Cloud Infotech Solutions Academy

Training / IT Consultant

8688253560==========

Terraform Real-Time Labs (Beginner → Advanced)

Lab 1: Install Terraform

Prerequisites: Linux VM (Ubuntu 20.04/22.04), AWS Free Tier account, IAM user with programmatic access, `awscli` installed.

Steps:

```
# Download Terraform
sudo apt update && sudo apt install -y wget unzip

wget
https://releases.hashicorp.com/terraform/1.9.0/terraform_1.9.0_linux_a
md64.zip
unzip terraform_1.9.0_linux_amd64.zip
sudo mv terraform /usr/local/bin/

# Verify
terraform -version
```

Expected Output:

```
Terraform v1.9.0
```

Lab 2: Configure AWS CLI

Steps:

```
aws configure
```

Enter:

- Access Key ID
- Secret Key
- Default region (`ap-south-1` for Mumbai)
- Output format: `json`

Check:

```
aws s3 ls
```

Lab 3: First Terraform Project (EC2 Instance)

Files: `main.tf`

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

resource "aws_instance" "myec2" {
```


```
  ami           = "ami-0e742cca61fb65051" # Ubuntu 22.04 (Mumbai)
  instance_type = "t2.micro"
  tags = {
    Name = "MyFirstEC2"
  }
}
```

Steps:

```
terraform init
terraform plan
terraform apply -auto-approve
```

Expected Output:

- EC2 instance launched in AWS.
- Console shows `Apply complete! Resources: 1 added.`

Lab 4: Using Variables

Files: `variables.tf`

```
variable "instance_type" {
  default = "t2.micro"
}
```

Update `main.tf`:

```
instance_type = var.instance_type
```

Steps:

```
terraform apply -var "instance_type=t2.small"
```

Output: Instance launches with **t2.small**.

Lab 5: Outputs

Add to `outputs.tf`:

```
output "instance_ip" {
  value = aws_instance.myec2.public_ip
}
```

Steps:

```
terraform apply
```

Expected Output: Shows Public IP of EC2 after apply.

Lab 6: Terraform State

Steps:

```
terraform state list
terraform state show aws_instance.myec2
```

Expected Output: State details of EC2 resource.

Lab 7: Remote Backend (S3)

Files: `backend.tf`

```
terraform {
  backend "s3" {
    bucket = "my-terraform-state-akbar"
    key    = "dev/terraform.tfstate"
    region = "ap-south-1"
  }
}
```

Steps:

```
terraform init
```

Output: State file stored in **S3 bucket**.

Cloud Infotech Solutions Academy

Training / IT Consultant

8688253560==========

Lab 8: Create VPC

Files: `vpc.tf`

```
resource "aws_vpc" "myvpc" {
  cidr_block = "10.0.0.0/16"
  tags = {
    Name = "MyVPC"
  }
}
```

Steps:

```
terraform apply
```

Output: VPC created in AWS.

Lab 9: Subnets + Route Tables

```
resource "aws_subnet" "public_subnet" {
  vpc_id     = aws_vpc.myvpc.id
  cidr_block = "10.0.1.0/24"
  availability_zone = "ap-south-1a"
  tags = { Name = "PublicSubnet" }
}

resource "aws_internet_gateway" "gw" {
  vpc_id = aws_vpc.myvpc.id
```


```
}

resource "aws_route_table" "public_rt" {
  vpc_id = aws_vpc.myvpc.id
  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.gw.id
  }
}

resource "aws_route_table_association" "a" {
  subnet_id      = aws_subnet.public_subnet.id
  route_table_id = aws_route_table.public_rt.id
}
```

Expected Output: Subnet + Internet Gateway + Route Table.

Lab 10: Security Groups (Dynamic Block)

```
variable "ports" {
  default = [22, 80, 443]
}

resource "aws_security_group" "web_sg" {
  vpc_id = aws_vpc.myvpc.id
  dynamic "ingress" {
    for_each = var.ports
    content {
```

```
      from_port   = ingress.value
      to_port     = ingress.value
      protocol    = "tcp"
      cidr_blocks = ["0.0.0.0/0"]
    }
  }
  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }
}
```

Expected Output: Security group allowing SSH, HTTP, HTTPS.

Lab 11: User Data (Install Apache on EC2)

```
resource "aws_instance" "web" {
  ami           = "ami-0e742cca61fb65051"
  instance_type = "t2.micro"
  subnet_id     = aws_subnet.public_subnet.id
  vpc_security_group_ids = [aws_security_group.web_sg.id]

  user_data = <<-EOF
              #!/bin/bash
              apt update -y
              apt install apache2 -y
```

```
              systemctl start apache2
              echo "Hello Terraform" > /var/www/html/index.html
              EOF

  tags = { Name = "WebServer" }
}
```

Expected Output: EC2 running Apache, accessible at Public IP.

Lab 12: Workspaces (Dev, Test, Prod)

```
terraform workspace new dev
terraform workspace new test
terraform workspace new prod

terraform workspace list
```

Output: Different state files for each environment.

Lab 13: RDS Instance

```
resource "aws_db_instance" "mydb" {
  allocated_storage    = 20
  engine               = "mysql"
  engine_version       = "5.7"
  instance_class       = "db.t3.micro"
  name                 = "schooldb"
```


```
  username             = "admin"
  password             = "password123"
  skip_final_snapshot  = true
}
```

Expected Output: MySQL DB in AWS RDS.

Lab 14: Module Usage (EC2 Module)

Files: `modules/ec2/main.tf`

```
resource "aws_instance" "module_ec2" {
  ami           = var.ami
  instance_type = var.instance_type
  tags = { Name = var.name }
}
```

Root main.tf:

```
module "ec2" {
  source        = "./modules/ec2"
  ami           = "ami-0e742cca61fb65051"
  instance_type = "t2.micro"
  name          = "ModuleEC2"
}
```

Expected Output: EC2 created via module.

Cloud Infotech Solutions Academy

Training / IT Consultant

8688253560==========

Lab 15: Destroy Infrastructure

```
terraform destroy -auto-approve
```

Expected Output: All AWS resources deleted.

✅ These 15 labs give a **complete real-world Terraform journey**: installation → EC2 → VPC → security → userdata → RDS → modules → workspaces → destruction.