Lab 8 – Terraform Locals (Clean, Simple, Beginner-Friendly)

Creator: Sandeep Kumar Sharma

Learning Objectives

- Understand what **locals** are in Terraform.
- Learn how locals help simplify and clean your Terraform configuration.
- Learn how to declare locals inside Terraform.
- Learn how to reuse local values across multiple resources.
- Deploy AWS resources using locals to manage repeated values.

Learning Outcome

By the end of this lab, the learner will be able to: - Explain the purpose and use of Terraform locals. - Create and use local values in a real Terraform project. - Reduce code duplication using locals. - Write cleaner, more readable Terraform configurations.

Concept Explanation (Natural Style)

Let's understand locals with a simple real-life example.

Imagine you are filling out a form where you must write your **full name** in 8 different places — header, signature, declaration, etc. Instead of writing it again and again, what if you could simply store your name once and reuse it everywhere?

That is exactly what Terraform **locals** do.

👉What are locals?

Locals are **internal constants** or **temporary values** that you define once and reuse many times in your configuration.

👉Why locals?

Locals help you avoid: - repeating the same strings multiple times - writing long, complex expressions again and again - mistakes caused by typos - unnecessary variables

Locals make your code: - clean - readable - maintainable - production friendly

👉Variables vs Locals

Feature	Variables	Locals
Purpose	Accept values from outside	Store reusable internal values
Changeable?	Yes, user can modify	No, fixed inside code
Use case	Different environments	Naming conventions, tags, common values

So variables are **inputs**, but locals are **internal helpers**.

😡Part 1: Project Setup

```
mkdir terraform-lab8-locals
cd terraform-lab8-locals
```

Create one file:

```
touch main.tf
```

😡Part 2: Write Terraform Code Using Locals

Open **main.tf** and add:

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

# 👇 Declare locals
locals {
  project_name = "training-project"
  environment  = "dev"
  instance_tag = "Terraform-Lab8-EC2"
}

# 👇 Create EC2 instance using locals
resource "aws_instance" "lab8_ec2" {
  ami           = "ami-052c08d70def0ac62"
  instance_type = "t2.micro"

  tags = {
    Name        = local.instance_tag
    Project     = local.project_name
    Environment = local.environment
  }
}
```

Explanation

👉🏽 `locals` block

We created three local values: - project_name - environment - instance_tag

These are reused inside the EC2 resource.

👉🏽Why this is useful?

Because if tomorrow you want to change: - project name - environment - naming convention

You only change **one line**.

Without locals, you would have changed it in many places.

😠Part 3: Run Terraform

Initialize

```
terraform init
```

Plan

```
terraform plan
```

You will see the EC2 instance configuration using locals.

Apply

```
terraform apply
```

Type **yes**.

Terraform will create an EC2 instance with tags coming from locals.

😠Part 4: Validate in AWS Console

Go to: - EC2 → Instances → Terraform-Lab8-EC2

Check the **Tags** section: - Name - Project - Environment

All values came from **locals**, not hardcoded.

😠Part 5: Destroy (Optional)

```
terraform destroy
```

Type **yes**.

Summary

In this lab, you learned: - What locals are in Terraform. - How locals help reduce duplication. - How to use locals for naming, tags, and reusable values. - How locals improve the readability and maintainability of Terraform code.

Locals are extremely useful when we move into **modules** (next labs). They help centralize formulas, naming patterns, and logic.

End of Lab 8