Terraform\_Variables\_Output\_input  
<https://spacelift.io/blog/how-to-use-terraform-variables#what-are-terraform-variables>

* We can create an instance by writing the below code.
* In this case, we will be passing the ami and instance type as variables

|  |
| --- |
| provider "aws" {  region = "ap-south-1"  }  resource "aws\_instance" "first" {  ami = "ami-00bb6a80f01f03502"  instance\_type = "t2.micro"    } |

* Create a new file for variable (variables.tf)
* main.tf

|  |
| --- |
| provider "aws" {  region = "ap-south-1"  }  resource "aws\_instance" "first" {  ami = var.ami\_value  instance\_type = var.type\_value    } |

* Variables.tf

|  |
| --- |
| variable "ami\_value" {  default = "ami-00bb6a80f01f03502"  description = "image type value"  sensitive = false  }  variable "type\_value" {  default = "t2.micro"  description = "The type of ths instance"  sensitive = false    } |

* Here the values will be picked up from the variables file and from the default value
* The default values can be overridden in two ways:
  + Passing the values in CLI as -var argument.
  + Using .tfvars file to set variable values explicitly.
* CLI
  + terraform apply -var “type\_value=t2.nano”
  + type\_value is a varaible which we have defined in the variables.tf
  + Here instead of taking the default value “t2.micro”, we are over riding it with t2.nano
  + The rest of the variables will be picked up from the variables.tf
* .tfvars
  + Create terraform.tfvars file and add the below code

|  |
| --- |
| type\_value = "t2.nano" |

* + Now from the variables file, we have t2.micro as the default value and we are trying to override it with t2.nano
  + We can also create a custom .tfvars file
  + create test.tfvars and place the below code

|  |
| --- |
| type\_value = "t2.small" |

* + Now, we have variables.tf (t2.micro), terraform.tfvars(t2.nano) and test.tfvars(t2.small)
  + if we use the below command, it will pick up the value from the terraform.tfvars  
    terraform apply
  + If we want the value to be picked up from the custom file, use the below command  
    terraform apply -var-file test.tfvars

**Note:** The -var-file argument is great if you have multiple .tfvars files with variations in values. However, if you do not wish to provide the file path every time you run plan or apply, simply name the file as <filename>.auto.tfvars. This file is then automatically chosen to supply input variable values or you can name it as terraform.tfvars

* + If we have files with both terraform.tfvars and <filename>.auto.tfvars, then the value will be picked up from the <filename>.auto.tfvars file.
  + Input variable values can also be set using Terraform environment variables. To do so, simply set the environment variable in the format TF\_VAR\_<variable name>.
  + Variable Precedence
    - The precedence is given to values passed via CLI arguments. This is followed by values passed using the .tfvars file and lastly, the default values are considered.
  + If the values are not provided in the .tfvars file, or as defaults, or as CLI arguments, it falls back on TF\_VAR\_ environment variables.
  + Additionally, if we don’t provide the values in any of the forms discussed above, Terraform would ask for the same in interactive mode when plan or apply commands are run.