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:

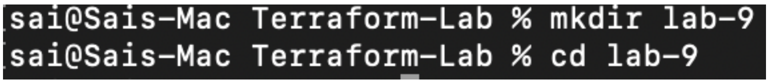
- Terraform installed on your machine.
- AWS CLI configured with the necessary credentials.

Steps:

1. Create a Terraform Directory:

```
mkdir lab-9
cd lab-9
```

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

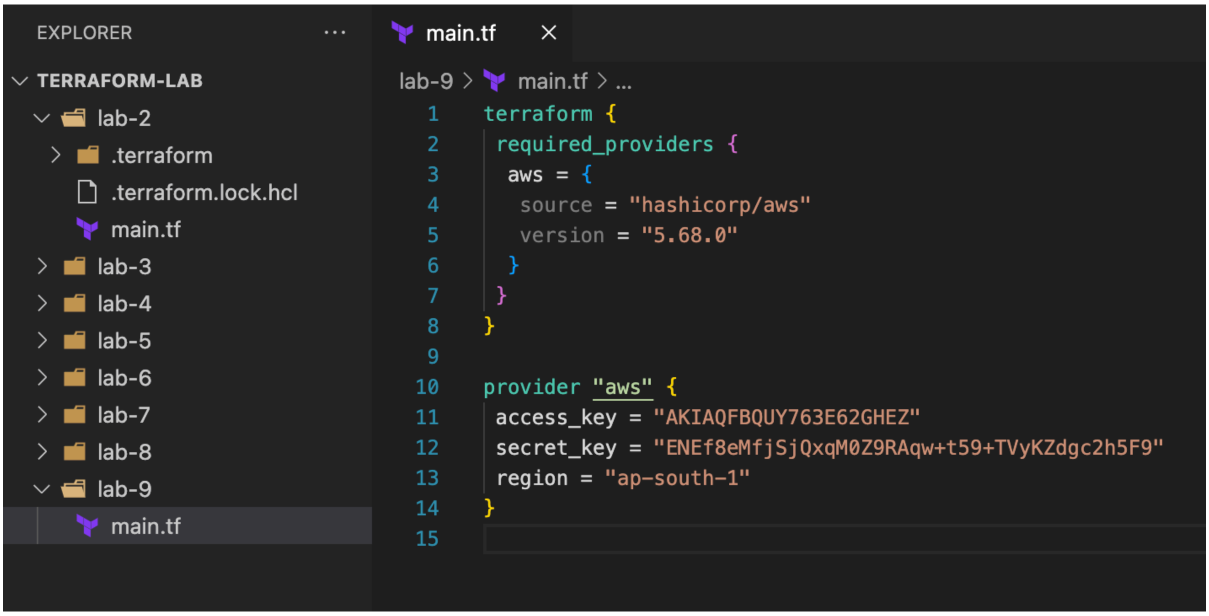


```
sai@Sais-Mac Terraform-Lab % mkdir lab-9
sai@Sais-Mac Terraform-Lab % cd lab-9
```

