Module-11: Provisioning Infrastructure using Terraform Part-I

Demo Document - 4

edureka!

DEMO-4: Modifying Resources in Terraform

To demonstrate this, we will create a new aws instance and change the tag name of the existing instance.

1. Modify the existing terraform configuration file using a text editor of your choice

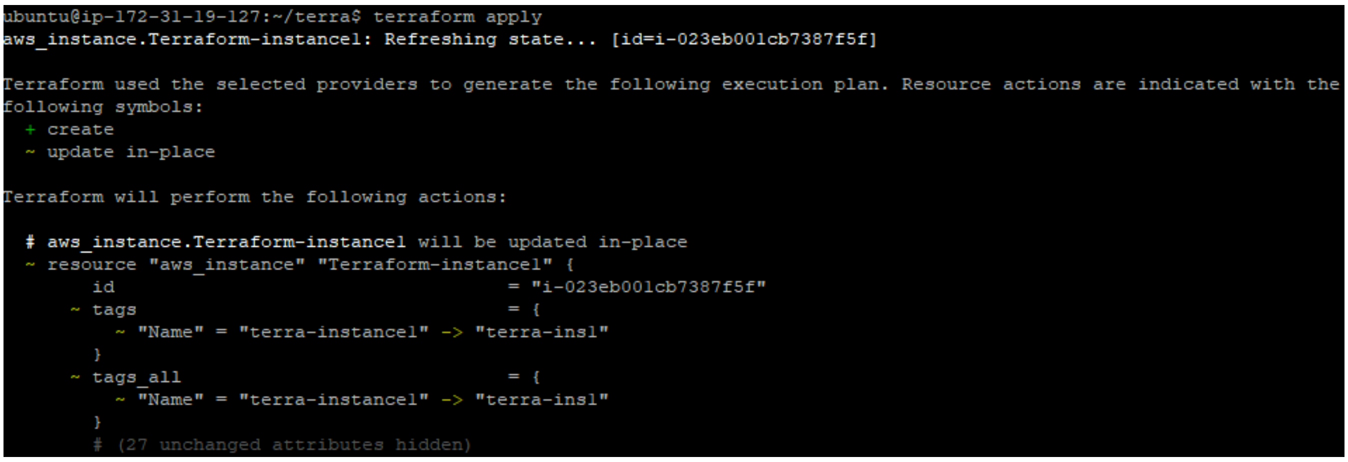
   Syntax: vi filename.tf
2. Edit the file with the following configuration

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      # optional
      version = "~> 3.0"
    }
  }
}

# Configuring provider
provider "aws" {
  region = "us-east-2"
  access_key = "my-access-key"
  secret_key = "my-secret-key"
}

