

## Lab Exercise 3–Provisioning an EC2 Instance on AWS

**Prerequisites: Terraform Installed:** Make sure you have Terraform installed on your machine. Follow the official installation guide if needed.

**AWS Credentials:** Ensure you have AWS credentials (Access Key ID and Secret Access Key) configured. You can set them up using the AWS CLI or by setting environment variables.

### Exercise Steps:

#### Step 1: Create a New Directory:

Create a new directory for your Terraform configuration:

```
mkdir aws-terraform-demo  
cd aws-terraform-demo
```

#### Step 2: Create Terraform Configuration File (main.tf):

Create a file named main.tf with the following content:

```
terraform {  
  required_providers {  
    aws = {  
      source = "hashicorp/aws"  
      version = "5.31.0"  
    }  
  }  
}  
  
provider "aws" {  
  region = "ap-south-1"}
```

```

access_key = "your IAM access key"

secret_key = "your secret access key"
}

```

```

main.tf > provider "aws"
1  terraform {
2      required_providers {
3          aws = {
4              source = "hashicorp/aws"
5              version = "5.83.0"
6          }
7      }
8  }
9
10 provider "aws" {
11     region = "us-east-1"
12     access_key = "AKIAQMEY6IA6VI75R3U4"
13     secret_key = "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"
14 }

```

This script defines an AWS provider and provisions an EC2 instance.

### Step 3: Initialize Terraform:

Run the following command to initialize your Terraform working directory:

#### terraform init

```

palakgupta@Palaks-MacBook-Air Terraform % terraform init
Initializing the backend...
Initializing provider plugins...
- Finding hashicorp/aws versions matching "5.83.0"...
- Installing hashicorp/aws v5.83.0...
- Installed hashicorp/aws v5.83.0 (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.
palakgupta@Palaks-MacBook-Air Terraform % pwd
/Users/palakgupta/Documents/sem6/SPCM/Terraform
palakgupta@Palaks-MacBook-Air Terraform %

```

#### Step 4: Create Terraform Configuration File for EC2 instance (instance.tf):

Create a file named instnace.tf with the following content:

```
resource "aws_instance" "My-instance" {  
  
    instance_type = "t2.micro"  
  
    ami = "ami-o3f4878755434977f"  
  
    count = 1  
  
    tags = {  
  
        Name = "UPES-EC2-Instnace"  
  
    }  
  
}
```

```
instance.tf > resource "aws_instance" "my_instance_palak"  
1  resource "aws_instance" "my_instance_palak" {  
2      instance_type = "t2.micro"  
3      ami = "ami-0e2c8caa4b6378d8c"  
4      count = 2  
5      tags = {  
6          Name = "my_instance"  
7      }  
8  }
```

#### Step 5: Review Plan:

Run the following command to see what Terraform will do:

```
terraform plan
```

Review the plan to ensure it aligns with your expectations.

```
[palakgupta@Palaks-MacBook-Air Terraform % terraform validate  
Success! The configuration is valid.  
palakgupta@Palaks-MacBook-Air Terraform %
```

```
+ instance_type                = "t2.micro"
+ ipv6_address_count           = (known after apply)
+ ipv6_addresses               = (known after apply)
+ key_name                     = (known after apply)
+ monitoring                   = (known after apply)
+ outpost_arn                  = (known after apply)
+ password_data                = (known after apply)
+ placement_group              = (known after apply)
+ placement_partition_number    = (known after apply)
+ primary_network_interface_id = (known after apply)
+ private_dns                   = (known after apply)
+ private_ip                   = (known after apply)
+ public_dns                   = (known after apply)
+ public_ip                     = (known after apply)
+ secondary_private_ips         = (known after apply)
+ security_groups               = (known after apply)
+ source_dest_check             = true
+ spot_instance_request_id      = (known after apply)
+ subnet_id                     = (known after apply)
+ tags                         = {
  + "Name" = "my_instance"
}
+ tags_all                     = {
  + "Name" = "my_instance"
}
+ tenancy                      = (known after apply)
+ user_data                    = (known after apply)
+ user_data_base64             = (known after apply)
+ user_data_replace_on_change  = false
+ vpc_security_group_ids       = (known after apply)

+ capacity_reservation_specification (known after apply)

+ cpu_options (known after apply)

+ ebs_block_device (known after apply)

+ enclave_options (known after apply)

+ ephemeral_block_device (known after apply)

+ instance_market_options (known after apply)

+ maintenance_options (known after apply)

+ metadata_options (known after apply)

+ network_interface (known after apply)

+ private_dns_name_options (known after apply)

+ root_block_device (known after apply)
}
```

Plan: 2 to add, 0 to change, 0 to destroy.

