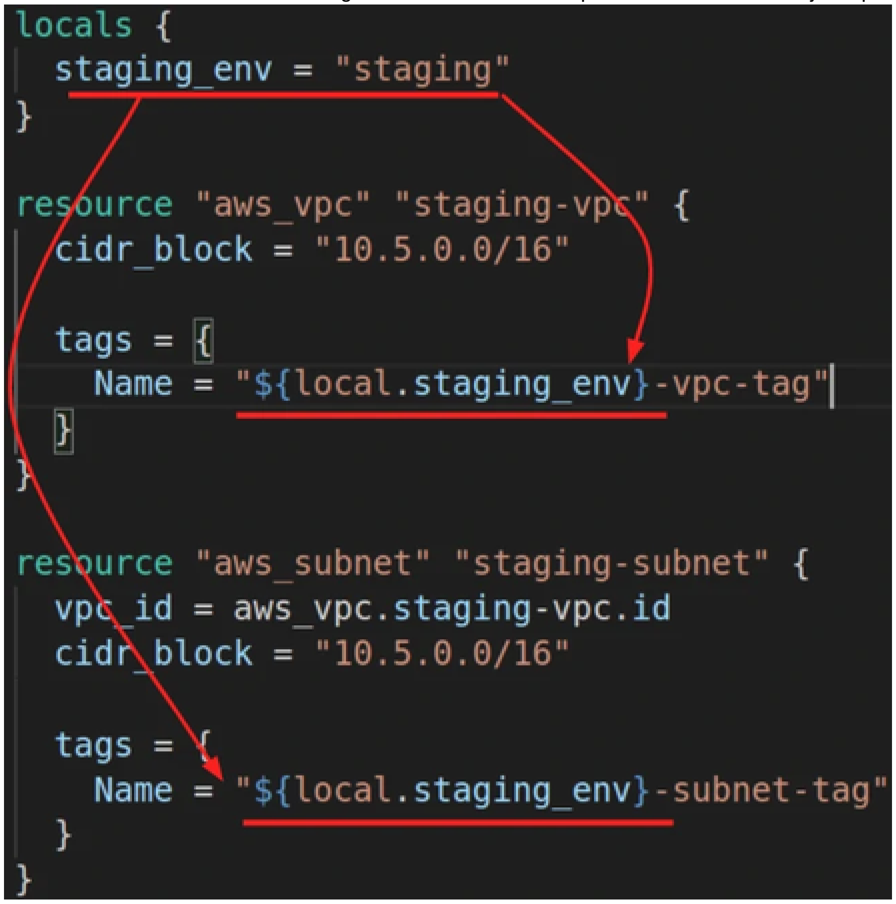
Terraform locals

Table of Content

Terraform locals are quite similar to Terraform variables but Terraform locals do not change their value. On the other hand, if you talk about Terraform input variables then it is dependent on user input and it can change its value.

If you have a very large Terraform file where you need to use the same values or expressions multiple times then Terraform local can be useful for you.

If you look at the following diagram, here I am trying to create a terraform local in which I am going to Define The server name so in the future if there is a change in the server name so I do not need to change the server name at all places but instead I can just update my locals.



```
locals {
  staging_env = "staging"
}

resource "aws_vpc" "staging-vpc" {
  cidr_block = "10.5.0.0/16"

  tags = {
    Name = "${local.staging_env}-vpc-tag"
  }
}

resource "aws_subnet" "staging-subnet" {
  vpc_id = aws_vpc.staging-vpc.id
  cidr_block = "10.5.0.0/16"

  tags = {
    Name = "${local.staging_env}-subnet-tag"
  }
}
```

1. Create your first Terraform Local

Let's create our first Terraform local and in this terraform local we are going to Define the server name we're going to deploy or where we are going to apply our terraform configuration.

```
locals {
  staging_env = "staging"
}
```

As you can see in the above syntax we have created a local environment. And we're going to use the same local throughout our Terraform configuration

Here is my Terraform configuration for my AWS environment

```
provider "aws" {
   region     = "eu-central-1"
   access_key = "AKIATQ37NXB2G2LXXXXX"
   secret_key = "r1oaShokKPw+YY7qaHxj8mD2T8BpxRUVXXXXXXXX"
}

locals {
  staging_env = "staging"
}

resource "aws_vpc" "staging-vpc" {
  cidr_block = "10.5.0.0/16"

  tags = {
    Name = "${local.staging_env}-vpc-tag"
  }
}

resource "aws_subnet" "staging-subnet" {
  vpc_id = aws_vpc.staging-vpc.id
  cidr_block = "10.5.0.0/16"

  tags = {
    Name = "${local.staging_env}-subnet-tag"
  }
}

resource "aws_instance" "ec2_example" {

   ami           = "ami-0767046d1677be5a0"
   instance_type = "t2.micro"
   subnet_id = aws_subnet.staging-subnet.id

   tags = {
           Name = "${local.staging_env} - Terraform EC2"
   }
}
```

2. Combine terraform local with terraform variable

now we know how to use terraform local the next thing which we are going to try is to combine terraform local along with Terraform variable.

First, let’s create a few terraform variables -

```
variable location {
  description = "Location of server"
  type        = string
  default     = "finland"
}
```

```
variable server_name {
  description = "Name of server"
  type        = string
  default     = "primary-app-server"
}
```

Now let's create a terraform local, in which we are going to combine the two variables which we have created just above

```
locals {
    server_details = "${var.location}-${var.server_name}"
}
```

And now we can use this local inside our terraform configuration where we are going to define the server detail.

Best practices for using locals

1. You should use terraform locals excessively inside your Terraform configuration
2. Always keep in mind to use terraform local where you think that value is going to be changed in the future.
3. Always think of Terraform Locals as a central place for storing configuration values.