LAB-7
Creating Multiple IAM Users Using Terraform

Step 1: Create a file name main.tf

main.tf

main.tf > resource "aws_iam_user" "iam_users"

```
terraform {
  required_providers {
    aws = {
        source = "hashicorp/aws"
        version = "5.35.0"
    }
  }
}
provider "aws" {
  region = "ap-south-1"
  access_key = "[redacted]"
  secret_key = "[redacted]"
}
variable "iam_users" {
  type = list(string)
  default = [ "user1", "user2", "user3" ]
}
resource "aws_iam_user" "iam_users" {
  count = length(var.iam_users)
  name = var.iam_users[count.index]
  tags = {
    Name = "${var.iam_users[count.index]}-users"
  }
}
```

Step 2: Use terraform init to initalize terraform

```
❯ terraform init

Initializing the backend...

Initializing provider plugins...
- Finding hashicorp/aws versions matching "5.35.0"...
- Installing hashicorp/aws v5.35.0...
- Installed hashicorp/aws v5.35.0 (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.

~/Desktop/lab7
```

Step 3: Use terraform validate to check any error in HCL script

```
❯ terraform validate
Success! The configuration is valid.

~/Desktop/lab7
```

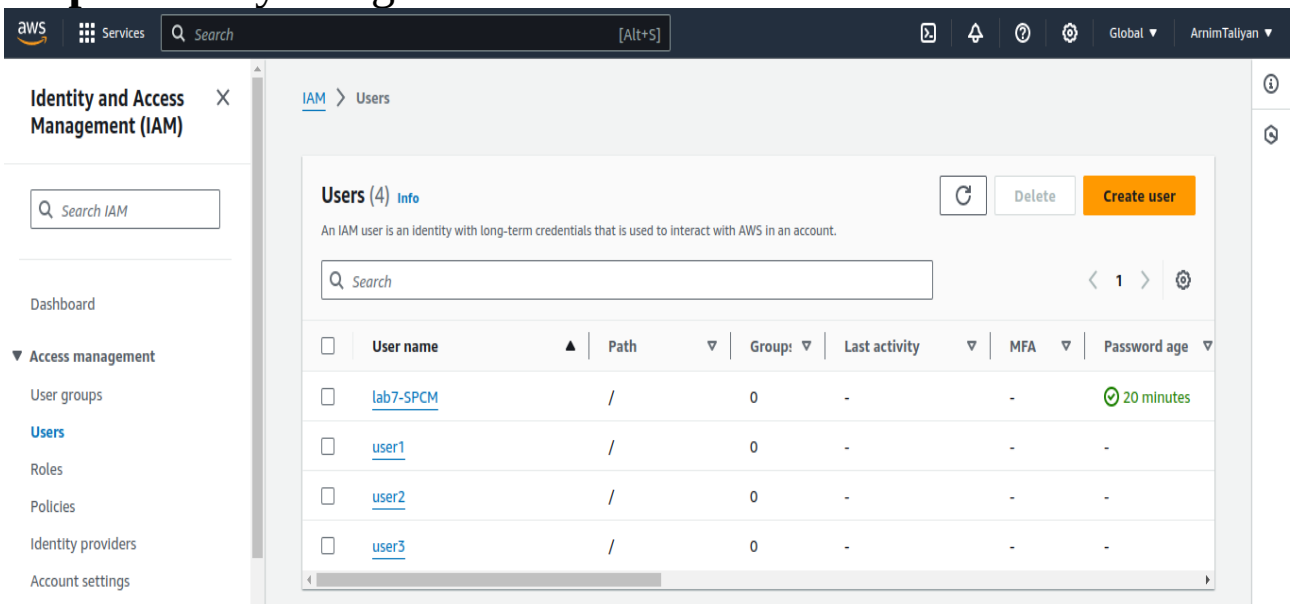
Step 4: Use terraform plan to review the provided resources

```
❯ 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_user.iam_users[0] will be created
  + resource "aws_iam_user" "iam_users" {
      + arn           = (known after apply)
      + force_destroy = false
      + id            = (known after apply)
      + name          = "user1"
      + path          = "/"
      + tags          = {
          + "Name" = "user1-users"
        }
      + tags_all      = {
          + "Name" = "user1-users"
        }
      + unique_id     = (known after apply)
    }

  # aws_iam_user.iam_users[1] will be created
  + resource "aws_iam_user" "iam_users" {
      + arn           = (known after apply)
      + force_destroy = false
      + id            = (known after apply)
      + name          = "user2"
      + path          = "/"
      + tags          = {
          + "Name" = "user2-users"
        }
      + tags_all      = {
          + "Name" = "user2-users"
        }
      + unique_id     = (known after apply)
    }

  # aws_iam_user.iam_users[2] will be created
```

Step 5: Use terraform apply to apply the changes

```
❯ 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_user.iam_users[0] will be created
  + resource "aws_iam_user" "iam_users" {
      + arn           = (known after apply)
      + force_destroy = false
      + id            = (known after apply)
      + name          = "user1"
      + path          = "/"
      + tags          = {
          + "Name" = "user1-users"
        }
      + tags_all      = {
          + "Name" = "user1-users"
        }
      + unique_id     = (known after apply)
    }

  # aws_iam_user.iam_users[1] will be created
  + resource "aws_iam_user" "iam_users" {
      + arn           = (known after apply)
      + force_destroy = false
      + id            = (known after apply)
      + name          = "user2"
      + path          = "/"
      + tags          = {
          + "Name" = "user2-users"
```