# main.tf

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

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

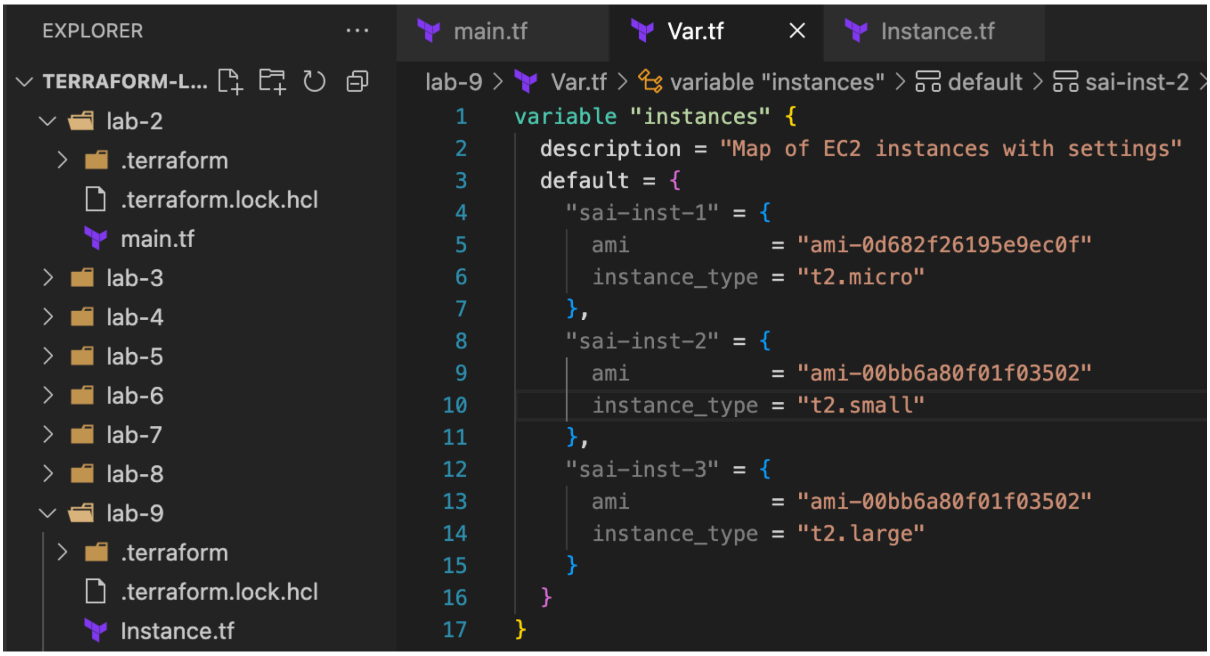


```
terraform {
 required_providers {
  aws = {
   source = "hashicorp/aws"
   version = "5.68.0"
  }
 }
}

provider "aws" {
 access_key = "AKIAQFBQUY763E62GHEZ"
 secret_key = "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"
 region = "ap-south-1"
}
```

#Var.tf

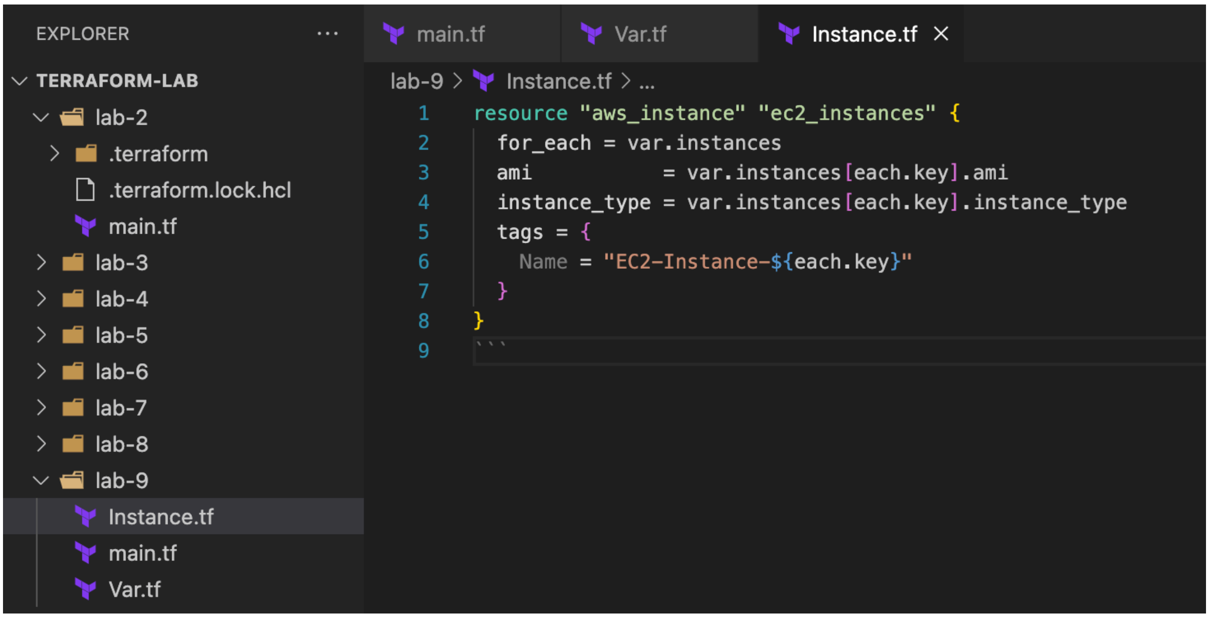
```
variable "instances" {
 description = "Map of EC2 instances with settings"
 default = {
  "instance1" = {
   ami           = "ami-0c55b159cbfafe1f0"
   instance_type = "t2.micro"
  },
  "instance2" = {
   ami          = "ami-0123456789abcdef0"
   instance_type = "t2. small "
  },
  "instance3" =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



