Lab 11 – Using a Terraform Registry Module to Create an EC2 Instance on AWS

Creator: Sandeep Kumar Sharma

Learning Objectives

- Understand what Terraform public (registry) modules are.
- Learn how to search, select, and use Terraform-managed modules.
- Learn how to use AWS EC2 module from the official Terraform Registry.
- Deploy an EC2 instance using only a few lines of code by consuming a pre-built module.

Learning Outcome

By the end of this lab, the learner will be able to: - Locate and use Terraform public modules. - Understand how module input variables and outputs work. - Deploy AWS EC2 instances using official registry modules. - Write clean, reusable, production-ready Terraform configurations.

Concept Explanation (Natural Style)

So far, we created our own modules, which is great. But in real-world projects, companies do not always build everything from scratch.

Just like we reuse: - ready-made furniture, - ready-made templates, - ready-made components in software…

Terraform also provides **ready-made modules** created and maintained by experts.

These modules are available on the **Terraform Registry**, which is like the "Play Store" of Terraform infrastructure modules.

You simply search for a module → provide inputs → Terraform creates the entire architecture.

This saves time, avoids mistakes, and follows best practices.

⭐Example: Using the Official AWS EC2 Instance Module

Terraform maintains a high-quality EC2 module:

https://registry.terraform.io/modules/terraform-aws-modules/ec2-instance/aws/latest

This module: - automatically handles AMIs - manages tags - supports VPC, key pair, and networking - follows AWS best practices

And we only need to supply a few values.

⭐Part 1: Create the Project Structure

```
mkdir terraform-lab11-public-module
cd terraform-lab11-public-module
```

Create one file:

```
touch main.tf
```

⭐Part 2: Write Code Using the Official EC2 Module

Open **main.tf** and add:

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 5.0"
    }
  }

  required_version = ">= 1.4.0"
}

provider "aws" {
  region = "ap-south-1"
}

# ---------------------------------------------
# Calling the official EC2 module from registry
# ---------------------------------------------
module "ec2_instance" {
  source  = "terraform-aws-modules/ec2-instance/aws"
  version = "5.5.0"        # use the latest stable version

  name = "terraform-lab11-ec2"

  ami                     = "ami-052c08d70def0ac62"    # Amazon Linux 2 AMI
  instance_type           = "t2.micro"
  monitoring              = true

  # Tagging
  tags = {
    Project     = "Terraform-Lab11"
    Environment = "dev"
  }
}

# --------------------
# Outputs
# --------------------
output "instance_id" {
  value = module.ec2_instance.id
}

output "public_ip" {
  value = module.ec2_instance.public_ip
}
```

⭐Explanation of What We Just Did

🦝We used the official EC2 module:

```
source = "terraform-aws-modules/ec2-instance/aws"
```

This tells Terraform to download the module from the Terraform Registry.

🦝We passed the required inputs:

- name
- AMI
- instance type
- tags

🦝**The module internally creates:**

- EC2 instance
- IAM Role (if needed)
- Security group (optional)
- Metadata handling
- Tags

The best part? You did **not** need to write 30–40 lines of EC2 resource configuration.

⭐Part 3: Initialize Terraform (Download the Module)

```
terraform init
```

Terraform will automatically: - Download the EC2 module - Download AWS provider - Prepare everything to run

---

⭐Part 4: Plan

```
terraform plan
```

You will see Terraform is planning to: - Create an EC2 instance - Apply module settings

---

⭐Part 5: Apply

```
terraform apply
```

Type **yes**.

After a few seconds, Terraform will show:

```
Outputs:
instance_id = "i-12345abcde"
public_ip   = "43.204.x.x"
```

⭐Part 6: Validate in AWS Console

Go to: - **EC2 → Instances**

Look for: - Name = terraform-lab11-ec2 - Tags - Instance Type - Public IP - Status = Running

Everything was created using the **public module**.

⭐Part 7: Destroy (Optional)

```
terraform destroy
```

Type **yes**.

Summary

In this lab, you learned: - What Terraform public (registry) modules are. - How to search and use modules from Terraform Registry. - How to call the official AWS EC2 module. - How to deploy an EC2 instance using just a few lines of code. - How the module automatically manages complex AWS configurations.

In the next labs, we will use more advanced modules like: - VPC module - Security Group module - Autoscaling module - Load balancer module

These will help you build full production architectures.

End of Lab 11