System Provisioning and Configuration Management LAB

SUBMITTED TO

Dr. Hitesh Kumar Sharma

SUBMITTED BY

Siddharth Agarwal

500107594

R2142220663

Btech CSE DevOps B1

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.

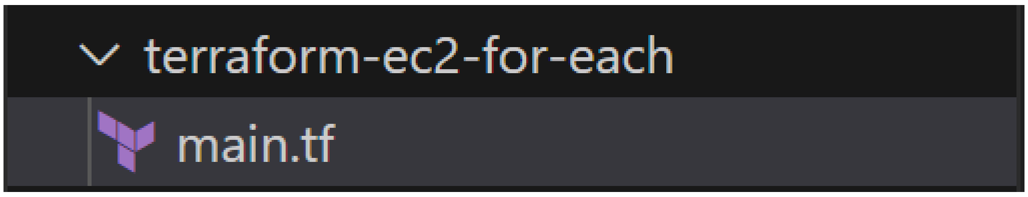
Prerequisites:

- 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
```

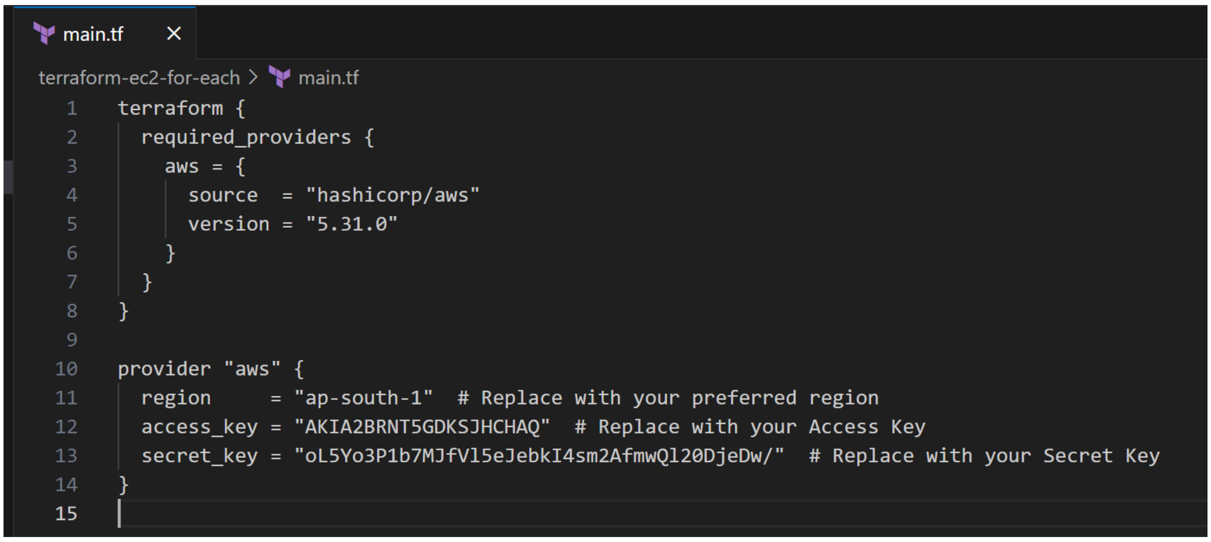


```
terraform-ec2-for-each
    main.tf
```

- 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"
}
```