```
variable "instances" {
  description = "Map of EC2 instances with settings"
  default = {
    "sai-inst-1" = {
      ami           = "ami-0d682f26195e9ec0f"
      instance_type = "t2.micro"
    },
    "sai-inst-2" = {
      ami           = "ami-00bb6a80f01f03502"
      instance_type = "t2.small"
    },
    "sai-inst-3" = {
      ami           = "ami-00bb6a80f01f03502"
      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}"
 }
}
```



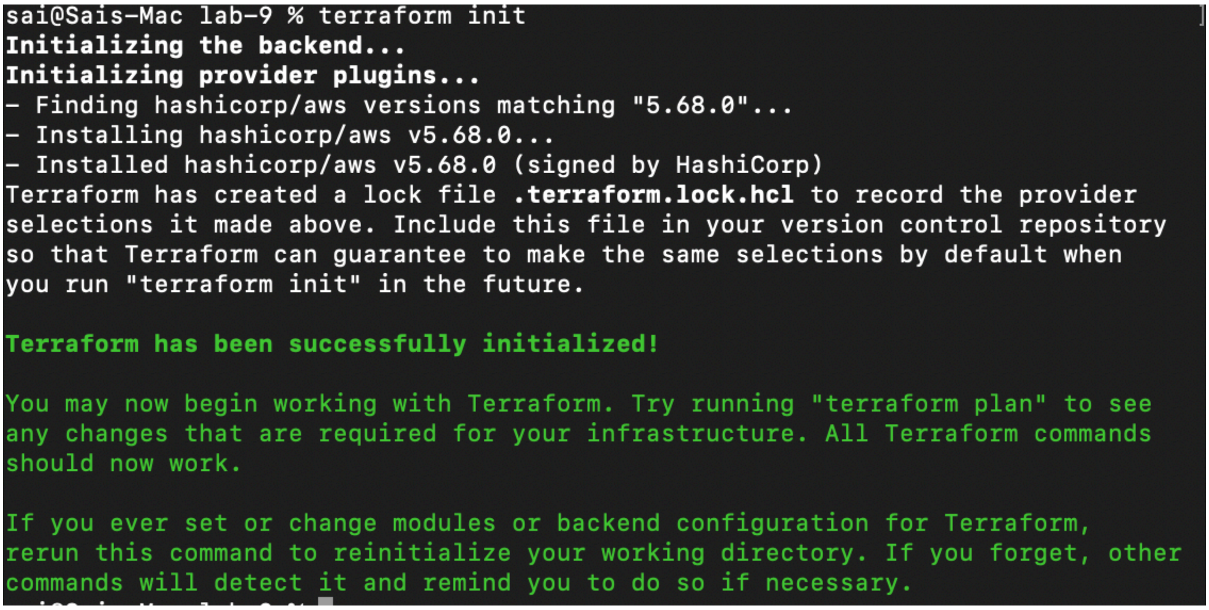
~~~~
resource "aws_instance" "ec2_instances" {
  for_each = var.instances
  ami           = var.instances[each.key].ami
  instance_type = var.instances[each.key].instance_type
  tags = {
    Name = "EC2-Instance-${each.key}"
  }
}
```
~~~~

