

Terraform variables

Variables in [Terraform](#) are a great way to define centrally controlled reusable values. The information in Terraform variables is saved independently from the deployment plans, which makes the values easy to read and edit from a single file.

We have 3 types of variables:

1. **Input variables:** Sends values to Terraform.
2. **Output variables:** Retrieves values from Terraform.
3. **Local variables:** Assigns names to an expression.

Input variables

Input variables are usually defined by stating a name, type and default value. However, the type and default values are not strictly necessary.

We use input variables to pass certain values from outside of the configuration or module. Variables can be predetermined in a file or included in the command-line options.

The variable declaration can optionally include three arguments:

- description: briefly explain the purpose of the variable and what kind of value is expected.
- type: specifies the type of value such as string, number, bool, map, list, etc.
- default: If present, the variable is considered to be optional and if no value is set, the default value is used.

Variable example =>

```
Variable "your_variable_name" {  
    Description = "define instance type"  
    Type = "string"  
    Default = "t2.micro"  
}
```

variable name

description |

ex: string, list, map, number..

variable default value..

Input variables **support multiple data types**. They are broadly categorized as simple and complex. `String`, `number`, `bool` are simple data types, whereas `list` and `map` are complex data types.

Here are some examples of how each type are defined and used.

Primitive Types

A primitive type is a simple type that isn't made from any other type. The available primitive types are:

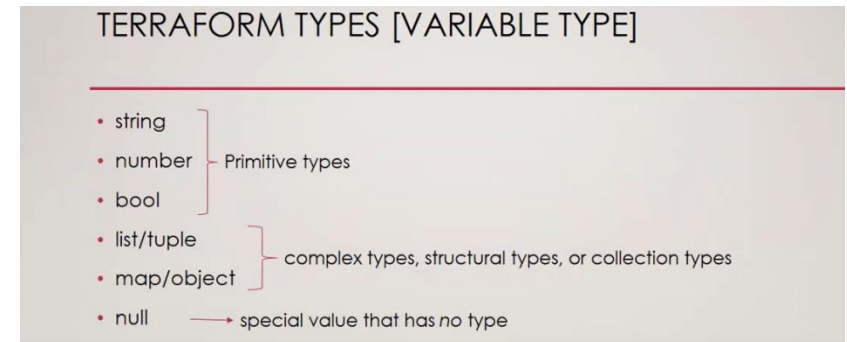
1. **string**: a sequence of characters representing some text, such as "hello".
2. **number**: a numeric value. The number type can represent both whole numbers like 15 and fractional values such as 6.28318.
3. **bool**: either true or false..

Complex Types

A *complex* type is a type that groups multiple values into a single value. These values could be of a similar type or different types.

List:

Another type of Terraform variables lists. They work much like a numbered catalogue of values. Each value can be called by its corresponding index in the list. Here is an example of a list variable definition.



```
variable "users" {  
  type    = list  
  default = ["root", "user1", "user2"]  
}
```

Lists can be used in the resource plans similarly to strings, but you'll also need to denote the index of the value you are looking for.

```
username = var.users[0]
```

Map:

Maps are a collection of string keys and string values. These can be useful for selecting values based on predefined parameters such as the server configuration by the monthly price.

```
variable "plans" {  
  type = map  
  default = {  
    "5USD" = "1xCPU-1GB"  
    "10USD" = "1xCPU-2GB"  
    "20USD" = "2xCPU-4GB"  
  }  
}
```

You can access the right value by using the matching key. For example, the variable below would set the plan to "1xCPU-1GB".

```
plan = var.plans["5USD"]
```

Assign Values To Input Variables

There are multiple ways to assign values to variables. The following is the descending order of precedence in which variables are considered.

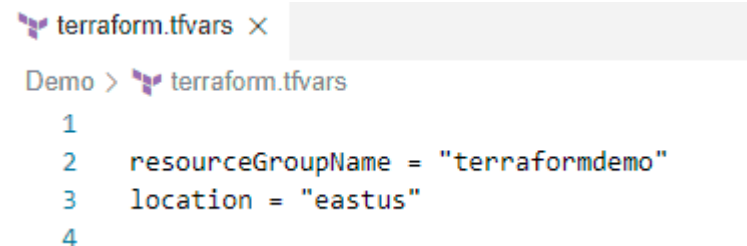
1. Command-line flags

The most simple way to assign value to a variable is using the `-var` option in the command line when running the `terraform plan` and `terraform apply` commands.

```
$ terraform apply -var="location=eastus"
```

2. Variable Definition (.tfvars) Files

If there are many variable values to input, we can define them in a variable definition file. Terraform also automatically loads a number of variable definitions files if they are present:



```
terraform.tfvars ×  
Demo > terraform.tfvars  
1  
2 resourceGroupName = "terraformdemo"  
3 location = "eastus"  
4
```

3. Environment Variables

Terraform searches the environment of its own process for environment variables named **TF_VAR_<var-name>** followed by the name of a declared variable. Terraform scans all variables starting with `TF_VAR` and use those as variable values for Terraform.

```
$ export TF_VAR_location=eastus
```