

Lab Exercise 9– Creating Multiple EC2 Instances with for_each in Terraform

Aditya Tomar

500106015

R2142221060

Batch-2(DevOps)

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
```


```
adityatomar@Adityas-MacBook-Air-3 ~ % mkdir terraform-ec2-for-each
adityatomar@Adityas-MacBook-Air-3 ~ % cd terraform-ec2-for-each
adityatomar@Adityas-MacBook-Air-3 terraform-ec2-for-each % touch main.tf
```

- 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 > terraform  
1 terraform {  
2   required_providers {  
3     aws = {  
4       source = "hashicorp/aws"  
5       version = "5.68.0"  
6     }  
7   }  
8 }  
9  
10 provider "aws" {  
11   access_key = "AKIATTSKF2XAPM4CHJE0"  
12   secret_key = "kBjGoZ9wdF03Yyn3RudmKmzJso7tcSwLU8c01kkz"  
13   region = "ap-south-1"  
14 }  
15
```

#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-9876543210fedcba0"
instance_type = "t2. large "
}
}
}
```

```
var.tf > ...
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
```

- 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

```

aws ec2 run-instances --profile default --region us-east-1 --terraform-excl-for-each N terraform-excl-for-each N
Initiating the backend...
Installing provider plugin...
- Downloading version 0.12.0 of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v0.12.0

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.

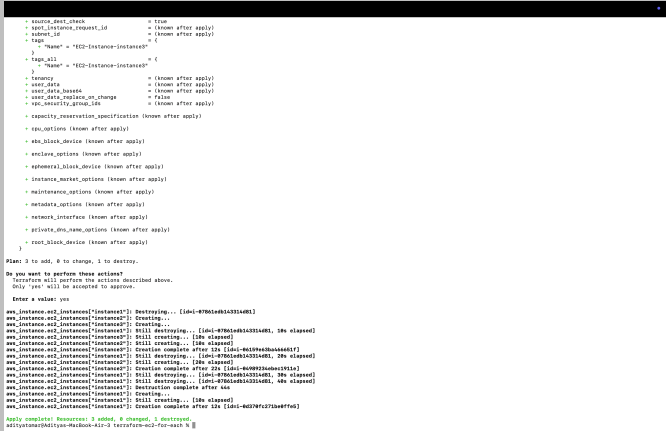
edits@terracore-us-east-1:~/tf-ec2-terraform-excl-for-each N$ terraform apply
aws_instance["instance1"] refreshing state... [local-07861404143344621]

Terraform will use the selected providers to generate the following execution plan. Resource actions are indicated with the following
symbols:
  ~ create
  ~ destroy and then create replacement

Terraform will perform the following actions:

# aws_instance["ec2-instance1"] will be created, so will be replaced
+ resource "aws_instance" "ec2_instance1" {
  ami              = "ami-0c55b1e9-99279096-13d3-899399209090:linux/ubuntu-18.04-lts-ami-20190801" ~ (known after apply)
  architecture     = "x86_64" ~ (known after apply)
  associate_public_ip_address = true ~ (known after apply)
  availability_zone = "us-east-1a" ~ (known after apply)
  cpu_credits       = "standard" ~ (known after apply)
  cpu_core_count    = 1 ~ (known after apply)
  cpu_threads_per_core = 1 ~ (known after apply)
  disable_api_termination = false ~ (known after apply)
  disable_auto_termination = false ~ (known after apply)
  ebs_optimized      = false ~ (known after apply)
  elb_optimized      = false ~ (known after apply)
  hibernation        = false ~ (known after apply)
  host_id            = (known after apply)
  host_resource_group_arn = (known after apply)
  iam_instance_profile = (known after apply)
  id                 = "i-07861404143344621" ~ (known after apply)
  instance_initiated_shutdown_behavior = "stop" ~ (known after apply)
  instance_lifecycle = (known after apply)
  instance_state     = "pending" ~ (known after apply)
  ipv6_address_count = 0 ~ (known after apply)
  ipv6_addresses     = (known after apply)
  key_name            = (known after apply)
  monitoring          = false ~ (known after apply)
  outpost_arn         = (known after apply)
  password_data       = (known after apply)
  placement_group     = (known after apply)
  placement_partition_number = 0 ~ (known after apply)
  primary_network_interface_id = "eni-0d891814a818219a78" ~ (known after apply)
  private_dns         = "ip-172-31-46-198.us-east-1.compute.amazonaws.com" ~ (known after apply)
  private_ip          = "172.31.46.198" ~ (known after apply)
  private_ip_prefix = "ip-172-31-46-198.us-east-1.compute.amazonaws.com" ~ (known after apply)
  public_ip           = "52.66.164.111" ~ (known after apply)
  secondary_private_ips = [ ] ~ (known after apply)
  security_groups     = [ "sg-0a1c1c1c" ] ~ (known after apply)
  subnet_id           = "subnet-b0d27c3ae8298264" ~ (known after apply)
  tags                = [ ]
  name               = "EC2-Instance-Instance1"
}

```



- ### 3. Verify Instances in AWS Console:

- #### 4. Update Instance Configuration:

- terraform apply

5. Clean Up:

- terraform destroy

terraform destroy

```
aws_instance.ec2_instances["instance2"]: Destroying... [id=i-04989234ebec1911e]
aws_instance.ec2_instances["instance1"]: Destroying... [id=i-0d378fc271be0ffe5]
aws_instance.ec2_instances["instance3"]: Destroying... [id=i-06159e63ba466651f]
aws_instance.ec2_instances["instance1"]: Still destroying... [id=i-0d378fc271be0ffe5, 10s elapsed]
aws_instance.ec2_instances["instance2"]: Still destroying... [id=i-04989234ebec1911e, 10s elapsed]
aws_instance.ec2_instances["instance3"]: Still destroying... [id=i-06159e63ba466651f, 10s elapsed]
aws_instance.ec2_instances["instance2"]: Still destroying... [id=i-06159e63ba466651f, 20s elapsed]
aws_instance.ec2_instances["instance1"]: Still destroying... [id=i-04989234ebec1911e, 20s elapsed]
aws_instance.ec2_instances["instance1"]: Still destroying... [id=i-0d378fc271be0ffe5, 20s elapsed]
aws_instance.ec2_instances["instance2"]: Still destroying... [id=i-0d378fc271be0ffe5, 30s elapsed]
aws_instance.ec2_instances["instance3"]: Still destroying... [id=i-04989234ebec1911e, 30s elapsed]
aws_instance.ec2_instances["instance3"]: Still destroying... [id=i-06159e63ba466651f, 30s elapsed]
aws_instance.ec2_instances["instance2"]: Destruction complete after 30s
aws_instance.ec2_instances["instance1"]: Destruction complete after 31s
aws_instance.ec2_instances["instance3"]: Still destroying... [id=i-06159e63ba466651f, 40s elapsed]
aws_instance.ec2_instances["instance3"]: Still destroying... [id=i-06159e63ba466651f, 50s elapsed]
aws_instance.ec2_instances["instance3"]: Destruction complete after 51s

Destroy complete! Resources: 3 destroyed.
```

- 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.