Lab 14 – Terraform Conditionals & Expressions (Make Your Code Smart)

Creator: Sandeep Kumar Sharma

Learning Objectives

- Understand what Terraform **conditionals** are.
- Learn how to make Terraform resources behave differently based on conditions.
- Learn how to use `count` and `ternary` operators.
- Learn how to enable/disable resources using conditions.
- Deploy EC2 only when a condition is true.

Learning Outcome

By the end of this lab, the learner will be able to: - Write condition-based Terraform code. - Use conditional expressions to choose values dynamically. - Enable/disable resources using `count`. - Build smarter Terraform configurations that adapt to environment needs.

Concept Explanation (Natural Style)

Let’s understand Terraform conditionals in a simple way.

Imagine you're organizing an event. - If the event is **online**, you book Zoom. - If the event is **offline**, you book a conference hall.

Same logic → different result.

Terraform conditionals work exactly like this. Terraform lets you write **smart code** so that: - resources are created only when needed, - values change based on environment, - naming changes automatically.

This makes Terraform flexible and avoids duplication.

⭐Core Terraform Conditionals

🦝Ternary Expression

Terraform's conditional syntax:

```
condition ? value_if_true : value_if_false
```

Example:

```
var.env == "prod" ? "t3.large" : "t2.micro"
```

🦝 `count` (Enable/Disable Resources)

```
count = var.enable_instance ? 1 : 0
```

If condition is false → resource is not created.

⭐Part 1: Project Setup

```
mkdir terraform-lab14-conditionals
cd terraform-lab14-conditionals
```

Create files:

```
touch main.tf
```

⭐Part 2: Write Terraform Code (Using Conditionals)

We will: - Declare an environment variable (dev or prod) - Decide instance type based on environment - Create EC2 only when user says enable=true

---

main.tf

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

# --------------------------------
# Declare Variables
# --------------------------------
variable "environment" {
  type        = string
  description = "Environment name (dev/prod)"
  default     = "dev"
}

variable "enable_ec2" {
  type        = bool
  description = "Whether EC2 should be created"
  default     = true
}

# --------------------------------
# Conditional Instance Type
# --------------------------------
locals {
  instance_type = var.environment == "prod" ? "t3.medium" : "t2.micro"
}

# --------------------------------
# Conditional EC2 Resource
# --------------------------------
resource "aws_instance" "lab14_ec2" {
  count         = var.enable_ec2 ? 1 : 0
  ami           = "ami-052c08d70def0ac62"
  instance_type = local.instance_type

  tags = {
    Name        = "Terraform-Lab14-EC2"
    Environment = var.environment
  }
}

# Outputs
output "created" {
  value = var.enable_ec2 ? "EC2 was created" : "EC2 was NOT created"
}
```

```
output "instance_type" {
  value = local.instance_type
}
```

⭐Explanation of What We Did

🦝 `var.environment` **decides instance type**

- If **prod** → EC2 gets **t3.medium**
- If **dev** → EC2 gets **t2.micro**

🦝 `var.enable_ec2` **decides if EC2 should be created**

```
count = var.enable_ec2 ? 1 : 0
```

If false → Terraform creates 0 EC2.

🦝 `locals` **used for dynamic value**

Instance type is chosen based on environment.

⭐Part 3: Initialize Terraform

```
terraform init
```

⭐Part 4: Plan (Default Values)

```
terraform plan
```

Terraform will: - Create 1 EC2 - Use `t2.micro`

⭐Part 5: Apply

```
terraform apply
```

Type **yes**.

Outputs:

```
created = "EC2 was created"
instance_type = "t2.micro"
```

⭐Part 6: Try Production Mode

Edit the variable at plan time:

```
terraform plan -var="environment=prod"
```

Instance type becomes **t3.medium**.

⭐Part 7: Disable EC2 Completely

```
terraform plan -var="enable_ec2=false"
```

EC2 count becomes **0**.

This is extremely useful when: - toggling optional components - using dev vs prod - turning features ON/OFF

⭐Part 8: Destroy (Optional)

```
terraform destroy
```

Type **yes**.

Summary

In this lab, you learned: - What Terraform conditional expressions are. - How to use ternary operators to choose values. - How to use `count` to enable/disable resources. - How to make Terraform dynamic, flexible, and environment-aware.

In the next lab, we will learn **Terraform loops (for-each and count)** to create multiple resources dynamically.

End of Lab 14