# Changing the name of the old instance
resource "aws_instance" "Terraform-instance1" {
  ami           = "ami-00399ec92321828f5"
  instance_type = "t2.micro"
  tags = {
    Name = "terra-ins1"
  }
}
```

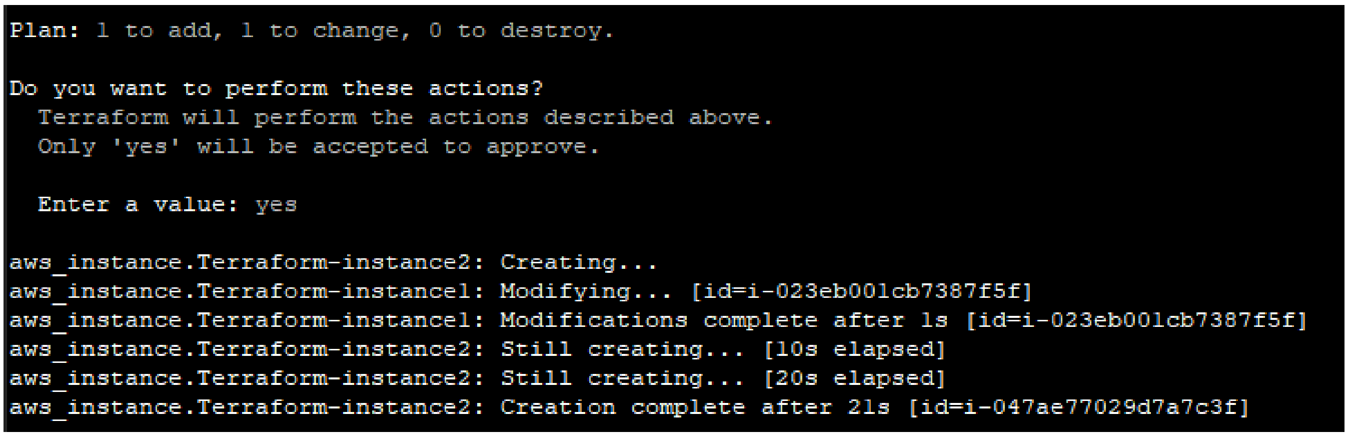
```
# Deploying a new instance
resource "aws_instance" "Terraform-instance2" {
  ami           = "ami-00399ec92321828f5"
  instance_type = "t2.micro"
  tags = {
    Name = "terra-ins2"
  }
}
```

3. Now enter the apply command and check the list of changes the updated configuration is going to make

   Syntax: terraform apply

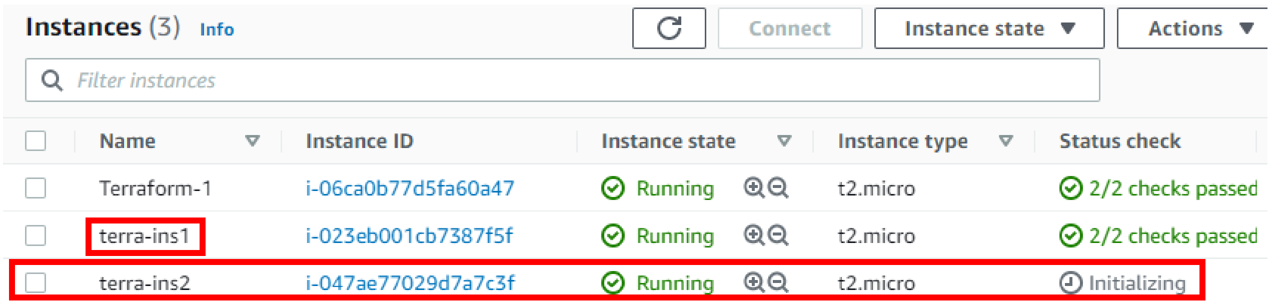


```
ubuntu@ip-172-31-19-127:~/terra$ terraform apply
aws_instance.Terraform-instance1: Refreshing state... [id=i-023eb001cb7387f5f]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the
following symbols:
  + create
  ~ update in-place

Terraform will perform the following actions:

  # aws_instance.Terraform-instance1 will be updated in-place
  ~ resource "aws_instance" "Terraform-instance1" {
        id                                   = "i-023eb001cb7387f5f"
      ~ tags                                 = {
          ~ "Name" = "terra-instance1" -> "terra-ins1"
        }
      ~ tags_all                             = {
          ~ "Name" = "terra-instance1" -> "terra-ins1"
        }
        # (27 unchanged attributes hidden)
```



