

# terraform variables

- <https://developer.hashicorp.com/terraform/tutorials/configuration-language/variables>

## declaración

- recomendado en fichero aparte **variables.tf**

declaración:

variables.tf

```
variable "aws_region" {
  description = "AWS region"
  type        = string
  default     = "us-west-2"
}

variable "instance_count" {
  description = "Number of instances to provision."
  type        = number
  default     = 2
}

variable "enable_vpn_gateway" {
  description = "Enable a VPN gateway in your VPC."
  type        = bool
  default     = false
}

variable "resource_tags" {
  description = "Tags to set for all resources"
  type        = map(string)
  default     = {
    project = "project-alpha",
    environment = "dev"
  }
}
```

uso:

```
provider "aws" {
  region = var.aws_region
}

module "ec2_instances" {
  source = "../modules/aws-instance"

  depends_on = [module.vpc]

  instance_count = var.instance_count
}
```

```
#tags = {
#  project      = "project-alpha",
#  environment = "dev"
#}
tags = var.resource_tags
}
```

## list, map, set

- **list**: lista de valores del mismo tipo, retornados en el mismo orden.
  - Admite el uso de varios tipos: `list(string)`, `list(list)`, `list(map)`, ...

◦ `list("a", "b", "c") # <=0.11`

◦ `["a", "b", "c"] # >=0.12`

◦ variable "public\_subnet\_cidr\_blocks" {  
   description = "Available cidr blocks for public subnets."  
   type = list(string)  
   default = [  
     "10.0.1.0/24",  
     "10.0.2.0/24",  
     "10.0.3.0/24",  
     "10.0.4.0/24",  
     "10.0.5.0/24",  
     "10.0.6.0/24",  
     "10.0.7.0/24",  
     "10.0.8.0/24",  
   ]  
}

- **map**: colección de valores key=value, identificado cada uno con una cadena

◦ `map("a", "b", "c", "d")`  
 {  
   "a" = "b"  
   "c" = "d"  
 }

◦ `{"a" = "b", "c" = "d"}`  
 {  
   "a" = "b"  
   "c" = "d"  
 }

- **set**: colección de valores únicos sin identificadores secundarios y sin orden

## funciones

- `slice()`: obtener parte de una lista.

## más tipos

- **object:** colección de atributos identificados, con posibilidad de diferentes tipos

```
◦ {  
  name = "John"  
  age  = 52  
}  
  
◦ variable "wks_rg_name" {  
  type = object({  
    pre    = string,  
    int    = string,  
    default = string  
  })  
}  
wks_rg_name = {  
  pre    = "prepre",  
  int    = "intint",  
  default = ""  
}  
  
resource "azurerm_resource_group" "rg" {  
  name      = "rg1-rg-${var.wks_rg_name[terraform.workspace]}"  
  location  = "${var.location}"  
}
```

- **tuple:** colección de elementos consecutiva de diferentes tipos

```
◦ ["a", 15, true]
```

/via: <https://www.terraform.io/docs/configuration/types.html>

## input variables

- bloque declaración/valor por defecto:

```
variable "image_id" {  
  type = string  
}  
  
variable "availability_zone_names" {  
  type    = list(string)  
  default = ["us-west-1a"]  
}  
  
variable "docker_ports" {  
  type = list(object({  
    internal = number  
    external = number  
    protocol = string  
  }))  
  default = [  

```

```

    {
      internal = 8300
      external = 8300
      protocol = "tcp"
    }
  ]
}

```

- type (constraints) = bool, number, string, list(), set(), map(), object(), tuple([...])
- default
- description
- validation {condition,error\_message} → en pruebas
- uso:

```
var.<NOMBRE_DECLARATIVO>
```