- 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
```



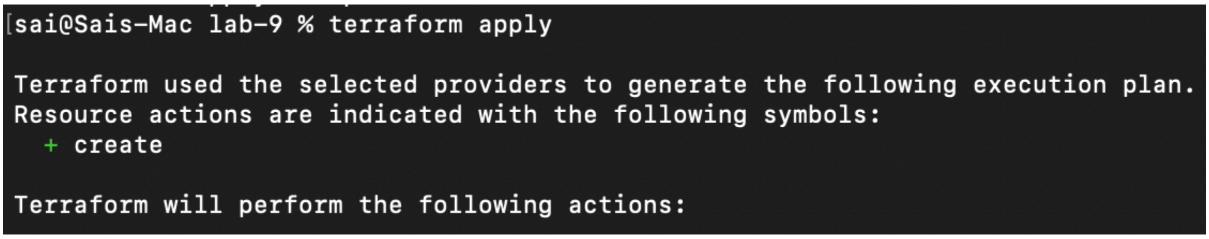
```
sai@Sais-Mac lab-9 % terraform init
Initializing the backend...
Initializing provider plugins...
- Finding hashicorp/aws versions matching "5.68.0"...
- Installing hashicorp/aws v5.68.0...
- Installed hashicorp/aws v5.68.0 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl 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.
```

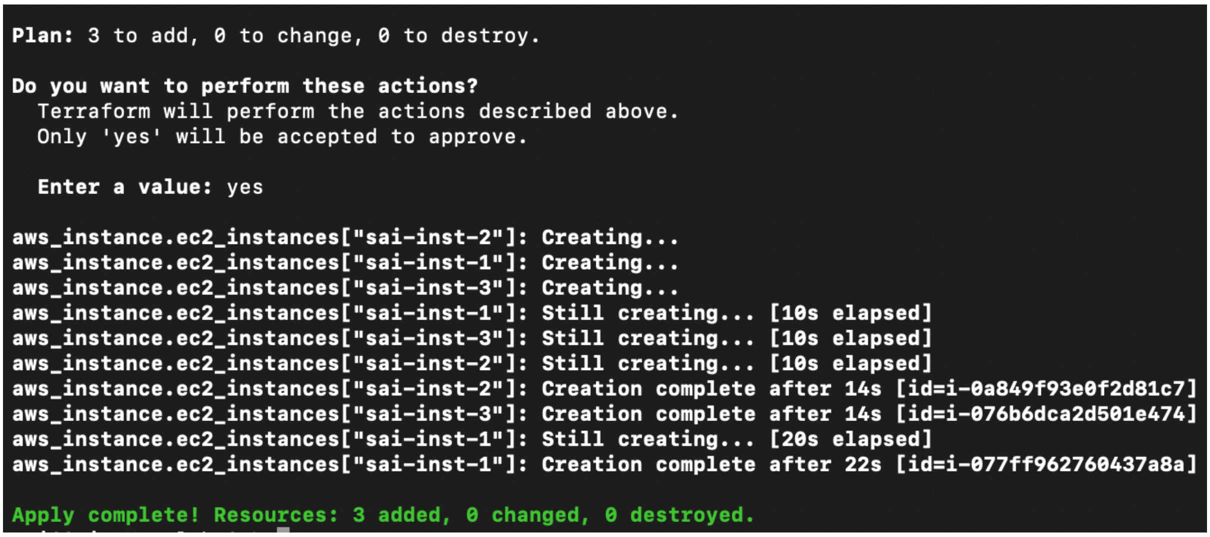
- Terraform will prompt you to confirm the creation of EC2 instances. Type yes and press Enter.



