Terraform Session 8

Summary 2 June 2024

- In terraform we have to write code to automate something that's why it is known as Infrastructure as a code.
- With the help of terraform we can automate almost every infrastructure.
- Terraform also gives you a console like a live interpreter , it is almost like a python terminal.

```
C:\Users\Vimal Daga>terraform console
> max(2,5)
5
>
```

- You can search terraform functions on your browser and it will give you a lot of terraform built in functions.
- If you want to split some information from a string then you can use **split( )** function.
- Split function wants two things: the first is the delimiter and second is **string.**

```
1   output "o1" {
2       value = split( "-" , "ap-south-1a" )
3   }
```

```
Outputs:

o1 = tolist([
  "ap",
  "south",
  "1a",
])
```

- It will automatically convert your data into a list.
- If you want to split a particular part then you can apply array operation.

```
output "o1" {
    value = split( "-" , "ap-south-1a" )[2]
}
```

- We can store this value in a variable also, so we can use it further.

```
locals {
  myAZ = split( "-" , "ap-south-1a" )[2]
}
output "o1" {
    value = local.myAZ
}
```

- In Terraform, the **Join** function is used to concatenate elements of a list into a single string with a specified delimiter. It's useful when you need to combine multiple values into a single string, often for constructing configurations like IP addresses, paths, or other delimited strings.

```
  myRegion =  join ( "-" , ["ap" , "south", "1b"] )
}

output "o1" {
    value = local.myRegion
}
```

- In Terraform, the **Lower** function is used to convert upper case to lower case.

```
C:\Users\Vimal Daga>terraform console
> lower("HELLO")
"hello"
>
```

- We can pass a function in a function in terraform.

```
  myAZ1 =  lower ( substr( "Welcome to Linux World", 3, 15 ) )
}
```

- In Terraform, regular expressions (regex) can be used for string manipulation, validation, and pattern matching. Terraform provides several functions that allow you to work with regex, such as **regex.**

```
> regex("[a-z]+", "53453453.345345aaabbbccc23454")
"aaabbbccc"
```

```
> regex("(\\d\\d\\d\\d)-(\\d\\d)-(\\d\\d)", "2019-02-01")
[
  "2019",
  "02",
  "01",
]
```

- We can make or retrieve a key from a string through regex in terraform.
- In Terraform, the **endswith( )** function is used to determine if a given string ends with a specified substring. This function returns **true** if the string ends with the substring, and **false** otherwise.

```
> endswith("hello world", "world")
true

> endswith("hello world", "hello")
false
```

- The **replace( )** function in Terraform is used to perform string replacements. It takes a string, searches for a pattern (either a literal string or a regular expression), and replaces it with a specified replacement string.

```
> replace("1 + 2 + 3", "+", "-")
1 - 2 - 3
```

- In Terraform, the **contains( )** function is used to check whether a list contains a specific element. It returns **true** if the element is found in the list and **false** if it is not.

```
> contains(["a", "b", "c"], "a")
true
> contains(["a", "b", "c"], "d")
false
```

- In Terraform, the **flatten( )** function is used to collapse a list of lists into a single flat list. It takes a nested list (a list containing other lists) and returns a single list with all the elements in the original nested lists.

```
variable "nested_list" {
  default = [["apple", "banana"], ["orange", "grape"], ["mango"]]
}

output "flat_list" {
  value = flatten(var.nested_list)
}
```

```
flat_list = ["apple", "banana", "orange", "grape", "mango"]
```

- In Terraform, the **tolist( )** function is used to convert a given value into a list. This is particularly useful when you have a value that is not explicitly a list but needs to be treated as one, such as when working with a single value or a set that you need to convert to a list.
- The **lookup( )** function in Terraform is a utility that retrieves a value from a map using a given key. If the key is not present in the map, a default value can be returned. This function is particularly useful for retrieving values from maps in a way that allows for safe fallbacks if a key isn't found.

```
> lookup({dev="t2.micro", prod="t2.large"}, "dev", "t2.small")
"t2.micro"
```

- If you use the list and try to retrieve some value it will fail. It will give an out of index error because the list does not give you a wrapped around function for that you have to use **element( )** function in terraform.

```
> ["a", "b", "c"][3]

Error: Invalid index

  on <console-input> line 1:
  (source code not available)
```

```
> element(["a", "b", "c"], 3)
a
```

- In Terraform, the **slice( )** function is used to extract a subset of elements from a list, starting at a specified index and ending just before another specified index. This function is useful when you need to work with only a portion of a list.
- In Terraform, the **urlencode( )** function is used to encode a string so that it is safe to be included in a URL. This means that special characters within the string (such as spaces, slashes, or punctuation) are converted into their corresponding percent-encoded form, making the string compliant with URL standards.

```
> urlencode("Hello World!")
"Hello+World%21"
```

- In Terraform, the **base64encode( )** function is used to encode a string in Base64 format. This is commonly used when you need to transmit binary data or sensitive information as a text string, such as in configuration files or API requests.

```
> base64encode("Hello World")
"SGVsbG8gV29ybGQ="
```

- In Terraform, the **abspath( )** function is used to return the absolute path of a given relative path. This function is useful when you need to resolve a path relative to the current working directory or module path, and you want to ensure that you're working with an absolute path.

```
> abspath(path.root)
"C:/Users/Vimal Daga"
```

- In Terraform, the **timestamp( )** function returns the current date and time in UTC, formatted as an RFC 3339 string (which is a subset of ISO 8601). This is useful for generating time-based metadata, logging, or creating unique identifiers.

```
> timestamp()
"2024-06-02T08:52:47Z"
```

- In Terraform, the **cidrhost( )** function is used to calculate the IP address for a specific host within a given CIDR (Classless Inter-Domain Routing) block. This function is particularly useful when you need to assign specific IP addresses within a subnet or CIDR range.

```
> cidrhost("10.12.112.0/20", 16)
10.12.112.16
```

- Hardcoding is not a good practice, repeating is also not a good practice.So if you want to generalize your project you have to make it more dynamic, more robustic by using some of these functions