Lab: Introduction to the Terraform Locals Block

Locals blocks (often referred to as locals) are defined values in Terraform that are used to reduce repetitive references to expressions or values. Locals are very similar to traditional input variables and can be referred to throughout your Terraform configuration. Locals are often used to give a name to the result of an expression to simplify your code and make it easier to read.

Locals are not set directly by the user/machine executing the Terraform configuration, and the values don't change between or during the Terraform workflow (`init`, `plan`, `apply`).

Locals are defined in a `locals` block (plural) and include named local variables with their defined values. Each locals block can contain one or more local variables. Locals are then referenced in your configuration using interpolation using `local.<name>` (note `local` and not `locals`). The syntax of a locals block is as follows:

Template

```
locals {
  # Block body
  local_variable_name = <EXPRESSION OR VALUE>
  local_variable_name = <EXPRESSION OR VALUE>
}
```

Example

```
locals {
  time = timestamp()
  application = "api_server"
  server_name = "${var.account}-${local.application}"
```

- Task 1: Define the Name of an EC2 Instance using a Local Variable

Task 1: Define the Name of an EC2 Instance using a Local Variable

Before we can use a locals variable, it needs to be defined in our configuration file. Locals are often defined right in the configuration file where they will be used. In the `main.tf` file, add the following locals block to the configuration file:

```
locals {
  team = "api_mgmt_dev"
  application = "corp_api"
  server_name = "ec2-${var.environment}-api-${var.variables_sub_az}"
}
```

Next, let's update the configuration of our web server resource to use the locals variable to define the name. Modify the web_server resource in `main.tf` so it matches the following:

```
resource "aws_instance" "web_server" {
  ami           = data.aws_ami.ubuntu.id
  instance_type = "t2.micro"
  subnet_id     = aws_subnet.public_subnets["public_subnet_1"].id
  tags = {
    Name = local.server_name
    Owner = local.team
    App = local.application
  }
}
```

Task 1.1

Run a `terraform plan` to see the changes to our web server. You should see that the EC2 instance will not be destroyed, but the name of the server (tag) will be updated to reflect the value of our local variables.

```
terraform plan
```

```
<add output here>
```

Task 1.2

Let's go ahead and commit our code to validate Terraform makes the desired changes to the AWS infrastructure. Run a `terraform apply` to apply the changes. Log into AWS and validate the changes were applied.

Once validated, feel free to remove the `locals` block and tags on the EC2 instance and then apply the configuration to revert back to the original values. However, you can leave everything in your Terraform configuration if you'd like.