Infrastructure Automation on AWS using Terraform — EC2 Deployment Project

Objective

The objective of the project is to automate the deployment of cloud resources using Terraform, an Infrastructure as Code (IaC) tool.
This project provisions:

- One EC2 instance
- One Security Group allowing SSH access

without manually using the AWS console.

Steps:

Step 1: Configure AWS CLI

Enter:

- AWS Access Key
- Secret
- KeyRegion
- Output format

Step 2: Create one folder for example ***terraform-new.***

Step 3: Create a file called variable.tf which holds configurable values:

```
variable "region" {
  default = "us-east-1"
}

variable "ami_id" {
  default = "ami-0c0------------"
}

variable "instance_type" {
  default = "t3.micro"
}

variable "key_name" {
  default     = "your-existing-key-name"
}
```

This code tells about in which region ec2 is created, ami id means ubuntu or linux id for that region, instance type and key value name.

Step 4: Create a file called main.tf :

```
provider "aws" {
  region = var.region
}

resource "aws_security_group" "allow_ssh" {
  name        = "allow-ssh"
  description = "Allow SSH from anywhere"

ingress {
    description = "SSH"
    from_port   = 22
    to_port     = 22            -------> Inbound Rules
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  egress {
    from_port   = 0
    to_port     = 0            --------> Outbound Rules
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }
}

resource "aws_instance" "my_ec2" {
  ami           = var.ami_id
  instance_type = var.instance_type
  key_name      = var.key_name

  vpc_security_group_ids = [
    aws_security_group.allow_ssh.id
  ]

  tags = {
    Name = "Terraform-EC2"
  }
}
```

This code tells about SSH security .

