

## Enunciado 4

### Ahora realizo el ejercicio 4 y las validaciones

```
type          = string
validation {
  condition    = var.environment_tag !=null && var.environment_tag !=
"" && contains(["dev", "prod", "test"], lower(var.environment_tag))
  error_message = "El valor de environment_tag debe ser 'dev', 'prod' o
'test'"
}
```

Esta es un ejemplo, lanzará un error si la variable es nula o esta vacía y si contiene algo diferente a dev prod y test. Con lower permitimos que si se escribe DEV no salte el error por las mayúsculas porque primero los valores se van a pasar a minúsculas.

El pattern usado para el nombre de vnet:

```
condition    = can(regex("^vnet[a-z]{2,}tfexercise\\d{2,}$",
var.vnet_name))
```

uso can porque así verifica si existe o es null antes del regex

```
^vnet[a-z]{2,}tfexercise\\d{2,}$
```

Vnet+ dos letra minimo en miusculas +tfexercise+dos dígitos minimo

Un ejemplo de la validación:

```
Planning failed. Terraform encountered an error while generating this plan.
```

```
Error: Invalid value for variable
```

```
on variables.tf line 5:
```

```
5: variable "vnet_name" {
```

```
    var.vnet_name is "vnetpalonsotexercise02"
```

```
El nombre de la Virtual Network debe seguir el formato especificado.
```

```
This was checked by the validation rule at variables.tf:8,3-13.
```

```
PS C:\Users\palonso\Desktop\Workspace_VSC\TERRAFORM\ejercicio_terraform>
```

Para validar el mapa habrá que recorrer los valores que hay dentro y crear una lista auxiliar con la condición de que si esta vacío o es null se guarde el elemento en la lista.

Si la lista es >0 entonces es que ha encontrado algún valor nulo:

```
condition      = var.vnet_tags != null && length([for v in  
values(var.vnet_tags) : v if v == null || v == ""]) == 0
```