```
[sai@Sais-Mac lab-9 % 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:
```



```
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.

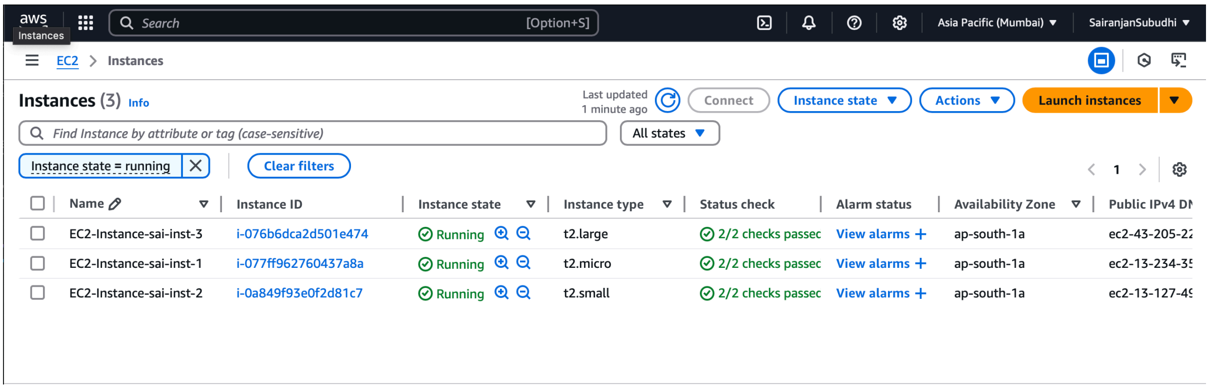
  Enter a value: yes

aws_instance.ec2_instances["sai-inst-2"]: Creating...
aws_instance.ec2_instances["sai-inst-1"]: Creating...
aws_instance.ec2_instances["sai-inst-3"]: Creating...
aws_instance.ec2_instances["sai-inst-1"]: Still creating... [10s elapsed]
aws_instance.ec2_instances["sai-inst-3"]: Still creating... [10s elapsed]
aws_instance.ec2_instances["sai-inst-2"]: Still creating... [10s elapsed]
aws_instance.ec2_instances["sai-inst-2"]: Creation complete after 14s [id=i-0a849f93e0f2d81c7]
aws_instance.ec2_instances["sai-inst-3"]: Creation complete after 14s [id=i-076b6dca2d501e474]
aws_instance.ec2_instances["sai-inst-1"]: Still creating... [20s elapsed]
aws_instance.ec2_instances["sai-inst-1"]: Creation complete after 22s [id=i-077ff962760437a8a]

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.

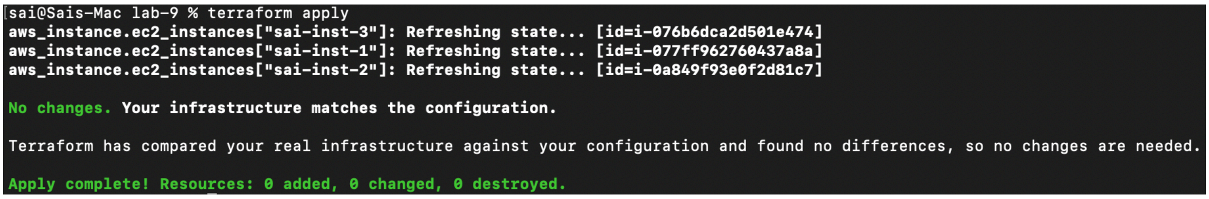


□	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DN
□	EC2-Instance-sai-inst-3	i-076b6dca2d501e474	✓ Running	t2.large	✓ 2/2 checks passec	View alarms +	ap-south-1a	ec2-43-205-22
□	EC2-Instance-sai-inst-1	i-077ff962760437a8a	✓ Running	t2.micro	✓ 2/2 checks passec	View alarms +	ap-south-1a	ec2-13-234-35
□	EC2-Instance-sai-inst-2	i-0a849f93e0f2d81c7	✓ Running	t2.small	✓ 2/2 checks passec	View alarms +	ap-south-1a	ec2-13-127-49

