Lab 15 – Terraform Loops (count, for_each, and dynamic blocks)

Creator: Sandeep Kumar Sharma

Learning Objectives

- Understand why loops are needed in Terraform.
- Learn the difference between **count** and **for_each** loops.
- Learn when to use each type of loop.
- Learn how to create multiple AWS resources (EC2, S3, SG rules) dynamically.
- Understand dynamic blocks for repeated nested configurations.

Learning Outcome

By the end of this lab, the learner will be able to: - Use `count` to create multiple identical resources. - Use `for_each` to create map- or list-based resources. - Use dynamic blocks for repeated sub-configurations. - Build scalable Terraform infrastructure using loops.

Concept Explanation (Natural Style)

Imagine you are ordering 5 laptops for your team. Instead of filling 5 separate order forms, you simply say:

> "Give me 5 laptops of the same model."

This is exactly what Terraform loops do.

Terraform loops help you create: - multiple EC2 instances - multiple subnets - multiple S3 buckets - repeated firewall rules

...without duplicating your code.

⭐Types of Loops in Terraform

🐓count (best for identical resources)

- Simple
- Uses index numbers

- Great for creating *N number* of similar resources

🐕 for_each (best for named resources)

- Works with maps and sets
- Lets you uniquely identify each resource
- Perfect for tagging, subnet creation, etc.

🐗dynamic blocks (for nested repeated blocks)

- Used when a resource needs repeating rules under a single block (e.g., multiple ingress rules)

⭐Part 1 – Setup

```
mkdir terraform-lab15-loops
cd terraform-lab15-loops
```

Create file:

```
touch main.tf
```

⭐Part 2 – Using `count` to Create Multiple EC2 Instances

Add this to **main.tf**:

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

variable "instance_count" {
  type    = number
  default = 3
}

resource "aws_instance" "loop_ec2" {
  count         = var.instance_count
  ami           = "ami-052c08d70def0ac62"
  instance_type = "t2.micro"
  tags = {
    Name = "lab15-ec2-${count.index}"
  }
}
```

🦝Explanation

- `count = 3` → Terraform creates 3 EC2 instances.
- `count.index` gives 0, 1, 2 → used in naming.

⭐Part 3 – Using `for_each` to Create Named S3 Buckets

Append this:

```
variable "bucket_names" {
  type = set(string)
  default = [
    "lab15-bucket-a",
    "lab15-bucket-b",
    "lab15-bucket-c"
  ]
}

resource "aws_s3_bucket" "loop_buckets" {
  for_each = var.bucket_names

  bucket = each.value
}
```

🦝Explanation

- `for_each` loops through the list of bucket names
- Creates a bucket for each name

This is better than `count` because each item has a unique identity.

⭐Part 4 – Using a Dynamic Block (for Nested Loops)

Now create a security group with multiple ingress rules using loops.

```
resource "aws_security_group" "lab15_sg" {
  name   = "lab15-security-group"
  vpc_id = null

  dynamic "ingress" {
    for_each = [22, 80, 443]
    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"]
  }
}
```

🦝Explanation

- Dynamic block loops through ports list
- Creates one ingress rule each for ports:
- 22 (SSH)
- 80 (HTTP)
- 443 (HTTPS)

This avoids copy-pasting rule blocks.

⭐Part 5 – Initialize Terraform

```
terraform init
```

⭐Part 6 – Run Plan

```
terraform plan
```

Terraform will show: - 3 EC2 instances - 3 S3 buckets - 3 ingress rules in the SG

⭐Part 7 – Apply

```
terraform apply
```

Type **yes**.

AWS resources will be created automatically using loops.

⭐Part 8 – Validate in AWS Console

🦝EC2

Check 3 instances: - lab15-ec2-0 - lab15-ec2-1 - lab15-ec2-2

🦝S3

Check buckets: - lab15-bucket-a - lab15-bucket-b - lab15-bucket-c

🦝Security Group

Check inbound rules: - SSH (22) - HTTP (80) - HTTPS (443)

All generated using loops.

⭐Part 9 – Destroy (Optional)

```
terraform destroy
```

Type **yes**.

Summary

In this lab, you learned: - The difference between `count` and `for_each`. - How to create multiple EC2 instances with a loop. - How to create multiple S3 buckets using `for_each`. - How to use dynamic blocks for repeated nested structures.

Loops make Terraform: - scalable - flexible - clean - production-ready

In the next lab, we will learn **Terraform Workspaces** to manage multiple environments (dev, qa, prod).

End of Lab 15