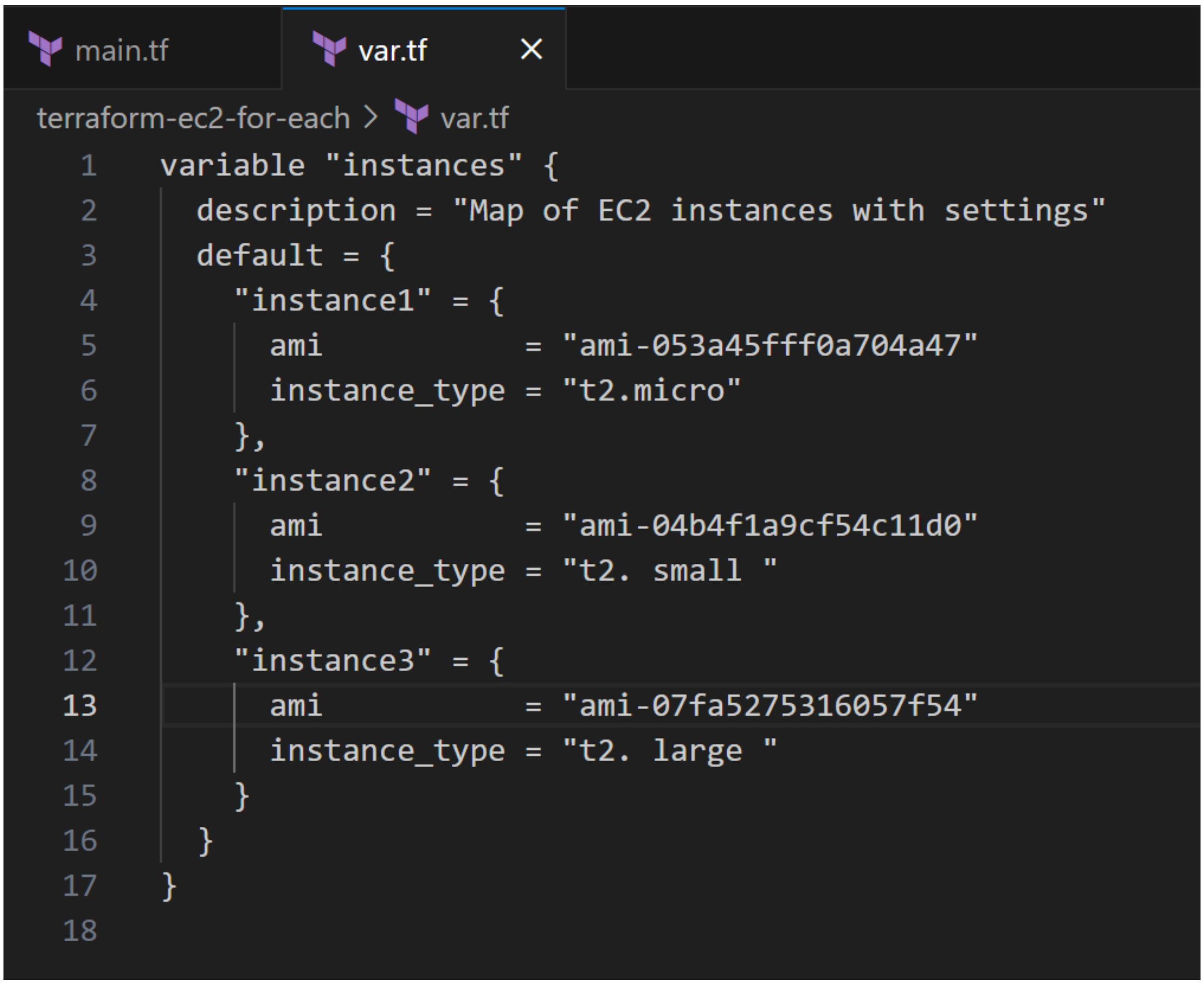

main.tf

terraform-ec2-for-each > main.tf

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "5.31.0"
    }
  }
}

provider "aws" {
  region     = "ap-south-1"  # Replace with your preferred region
  access_key = "AKIA2BRNT5GDKSJHCHAQ"  # Replace with your Access Key
  secret_key = "oL5Yo3P1b7MJfVl5eJebkI4sm2AfmwQl20DjeDw/"  # Replace with your Secret Key
}

```

#Var.tf

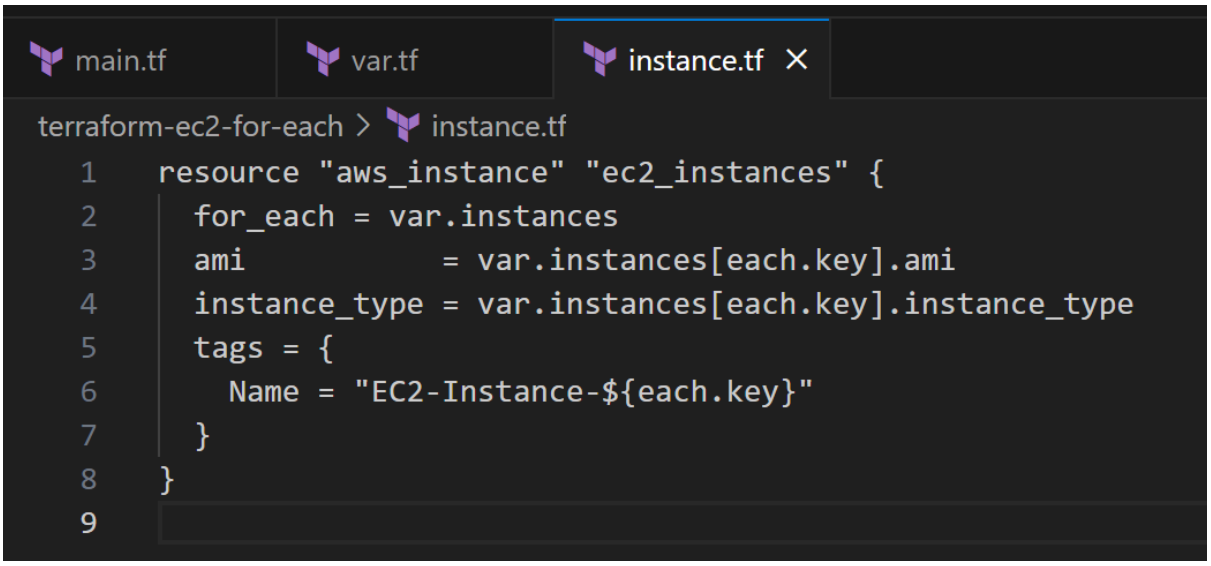
```
variable "instances" {
  description = "Map of EC2 instances with settings"
  default = {
    "instance1" = {
      ami           = "ami-0c55b159cbfafe1f0"
      instance_type = "t2.micro"
    },
    "instance2" = {
      ami           = "ami-0123456789abcdef0"
      instance_type = "t2. small "
    },
    "instance3" = {
      ami           = "ami-9876543210fedcba0"
    instance_type = "t2. large "
   }
  }
}
```



```