## interpolación

```
name = "web-sg-${var.resource_tags["project"]}-
${var.resource_tags["environment"]}"
```

## validación

```
variable "resource_tags" {
  description = "Tags to set for all resources"
  type        = map(string)
  default     = {
    project = "my-project",
    environment = "dev"
  }

  validation {
    condition = length(var.resource_tags["project"]) <= 16 &&
length(regexall("[^a-zA-Z0-9-]", var.resource_tags["project"])) == 0
    error_message = "The project tag must be no more than 16 characters, and only
contain letters, numbers, and hyphens."
  }

  validation {
    condition = length(var.resource_tags["environment"]) <= 8 &&
length(regexall("[^a-zA-Z0-9-]", var.resource_tags["environment"])) == 0
    error_message = "The environment tag must be no more than 8 characters, and
only contain letters, numbers, and hyphens."
  }
}
```

## asignación de valores en variables del módulo raíz (Root Module)

- mediante parámetro en el cli -var='<KEY>=<<VALUE>>'
- usando ficheros de definición de variables: \*.tfvars o \*.tfvars.json

- que se pueden cargar automáticamente

```
image_id = "ami-abc123"
availability_zone_names = [
    "us-east-1a",
    "us-west-1c",
]
```

- **terraform.tfvars** o **terraform.tfvars.json**
- **\*.auto.tfvars** o **\*.auto.tfvars.json**
- o especificado como parámetro en el cli
  - en fichero específico -var-file=«<FILE.TFVARS>»
  - en el propio comando -var «server=web»
- en estos ficheros solo se realiza la asignación de la variable
- los ficheros acabados en JSON se parsean como objeto

```
{
  "image_id": "ami-abc123",
  "availability_zone_names": ["us-west-1a", "us-west-1c"]
}
```

- variables de entorno:
  - cualquier variables que empiece por **TF\_VAR\_**
- orden de preferencia (orden en el cargan, el posterior prevalece al anterior):
  1. terraform.tfvars
  2. terraform.tfvars.json
  3. \*.auto.tfvars o \*.auto.tfvars, alfabéticamente
  4. -var o -var-file

## output

como valores de retorno de un módulo o recurso

## usos

- usos
  - módulo hijo expone variables al módulo padre
  - exponer variables en el cli en la ejecución de terraform apply
  - cuando se usa **remote state**, los outputs pueden ser accesibles mediante **terraform\_remote\_state**:

```
data "terraform_remote_state" "vpc" {
  backend = "remote"

  config = {
    organization = "hashicorp"
    workspaces = {
      name = "vpc-prod"
    }
  }
}

# Terraform >= 0.12
resource "aws_instance" "foo" {
```

```
# ...
subnet_id = data.terraform_remote_state.vpc.outputs.subnet_id
}
```

## declaración

```
output "instance_ip_addr" {
  value = aws_instance.server.private_ip
}
```

- **description:** string
- **sensitive:** [true | false] → oculta valores sensibles de la salida por consola, pero los mantiene en el estado
  - cuando se usa estado remoto, este se guarda en la memoria local del equipo
  - terraform output [ [-j-son] [-no-color] [-state=<path\_to\_state\_file>] ]  
<NOMBRE> \* '' -state se ignora si se trabaja con **remote state**
  - terraform output password password = <sensitive> ¿?
- **depends\_on:**
  - list
  - como último recurso en caso de tener algún problema con esa variable y el orden de ejecución

## acceso

- mediante **module.output\_name**

## Local values

- >=0.12

```
locals {
  service_name = "forum"
  owner        = "Community Team"
}
```

```
locals {
  # Ids for multiple sets of EC2 instances, merged together
  instance_ids = concat(aws_instance.blue.*.id, aws_instance.green.*.id)
}
```

```
locals {
  # Common tags to be assigned to all resources
  common_tags = {
    Service = local.service_name
    Owner   = local.owner
  }
}
```

```
resource "aws_instance" "example" {
  # ...

  tags = local.common_tags
}
```

```
}
```

From:  
<https://matebcn.eu/> - **miguel angel torres egea**

Permanent link:  
<https://matebcn.eu/tech:terraform:variables>

Last update: **10/06/2025 15:50**

