

## 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
```

```
(base) → MyLab git:(main) mkdir terraform-ec2-for-each-lab9  
(base) → MyLab git:(main) cd terraform-ec2-for-each-lab9  
(base) → terraform-ec2-for-each-lab9 git:(main) ls  
(base) → terraform-ec2-for-each-lab9 git:(main) |
```

- 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"  
}
```

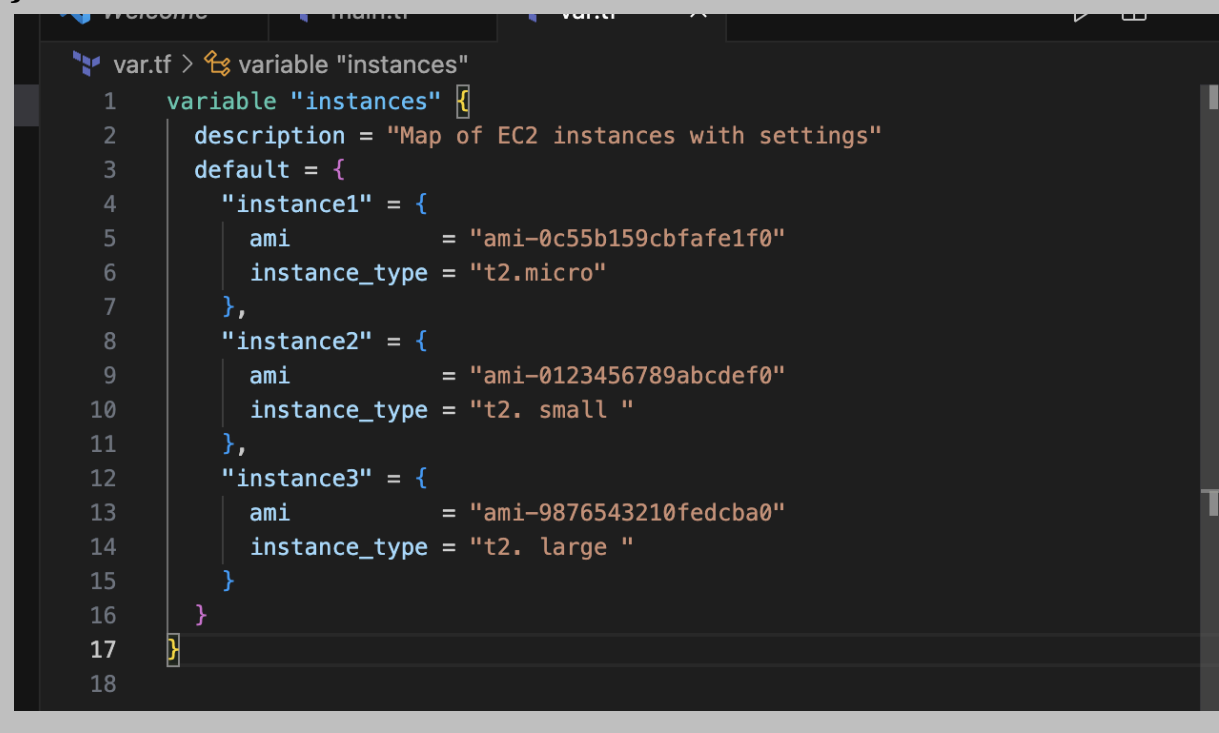


```
main.tf > ...  
1  ∨ terraform {  
2  ∨    required_providers {  
3  ∨      aws = {  
4      |   source = "hashicorp/aws"  
5      |   version = "5.68.0"  
6      | }  
7    }  
8  }  
9  
10  
11 ∨ provider "aws" {  
12 |   access_key = "AKIA4ZEEDP7HOCZXIMPT"  
13 |   secret_key = "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"  
14 |   region = "ap-south-1"  
15 | }  
16 |
```

#Var.tf

```
variable "instances" {  
  description = "Map of EC2 instances with settings"  
  default = {  
    "instance1" = {  
      ami      = "ami-oc55b159cbfafa1fo"  
      instance_type = "t2.micro"  
    },  
  },  
}
```

```
"instance2" = {  
  ami      = "ami-0123456789abcdef0"  
  instance_type = "t2. small "  
},  
"instance3" = {  
  ami      = "ami-9876543210fedcbao"  
  instance_type = "t2. large "  
}  
}  
}
```



The screenshot shows a code editor with a dark theme. At the top, there's a snippet of Terraform code defining three instance types. Below that, a terminal window shows the command 'var.tf > variable "instances"' and the resulting JSON structure for the 'instances' variable. The JSON structure is a map with three entries: 'instance1', 'instance2', and 'instance3', each containing 'ami' and 'instance\_type' values.

```
var.tf > variable "instances"  
1  variable "instances" {  
2    description = "Map of EC2 instances with settings"  
3    default = {  
4      "instance1" = {  
5        ami      = "ami-0c55b159cbfafa1f0"  
6        instance_type = "t2.micro"  
7      },  
8      "instance2" = {  
9        ami      = "ami-0123456789abcdef0"  
10       instance_type = "t2. small "  
11     },  
12     "instance3" = {  
13       ami      = "ami-9876543210fedcba0"  
14       instance_type = "t2. large "  
15     }  
16   }  
17 }  
18
```

