**12\_Terraform Functions and tfvar files**

**🔹 Functions in Terraform**

Functions are built-in helpers to transform/format data.  
Syntax:

<FUNCTION>(argument1, argument2, ...)

**Common Functions**

* **string functions** → upper(), lower(), trimspace(), replace()
* **numeric functions** → min(), max(), length()
* **collection functions** → element(), lookup(), concat(), tolist()
* **encoding functions** → base64encode(), base64decode()

📌 Example:

output "upper\_case" {

value = upper("terraform")

}

**🔹 tfvars Files**

.tfvars files store variable values separately.

**Example: variables.tf**

variable "instance\_type" {

type = string

default = "t2.micro"

}

variable "instance\_count" {

type = number

}

**Example: terraform.tfvars**

instance\_type = "t2.small"

instance\_count = 2

**Example: createinstance.tf**

resource "aws\_instance" "demo" {

count = var.instance\_count

ami = "ami-0bbdd8c17ed981ef9"

instance\_type = var.instance\_type

tags = {

Name = "demo-${count.index}"

}

}

**Run with:**

terraform apply -var-file="terraform.tfvars"

**🔹 Why tfvars?**

* Keeps .tf files clean.
* Easier to manage **multiple environments** (e.g., dev.tfvars, prod.tfvars).
* Prevents hardcoding values.

**🔹 Example: Using join() Function**

We’ll join multiple strings (or list items) into a single string.

**variables.tf**

variable "names" {

type = list(string)

default = ["app", "db", "cache"]

}

**createinstance.tf**

output "joined\_names" {

value = join("-", var.names)

}

**Result after apply**

joined\_names = "app-db-cache"

👉 join(separator, list)

* "-" → separator
* ["app", "db", "cache"] → list

= "app-db-cache"