Lab Exercise 10– Creating Multiple IAM Users in Terraform

Objective:

Learn how to use Terraform to create multiple IAM users with unique settings.

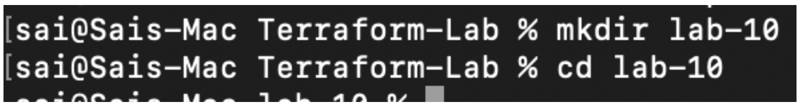
Prerequisites:

- Terraform installed on your machine.
- AWS CLI configured with the necessary credentials.

Steps:

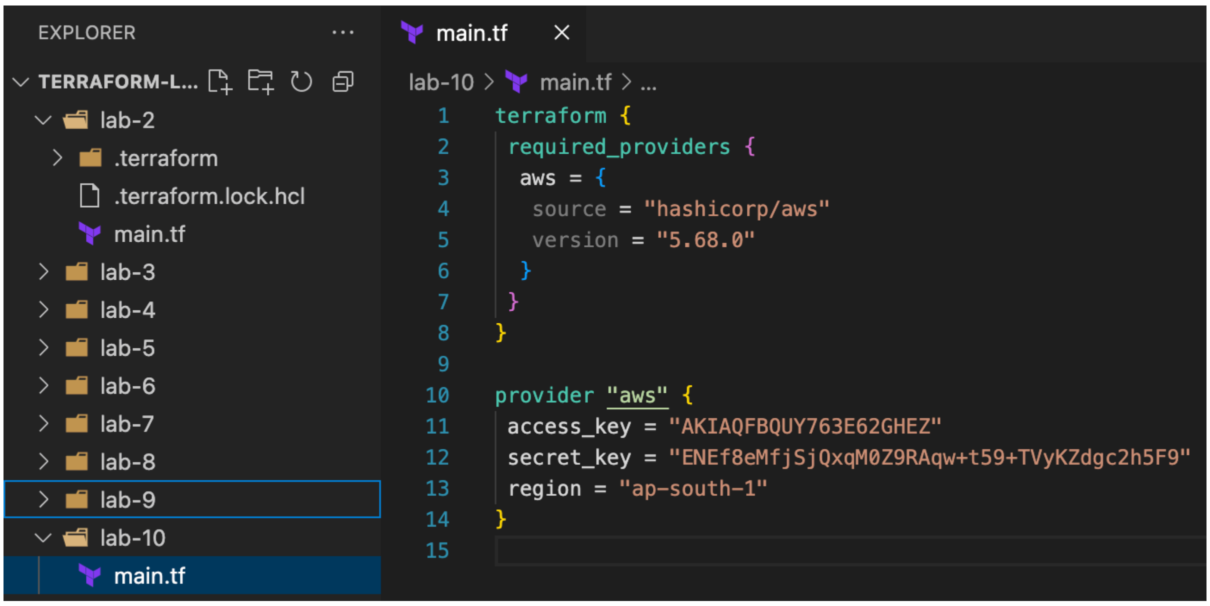
1. Create a Terraform Directory:

```
mkdir lab-10
cd lab-10
```



```
sai@Sais-Mac Terraform-Lab % mkdir lab-10
sai@Sais-Mac Terraform-Lab % cd lab-10
```

