

## Lab Exercise 9– Creating Multiple EC2 Instances with for\_each in Terraform

### Objective:

Learn how to use for\_each in Terraform to create multiple AWS EC2 instances with specific settings for each instance.

### Prerequisites:

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

### Steps:

#### 1. Create a Terraform Directory:

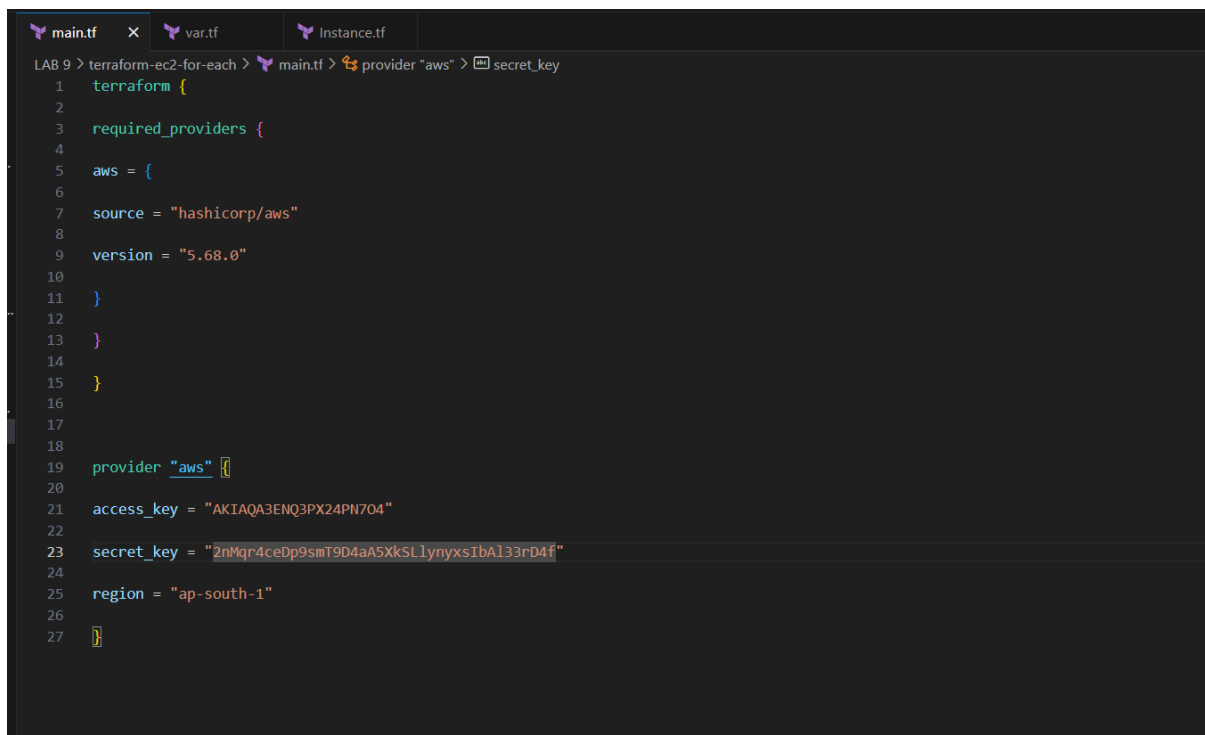
```
mkdir terraform-ec2-for-each  
cd terraform-ec2-for-each
```

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

## # main.tf

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

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



The screenshot shows a code editor with three tabs: main.tf, var.tf, and Instance.tf. The main.tf tab is active, displaying the Terraform configuration for the AWS provider. The code is as follows:

```
LAB 9 > terraform-ec2-for-each > main.tf > provider "aws" > secret_key
1  terraform {
2
3  required_providers {
4
5    aws = {
6
7      source = "hashicorp/aws"
8
9      version = "5.68.0"
10
11    }
12  }
13 }
14
15
16
17
18
19 provider "aws" {
20
21   access_key = "AKIAQA3ENQ3PX24PN7O4"
22
23   secret_key = "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"
24
25   region = "ap-south-1"
26
27 }
```

