**Terraform task  
Name : Varsha Yadav  
Batch 11  
Topic – Continuation Datatype**   
  
  
tuple

defined structure of array of elements

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

default=["adi",123,true,[1,2,3]]

error scenario

["adi",123"]

["adi",123,[1, 2 ,3],true]

var.varname[index]

var.varname[2][1]

object -> user defind datatype -> wrapper on map, it declares the key structure and type

type = object({

name = string

id = number

address = list(string)

})

default = {

name = "adi"

id = 123

address = ["marathalli","bangalore","560037"]

}

var.varname.keyname

var.varname.address[1]

tfvars

varname = { name = "test", id = 456789, address = ["a1","a2","123"]}

error

{ name = "test", id = "four" , address = ["a1","a2","123"]}

{ house = "test", id = 456 , address = ["a1","a2","123"]}

resource "local\_file" "f9" {

filename = var.filename9[3][1]

content = var.content

}

variable "filename9" {

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

default=["adi",123,true,[1,2,3]]

}

tfvars

["adi",123"]

["adi",123,true,["one",2,3]]

["adi",123,true,[4,5,6]]

resource "local\_file" "f10" {

filename = var.filename10.id

content = var.content

}

variable "filename10" {

type = object({

name = string

id = number

address = list(string)

})

default = {

name = "adi"

id = 123

address = ["marathalli","bangalore","560037"]

}

}

providers, resources

-------------------------------------------------------------

data source -> read only resourceswhicu you are not manging via terraform, they can be used to get information or data for the execution

data <restype> <datasourcename>{

datasource config

}

data "local\_file" "d1" {

filename = path

}

resource "local\_file" "f1" {

filename = "a1.txt"

content = data.local\_file.d1.content

}

vm creation -> resources

rg, vnet, subnet, nic, nsg, vm

rg, vnet, subnet, nsg

nic, vm

----------------------------------------

workspace -> place where all tf execution happens

default

terraform workspace new <ws name>

terraform workspace new dev -> creates a new ws dev and swtch to the workspace

terraform workspace select <wsname>

terraform workspace select default

terraform workspace select dev

**Question : task: random provider -> all datatypes**

**null provider  
  
🡪 Terraform Random Provider**

**The Random provider in Terraform introduces resources that generate random values during their creation and hold those values steady until the inputs are changed. This is particularly useful for creating unique identifiers, passwords, and other random data required in infrastructure provisioning.**[**Medium**](https://medium.com/%40vijayalakshmiyvl/random-provider-in-terraform-950134f9fe86?utm_source=chatgpt.com)

**Key Resources:**

* **random\_string: Generates a random string.**
* **random\_id: Generates a random ID.**
* **random\_password: Generates a random password.**
* **random\_pet: Generates a random pet name.**
* **random\_shuffle: Shuffles a list of strings.**
* **random\_integer: Generates a random integer.**
* **random\_float: Generates a random floating-point number.**
* **random\_uuid: Generates a random UUID.**

**Example Usage:**

**hcl**

**CopyEdit**

**provider "random" {}**

**resource "random\_string" "example" {**

**length = 16**

**special = true**

**}**

**resource "random\_password" "example" {**

**length = 20**

**special = true**

**}**

**output "random\_string" {**

**value = random\_string.example.result**

**}**

**output "random\_password" {**

**value = random\_password.example.result**

**}**

**In this example, Terraform generates a random string and a random password, outputting their values.**

**Terraform Null Provider**

**The Null provider in Terraform provides constructs that intentionally do nothing. These constructs are useful in various situations to help orchestrate tricky behavior or work around limitations. For instance, you can use the null\_resource to execute provisioners or trigger actions without managing any actual infrastructure.**

**Example Usage:**

**:contentReference[oaicite:45]{index=45}**

**In this example, Terraform executes a local command as part of the null\_resource provisioning process.**

**Evolution: From Null Provider to terraform\_data**

**Starting from Terraform v1.4, the functionality provided by the null\_resource has been integrated into a new built-in resource called terraform\_data. This change eliminates the need for the separate Null provider, simplifying configurations and reducing external dependencies.**[**InfoQ+3Medium+3Spacelift+3**](https://medium.com/%40thiagoalves/a-reason-to-stop-using-the-terraform-null-resource-51180b2339?utm_source=chatgpt.com)

**Example Usage of terraform\_data:**

**resource "terraform\_data" "example" {**

**triggers\_replace = [random\_string.example.result]**

**provisioner "local-exec" {**

**command = "echo Replaced!"**

**}**

**}**

**In this example, the terraform\_data resource replaces the need for the null\_resource, providing a cleaner and more integrated approach to executing provisioners.**

**Comparison Table**

| **Feature** | **Random Provider** | **Null Provider (Deprecated)** | **terraform\_data (v1.4+)** |
| --- | --- | --- | --- |
| **Purpose** | **Generate random values** | **Execute provisioners** | **Execute provisioners** |
| **Provider Required** | **Yes** | **Yes** | **No (Built-in)** |
| **Use Case** | **Unique IDs, passwords, etc.** | **Orchestrate actions** | **Orchestrate actions** |
| **Terraform Version Support** | **All versions** | **v0.12 to v1.3** | **v1.4+** |