

# How to use Terraform Variables - Locals,Input,Output

📅 Apr 1, 2021 · 8 min read · Share on: [🐦](#) [f](#) [in](#) [☰](#)

[Terraform variables](#) 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 explaining how variables are defined inside terraform-

```
variable "<YOUR_VARIABLE_NAME>" {  
  description = "Instance type t2.micro" ← Meaning full description  
  type        = string ← Ex - string, number, bool, list, set, map..  
  default     = "t2.micro" ← variable default value  
}
```

*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**

## Table of Content

1. [Types of Terraform Variables](#)
2. [Terraform Variables - string, number, bool](#)
3. [Terraform Variables - list, set, map](#)
4. [Terraform Output Variables](#)
5. [How to pass variable as `tfvars` file as command-line arguments using the `-var-file` flag?](#)

Terraform Input Variable (string, number, bool,...)



## 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](#))

# 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.

1. provider
2. resource

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

```
provider "aws" {  
    region      = "eu-central-1"  
    access_key  = "<INSERT_YOUR_ACCESS_KEY>"  
    secret_key  = "<INSERT_YOUR_SECRET_KEY>"  
}  
  
resource "aws_instance" "ec2_example" {  
  
    ami          = "ami-0767046d1677be5a0"  
    instance_type = "t2.micro"  
  
    tags = {  
        Name = "Terraform EC2"  
    }  
}
```

BASH

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
```

B

### 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
```

BASH

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

```
provider "aws" {  
    region      = "eu-central-1"  
    access_key  = "<INSERT_YOUR_ACCESS_KEY>"  
    secret_key  = "<INSERT_YOUR_SECRET_KEY>"  
}  
  
resource "aws_instance" "ec2_example" {  
  
    ami          = "ami-0767046d1677be5a0"  
    instance_type = var.instance_type  
  
    tags = {  
        Name = "Terraform EC2"  
    }  
}  
  
variable "instance_type" {  
    description = "Instance type t2.micro"  
    type        = string  
    default     = "t2.micro"  
}
```

BASH

BASH

```
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 -

BASH

```
variable "instance_count" {  
  description = "EC2 instance count"  
  type        = number  
  default     = 2  
}
```

Here is the final terraform file with instance count -

B

```
provider "aws" {  
  region      = "eu-central-1"  
  access_key  = "<INSERT_YOUR_ACCESS_KEY>"  
  secret_key  = "<INSERT_YOUR_SECRET_KEY>"  
}  
  
resource "aws_instance" "ec2_example" {  
  
  ami          = "ami-0767046d1677be5a0"  
  instance_type = "t2.micro"  
  count        = var.instance_count  
}
```

```
}  
}  
  
variable "instance_count" {  
  description = "EC2 instance count"  
  type        = number  
  default     = 2  
}
```

### 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      = "eu-central-1"  
  access_key  = "<INSERT_YOUR_ACCESS_KEY>"  
  secret_key  = "<INSERT_YOUR_SECRET_KEY>"  
}
```

BASH

```
ami          = "ami-0767046d1677be5a0"
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**



```
description = "IAM usernames"
type        = list(string)
default     = ["user1", "user2", "user3s"]
}
```

Here is our final terraform file with List variables -

```
BASH
provider "aws" {
  region      = "eu-central-1"
  access_key  = "<INSERT_YOUR_ACCESS_KEY>"
  secret_key  = "<INSERT_YOUR_SECRET_KEY>"
}

resource "aws_instance" "ec2_example" {

  ami          = "ami-0767046d1677be5a0"
  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"]
}
```

## 3.2 Map variable type

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-alpha",  
    environment  = "dev"  
  }  
}
```

*BASH*

Let's create a Terraform file

```
provider "aws" {  
  region      = "eu-central-1"  
  access_key  = "<INSERT_YOUR_ACCESS_KEY>"  
  secret_key  = "<INSERT_YOUR_SECRET_KEY>"  
}  
  
resource "aws_instance" "ec2_example" {  
  
  ami          = "ami-0767046d1677be5a0"  
  instance_type = "t2.micro"  
  
  tags = var.project_environment  
  
}  
  
variable "project_environment" {  
  description = "project name and environment"  
  type        = map(string)  
  default     = {  
    project      = "project-alpha",
```

*B*

## 4. Terraform Output Variables

In Terraform, [output variables](#) allow you to easily extract information about the resources that were created by Terraform. They allow you to easily reference the values of resources after Terraform has finished running.