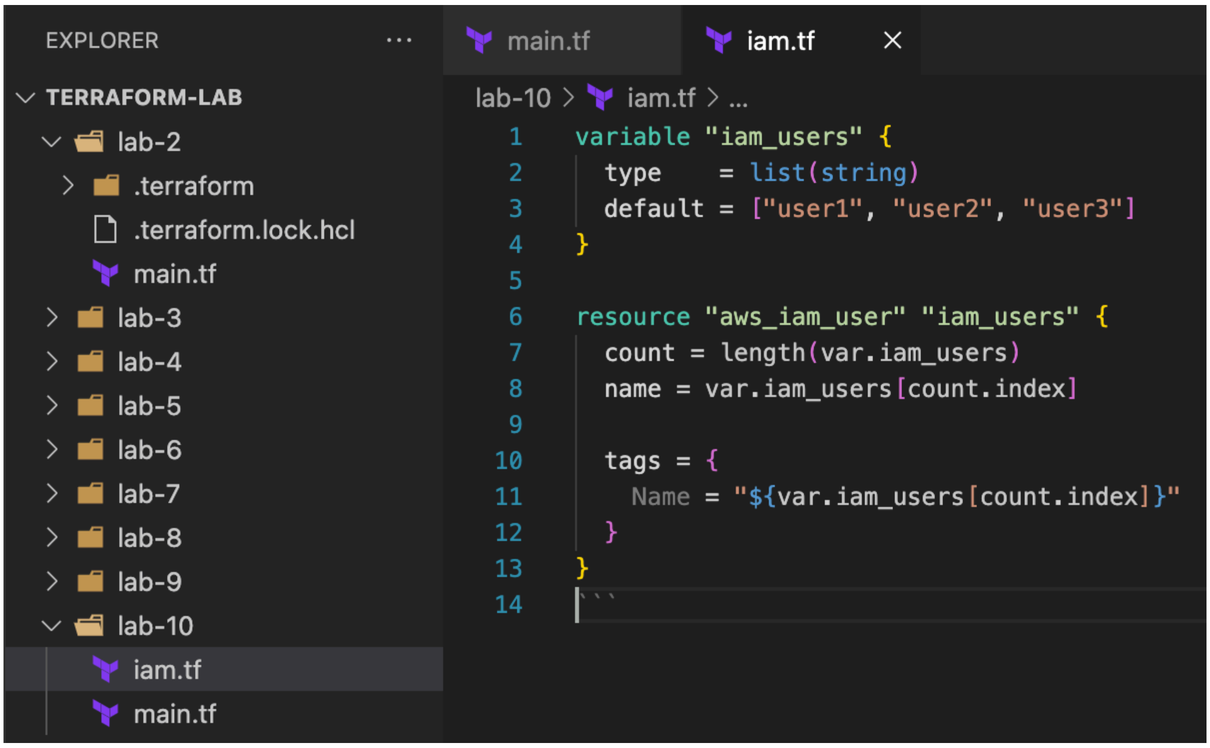
- Create Terraform Configuration Files:
- Create a file named main.tf:



```
terraform {
  required_providers {
    aws = {
     source = "hashicorp/aws"
     version = "5.68.0"
    }
  }
}

provider "aws" {
  access_key = "AKIAQFBQUY763E62GHEZ"
  secret_key = "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"
  region = "ap-south-1"
}
```

# iam.tf

```
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]}"
  }
}
```

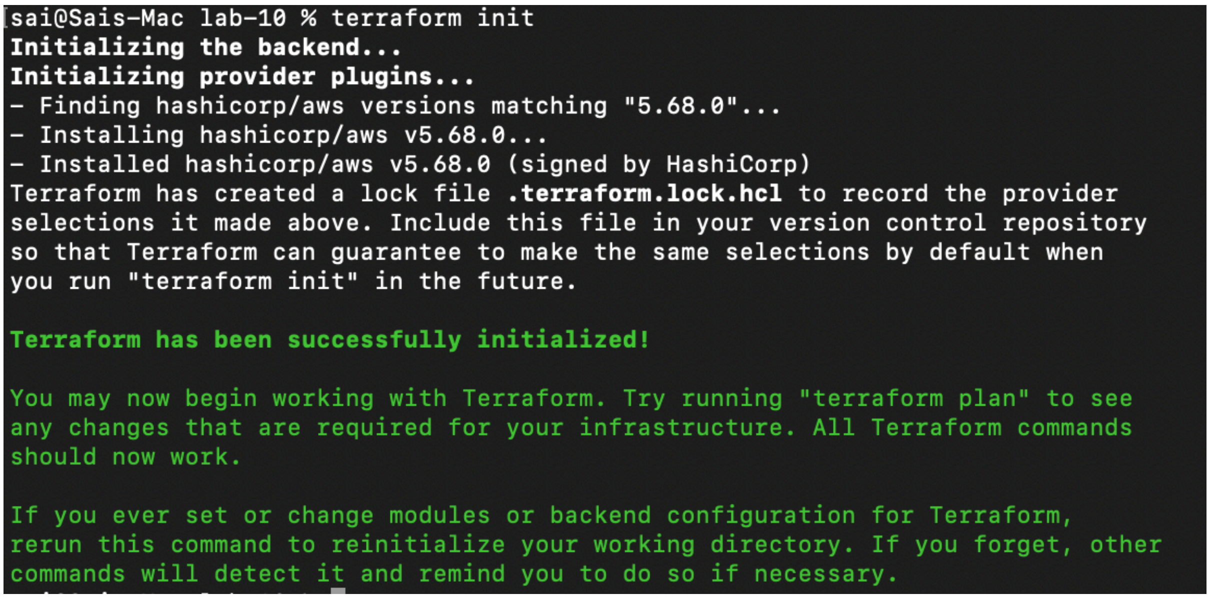


```
variable "iam_users" {
  type    = list(string)
  default = ["user1", "user2", "user3"]
}

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

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

In this configuration, we define a list variable iam_users containing the names of the IAM users we want to create. The aws_iam_user resource is then used in a loop to create users based on the values in the list.
```