Step 5：Initialize Terraform

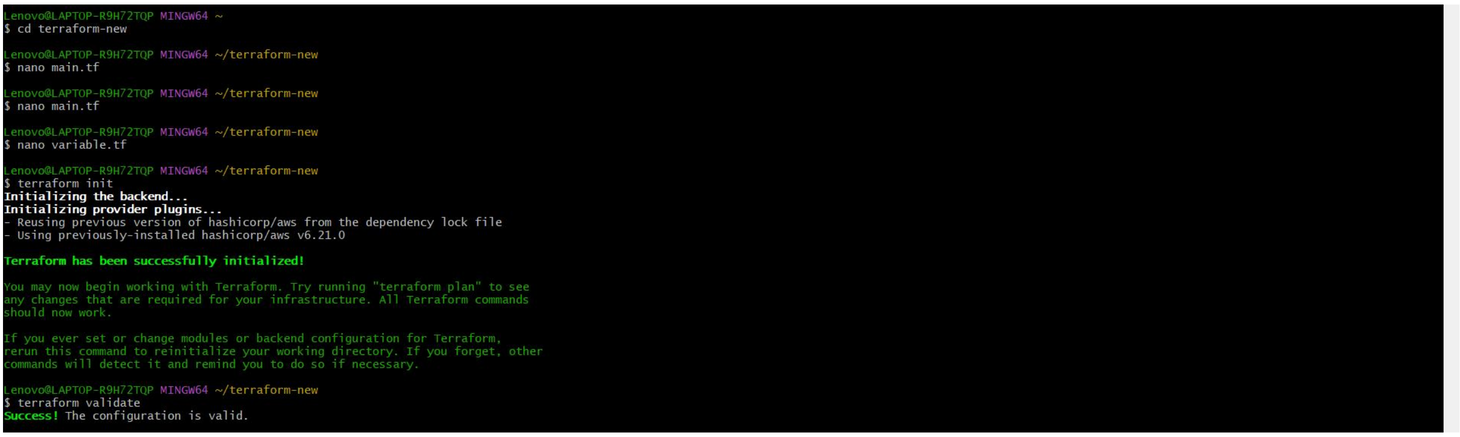
terraform init

Step 6: Validate Configuration

terraform validate

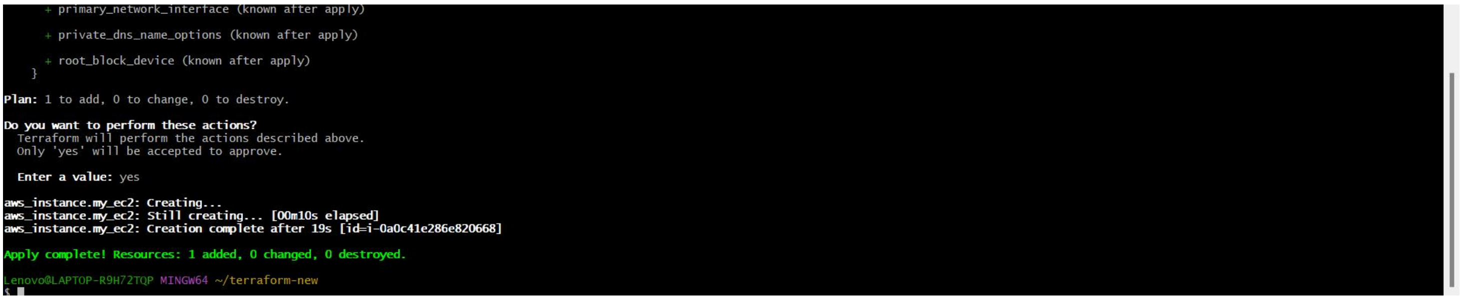
Step 7: Apply cofiguration

terraform apply



```
Lenovo@LAPTOP-R9H72TQP MINGW64 ~
$ cd terraform-new

Lenovo@LAPTOP-R9H72TQP MINGW64 ~/terraform-new
$ nano main.tf

Lenovo@LAPTOP-R9H72TQP MINGW64 ~/terraform-new
$ nano main.tf

Lenovo@LAPTOP-R9H72TQP MINGW64 ~/terraform-new
$ nano variable.tf

Lenovo@LAPTOP-R9H72TQP MINGW64 ~/terraform-new
$ terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v6.21.0

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.

Lenovo@LAPTOP-R9H72TQP MINGW64 ~/terraform-new
$ terraform validate
Success! The configuration is valid.
```

Fig 1: Snapshot of bash interface from step 2 to step 6



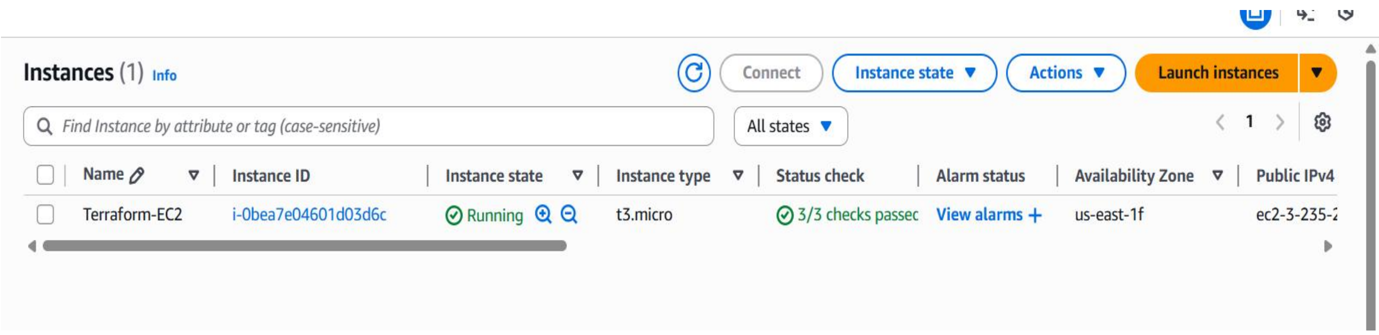
```
      + primary_network_interface (known after apply)

      + private_dns_name_options (known after apply)

      + root_block_device (known after apply)
    }

Plan: 1 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_ec2: Creating...
aws_instance.my_ec2: Still creating... [00m10s elapsed]
aws_instance.my_ec2: Creation complete after 19s [id=i-0a0c41e286e820668]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.

Lenovo@LAPTOP-R9H72TQP MINGW64 ~/terraform-new
$ 
```

Fig 2: Snapshot of the step 7

Step 8: Verify Resource in AWS

AWS Console → EC2 → Instances → instance will be running.

Finally EC2 instance is created using Terraform as shown in Fig 3.



□	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4
□	Terraform-EC2	i-0bea7e04601d03d6c	✓ Running	t3.micro	✓ 3/3 checks passec	View alarms +	us-east-1f	ec2-3-235-2

Fig 3: Snapshot of the created ec2 instance

Conclusion

This project shows that Terraform can easily automate cloud setups. With just a few HCL(HashiCorp Configuration Lang) files, a working EC2 instance and a security group were created. This proves that IaC makes cloud deployment faster, easier, and more reliable in Devops.