## Step 6: Apply Changes:

Apply the changes to create the AWS resources:

```
terraform apply
```

Type yes when prompted.

```

Plan: 2 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

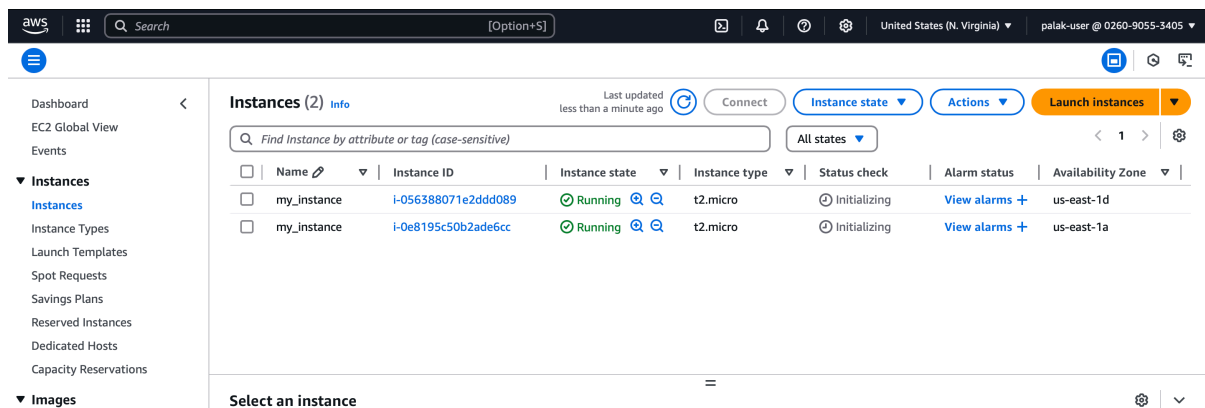
aws_instance.my_instance_palak[1]: Creating...
aws_instance.my_instance_palak[0]: Creating...
aws_instance.my_instance_palak[1]: Still creating... [10s elapsed]
aws_instance.my_instance_palak[0]: Still creating... [10s elapsed]
aws_instance.my_instance_palak[0]: Creation complete after 18s [id=i-056388071e2ddd089]
aws_instance.my_instance_palak[1]: Creation complete after 18s [id=i-0e8195c50b2ade6cc]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
palakgupta@Palaks-MacBook-Air Terraform %

```

## Step 7: Verify Resources:

After the terraform apply command completes, log in to your AWS Management Console and navigate to the EC2 dashboard. Verify that the EC2 instance has been created.



| Name        | Instance ID         | Instance state | Instance type | Status check | Alarm status  | Availability Zone |
|-------------|---------------------|----------------|---------------|--------------|---------------|-------------------|
| my_instance | i-056388071e2ddd089 | Running        | t2.micro      | Initializing | View alarms + | us-east-1d        |
| my_instance | i-0e8195c50b2ade6cc | Running        | t2.micro      | Initializing | View alarms + | us-east-1a        |

## Step 8: Cleanup Resources:

When you are done experimenting, run the following command to destroy the created resources:

```
terraform destroy
```

Type yes when prompted.

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.my_instance_palak[0]: Destroying... [id=i-056388071e2ddd089]
aws_instance.my_instance_palak[1]: Destroying... [id=i-0e8195c50b2ade6cc]
aws_instance.my_instance_palak[0]: Still destroying... [id=i-056388071e2ddd089, 10s elapsed]
aws_instance.my_instance_palak[1]: Still destroying... [id=i-0e8195c50b2ade6cc, 10s elapsed]
aws_instance.my_instance_palak[0]: Still destroying... [id=i-056388071e2ddd089, 20s elapsed]
aws_instance.my_instance_palak[1]: Still destroying... [id=i-0e8195c50b2ade6cc, 20s elapsed]
aws_instance.my_instance_palak[1]: Still destroying... [id=i-0e8195c50b2ade6cc, 30s elapsed]
aws_instance.my_instance_palak[0]: Still destroying... [id=i-056388071e2ddd089, 30s elapsed]
aws_instance.my_instance_palak[0]: Still destroying... [id=i-056388071e2ddd089, 40s elapsed]
aws_instance.my_instance_palak[1]: Still destroying... [id=i-0e8195c50b2ade6cc, 40s elapsed]
aws_instance.my_instance_palak[1]: Still destroying... [id=i-0e8195c50b2ade6cc, 50s elapsed]
aws_instance.my_instance_palak[0]: Still destroying... [id=i-056388071e2ddd089, 50s elapsed]
aws_instance.my_instance_palak[0]: Destruction complete after 54s
aws_instance.my_instance_palak[1]: Still destroying... [id=i-0e8195c50b2ade6cc, 1m0s elapsed]
aws_instance.my_instance_palak[1]: Destruction complete after 1m4s
```

**Destroy complete! Resources: 2 destroyed.**

palakgupta@Palaks-MacBook-Air Terraform %

The screenshot shows the AWS Management Console interface. The left sidebar contains navigation links for Dashboard, EC2 Global View, Events, and Instances. The main content area displays a table of EC2 instances. Two instances are listed, both with a status of 'Terminated'.

| Name        | Instance ID         | Instance state | Instance type | Status check | Alarm status  | Availability Zone |
|-------------|---------------------|----------------|---------------|--------------|---------------|-------------------|
| my_instance | i-056388071e2ddd089 | Terminated     | t2.micro      | -            | View alarms + | us-east-1d        |
| my_instance | i-0e8195c50b2ade6cc | Terminated     | t2.micro      | -            | View alarms + | us-east-1a        |

## Notes:

Customize the instance.tf file to provision different AWS resources.

Explore the Terraform AWS provider documentation for additional AWS resources and configuration options.

Always be cautious when running terraform destroy to avoid accidental resource deletion.

This exercise provides a basic introduction to using Terraform with the AWS provider. Feel free to explore more complex Terraform configurations and resources based on your needs.