variable "instances" {
  description = "Map of EC2 instances with settings"
  default = {
    "instance1" = {
      ami           = "ami-053a45fff0a704a47"
      instance_type = "t2.micro"
    },
    "instance2" = {
      ami           = "ami-04b4f1a9cf54c11d0"
      instance_type = "t2. small "
    },
    "instance3" = {
      ami           = "ami-07fa5275316057f54"
      instance_type = "t2. large "
    }
  }
}
```

#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}"
  }
}
```



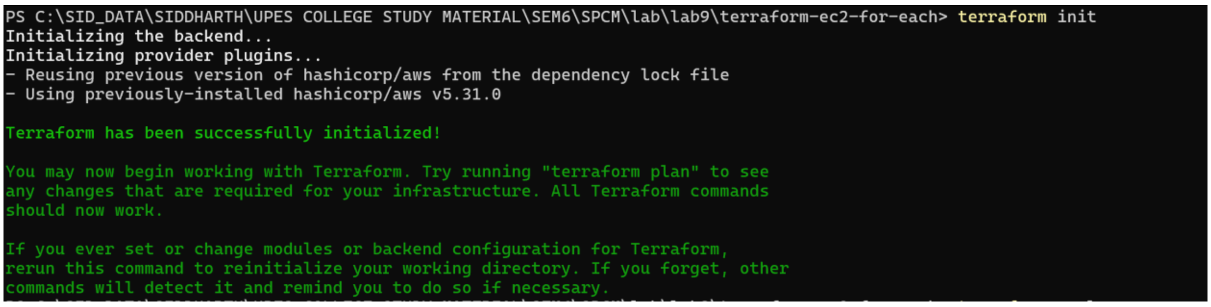
```
resource "aws_instance" "ec2_instances" {
  for_each = var.instances
  ami           = var.instances[each.key].ami
  instance_type = var.instances[each.key].instance_type
  tags = {
    Name = "EC2-Instance-${each.key}"
  }
}
```

- 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.

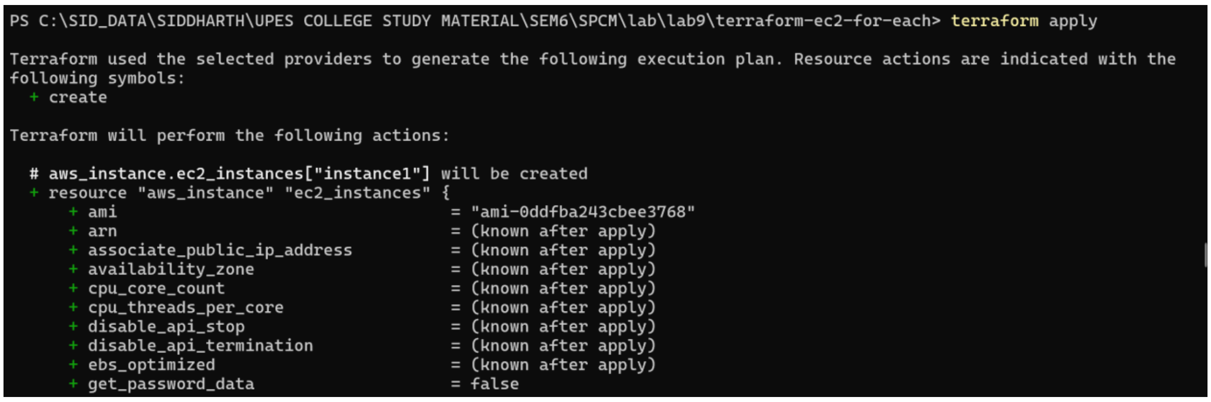
2. Initialize and Apply:

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

```
terraform init
terraform apply
```



```
PS C:\SID_DATA\SIDDHARTH\UPES COLLEGE STUDY MATERIAL\SEM6\SPCM\lab\lab9\terraform-ec2-for-each> terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.31.0

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.
```

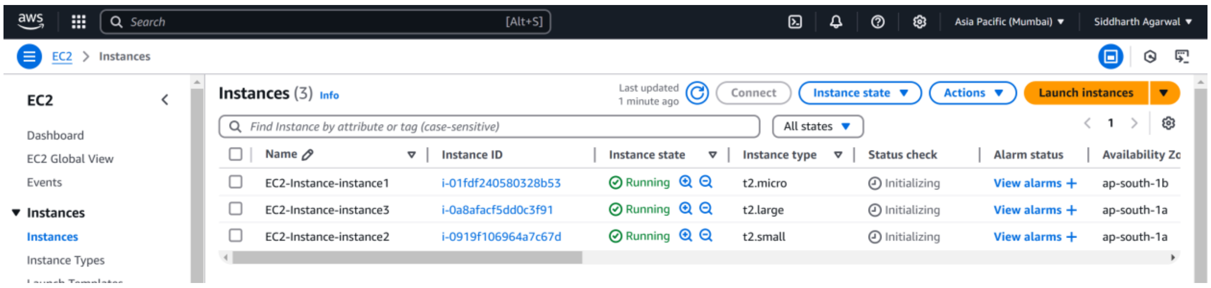


```
PS C:\SID_DATA\SIDDHARTH\UPES COLLEGE STUDY MATERIAL\SEM6\SPCM\lab\lab9\terraform-ec2-for-each> terraform apply

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:

  # aws_instance.ec2_instances["instance1"] will be created
  + resource "aws_instance" "ec2_instances" {
      + ami                                  = "ami-0ddfba243cbee3768"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = (known after apply)
      + cpu_core_count                       = (known after apply)
      + cpu_threads_per_core                 = (known after apply)
      + disable_api_stop                     = (known after apply)
      + disable_api_termination              = (known after apply)
      + ebs_optimized                        = (known after apply)
      + get_password_data                    = false
```

- 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.



