

Lab Exercise 7– Creating Multiple IAM Users in Terraform

Objective:

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

Prerequisites:

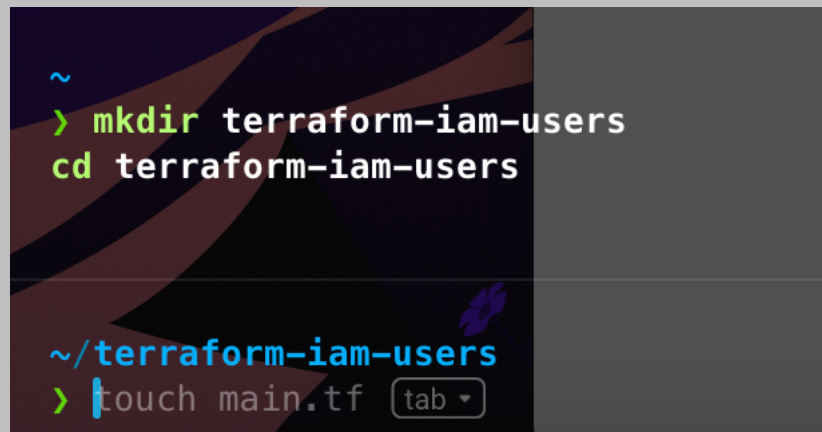
- Terraform installed on your machine.
- AWS CLI configured with the necessary credentials.

Steps:

1. Create a Terraform Directory:

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

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




```
~  
> mkdir terraform-iam-users  
cd terraform-iam-users  
  
~/terraform-iam-users  
> touch main.tf
```

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

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]}"  
  }  
}
```



The screenshot shows a terminal window with a dark background and light-colored text. The terminal title bar includes the text 'nvim' and 'type = list(string)'. The code is displayed with line numbers on the left side, ranging from 1 to 14. The code is a Terraform configuration for creating IAM users. It defines a variable 'iam_users' with a list of three users and a resource 'aws_iam_user' that creates a user for each element in the list. The resource has a 'count' attribute set to the length of the list and a 'name' attribute set to the user name. It also has a 'tags' block with a 'Name' tag.

```
13 variable "iam_users" {  
12   type = list(string)  
11   default = ["user1", "user2", "user3"]  
10 }  
9  
8 resource "aws_iam_user" "iam_users" {  
7   count = length(var.iam_users)  
6   name = var.iam_users[count.index]  
5  
4   tags = {  
3     Name = "${var.iam_users[count.index]}"  
2   }  
1 }  
14 |
```

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-iam-users
> terraform init

Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.94.1...
- Installed hashicorp/aws v5.94.1 (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!
```

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.

Following Terraform commands to initialize and apply the configuration:

1. Verify Users in AWS Console:

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

2. Update IAM Users:

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

terraform apply

```
~/terraform-iam-users
> terraform apply

# aws_iam_user.iam_users[2] will be created
+ resource "aws_iam_user" "iam_users" {
+   arn                = (known after apply)
+   force_destroy      = false
+   id                 = (known after apply)
+   name                = "user3"
+   path               = "/"
+   tags               = {
+     "Name" = "user3"
+   }
+   tags_all           = {
+     "Name" = "user3"
+   }
+   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

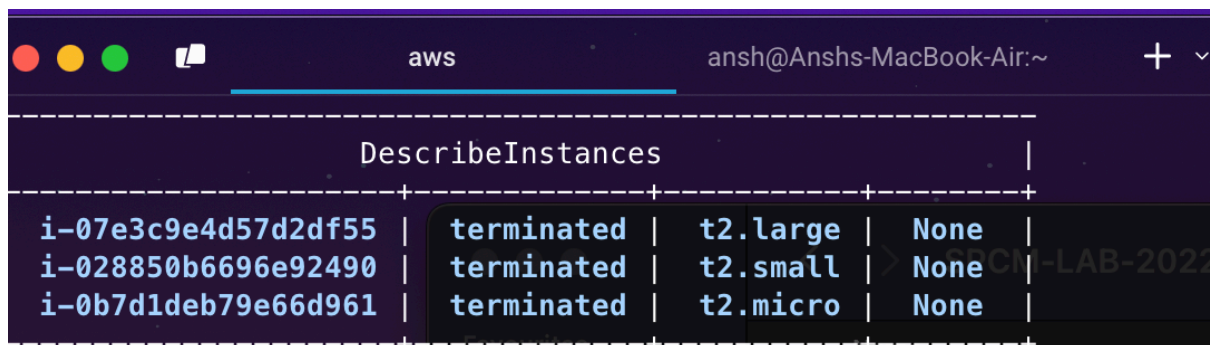
aws_iam_user.iam_users[1]: Creating...
aws_iam_user.iam_users[2]: Creating...
aws_iam_user.iam_users[0]: Creating...
aws_iam_user.iam_users[0]: Creation complete after 2s [id=user1]
aws_iam_user.iam_users[1]: Creation complete after 2s [id=user2]
aws_iam_user.iam_users[2]: Creation complete after 2s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

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.



DescribeInstances				
i-07e3c9e4d57d2df55	terminated	t2.large	None	
i-028850b6696e92490	terminated	t2.small	None	
i-0b7d1deb79e66d961	terminated	t2.micro	None	

4. Clean Up:

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

```
terraform destroy
```

```
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[1]: Destroying... [id=user2]
aws_iam_user.iam_users[0]: Destroying... [id=user1]
aws_iam_user.iam_users[2]: Destroying... [id=user3]
aws_iam_user.iam_users[2]: Destruction complete after 2s
aws_iam_user.iam_users[1]: Destruction complete after 2s
aws_iam_user.iam_users[0]: Destruction complete after 2s

Destroy complete! Resources: 3 destroyed.
```

- Confirm the destruction by typing yes.

5. 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.