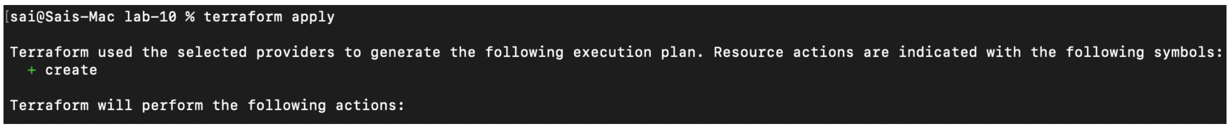
2. Initialize and Apply:

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

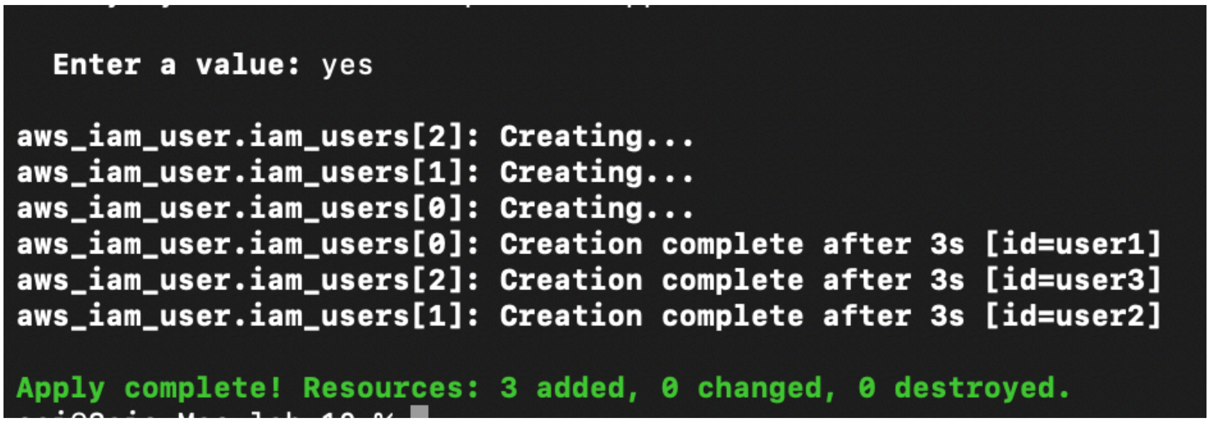
```
terraform init
terraform apply
```



```
sai@Sais-Mac lab-10 % terraform init
Initializing the backend...
Initializing provider plugins...
- Finding hashicorp/aws versions matching "5.68.0"...
- Installing hashicorp/aws v5.68.0...
- Installed hashicorp/aws v5.68.0 (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.
```



```
sai@Sais-Mac lab-10 % 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:
```



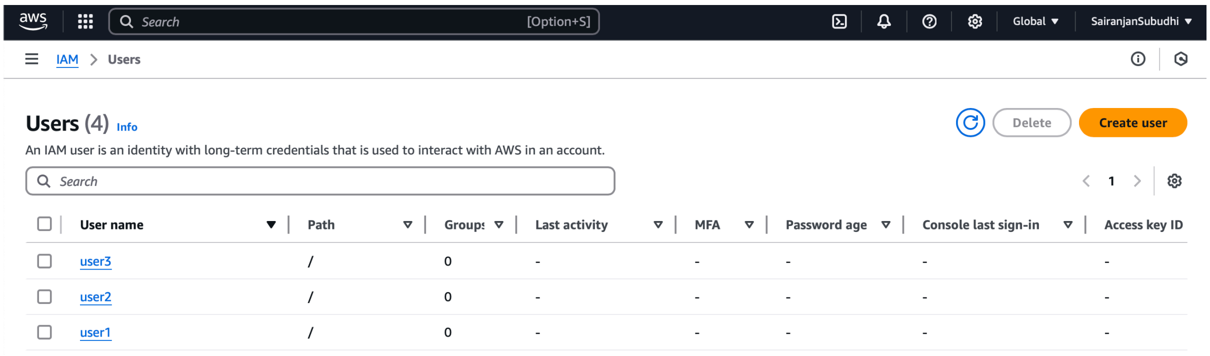
```
  Enter a value: yes

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

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

Terraform will prompt you to confirm the creation of IAM users. Type yes and press Enter.

3. Verify Users in AWS Console:

- Log in to the AWS Management Console and navigate to the IAM service.

- Verify that the IAM users with the specified names and tags have been created.



Users (4) Info

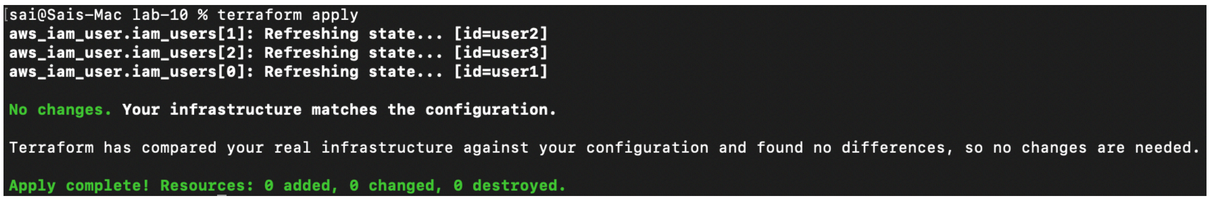
An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.

User name	Path	Groups	Last activity	MFA	Password age	Console last sign-in	Access key ID
user3	/	0	-	-	-	-	-
user2	/	0	-	-	-	-	-
user1	/	0	-	-	-	-	-

4. Update IAM Users:

- If you want to add or remove IAM users, modify the iam_users list in the main.tf file.
- Rerun the terraform apply command to apply the changes:

```
terraform apply
```

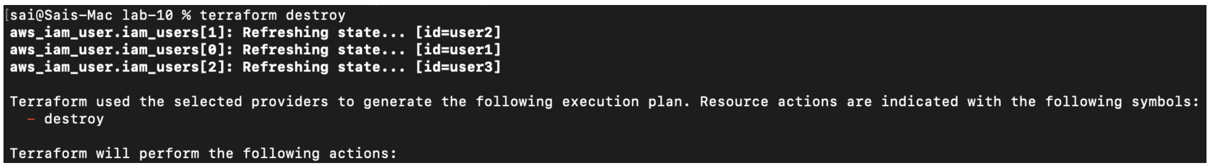


```
sai@Sais-Mac lab-10 % terraform apply
aws_iam_user.iam_users[1]: Refreshing state... [id=user2]
aws_iam_user.iam_users[2]: Refreshing state... [id=user3]
aws_iam_user.iam_users[0]: Refreshing state... [id=user1]

No changes. Your infrastructure matches the configuration.

Terraform has compared your real infrastructure against your configuration and found no differences, so no changes are needed.

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

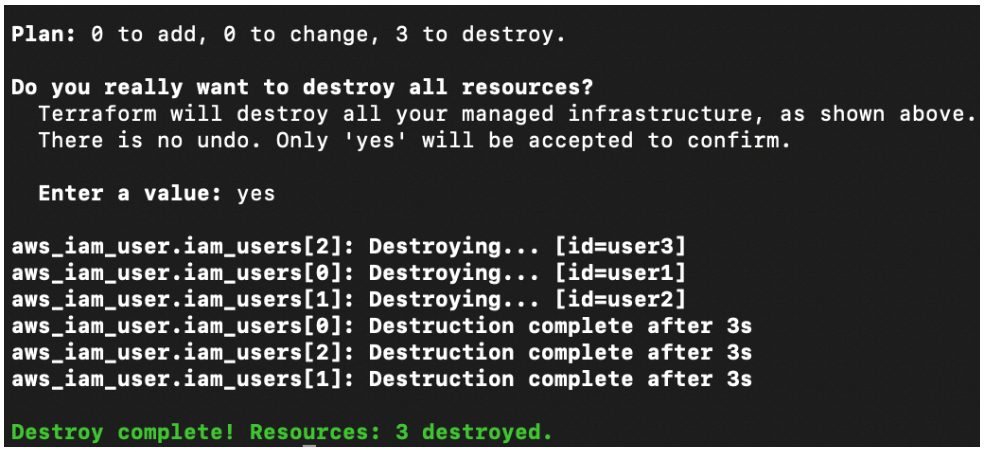
5. Clean Up:

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

```
terraform destroy
```



```
sai@Sais-Mac lab-10 % terraform destroy
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:
  - destroy

Terraform will perform the following actions:
```



```
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 3s
aws_iam_user.iam_users[2]: Destruction complete after 3s
aws_iam_user.iam_users[1]: Destruction complete after 3s

Destroy complete! Resources: 3 destroyed.
```

- Confirm the destruction by typing yes.

6. Conclusion:

This lab exercise demonstrates how to create multiple IAM users in AWS using Terraform. The use of variables and loops allows you to easily manage and scale the creation of IAM users. Experiment with different user names and settings in the main.tf file to understand how Terraform provisions resources based on your configuration.