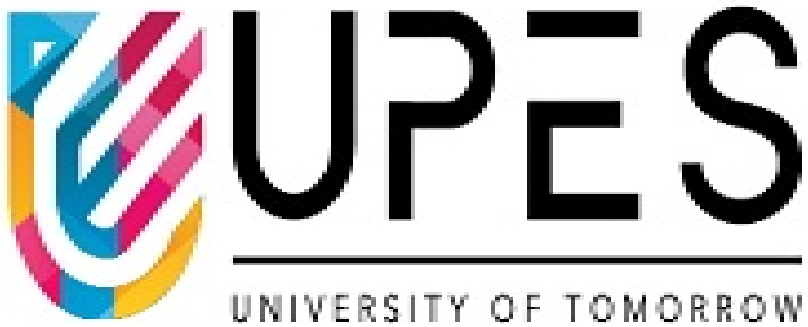


SYSTEM PROVISIONING AND CONFIGURATION MANAGEMENT

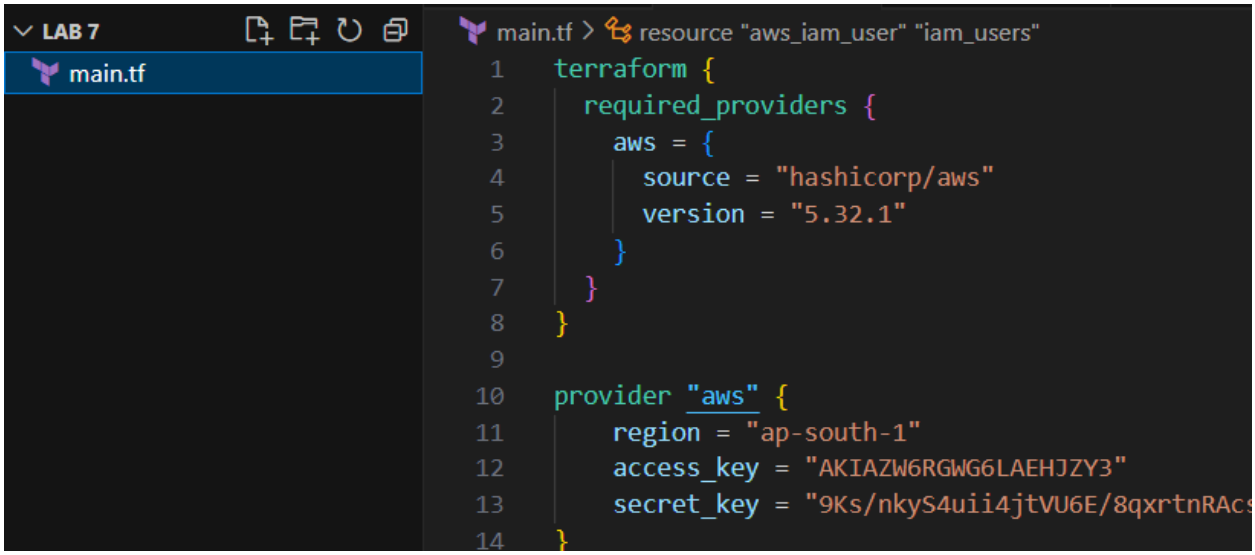
LAB FILE

NAME: SMRITI RAI
SAP ID: 500096396
BATCH: B3
SUBMITTED TO: Dr. Hitesh Kumar Sharma
SEMESTER: VI
ENROLLMENT NO.: R2142211212

EXPERIMENT 7:

Creating Multiple IAM users in Terraform

1. Create a file named main.tf



2. Write the following code for the generation of IAM users.

```
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]}-SMuser"
    }
}
```