Output variables are defined in the **outputs** block in the Terraform configuration. Here's an example of an output variable that references the IP address of an EC2 instance:

*TERRAFORM*

```
output "instance_ip" {  
    value = aws_instance.example.public_ip  
}
```

In this example, the output variable is named "instance\_ip" and its value is set to the public IP of an EC2 instance named "example" that is defined in the Terraform configuration.

### What is terraform output command?

You can use the terraform output command to access the value of an output variable:

*BASH*

```
terraform output instance_ip  
52.11.222.33
```

`t` function. For example:

```
resource "aws_security_group_rule" "example" {  
    ...  
    cidr_blocks = [output.instance_ip]  
}
```

TERRAFORM

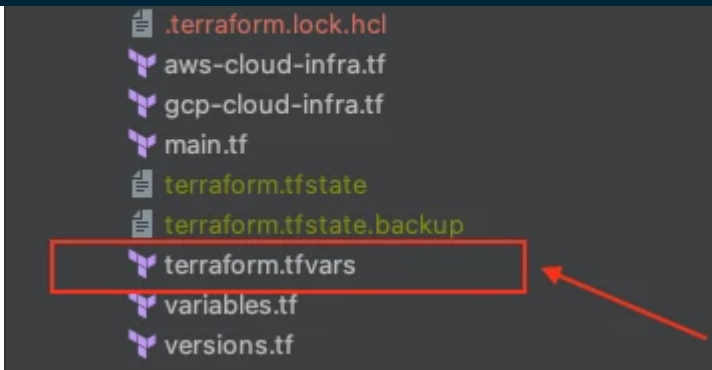
Output variables are especially useful when you need to pass information about the infrastructure that Terraform has created to other systems or scripts. They also enable you to access the values of resources that are not directly visible in the Terraform state, such as the IP address of an EC2 instance.

It's worth noting that you can also set the output variable to be sensitive, in that case, Terraform will mask the output value when it appears in output, making it more secure.

## 5. How to pass variable as `tfvars` file as command-line arguments using the `-var-file` flag?

In Terraform, you can pass variables from a `tfvars` file as command-line arguments using the `-var-file` flag. The `-var-file` flag allows you to specify a file containing variable values, which will be used when running Terraform commands.

Here is an example of one my terraform project where I have created `terraform.tfvars` file -



*terraform.tfvars file inside the terraform project*

Here's an example of how you might use the `-var-file` flag to pass variables from a `tfvars` file named `terraform.tfvars`.

BASH

```
# You need to supply variable during the terraform init
terraform init -var-file=terraform.tfvars

# You need to supply variable during the terraform plan
terraform plan -var-file=terraform.tfvars

# You need to supply variable during the terraform apply
terraform apply -var-file=terraform.tfvars
```

## What tfvars file should contain?

A typical `tfvars` file should contain the variables that you want to pass to Terraform. Each variable should be in the form of `variable_name = value`. For example

TERRAFORM

```
project_id = "gcp-terraform-307119"
location   = "europe-central2"
```

But you should also create a `variable.tf` file also to define the variable type -



*variables.tf and terraform.tfvars file containing the variables*

```
variable "project_id" {  
  type      = string  
  description = "The Project ID"  
}  
  
variable "location" {  
  description = "Region for project"  
  default     = "europe-central2"  
  type        = string  
}
```

TERRAFORM

### How to pass multiple variables files using `-var-file`?

You can also specify multiple variable files by using the `-var-file` flag multiple times on the command line. For example:

```
terraform apply -var-file=myvars-1.tfvars -var-file=myvars-2.tfvars
```

BASH

It's worth noting that variables defined in the command line options will have higher priority than the variables defined in the tfvars files.