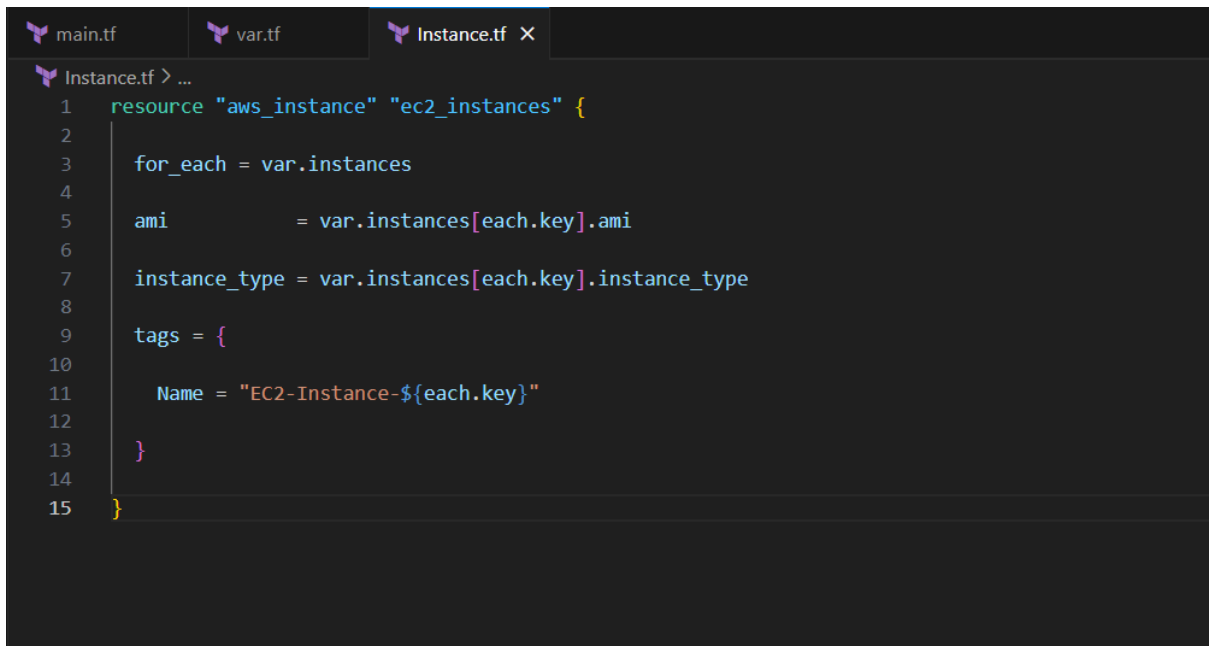
## #Var.tf

```
variable "instances" {  
  description = "Map of EC2 instances with settings"  
  default = {  
    "instance1" = {  
      ami      = "ami-0c55b159cbfafa1fo"  
      instance_type = "t2.micro"  
    },  
    "instance2" = {  
      ami      = "ami-0123456789abcdefo"  
      instance_type = "t2. small "  
    },  
    "instance3" = {  
      ami      = "ami-987654321ofedcbao"  
      instance_type = "t2. large "  
    }  
  }  
}
```

```
var.tf > variable "instances" > default > instance3 > ami
1  variable "instances" {
2
3      description = "Map of EC2 instances with settings"
4
5      default = {
6
7          "instance1" = {
8
9              ami          = "ami-0ddfb243cbee3768"
10
11              instance_type = "t2.micro"
12
13          },
14
15          "instance2" = {
16
17              ami          = "ami-00bb6a80f01f03502"
18
19              instance_type = "t2. small "
20
21          },
22
23          "instance3" = {
24
25              ami          = "ami-05a00967f06885a63"
26
27              instance_type = "t2. large "
28
29          }
30
31      }
32
33  }
```

## #Instance.tf

```
resource "aws_instance" "ec2_instances" {
  for_each = var.instances
  ami      = var.instances[each.key].ami
  instance_type = var.instances[each.key].instance_type
  tags = {
    Name = "EC2-Instance-${each.key}"
  }
}
```



```
main.tf  var.tf  Instance.tf X
Instance.tf > ...
1  resource "aws_instance" "ec2_instances" {
2
3      for_each = var.instances
4
5      ami          = var.instances[each.key].ami
6
7      instance_type = var.instances[each.key].instance_type
8
9      tags = {
10
11          Name = "EC2-Instance-${each.key}"
12      }
13  }
14
15 }
```

- Replace "your-key-pair-name" and "your-subnet-id" with your actual key pair name and subnet ID.
- In this configuration, we define a variable instances as a map containing settings for each EC2 instance. The aws\_instance resource is then used with for\_each to create instances based on the map.