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
```



```
sai@Sais-Mac lab-9 % terraform apply
aws_instance.ec2_instances["sai-inst-3"]: Refreshing state... [id=i-076b6dca2d501e474]
aws_instance.ec2_instances["sai-inst-1"]: Refreshing state... [id=i-077ff962760437a8a]
aws_instance.ec2_instances["sai-inst-2"]: Refreshing state... [id=i-0a849f93e0f2d81c7]

No changes. Your infrastructure matches the configuration.

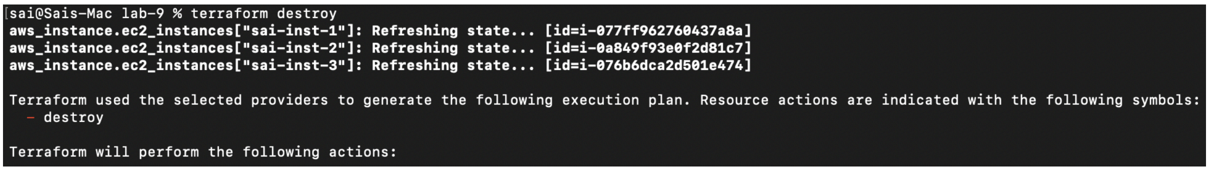
Terraform has compared your real infrastructure against your configuration and found no differences, so no changes are needed.

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

5. Clean Up:

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

```
terraform destroy
```

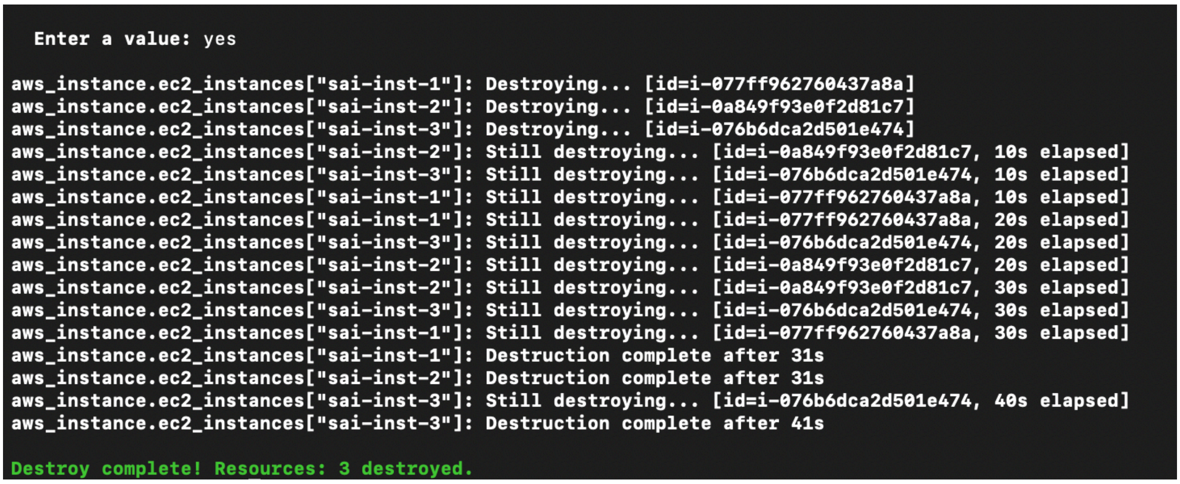


```
sai@Sais-Mac lab-9 % terraform destroy
aws_instance.ec2_instances["sai-inst-1"]: Refreshing state... [id=i-077ff962760437a8a]
aws_instance.ec2_instances["sai-inst-2"]: Refreshing state... [id=i-0a849f93e0f2d81c7]
aws_instance.ec2_instances["sai-inst-3"]: Refreshing state... [id=i-076b6dca2d501e474]

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:
```

