How to use Terraform locals?

In programming world we have concept of **variables**, similar to that in **Terraform** we also have concept of **Terraform Locals**.

**What is Terraform locals?**

Here is an very basic example code which show - *"How to create*[*locals*](https://www.terraform.io/)*in Terraform?"*

*## Example of local with static value*

locals {

my\_local = "value"

}

As you can see in the above code locals are very easy to define. In the later section of this blog we will take a look on *dynamic way* to assign the value.

**Here are the key properties of locals -**

1. **Static Value -** Terraform local can retain static value
2. **Dynamic Value -** You can also assign dynamic value to Terraform local.

*## Example of local with dynamic value*

locals {

my\_local = "${var.my\_variable\_value}"

}

1. Terraform Locals in a nutshell

Let's talk about the scope and visibility. Terraform Locals are only accessible within that functions or within the scope of terraform file. Here are few more points which you should take into consideration for the value assignment -

**1. Value assignment-** Terraform locals do not change their value once it is assigned, you have to re-assign a new value.

**2. Power of Expression-** Apart from static value assignment terraform locals can use the power of [expression](https://www.terraform.io/language/expressions), so instead of writing the same expression the multiple times through the terraform configuration, you can declare a *terraform locals* and use the same *terraform locals* at other places.

Example with expression -

locals {

my\_local = "${var.ec2\_instance\_name}"

}

*BASH*

Example without expression -

locals {

my\_local = "t2.micro"

}

*BASH*

2. Benefits of using Terraform Locals

If you are working on a large enterprise infrastructure then it is impossible to work without Terraform Locals. But let's see the benefits of using terraform locals in a bit more detail -

1. Terraform locals can be re-used multiple numbers of times in the terraform configuration file.
2. It can reduce the work of updating your terraform configuration at multiple places. With terraform locals you need to update its value once and it should reflect all over the place where it is referred.

3. Create your first Terraform Local on AWS

Let's create our first Terraform local and in this terraform local we will store the [Tag name](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/Using_Tags.html) for the AWS EC2 instance.

locals {

staging\_env = "staging"

}

*BASH*

As you can see in the above syntax we have created a tag name *Staging* for the EC2 instance. And we're going to use the same local throughout our Terraform configuration.

We use the terraform locals for -

1. Putting a tag on aws\_subnet
2. Putting a tag on aws\_instance for ec2

Here is my Terraform configuration for my AWS environment

provider "aws" {

region = "ap-south-1"

access\_key = "<YOUR\_AWS\_ACCESS\_KEY>"

secret\_key = "<YOUR\_AWS\_SECRET\_KEY>"

}

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-0cca134ec43cf708f"

instance\_type = "t2.micro"

subnet\_id = aws\_subnet.staging-subnet.id

tags = {

Name = "${local.staging\_env} - Terraform EC2"

}

}

*BASH*

4. 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 = "ap-south-1"

}

*BASH*

variable server\_name {

description = "Name of server"

type = string

default = "primary-app-server"

}

*BASH*

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}"

}

*BASH*

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

5. Reference locals within locals

On a large Terraform project you might end up in a situation where you need to create multiple Terraform locals and even you might need to reference locals with other locals.

So terraform does provide support to *reference Locals within Locals*. Here is an example code-

*# Here I have first created my\_first\_local*

locals {

my\_first\_local = "value-1"

}

*# Here my\_second\_local is referencing to my\_first\_local*

locals {

my\_second\_local = "value-2-${local.my\_first\_local}"

}

*BASH*

**Note-** You need to make sure to prefix *local* before referencing to other local. Refer to [this thread for more details](https://stackoverflow.com/questions/62537713/combine-terraform-locals-variables)

6. Best practices for using locals

1. You should never 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.
4. Try not too hard to code locals inside terraform configuration file instead pick the value of terraform locals from the .tfvars so that you have a single location to update in the future.