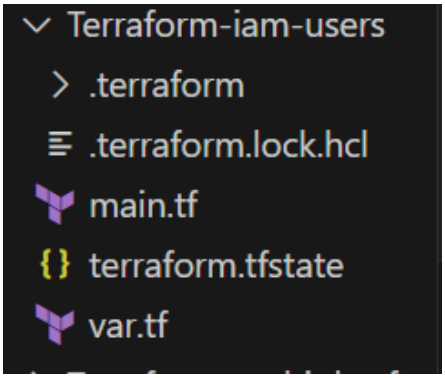
Lab Exercise 7– Creating Multiple IAM Users in Terraform

Steps:

1. Create a Terraform Directory:

```
mkdir terraform-iam-users
cd terraform-iam-users
```



- Create Terraform Configuration Files:
- Create a file named main.tf:

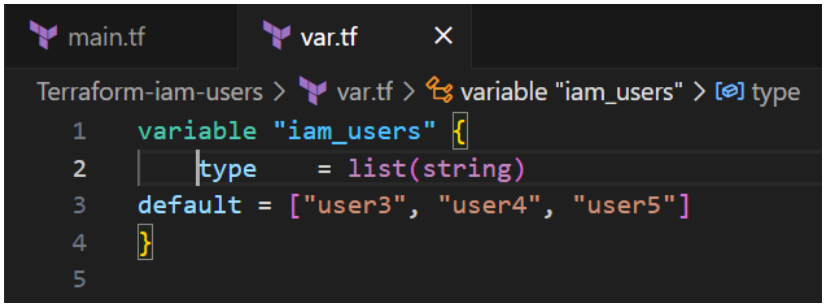
 # main.tf

main.tf var.tf

Terraform-iam-users > main.tf > resource "aws_iam_user" "iam_users"

```
terraform {
required_providers {
aws = {
source = "hashicorp/aws"
version = "5.31.0"
}
}
}

provider "aws" {
    region = "us-west-2"
    access_key = "AKIA5FTY77WSIB44R75Q"
    secret_key = "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"
}

resource "aws_iam_user" "iam_users" {
     count = length(var.iam_users)
name = var.iam_users[count.index]


tags = {
Name = "${var.iam_users[count.index]}-user"
}
}

```

#var.tf


main.tf
var.tf
Terraform-iam-users > var.tf > variable "iam_users" > type
variable "iam_users" {
type = list(string)
default = ["user3", "user4", "user5"]
}

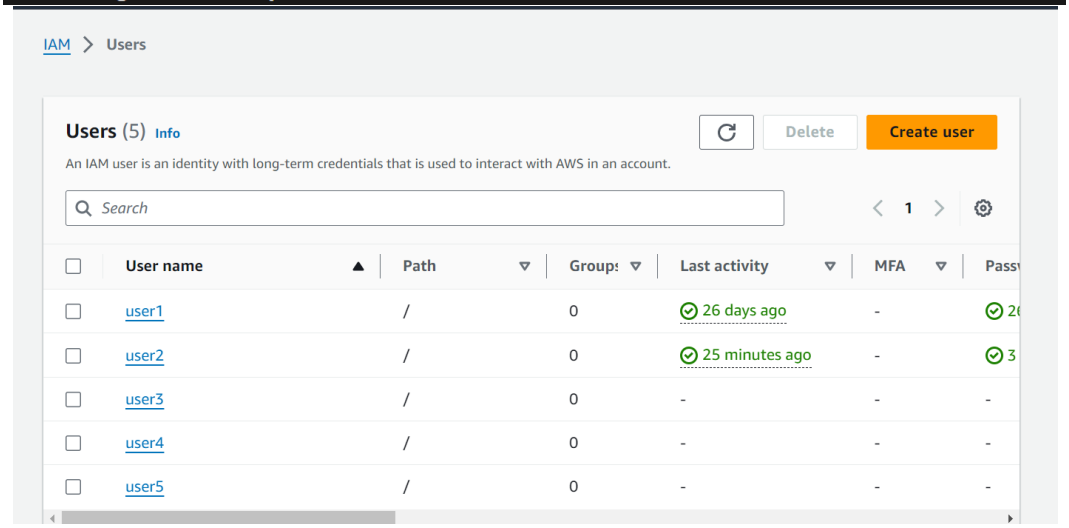
2. Initialize and Apply:

Run the following Terraform commands to initialize and apply the configuration:

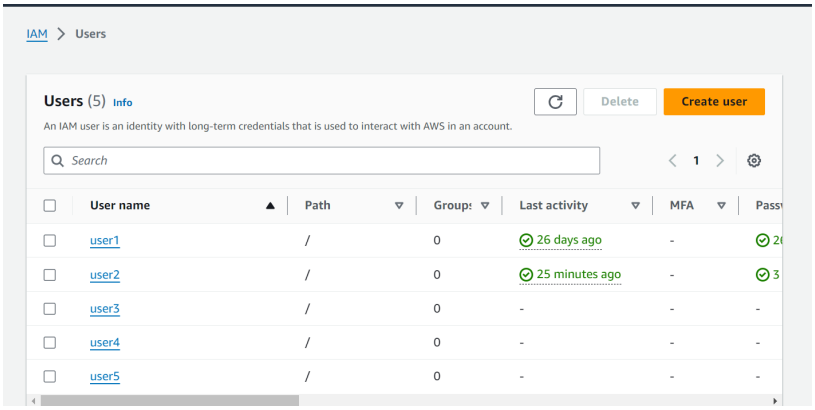
```
terraform init
terraform apply
```

```
PS C:\Desktop\DevOps\Sem6\SMCP\Lab Files\TERRAFORM LAB SCRIPTS\Terraform-iam-users> terraform init

Initializing the backend...
Initializing provider plugins...
- Finding hashicorp/aws versions matching "5.31.0"...
- Installing hashicorp/aws v5.31.0...
- Installed hashicorp/aws v5.31.0 (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!
```

```
PS C:\Desktop\DevOps\Sem6\SMCP\Lab Files\TERRAFORM LAB SCRIPTS\Terraform-iam-users> terraform apply

Terraform used the selected providers to generate the following execution plan. Resource actions are
  + 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          = "user3"
      + path          = "/"
      + tags          = {
```



3. Verify Users in AWS Console:



4. Clean Up:

- After testing, you can clean up the IAM users:

terraform destroy

```
PS C:\Desktop\DevOps\Sem6\SMCP\Lab Files\TERRAFORM LAB SCRIPTS\Terraform-iam-users> terraform destroy
aws_iam_user.iam_users[2]: Refreshing state... [id=user5]
aws_iam_user.iam_users[1]: Refreshing state... [id=user4]
aws_iam_user.iam_users[0]: Refreshing state... [id=user3]

Terraform used the selected providers to generate the following execution plan. Resource actions are i
  - 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::905418112420:user/user3" -> null
      - force_destroy = false -> null
      - id            = "user3" -> null
      - name          = "user3" -> null
      - path          = "/" -> null
```