Step 6: Verify using AWS Console



Step 7: Add or remove IAM Users by changing main.tf file

main.tf

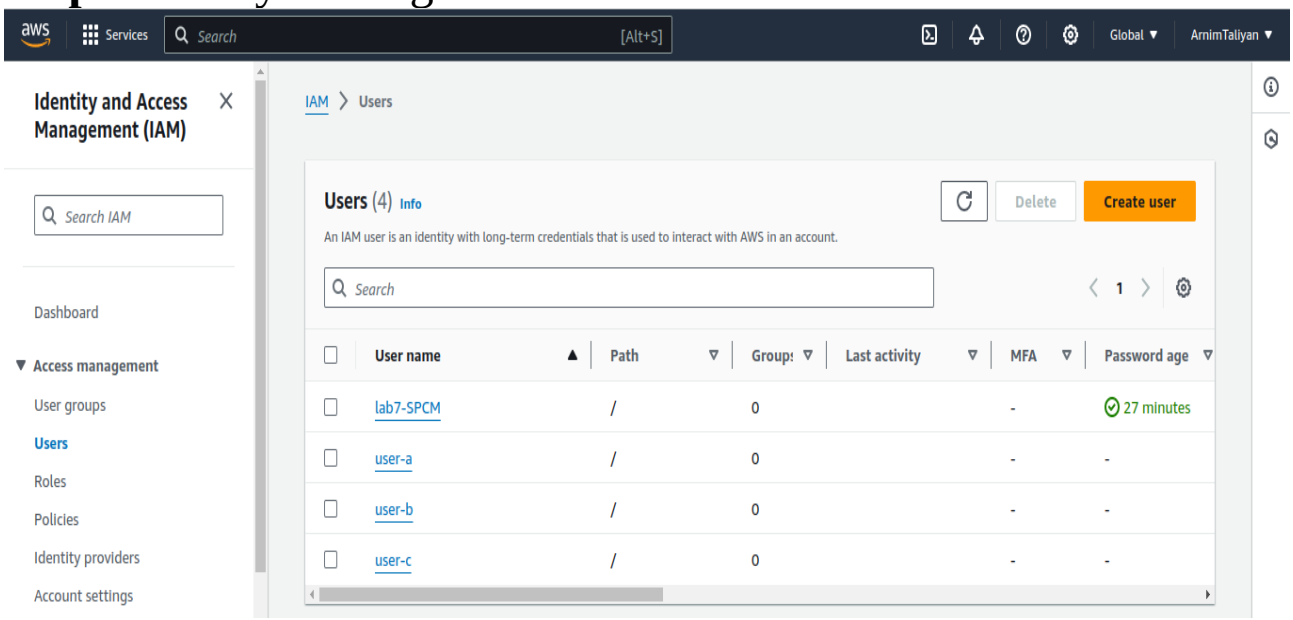
main.tf > resource "aws_iam_user" "iam_users"

```
terraform {
  required_providers {
    aws = {
        source = "hashicorp/aws"
        version = "5.35.0"
    }
  }
}
provider "aws" {
  region = "ap-south-1"
  access_key = "[redacted]"
  secret_key = "[redacted]"
}
variable "iam_users" {
  type = list(string)
  default = [ "user-a", "user-b", "user-c" ]
}
resource "aws_iam_user" "iam_users" {
  count = length(var.iam_users)
  name = var.iam_users[count.index]
  tags = {
    Name = "${var.iam_users[count.index]}-users"
  }
}
```

Step 8: Use terraform apply to apply changes

```
$ terraform apply
aws_iam_user.iam_users[1]: Refreshing state... [id=user2]
aws_iam_user.iam_users[0]: Refreshing state... [id=user1]
aws_iam_user.iam_users[2]: Refreshing state... [id=user3]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  ~ update in-place

Terraform will perform the following actions:

  # aws_iam_user.iam_users[0] will be updated in-place
  ~ resource "aws_iam_user" "iam_users" {
        id            = "user1"
      ~ name          = "user1" -> "user-a"
      ~ tags          = {
          ~ "Name" = "user1-users" -> "user-a-users"
        }
      ~ tags_all      = {
          ~ "Name" = "user1-users" -> "user-a-users"
        }
        # (4 unchanged attributes hidden)
    }

  # aws_iam_user.iam_users[1] will be updated in-place
  ~ resource "aws_iam_user" "iam_users" {
        id            = "user2"
      ~ name          = "user2" -> "user-b"
      ~ tags          = {
          ~ "Name" = "user2-users" -> "user-b-users"
        }
```

Step 9: Verify it using AWS Console



Step 10: Delete all IAM Users using terraform destroy

```
❯ terraform destroy
aws_iam_user.iam_users[2]: Refreshing state... [id=user-c]
aws_iam_user.iam_users[0]: Refreshing state... [id=user-a]
aws_iam_user.iam_users[1]: Refreshing state... [id=user-b]

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_user.iam_users[0] will be destroyed
  - resource "aws_iam_user" "iam_users" {
      - arn           = "arn:aws:iam::533266967718:user/user-a" -> null
      - force_destroy = false -> null
      - id            = "user-a" -> null
      - name          = "user-a" -> null
      - path          = "/" -> null
      - tags          = {
          - "Name" = "user-a-users"
        } -> null
      - tags_all      = {
          - "Name" = "user-a-users"
        } -> null
      - unique_id     = "AIDAXYKJQFSTIHRUUKLL2" -> null
    }

  # aws_iam_user.iam_users[1] will be destroyed
  - resource "aws_iam_user" "iam_users" {
      - arn           = "arn:aws:iam::533266967718:user/user-b" -> null
      - force_destroy = false -> null
      - id            = "user-b" -> null
      - name          = "user-b" -> null
      - path          = "/" -> null
      - tags          = {
```

Step 11: Verify it by AWS Console