This section has a text input field for 'User name' with the value 'new1' entered. Below the input field, a small note states: 'The user name can have up to 64 characters. Valid characters: A-Z, a-z, 0-9, and + = , _ @ _ (hyphen)'. There is an unchecked checkbox labeled 'Provide user access to the AWS Management Console - optional' with a sub-note: 'If you're providing console access to a person, it's a best practice to manage their access in IAM Identity Center.' Below this is a light blue information box with an icon and text: 'If you are creating programmatic access through access keys or service-specific credentials for AWS CodeCommit or Amazon Keyspaces, you can generate them after you create this IAM user. Learn more'. At the bottom right of the form, there are 'Cancel' and 'Next' buttons. The footer of the console shows 'CloudShell', 'Feedback', 'Language', and copyright information for Amazon Web Services India Private Limited.



add
a new user group with permission to have administrator access



then go to user and select create new access key. after selecting select create CLI and download the csv.



write this script and save it in Terraform Script folder. use the acces keys and secret key present in the csv files.

```
test2.tf - Notepad
File Edit Format View Help
provider "aws" {
  access_key="AKIAT2LE0ASA6QMGQZ02"
  secret_key="PEqoGDuRu+wpj9VLCztcWqjhCS1JSfMF/gwbYLXQ"
  region="ap-south-1"
}
resource "aws_instance" "Ubuntu" {
  ami="ami-0f5ee92e2d63afc18"
  instance_type="t2.micro"
}
```

open cmd and choose the path where you have stored the script and run the following commands init,plan,apply and destroy.

```

- root_block_device {
  - delete_on_termination = true -> null
  - device_name           = "/dev/sda1" -> null
  - encrypted             = false -> null
  - iops                  = 100 -> null
  - tags                  = {} -> null
  - throughput            = 0 -> null
  - volume_id             = "vol-0798195950b794d7d" -> null
  - volume_size           = 8 -> null
  - volume_type           = "gp2" -> null
}
}

Plan: 0 to add, 0 to change, 1 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_instance.Ubuntu: Destroying... [id=i-09e2dcb327124c5bf]
aws_instance.Ubuntu: Still destroying... [id=i-09e2dcb327124c5bf, 10s elapsed]
aws_instance.Ubuntu: Still destroying... [id=i-09e2dcb327124c5bf, 20s elapsed]
aws_instance.Ubuntu: Still destroying... [id=i-09e2dcb327124c5bf, 30s elapsed]
aws_instance.Ubuntu: Destruction complete after 32s

Destroy complete! Resources: 1 destroyed.

C:\Terraform Script>

```

```

C:\Terraform Script>terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.13.0...
- Installed hashicorp/aws v5.13.0 (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.

```

C:\Terraform Script>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_instance.Ubuntu will be created
+ resource "aws_instance" "Ubuntu" {
  + ami              = "ami-0f5ee92e2d63afc18"
  + arn              = (known after apply)
  + associate_public_ip_address = (known after apply)
  + availability_zone = (known after apply)
  + cpu_core_count   = (known after apply)
  + cpu_threads_per_core = (known after apply)
  + disable_api_stop  = (known after apply)
  + disable_api_termination = (known after apply)
  + ebs_optimized     = (known after apply)
  + get_password_data = false
  + host_id           = (known after apply)
  + host_resource_group_arn = (known after apply)
  + iam_instance_profile = (known after apply)
  + id                = (known after apply)
  + instance_initiated_shutdown_behavior = (known after apply)
  + instance_lifecycle = (known after apply)

```



