How to use Terraform Variables - Locals,Input,Output

[Terraform variables](https://www.blogger.com/blog/post/edit/3356037005243296072/8265609069474122517) are a way to store values that can be reused throughout your Terraform configuration.

They allow you to define a value once and reference it in multiple places throughout your configuration, making it easier to manage and update your infrastructure.

1.    Variables are defined in the variables block in your Terraform configuration file, where you can give a name and a default value. Please refer to the following screenshot exaplaining how variables are defined inside terraform-

terraform variables string, bool, number, list, set, map

2.    Terraform variables can have various type such as ***string, number, boolean, list, map*** etc.

3.    Variables can be set in the command line when running Terraform commands using the -var flag.

4.    Variables can also be set using a separate file, called a variable file, using the -var-file flag.

5.    Variables can be accessed in Terraform configuration files using the var function, for example var.example\_variable

6.    Variables are useful for storing values that may change between environments, for example, different values for test and production environments.

Pre-requisite

Before we start working with Terraform variables, here are the pre-requisites -

1.    You must install terraform *(*[*click here on how to install terraform*](https://www.blogger.com/blog/post/edit/3356037005243296072/8265609069474122517)*)*

2.    You must have either [AWS](https://www.blogger.com/blog/post/edit/3356037005243296072/8265609069474122517) Account.

1. Types of Terraform Variables

There are two types of variables in Terraform -

1.    **Simple values**

2.    **Collection Variable**

1.1 Simple Values variables

As the name suggests *Simple Values* variables are which hold only a single value. Here the types of *Simple Value* variables -

1.    string

2.    number

3.    bool

1.2 Collection Variable

In the collection variable, it consists of -

1.    List

2.    Map

3.    Set

2.Terraform Variables - *string, number, bool*

Let's take a simple example in which we are going to set up an EC2 instance on AWS.

So to create an EC2 instance we need two things -

1.    provider

2.    resource

Here is the main.tf which we are going to parameterized using terraform variables.

provider "aws" {

   region     = "ap-south-1"

   access\_key = "<INSERT\_YOUR\_ACCESS\_KEY>"

   secret\_key = "<INSERT\_YOUR\_SECRET\_KEY>"

}

resource "aws\_instance" "ec2\_example" {

   ami           = "ami-0cca134ec43cf708f"

"

   instance\_type = "t2.micro"

   tags = {

           Name = "Terraform EC2"

   }

}

2.1 string variable type - We are going parameterized *instance\_type = "t2.micro"*

The first rule to create a parameter in terraform file is by defining *variable block*

Example -

variable "instance\_type" {

   description = "Instance type t2.micro"

   type        = string

   default     = "t2.micro"

}

*BASH*

**For defining***variable block***you need**

1.    *description* : Small or short description about the purpose of the variable

2.    *type* : What type of variable it is going to be ex - string, bool, number ...

3.    *default* : What would be the default value of the variable

**Let's replace the hardcoded value of instance\_type with variable**

 instance\_type = var.instance\_type

Here is our final terraform file after replacing the hardcoded value of a variable -

provider "aws" {

   region     = "ap-south-1"

   access\_key = "<INSERT\_YOUR\_ACCESS\_KEY>"

   secret\_key = "<INSERT\_YOUR\_SECRET\_KEY>"

}

resource "aws\_instance" "ec2\_example" {

   ami           = "ami-0cca134ec43cf708f"

   instance\_type =  var.instance\_type

   tags = {

           Name = "Terraform EC2"

   }

}

variable "instance\_type" {

   description = "Instance type t2.micro"

   type        = string

   default     = "t2.micro"

}

And now you can apply your terraform configuration

terraform apply

2.2 number variable type - We are going parameterized *instance\_count = 2*

The next variable type we are going to take is number.

For example, we are going to increase the instance\_count of the ec2\_instances.

Let's create the variable first -

variable "instance\_count" {

  description = "EC2 instance count"

  type        = number

  default     = 2

}

*BASH*

Here is the final terraform file with instance count -

provider "aws" {

   region     = "ap-south-1"

   access\_key = "<INSERT\_YOUR\_ACCESS\_KEY>"

   secret\_key = "<INSERT\_YOUR\_SECRET\_KEY>"

}

resource "aws\_instance" "ec2\_example" {

   ami           = "ami-0cca134ec43cf708f"

   instance\_type =  "t2.micro"

   count = var.instance\_count

   tags = {

           Name = "Terraform EC2"

   }

}

variable "instance\_count" {

  description = "EC2 instance count"

  type        = number

  default     = 2

}

*BASH*

2.3 boolean variable type - We are going parameterized *enable\_vpn\_gateway = false*

The next variable type which we are going to discuss is bool.

The bool variable can be used to set true or false values inside your terraform file.

Here is an example to create your bool variable -

variable "enable\_public\_ip" {

  description = "Enable public IP address"

  type        = bool

  default     = true

}

*BASH*

Let's create a complete terraform file with bool variable -

provider "aws" {

   region     = "ap-south-1"

   access\_key = "<INSERT\_YOUR\_ACCESS\_KEY>"

   secret\_key = "<INSERT\_YOUR\_SECRET\_KEY>"

}

resource "aws\_instance" "ec2\_example" {

   ami           = "ami-0cca134ec43cf708f"

   instance\_type =  "t2.micro"

   count = 1

  associate\_public\_ip\_address = var.enable\_public\_ip

   tags = {

           Name = "Terraform EC2"

   }

}

variable "enable\_public\_ip" {

  description = "Enable public IP address"

  type        = bool

  default     = true

}

3. Terraform Variables - *list, set, map*

When it comes to collection input variables then we are talking about -

1.    List

2.    Map

3.    Set

3.1 List variable type

As the name suggests we are going to define a list that will contain more than one element in it.

Let's define our first List variable -

**Here is the list of IAM users**

variable "user\_names" {

  description = "IAM usernames"

  type        = list(string)

  default     = ["user1", "user2", "user3"]

}

Here is our final terraform file with List variables -

provider "aws" {

   region     = "ap-south-1"

   access\_key = "<INSERT\_YOUR\_ACCESS\_KEY>"

   secret\_key = "<INSERT\_YOUR\_SECRET\_KEY>"

}

resource "aws\_instance" "ec2\_example" {

   ami           = "ami-0cca134ec43cf708f"

   instance\_type =  "t2.micro"

   count = 1

   tags = {

           Name = "Terraform EC2"

   }

}

resource "aws\_iam\_user" "example" {

  count = length(var.user\_names)

  name  = var.user\_names[count.index]

}

variable "user\_names" {

  description = "IAM usernames"

  type        = list(string)

  default     = ["user1", "user2", "user3s"]

}

*BASH*

3.2 Map variable type

Terraform also supports the map variable type where you can define the key-valye pair.

Let's take an example where we need to define project and environment, so we can use the map variable to achieve that.

Here is an example of map variable -

variable "project\_environment" {

  description = "project name and environment"

  type        = map(string)

  default     = {

    project     = "project-cloud",

    environment = "dev"

  }

}

*BASH*

Let's create a Terraform file

provider "aws" {

   region     = "ap-south-1"

   access\_key = "<INSERT\_YOUR\_ACCESS\_KEY>"

   secret\_key = "<INSERT\_YOUR\_SECRET\_KEY>"

}

resource "aws\_instance" "ec2\_example" {

   ami           = "ami-0cca134ec43cf708f"

   instance\_type =  "t2.micro"

   tags = var.project\_environment

}

variable "project\_environment" {

  description = "project name and environment"

  type        = map(string)

  default     = {

    project     = "project-Cloud",

    environment = "dev"

  }

}