School of Computer Science

University of Petroleum and Energy Studies

System Provisioning & Configuration Management

Lab File (6th Sem)

Submitted By:-	Submitted To:-
Akshat Pandey 500101788 R2142220306 DevOps B1	Dr Hitesh Kumar Sharma

EXPERIMENT 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
```

```
PS C:\Users\aksha\Documents\SPCM_LAB> mkdir terraform-ec2-for-each


    Directory: C:\Users\aksha\Documents\SPCM_LAB


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        28-04-2025  01.12 PM                terraform-ec2-for-each


PS C:\Users\aksha\Documents\SPCM_LAB> cd terraform-ec2-for-each
PS C:\Users\aksha\Documents\SPCM_LAB\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"
}
```

```
main.tf
main.tf > provider "aws"
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = "5.68.0"
    }
  }
}

provider "aws" {
  access_key = "AKIASJ7PAFDUYXU4PLUQ"
  secret_key = "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"
  region = "ap-south-1"
}
```

#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 "
  }
 }
}
```

```
main.tf    Var.tf
Var.tf > variable "instances"
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 "
    }
  }
}
```

#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
Instance.tf > resource "aws_instance" "ec2_instances"
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

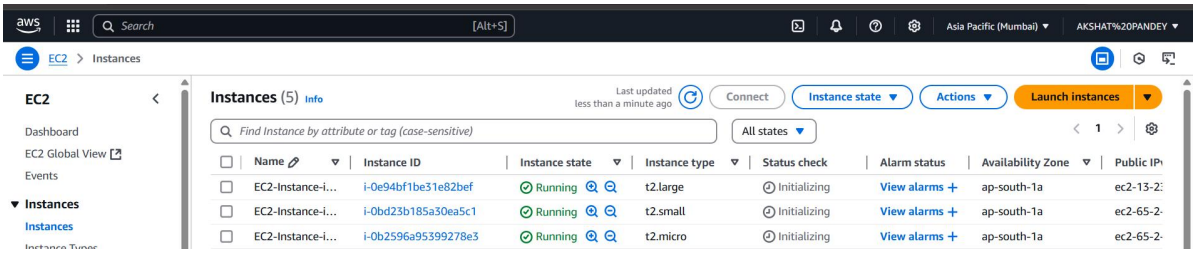
terraform apply

```
PS C:\Users\aksha\Documents\SPCM_LAB\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 v5.68.0

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:\Users\aksha\Documents\SPCM_LAB\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-03f4878755434977f"
      + 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
      + host_id                              = (known after apply)
      + host_resource_group_arn              = (known after apply)
      + iam_instance_profile                 = (known after apply)
      + id                                   = (known after apply)
      + instance_initiated_shutdown_behavior = (known after apply)
      + instance_lifecycle                   = (known after apply)
      + instance_state                       = (known after apply)
      + instance_type                        = "t2.micro"

      + network_interface (known after apply)

      + private_dns_name_options (known after apply)

      + root_block_device (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_instance.ec2_instances["instance1"]: Creating...
aws_instance.ec2_instances["instance2"]: Creating...
aws_instance.ec2_instances["instance3"]: Creating...
aws_instance.ec2_instances["instance1"]: Still creating... [10s elapsed]
aws_instance.ec2_instances["instance2"]: Still creating... [10s elapsed]
aws_instance.ec2_instances["instance3"]: Still creating... [10s elapsed]
aws_instance.ec2_instances["instance2"]: Creation complete after 13s [id=i-0bd23b185a30ea5c1]
aws_instance.ec2_instances["instance3"]: Still creating... [20s elapsed]
aws_instance.ec2_instances["instance1"]: Still creating... [20s elapsed]
aws_instance.ec2_instances["instance3"]: Creation complete after 22s [id=i-0e94bf1be31e82bef]
aws_instance.ec2_instances["instance1"]: Still creating... [30s elapsed]
aws_instance.ec2_instances["instance1"]: Creation complete after 32s [id=i-0b2596a95399278e3]

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

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

- Rerun the terraform apply command to apply the changes:

```
PS C:\Users\aksha\Documents\SPCM_LAB\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-03f4878755434977f"
      + 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
      + host_id                              = (known after apply)
      + host_resource_group_arn              = (known after apply)
      + iam_instance_profile                 = (known after apply)
      + id                                   = (known after apply)
      + instance_initiated_shutdown_behavior = (known after apply)
      + instance_lifecycle                   = (known after apply)
      + instance_state                       = (known after apply)
      + instance_type                        = "t2.micro"
```

```
      + network_interface (known after apply)