## 1. How to use Terragrunt?

## Posts in this Series

- [Managing strings in Terraform: A comprehensive guide](#)
- [How to use terraform depends\\_on meta argument?](#)
- [What is user\\_data in Terraform?](#)
- [Why you should not store terraform state file\(.tfstate\) inside Git Repository?](#)
- [How to import existing resource using terraform import comand?](#)
- [Terraform - A detailed guide on setting up ALB\(Application Load Balancer\) and SSL?](#)
- [Testing Infrastructure as Code with Terraform?](#)
- [How to remove a resource from Terraform state?](#)
- [What is Terraform null Resource?](#)
- [In terraform how to skip creation of resource if the resource already exist?](#)
- [How to setup Virtual machine on Google Cloud Platform](#)
- [How to use Terraform locals?](#)
- [Terraform Guide - Docker Containers & AWS ECR\(elastic container registry\)?](#)
- [How to generate SSH key in Terraform using tls\\_private\\_key?](#)
- [How to fix-Terraform Error acquiring the state lock ConditionalCheckFiledException?](#)
- [Terraform Template - A complete guide?](#)
- [How to use Terragrunt?](#)
- [Terraform and AWS Multi account Setup?](#)
- [Terraform and AWS credentials handling?](#)
- [How to fix-error configuring S3 Backend no valid credential sources for S3 Backend found?](#)
- [Terraform state locking using DynamoDB \(aws\\_dynamodb\\_table\)?](#)
- [Managing Terraform states?](#)

- [How to run specific terraform resource, module, target?](#)
- [How Terraform modules works?](#)
- [Terraform how to do SSH in AWS EC2 instance?](#)
- [What is terraform provisioner?](#)
- [Is terraform destroy needed before terraform apply?](#)
- [How to fix terraform error Your query returned no results. Please change your search criteria and try again?](#)
- [How to use Terraform Data sources?](#)
- [How to use Terraform resource meta arguments?](#)
- [How to use Terraform Dynamic blocks?](#)
- [Terraform - How to nuke AWS resources and save additional AWS infrastructure cost?](#)
- [Understanding terraform count, for\\_each and for loop?](#)
- [How to use Terraform output values?](#)
- [How to fix error configuring Terraform AWS Provider error validating provider credentials error calling sts GetCallerIdentity SignatureDoesNotMatch?](#)
- [How to fix Invalid function argument on line in provider credentials file google Invalid value for path parameter no file exists](#)
- [How to fix error value for undeclared variable a variable named was assigned on the command line?](#)
- [What is variable.tf and terraform.tfvars?](#)
- [How to use Terraform Variables - Locals,Input,Output](#)
- [Terraform create EC2 Instance on AWS](#)
- [How to fix Error creating service account googleapi Error 403 Identity and Access Management \(IAM\) API has not been used in project before or it is disabled](#)
- [Install terraform on Ubuntu 20.04, CentOS 8, MacOS, Windows 10, Fedora 33, Red hat 8 and Solaris 11](#)

## Categories



[ANSIBLE](#) 8[AWS](#) 7[BLOGGING](#) 6[SPRING-BOOT](#) 5[SSL](#) 5[QUARKUS](#) 4[GITHUB](#) 3[KUBESPRAY](#) 3[PROMETHEUS-GRAFANA](#) 3[VAGRANT](#) 3[ALL CATEGORIES](#)

## Series

[TERRAFORM](#) 44[KUBERNETES](#) 27[DOCKER](#) 24[HELM-CHART](#) 11[ANSIBLE](#) 8[AWS](#) 2[LINUX-COMMANDS](#) 1

## Tags

[KUBERNETES](#) 18[HELM-CHART](#) 10[BLOGGING](#) 4[QUARKUS](#) 4[DOCKER](#) 3[GITHUB](#) 3[SSL](#) 3[KUBESPRAY](#) 2[SPRING-BOOT](#) 2[ANSIBLE](#) 1[HADOOP](#) 1[INDEX](#) 1[NGINX](#) 1[TERRAFORM](#) 1

## Recent Posts

- [4 Ways to copy file from localhost to docker container](#)
- [Multiple commands execution in Docker Compose?](#)
- [Clone Public and Private Git Repositories with Ansible](#)
- [How to Limit Ansible playbook on Hosts, Groups and multiple hosts?](#)
- [Easy Fix for 'zsh command not found ansible' Error After Installing Ansible with Pip](#)
- [Demystifying Hosts, Inventory Roles, and Tasks](#)
- [Fixing-Unable to start service apache2 Job for apache2.service failed because the control process exited with error code?](#)
- [Why Ansible is the Ultimate Tool for DevOps Teams - A Beginner's Guide?](#)

## Rahul Wagh



Its all about Open Source and DevOps, here I talk about Kubernetes, Docker, Java, Spring boot and practices.