## #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}"  
  }  
}
```

```
}  
}  
  
instance.tf > ...  
1  resource "aws_instance" "ec2_instances" {  
2      for_each = var.instances  
3      ami      = var.instances[each.key].ami  
4      instance_type = var.instances[each.key].instance_type  
5      tags = {  
6          Name = "EC2-Instance-${each.key}"  
7      }  
8  }  
9  # Compare this snippet from output.tf:
```

- 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  
  
(base) → terraform-ec2-for-each-lab9 git:(main) × 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.  
(base) → terraform-ec2-for-each-lab9 git:(main) × |
```

### terraform apply

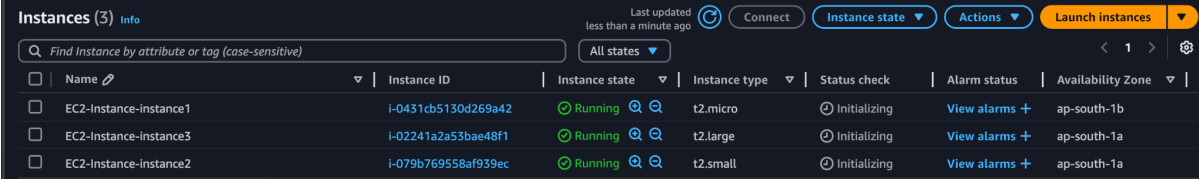
```
aws_instance.ec2_instances["instance3"]: Creating...
aws_instance.ec2_instances["instance2"]: Creating...
aws_instance.ec2_instances["instance3"]: Still creating... [10s elapsed]
aws_instance.ec2_instances["instance2"]: Still creating... [10s elapsed]
aws_instance.ec2_instances["instance3"]: Creation complete after 13s [id=i-02241a2a53bae48f1]
aws_instance.ec2_instances["instance2"]: Creation complete after 16s [id=i-079b769558af939ec]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
(base) → terraform-ec2-for-each-lab9 git:(main) ×
```

- 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.



Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
EC2-Instance-instance1	i-0431cb5130d269a42	Running	t2.micro	Initializing	View alarms +	ap-south-1b
EC2-Instance-instance3	i-02241a2a53bae48f1	Running	t2.large	Initializing	View alarms +	ap-south-1a
EC2-Instance-instance2	i-079b769558af939ec	Running	t2.small	Initializing	View alarms +	ap-south-1a

## 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

## 5. Clean Up:

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

### terraform destroy

```
aws_instance.ec2_instances["instance2"]: Destroying... [id=i-079b769558af939ec]
aws_instance.ec2_instances["instance1"]: Destroying... [id=i-0431cb5130d269a42]
aws_instance.ec2_instances["instance3"]: Destroying... [id=i-02241a2a53bae48f1]
aws_instance.ec2_instances["instance2"]: Still destroying... [id=i-079b769558af939ec, 10s elapsed]
aws_instance.ec2_instances["instance1"]: Still destroying... [id=i-0431cb5130d269a42, 10s elapsed]
aws_instance.ec2_instances["instance3"]: Still destroying... [id=i-02241a2a53bae48f1, 10s elapsed]
aws_instance.ec2_instances["instance3"]: Still destroying... [id=i-02241a2a53bae48f1, 20s elapsed]
aws_instance.ec2_instances["instance2"]: Still destroying... [id=i-079b769558af939ec, 20s elapsed]
aws_instance.ec2_instances["instance1"]: Still destroying... [id=i-0431cb5130d269a42, 20s elapsed]
aws_instance.ec2_instances["instance1"]: Still destroying... [id=i-0431cb5130d269a42, 30s elapsed]
aws_instance.ec2_instances["instance3"]: Still destroying... [id=i-02241a2a53bae48f1, 30s elapsed]
aws_instance.ec2_instances["instance2"]: Still destroying... [id=i-079b769558af939ec, 30s elapsed]
aws_instance.ec2_instances["instance2"]: Destruction complete after 32s
aws_instance.ec2_instances["instance1"]: Still destroying... [id=i-0431cb5130d269a42, 40s elapsed]
aws_instance.ec2_instances["instance3"]: Still destroying... [id=i-02241a2a53bae48f1, 40s elapsed]
aws_instance.ec2_instances["instance3"]: Destruction complete after 43s
aws_instance.ec2_instances["instance1"]: Still destroying... [id=i-0431cb5130d269a42, 50s elapsed]
aws_instance.ec2_instances["instance1"]: Still destroying... [id=i-0431cb5130d269a42, 1m0s elapsed]
aws_instance.ec2_instances["instance1"]: Destruction complete after 1m5s

Destroy complete! Resources: 3 destroyed.
(base) → terraform-ec2-for-each-lab9 git:(main) ×
```

- Confirm the destruction by typing yes.

## 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.