      + private_dns_name_options (known after apply)

      + root_block_device (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_instance.ec2_instances["instance1"]: Creating...
aws_instance.ec2_instances["instance2"]: Creating...
aws_instance.ec2_instances["instance3"]: Creating...
aws_instance.ec2_instances["instance1"]: Still creating... [10s elapsed]
aws_instance.ec2_instances["instance2"]: Still creating... [10s elapsed]
aws_instance.ec2_instances["instance3"]: Still creating... [10s elapsed]
aws_instance.ec2_instances["instance2"]: Creation complete after 13s [id=i-0bd23b185a30ea5c1]
aws_instance.ec2_instances["instance3"]: Still creating... [20s elapsed]
aws_instance.ec2_instances["instance1"]: Still creating... [20s elapsed]
aws_instance.ec2_instances["instance3"]: Creation complete after 22s [id=i-0e94bf1be31e82bef]
aws_instance.ec2_instances["instance1"]: Still creating... [30s elapsed]
aws_instance.ec2_instances["instance1"]: Creation complete after 32s [id=i-0b2596a95399278e3]

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

5. Clean Up:

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

terraform destroy

```
PS C:\Users\aksha\Documents\SPCM_LAB\terraform-ec2-for-each> terraform destroy
aws_instance.ec2_instances["instance3"]: Refreshing state... [id=i-0e94bf1be31e82bef]
aws_instance.ec2_instances["instance2"]: Refreshing state... [id=i-0bd23b185a30ea5c1]
aws_instance.ec2_instances["instance1"]: Refreshing state... [id=i-0b2596a95399278e3]

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-03f4878755434977f" -> null
      - arn                                  = "arn:aws:ec2:ap-south-1:158878148841:instance/i-0b2596a95399278e3" -> nul
l
      - associate_public_ip_address          = true -> null
      - availability_zone                    = "ap-south-1a" -> 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
      - get_password_data                    = false -> null
      - hibernation                          = false -> null
      - id                                   = "i-0b2596a95399278e3" -> null
      - instance_initiated_shutdown_behavior = "stop" -> null
      - instance_state                       = "running" -> null
```

```
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_instance.ec2_instances["instance1"]: Destroying... [id=i-0b2596a95399278e3]
aws_instance.ec2_instances["instance2"]: Destroying... [id=i-0bd23b185a30ea5c1]
aws_instance.ec2_instances["instance3"]: Destroying... [id=i-0e94bf1be31e82bef]
aws_instance.ec2_instances["instance2"]: Still destroying... [id=i-0bd23b185a30ea5c1, 10s elapsed]
aws_instance.ec2_instances["instance3"]: Still destroying... [id=i-0e94bf1be31e82bef, 10s elapsed]
aws_instance.ec2_instances["instance1"]: Still destroying... [id=i-0b2596a95399278e3, 10s elapsed]
aws_instance.ec2_instances["instance2"]: Still destroying... [id=i-0bd23b185a30ea5c1, 20s elapsed]
aws_instance.ec2_instances["instance1"]: Still destroying... [id=i-0b2596a95399278e3, 20s elapsed]
aws_instance.ec2_instances["instance3"]: Still destroying... [id=i-0e94bf1be31e82bef, 20s elapsed]
aws_instance.ec2_instances["instance3"]: Still destroying... [id=i-0e94bf1be31e82bef, 30s elapsed]
aws_instance.ec2_instances["instance2"]: Still destroying... [id=i-0bd23b185a30ea5c1, 30s elapsed]
aws_instance.ec2_instances["instance1"]: Still destroying... [id=i-0b2596a95399278e3, 30s elapsed]
aws_instance.ec2_instances["instance3"]: Destruction complete after 30s
aws_instance.ec2_instances["instance1"]: Destruction complete after 30s
aws_instance.ec2_instances["instance2"]: Still destroying... [id=i-0bd23b185a30ea5c1, 40s elapsed]
aws_instance.ec2_instances["instance2"]: Still destroying... [id=i-0bd23b185a30ea5c1, 50s elapsed]
aws_instance.ec2_instances["instance2"]: Still destroying... [id=i-0bd23b185a30ea5c1, 1m0s elapsed]
aws_instance.ec2_instances["instance2"]: Still destroying... [id=i-0bd23b185a30ea5c1, 1m10s elapsed]
aws_instance.ec2_instances["instance2"]: Still destroying... [id=i-0bd23b185a30ea5c1, 1m20s elapsed]
aws_instance.ec2_instances["instance2"]: Destruction complete after 1m20s

Destroy complete! Resources: 3 destroyed.
```

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.