```

+ instance_initiated_shutdown_behavior = (known after apply)
+ instance_lifecycle                   = (known after apply)
+ instance_state                       = (known after apply)
+ instance_type                       = "t2.micro"
+ ipv6_address_count                   = (known after apply)
+ ipv6_addresses                      = (known after apply)
+ key_name                             = (known after apply)
+ monitoring                           = (known after apply)
+ outpost_arn                         = (known after apply)
+ password_data                       = (known after apply)
+ placement_group                     = (known after apply)
+ placement_partition_number          = (known after apply)
+ primary_network_interface_id        = (known after apply)
+ private_dns                         = (known after apply)
+ private_ip                          = (known after apply)
+ public_dns                          = (known after apply)
+ public_ip                           = (known after apply)
+ secondary_private_ips               = (known after apply)
+ security_groups                     = (known after apply)
+ source_dest_check                   = true
+ spot_instance_request_id            = (known after apply)
+ subnet_id                           = (known after apply)
+ tags_all                            = (known after apply)
+ tenancy                             = (known after apply)
+ user_data                           = (known after apply)
+ user_data_base64                   = (known after apply)
+ user_data_replace_on_change         = false
+ vpc_security_group_ids              = (known after apply)
}

```

Plan: 1 to add, 0 to change, 0 to destroy.

instance has been created.

Instances (1) [Info](#) Refresh Connect Instance state ▼ Actions ▼ Launch instances ▼

Instance state = running X Clear filters < 1 > ⚙

<input type="checkbox"/>	Name ▼	Instance ID	Instance state ▼	Instance type ▼	Status check	Alarm status	Availability Zone
<input type="checkbox"/>	-	i-043313e67db448f8e	Running	t2.micro	✓	🔔	ap-south-1a

```

C:\Terraform>terraform destroy
aws_instance.Ubuntu: Refreshing state... [id=i-043313e67db448fbe]

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_instance.Ubuntu will be destroyed
- resource "aws_instance" "Ubuntu" {
  ami              = "ami-0f5ee92e2d63afc18" -> null
  arn              = "arn:aws:ec2:ap-south-1:262740509825:instance/i-043313e67db448fbe" -> null
  associate_public_ip_address = true -> null
  availability_zone = "ap-south-1a" -> null
  cpu_core_count    = 1 -> null
  cpu_threads_per_core = 1 -> null
  disable_api_stop   = false -> null
  disable_api_termination = false -> null
  ebs_optimized      = false -> null
  get_password_data   = false -> null
  hibernation         = false -> null
  id                 = "i-043313e67db448fbe" -> null
  instance_initiated_shutdown_behavior = "stop" -> null
  instance_state     = "running" -> null
  instance_type      = "t2.micro" -> null
  ipv6_address_count = 0 -> null
  ipv6_addresses     = [] -> null
  monitoring         = false -> null
  placement_partition_number = 0 -> null
  primary_network_interface_id = "eni-060f15a4b86e45f2f" -> null
  private_dns        = "ip-172-31-37-169.ap-south-1.compute.internal" -> null
  private_ip         = "172.31.37.169" -> null
  public_dns         = "ec2-3-111-147-14.ap-south-1.compute.amazonaws.com" -> null
  public_ip          = "3.111.147.14" -> null
}

```

```

Plan: 0 to add, 0 to change, 1 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.

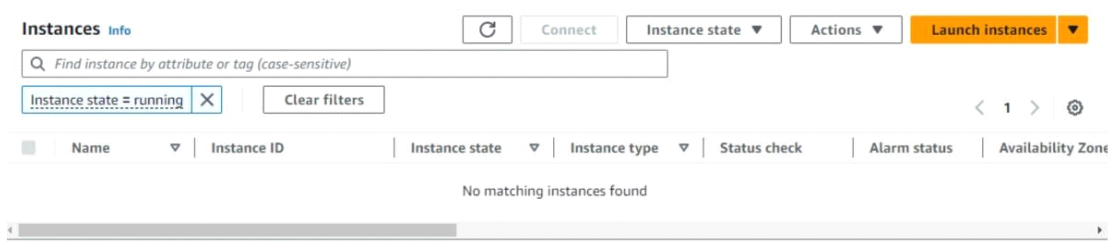
Enter a value: yes

aws_instance.Ubuntu: Destroying... [id=i-043313e67db448fbe]
aws_instance.Ubuntu: Still destroying... [id=i-043313e67db448fbe, 10s elapsed]
aws_instance.Ubuntu: Still destroying... [id=i-043313e67db448fbe, 20s elapsed]
aws_instance.Ubuntu: Still destroying... [id=i-043313e67db448fbe, 30s elapsed]
aws_instance.Ubuntu: Destruction complete after 30s

Destroy complete! Resources: 1 destroyed.

```

instance has been destroyed



once terraform destroy command has been run and completed you can delete all security key and csv files and delete users and user groups.

Conclusion:

In this assignment we learnt how to install terraform and use it to run scripts.