**Understanding the Variables Block in Terraform**

**Introduction**

In this lecture, we will take a close look at the variables block in Terraform. We will explore the different arguments that a variable block uses, along with various variable types supported in Terraform.

**Arguments in a Variable Block**

The variable block in Terraform accesses three key parameters:

1. **Default**: Specifies the default value for a variable.
2. **Type**: Enforces the type of variable being used (optional).
3. **Description**: Describes what the variable is used for (optional but recommended for better readability).

**Basic Variable Types**

Terraform supports three basic variable types:

1. **String**: Accepts a single alphanumeric value.
2. **Number**: Accepts a single numeric value (positive or negative).
3. **Boolean**: Accepts a value of either true or false.

If the type parameter is not specified, Terraform sets it to any by default.

**Advanced Variable Types**

Terraform also supports additional complex variable types such as:

**List**

A list is a numbered collection of values. Example:

variable "prefix" {

type = list(string)

default = ["Mr", "Mrs", "Sir"]

}

Accessing list elements by index:

var.prefix[0] # Outputs "Mr"

var.prefix[1] # Outputs "Mrs"

var.prefix[2] # Outputs "Sir"

**Map**

A map stores key-value pairs. Example:

variable "file\_content" {

type = map(string)

default = {

statement1 = "Hello, Terraform!"

statement2 = "Learning Terraform variables."

}

}

Accessing map values:

var.file\_content["statement2"] # Outputs "Learning Terraform variables."

**Combining Type Constraints**

Terraform allows type constraints for lists and maps.

* **List of Strings:**

variable "names" {

type = list(string)

default = ["Alice", "Bob", "Charlie"]

}

* **List of Numbers:**

variable "ages" {

type = list(number)

default = [25, 30, 35]

}

* **Map of Strings:**

variable "user\_roles" {

type = map(string)

default = {

admin = "full-access"

editor = "write-access"

}

}

* **Map of Numbers:**

variable "user\_ids" {

type = map(number)

default = {

Alice = 1001

Bob = 1002

}

}

**Set**

A set is similar to a list but does not allow duplicate elements.

variable "unique\_names" {

type = set(string)

default = ["Alice", "Bob", "Charlie"] # No duplicates allowed

}

Using duplicates in a set will result in an error.

**Object**

An object allows defining multiple attributes with different data types.

variable "bella" {

type = object({

name = string

color = string

age = number

food = list(string)

favorite\_pet = bool

})

default = {

name = "Bella"

color = "Brown"

age = 7

food = ["Fish", "Chicken", "Turkey"]

favorite\_pet = true

}

}

**Tuple**

A tuple is a sequence of elements with different types.

variable "pet" {

type = tuple([string, number, bool])

default = ["Cat", 7, true]

}

If additional elements are added or the type does not match, Terraform will return an error.

**Conclusion**

Terraform provides a variety of variable types to manage infrastructure configurations efficiently. Understanding and using them correctly can greatly improve code maintainability and reusability.