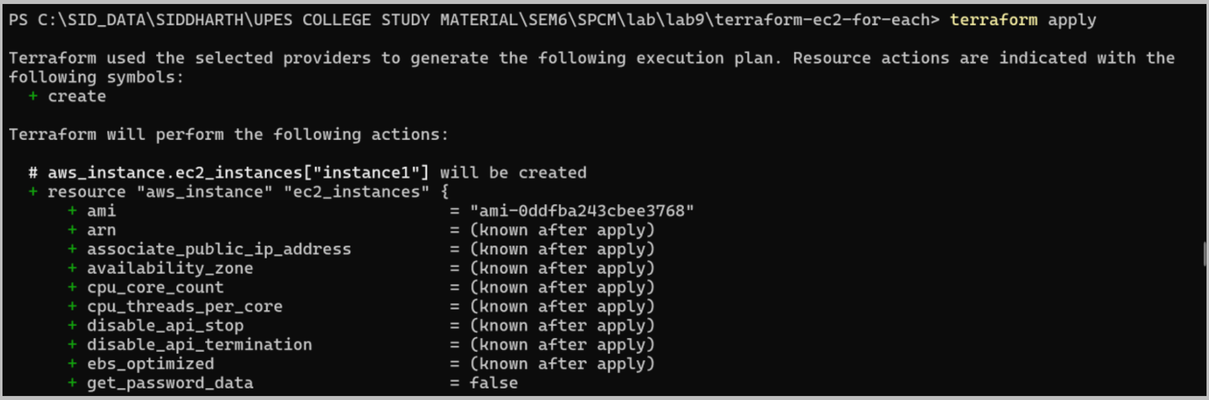
Instances (3) Info

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zo
EC2-Instance-instance1	i-01fdf240580328b53	Running	t2.micro	Initializing	View alarms +	ap-south-1b
EC2-Instance-instance3	i-0a8afacf5dd0c3f91	Running	t2.large	Initializing	View alarms +	ap-south-1a
EC2-Instance-instance2	i-0919f106964a7c67d	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
```

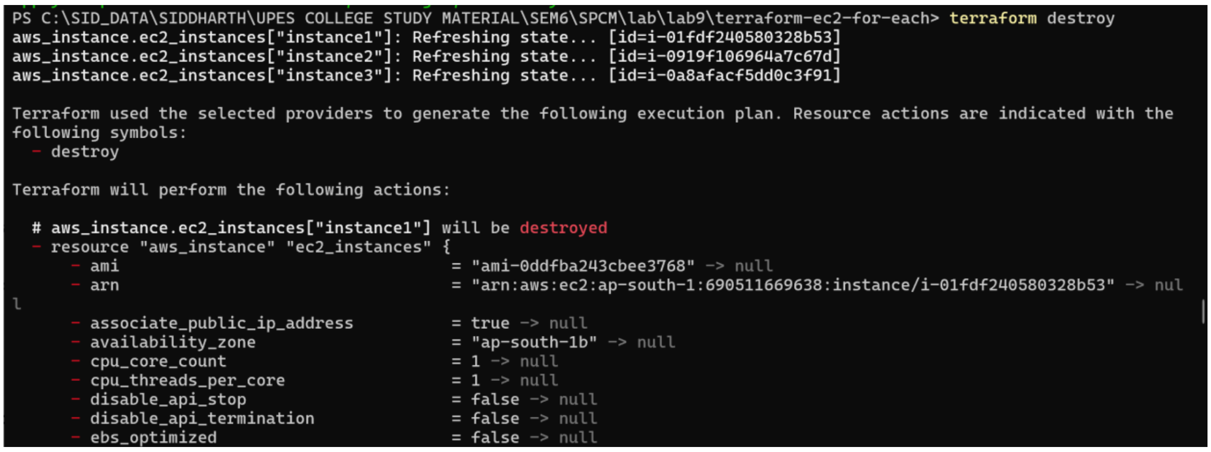


```
PS C:\SID_DATA\SIDDHARTH\UPES COLLEGE STUDY MATERIAL\SEM6\SPCM\lab\lab9\terraform-ec2-for-each> terraform apply

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:

  # aws_instance.ec2_instances["instance1"] will be created
  + resource "aws_instance" "ec2_instances" {
      + ami                                  = "ami-0ddfba243cbee3768"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = (known after apply)
      + cpu_core_count                       = (known after apply)
      + cpu_threads_per_core                 = (known after apply)
      + disable_api_stop                     = (known after apply)
      + disable_api_termination              = (known after apply)
      + ebs_optimized                        = (known after apply)
      + get_password_data                    = false
```

5. Clean Up:

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

```
terraform destroy
```



```
PS C:\SID_DATA\SIDDHARTH\UPES COLLEGE STUDY MATERIAL\SEM6\SPCM\lab\lab9\terraform-ec2-for-each> terraform destroy
aws_instance.ec2_instances["instance1"]: Refreshing state... [id=i-01fdf240580328b53]
aws_instance.ec2_instances["instance2"]: Refreshing state... [id=i-0919f106964a7c67d]
aws_instance.ec2_instances["instance3"]: Refreshing state... [id=i-0a8afacf5dd0c3f91]

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:

  # aws_instance.ec2_instances["instance1"] will be destroyed
  - resource "aws_instance" "ec2_instances" {
      - ami                                  = "ami-0ddfba243cbee3768" -> null
      - arn                                  = "arn:aws:ec2:ap-south-1:690511669638:instance/i-01fdf240580328b53" -> nul
l
      - associate_public_ip_address          = true -> null
      - availability_zone                    = "ap-south-1b" -> null
      - cpu_core_count                       = 1 -> null
      - cpu_threads_per_core                 = 1 -> null
      - disable_api_stop                     = false -> null
      - disable_api_termination              = false -> null
      - ebs_optimized                        = false -> null
```

- 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.