EXPERIMENT-5 (TERRAFORM AND AWS)

Step 1 : Terraform Initializing

```
Microsoft Windows [Version 10.0.18362.1016]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>cd C:\Terraform

C:\Terraform>terraform --version
Terraform v0.13.1
+ provider registry.terraform.io/hashicorp/aws v3.4.0

C:\Terraform>terraform init

Initializing the backend...

Initializing provider plugins...
- Using previously-installed hashicorp/aws v3.4.0

The following providers do not have any version constraints in configuration,
so the latest version was installed.

To prevent automatic upgrades to new major versions that may contain breaking
changes, we recommend adding version constraints in a required_providers block
in your configuration, with the constraint strings suggested below.

* hashicorp/aws: version = "~> 3.4.0"

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.
```

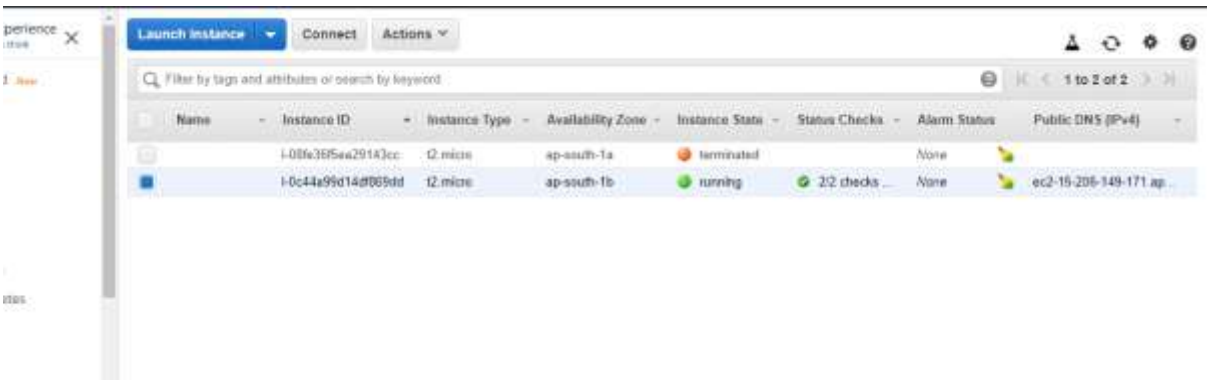
Step 2 : Terraform Planing

```
C:\Terraform>terraform plan
Refreshing Terraform state in-memory prior to plan...
The refreshed state will be used to calculate this plan, but will not be
persisted to local or remote state storage.

aws_instance.example: Refreshing state... [id=i-0c44a99d14df869dd]

------------------------------------------------------------------------

No changes. Infrastructure is up-to-date.

This means that Terraform did not detect any differences between your
configuration and real physical resources that exist. As a result, no
actions need to be performed.
```

Step 3 : Terraform Applying

```
C:\Terraform>terraform apply
aws_instance.example: Refreshing state... [id=i-0c44a99d14df869dd]

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

Instance Running



Step 4 : Terraform Destroying

```
C:\Terraform>terraform destroy
aws_instance.example: Refreshing state... [id=i-0c44a99d14df869dd]

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
    destroy

Terraform will perform the following actions:

  # aws_instance.example will be destroyed
  - resource "aws_instance" "example" {
      - ami                          = "ami-0ebc1ac48dfd14136" -> null
      - arn                          = "arn:aws:ec2:ap-south-1:040003432958:instance/i-0c44a99d14df869dd" -> null
      - 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_termination      = false -> null
      - ebs_optimized                = false -> null
      - get_password_data            = false -> null
      - hibernation                  = false -> null
      - id                           = "i-0c44a99d14df869dd" -> null
      - instance_state               = "running" -> null
      - instance_type                = "t2.micro" -> null
      - ipv6_address_count           = 0 -> null
      - ipv6_addresses               = [] -> null
      - monitoring                   = false -> null
      - primary_network_interface_id = "eni-02c263b502d9ad897" -> null
      - private_dns                  = "ip-172-31-13-199.ap-south-1.compute.internal" -> null
      - private_ip                   = "172.31.13.199" -> null
      - public_dns                   = "ec2-15-206-149-171.ap-south-1.compute.amazonaws.com" -> null
      - public_ip                    = "15.206.149.171" -> null
      - secondary_private_ips        = [] -> null
      - security_groups              = [
          "default",
      ] -> null
      - source_dest_check            = true -> null
      - subnet_id                    = "subnet-accab7e0" -> null
      - tags                         = {} -> null
      - tenancy                      = "default" -> null
      - volume_tags                  = {} -> null
      - vpc_security_group_ids       = [
          "sg-71002615",
```

```
      - credit_specification {
          - cpu_credits = "standard" -> null
        }

      - metadata_options {
          - http_endpoint               = "enabled" -> null
          - http_put_response_hop_limit = 1 -> null
          - http_tokens                 = "optional" -> null
        }

      - root_block_device {
          - delete_on_termination = true -> null
          - device_name           = "/dev/xvda" -> null
          - encrypted             = false -> null
          - iops                  = 100 -> null
          - volume_id             = "vol-0fd00a4d1336f14c2" -> null
          - volume_size           = 8 -> null
          - volume_type           = "gp2" -> null
        }
    }

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.example: Destroying... [id=i-0c44a99d14df869dd]
aws_instance.example: Still destroying... [id=i-0c44a99d14df869dd, 10s elapsed]
aws_instance.example: Still destroying... [id=i-0c44a99d14df869dd, 20s elapsed]
aws_instance.example: Still destroying... [id=i-0c44a99d14df869dd, 30s elapsed]
aws_instance.example: Destruction complete after 32s

Destroy complete! Resources: 1 destroyed.

C:\Terraform>
```