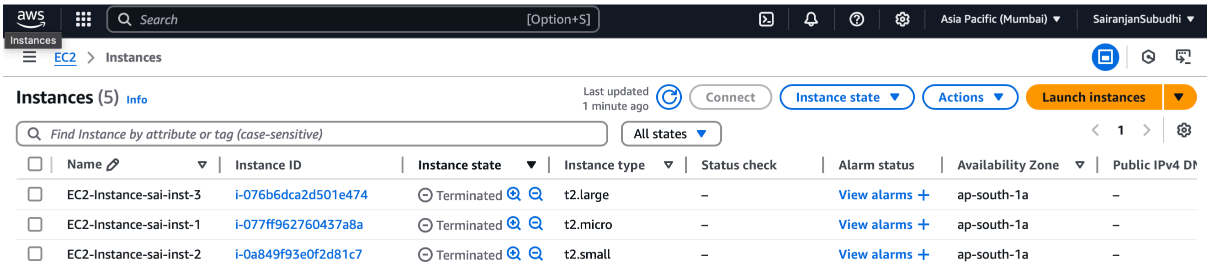
- Confirm the destruction by typing yes.



```
  Enter a value: yes

aws_instance.ec2_instances["sai-inst-1"]: Destroying... [id=i-077ff962760437a8a]
aws_instance.ec2_instances["sai-inst-2"]: Destroying... [id=i-0a849f93e0f2d81c7]
aws_instance.ec2_instances["sai-inst-3"]: Destroying... [id=i-076b6dca2d501e474]
aws_instance.ec2_instances["sai-inst-2"]: Still destroying... [id=i-0a849f93e0f2d81c7, 10s elapsed]
aws_instance.ec2_instances["sai-inst-3"]: Still destroying... [id=i-076b6dca2d501e474, 10s elapsed]
aws_instance.ec2_instances["sai-inst-1"]: Still destroying... [id=i-077ff962760437a8a, 10s elapsed]
aws_instance.ec2_instances["sai-inst-1"]: Still destroying... [id=i-077ff962760437a8a, 20s elapsed]
aws_instance.ec2_instances["sai-inst-3"]: Still destroying... [id=i-076b6dca2d501e474, 20s elapsed]
aws_instance.ec2_instances["sai-inst-2"]: Still destroying... [id=i-0a849f93e0f2d81c7, 20s elapsed]
aws_instance.ec2_instances["sai-inst-2"]: Still destroying... [id=i-0a849f93e0f2d81c7, 30s elapsed]
aws_instance.ec2_instances["sai-inst-3"]: Still destroying... [id=i-076b6dca2d501e474, 30s elapsed]
aws_instance.ec2_instances["sai-inst-1"]: Still destroying... [id=i-077ff962760437a8a, 30s elapsed]
aws_instance.ec2_instances["sai-inst-1"]: Destruction complete after 31s
aws_instance.ec2_instances["sai-inst-2"]: Destruction complete after 31s
aws_instance.ec2_instances["sai-inst-3"]: Still destroying... [id=i-076b6dca2d501e474, 40s elapsed]
aws_instance.ec2_instances["sai-inst-3"]: Destruction complete after 41s

Destroy complete! Resources: 3 destroyed.
```



Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DN
EC2-Instance-sai-inst-3	i-076b6dca2d501e474	⊖ Terminated	t2.large	–	View alarms +	ap-south-1a	–
EC2-Instance-sai-inst-1	i-077ff962760437a8a	⊖ Terminated	t2.micro	–	View alarms +	ap-south-1a	–
EC2-Instance-sai-inst-2	i-0a849f93e0f2d81c7	⊖ Terminated	t2.small	–	View alarms +	ap-south-1a	–

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.