```
Plan: 1 to add, 1 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.Terraform-instance2: Creating...
aws_instance.Terraform-instance1: Modifying... [id=i-023eb001cb7387f5f]
aws_instance.Terraform-instance1: Modifications complete after 1s [id=i-023eb001cb7387f5f]
aws_instance.Terraform-instance2: Still creating... [10s elapsed]
aws_instance.Terraform-instance2: Still creating... [20s elapsed]
aws_instance.Terraform-instance2: Creation complete after 21s [id=i-047ae77029d7a7c3f]
```

4. We can verify the changes made on the aws ec2 console



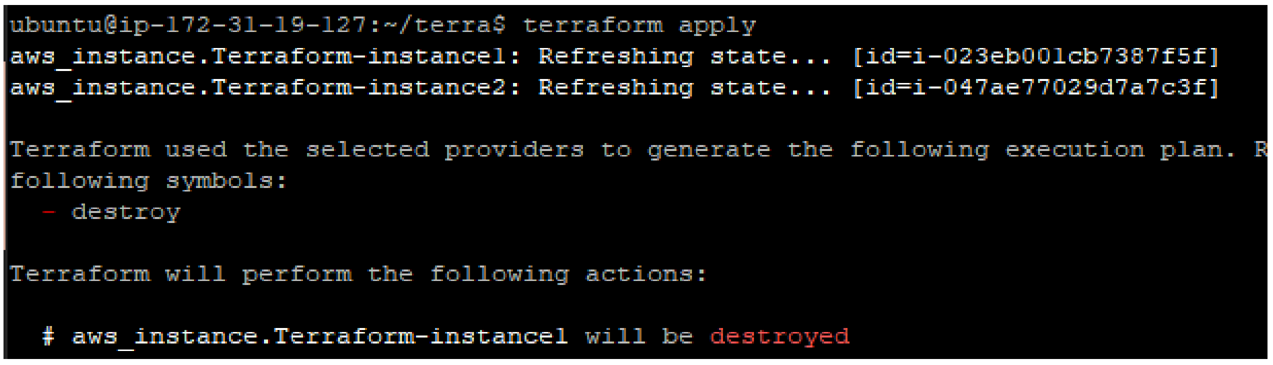
	Name	Instance ID	Instance state	Instance type	Status check
☐	Terraform-1	i-06ca0b77d5fa60a47	Running	t2.micro	2/2 checks passed
☐	terra-ins1	i-023eb001cb7387f5f	Running	t2.micro	2/2 checks passed
☐	terra-ins2	i-047ae77029d7a7c3f	Running	t2.micro	Initializing

Destroying your terraform infrastructure

1. To simply remove a resource, we can remove the code for it in Terraform and apply the changes. In this example we will comment out the first instance code in our configuration and apply the changes

```
/*resource "aws_instance" "Terraform-instance1" {

  ami           = "ami-00399ec92321828f5"

  instance_type = "t2.micro"

  tags = {

    Name = "terra-ins1"

  }

} */
```

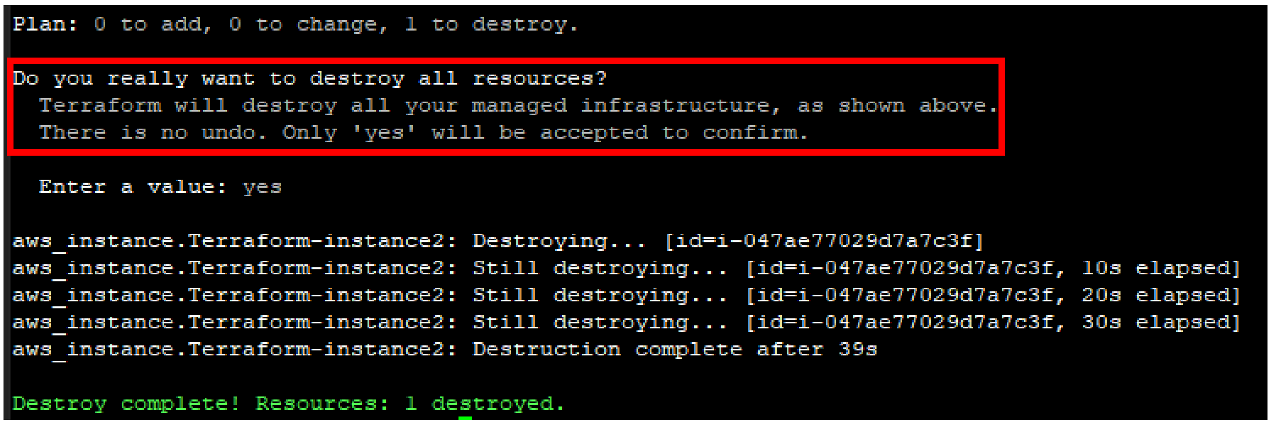
Apply the changes



```
ubuntu@ip-172-31-19-127:~/terra$ terraform apply
aws_instance.Terraform-instance1: Refreshing state... [id=i-023eb001cb7387f5f]
aws_instance.Terraform-instance2: Refreshing state... [id=i-047ae77029d7a7c3f]

Terraform used the selected providers to generate the following execution plan. R
following symbols:
  - destroy

Terraform will perform the following actions:

  # aws_instance.Terraform-instance1 will be destroyed
```

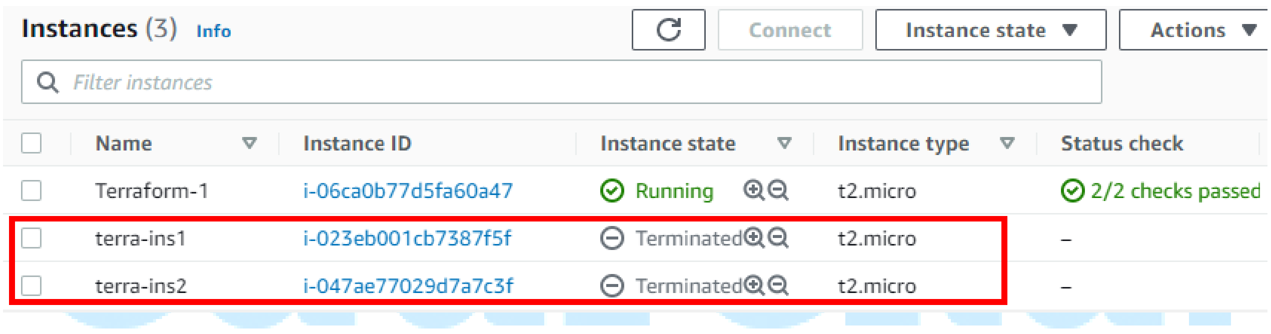
2. To destroy the entire infrastructure created by the configuration, we can use the destroy command

   Syntax: terraform destroy



```
Plan: 0 to add, 0 to change, 1 to destroy.

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.Terraform-instance2: Destroying... [id=i-047ae77029d7a7c3f]
aws_instance.Terraform-instance2: Still destroying... [id=i-047ae77029d7a7c3f, 10s elapsed]
aws_instance.Terraform-instance2: Still destroying... [id=i-047ae77029d7a7c3f, 20s elapsed]
aws_instance.Terraform-instance2: Still destroying... [id=i-047ae77029d7a7c3f, 30s elapsed]
aws_instance.Terraform-instance2: Destruction complete after 39s

Destroy complete! Resources: 1 destroyed.
```

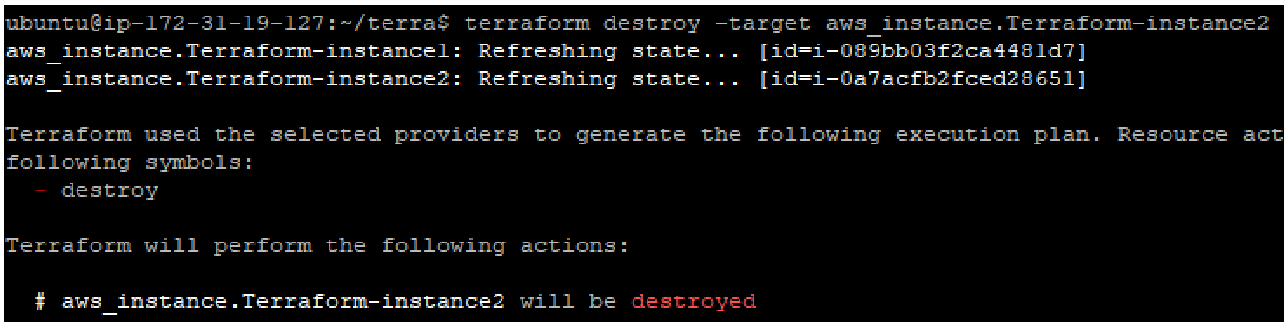
3. Verification can be done on the ec2 console



Name	Instance ID	Instance state	Instance type	Status check
Terraform-1	i-06ca0b77d5fa60a47	Running	t2.micro	2/2 checks passed
terra-ins1	i-023eb001cb7387f5f	Terminated	t2.micro	–
terra-ins2	i-047ae77029d7a7c3f	Terminated	t2.micro	–

4. To target a specific resource for deletion, -target flag can be used

   Syntax: terraform destroy -target aws_instance.Terraform-instance2



```
ubuntu@ip-172-31-19-127:~/terra$ terraform destroy -target aws_instance.Terraform-instance2
aws_instance.Terraform-instance1: Refreshing state... [id=i-089bb03f2ca4481d7]
aws_instance.Terraform-instance2: Refreshing state... [id=i-0a7acfb2fced28651]

Terraform used the selected providers to generate the following execution plan. Resource act
following symbols:
  - destroy

Terraform will perform the following actions:

  # aws_instance.Terraform-instance2 will be destroyed
```