3. Open command prompt and run terraform init.

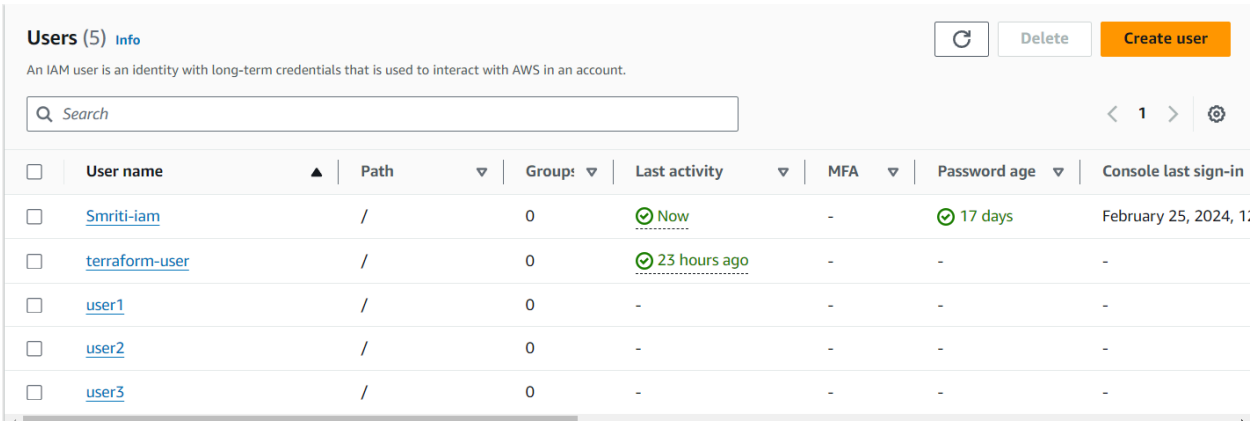
```
D:\docss\UPES\sem 6\SPCM Lab\lab 7>terraform init

Initializing the backend...

Initializing provider plugins...
- Finding hashicorp/aws versions matching "5.32.1"...
- Installing hashicorp/aws v5.32.1...
- Installed hashicorp/aws v5.32.1 (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:\docss\UPES\sem 6\SPCM Lab\lab 7>
```

4. Then run terraform apply.

```
D:\docss\UPES\sem 6\SPCM Lab\lab 7>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-SMuser"
        }
      + tags_all      = {
          + "Name" = "user1-SMuser"
        }
      + 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)
```

```
      + id            = (known after apply)
      + name          = "user3"
      + path          = "/"
      + tags          = {
          + "Name" = "user3-SMuser"
        }
      + tags_all      = {
          + "Name" = "user3-SMuser"
        }
      + unique_id     = (known after apply)
    }

Plan: 3 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.iam_users[0]: Creating...
aws_iam_user.iam_users[2]: Creating...
aws_iam_user.iam_users[1]: Creating...
aws_iam_user.iam_users[2]: Creation complete after 1s [id=user3]
aws_iam_user.iam_users[1]: Creation complete after 2s [id=user2]
aws_iam_user.iam_users[0]: Creation complete after 2s [id=user1]

Apply complete! Resources: 3 added, 0 changed, 0 destroyed.
```

5. Go to AWS Console and navigate to IAM users



6. Now clean up your resources using terraform destroy

```
D:\docss\UPES\sem 6\SPCM Lab\lab 7>terraform destroy
aws_iam_user.iam_users[1]: Refreshing state... [id=user2]
aws_iam_user.iam_users[2]: Refreshing state... [id=user3]
aws_iam_user.iam_users[0]: Refreshing state... [id=user1]

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::667769287100:user/user1" -> null
      - force_destroy = false -> null
      - id            = "user1" -> null
      - name          = "user1" -> null
      - path          = "/" -> null
      - tags          = {
          - "Name" = "user1-SMuser"
        } -> null
      - tags_all      = {
          - "Name" = "user1-SMuser"
        } -> null
      - unique_id     = "AIDAZW6RGWG6C266IXIKU" -> null
    }

  # aws_iam_user.iam_users[1] will be destroyed
  - resource "aws_iam_user" "iam_users" {
```

```
      - id            = "user3" -> null
      - name          = "user3" -> null
      - path          = "/" -> null
      - tags          = {
          - "Name" = "user3-SMuser"
        } -> null
      - tags_all      = {
          - "Name" = "user3-SMuser"
        } -> null
      - unique_id     = "AIDAZW6RGWG6KWF6D7XJ2" -> null
    }

Plan: 0 to add, 0 to change, 3 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_user.iam_users[2]: Destroying... [id=user3]
aws_iam_user.iam_users[0]: Destroying... [id=user1]
aws_iam_user.iam_users[1]: Destroying... [id=user2]
aws_iam_user.iam_users[0]: Destruction complete after 2s
aws_iam_user.iam_users[2]: Destruction complete after 2s
aws_iam_user.iam_users[1]: Destruction complete after 2s

Destroy complete! Resources: 3 destroyed.
```