## 2. Initialize and Apply:

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

```
terraform init
terraform apply
```

```
PS C:\Users\Lenovo\OneDrive\Desktop\SPCM Lab\LAB 9> cd .\terraform-ec2-for-each\  
PS C:\Users\Lenovo\OneDrive\Desktop\SPCM Lab\LAB 9\terraform-ec2-for-each> 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.  
PS C:\Users\Lenovo\OneDrive\Desktop\SPCM Lab\LAB 9\terraform-ec2-for-each> |
```

```
PS C:\Users\Lenovo\OneDrive\Desktop\SPCM Lab\LAB 9\terraform-ec2-for-each> terraform plan  
  
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.  
PS C:\Users\Lenovo\OneDrive\Desktop\SPCM Lab\LAB 9\terraform-ec2-for-each> |
```

- Terraform will prompt you to confirm the creation of EC2 instances. Type yes and press Enter.

### 3. Verify Instances in AWS Console:

- Log in to the AWS Management Console and navigate to the EC2 service.
- Verify that the specified EC2 instances with the specified names and settings have been created.

### 4. Update Instance Configuration:

- If you want to modify the EC2 instance configuration, update the main.tf file with the desired changes.
- Rerun the terraform apply command to apply the changes:

```
terraform apply
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

aws_instance.ec2_instances["instance3"]: Still creating... [10s elapsed]
aws_instance.ec2_instances["instance1"]: Still creating... [20s elapsed]
aws_instance.ec2_instances["instance2"]: Still creating... [20s elapsed]
aws_instance.ec2_instances["instance3"]: Still creating... [20s elapsed]
aws_instance.ec2_instances["instance1"]: Still creating... [30s elapsed]
aws_instance.ec2_instances["instance2"]: Still creating... [30s elapsed]
aws_instance.ec2_instances["instance3"]: Still creating... [30s elapsed]
aws_instance.ec2_instances["instance2"]: Creation complete after 32s [id=i-03999b333f855a9be]
aws_instance.ec2_instances["instance3"]: Creation complete after 32s [id=i-0ee8795c8fe009d1b]
aws_instance.ec2_instances["instance1"]: Creation complete after 32s [id=i-0de528d7d2b8b4231]

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

## 5. Clean Up:

- After testing, you can clean up the EC2 instances:

```
terraform destroy
```

- Confirm the destruction by typing yes.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

aws_instance.ec2_instances["instance1"]: Still destroying... [id=i-0de528d7d2b8b4231, 10s elapsed]
aws_instance.ec2_instances["instance2"]: Still destroying... [id=i-03999b333f855a9be, 20s elapsed]
aws_instance.ec2_instances["instance3"]: Still destroying... [id=i-0ee8795c8fe009d1b, 20s elapsed]
aws_instance.ec2_instances["instance1"]: Still destroying... [id=i-0de528d7d2b8b4231, 20s elapsed]
aws_instance.ec2_instances["instance2"]: Still destroying... [id=i-03999b333f855a9be, 30s elapsed]
aws_instance.ec2_instances["instance3"]: Still destroying... [id=i-0ee8795c8fe009d1b, 30s elapsed]
aws_instance.ec2_instances["instance1"]: Still destroying... [id=i-0de528d7d2b8b4231, 30s elapsed]
aws_instance.ec2_instances["instance2"]: Destruction complete after 30s
aws_instance.ec2_instances["instance3"]: Still destroying... [id=i-0ee8795c8fe009d1b, 40s elapsed]
aws_instance.ec2_instances["instance1"]: Still destroying... [id=i-0de528d7d2b8b4231, 40s elapsed]
aws_instance.ec2_instances["instance1"]: Destruction complete after 40s
aws_instance.ec2_instances["instance3"]: Destruction complete after 40s

Destroy complete! Resources: 3 destroyed.
PS C:\Users\Lenovo\OneDrive\Desktop\SPCM Lab\LAB 9\terraform-ec2-for-each>
```

## 6. Conclusion:

This lab exercise demonstrates how to use the `for_each` construct in Terraform to create multiple AWS EC2 instances with specific settings for each instance. The use of a map allows you to define and manage settings for each instance individually. Experiment with different instance types, AMIs, and settings in the `main.tf` file to observe how Terraform provisions resources based on your configuration.

