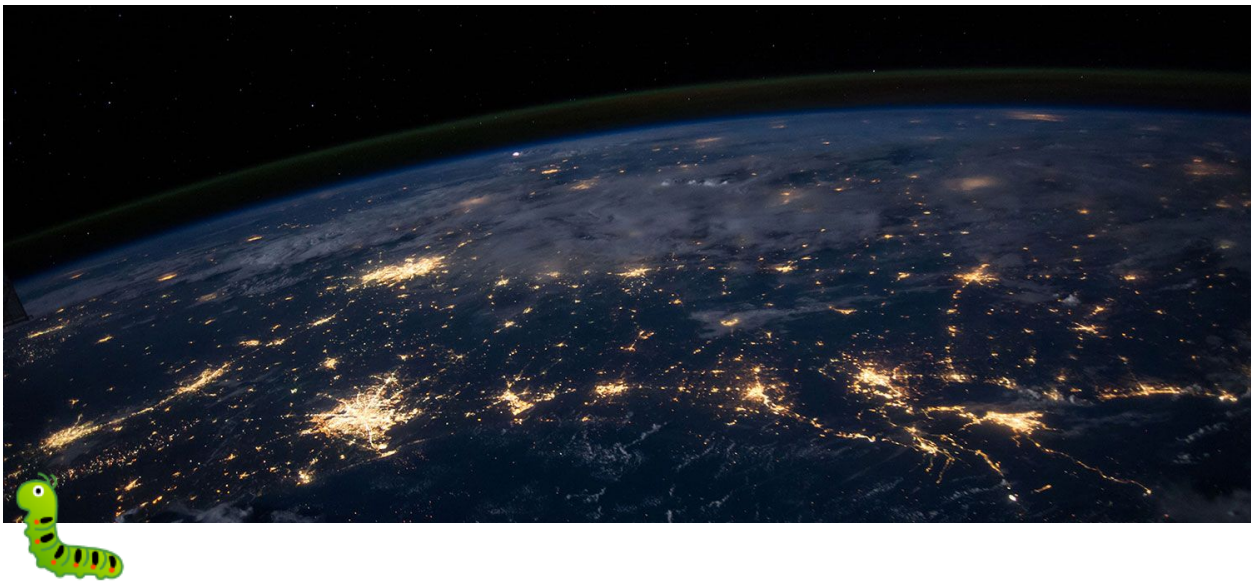

Terraform HCL Blocks

A `Terraform configuration` is a complete document in the Terraform language that tells Terraform how to manage a given collection of infrastructure. Terraform language is made up of the below elements.

- Blocks
- Arguments
- Expressions

The format of HCL is :

```
<BLOCK TYPE> "<BLOCK LABEL>" "<BLOCK LABEL>" {
  # Block body
  <IDENTIFIER> = <EXPRESSION> # Argument
}
```

The `Blocks` acts as a container. Blocks have a `block type`, can have zero or more labels, and have a `body` that contains any number of arguments and nested blocks. `Arguments` assign a value to a name. `Expressions` represent a value of Argument.

For example, let us consider below configuration file

```
resource "aws_instance" "hello" {
  ami = "xyz"

  network_interface {
    # ...
  }
}
```

Here, the Block Type is a `resource`. The resource block type expects two labels, which are `aws_instance` and `hello` in the example above. For some of the resources like `network_interface` defined above, does not expect any labels.

The code in the Terraform language is stored in plain text files with the `.tf` file extension. A collection of `.tf` files kept together in a directory is called the `modules`. Terraform always runs in the context of a single root module. A complete Terraform configuration consists of a root module and the tree of child modules.

Block Type: resource

Resource block is the most important fundamental block type which is used to describes one or more infrastructure objects. The format for resources is:

```
resource "aws_lambda_function" "test_lambda" {
  function_name = "hello-lambda"
  handler       = "index.test"
}
```

A `resource` block declares a resource of a given `type` ("aws_lambda_function") with a given `local name` ("test_lambda"). The name is used to refer to this resource from elsewhere in the same Terraform module. The resource type and name together serve as an `identifier` for a given resource and so must be unique within a module.

Each resource is associated with a single resource type, which determines the kind of infrastructure object it manages and what arguments and other attributes the resource supports. When we call the resource elsewhere, we do it by calling `aws_lambda_function.test_lambda`

Block Type: data

Data blocks allow Terraform to use the information defined outside of Terraform. Each provider like AWS, Azure may offer data sources alongside its resource types.

```
data "aws_lambda_function" "existing" {
  function_name = "test_lambda"
}
```

A `data` block requests that Terraform read from a given `data source` ("aws_lambda_function") and export the result under the given `local name` ("existing"). The name is used to refer to this resource from elsewhere in the same Terraform module.

Block Type: module

Modules are containers for multiple resources that are used together. A module consists of a collection of `.tf` files kept together in a directory. Modules are the main way to package and reuse resource configurations with Terraform.

Every Terraform configuration has at least one module, known as its `root module`, which consists of the resources defined in the `.tf` files in the main working directory.

A Terraform module can call other modules to include their resources into the configuration. A module that has been called by another module is often referred to as a `child module.`

Block Type: Input variables and Output

Input:

Input variables are like function arguments. Input variables are often referred to as just `variables`. Each input variable accepted by a module must be declared using a variable block:

```
variable "aws_profile" {
  type = list(string)
  default = ["ap-south-1"]
}
```

The `label` (availability_zone_names) after the `variable` keyword is a name for the variable, which must be unique among all variables in the same module.

Output:

Output values are like the return values of a Terraform module.Each output value exported by a module must be declared using an output block:

```
output "test_lambda_arn" {
  value = aws_lambda_function.arn}
```

The label immediately after the `output` keyword is the name, which must be a valid identifier. The `value` argument takes an expression whose result is to be returned to the user. In the above example, the expression refers to the `arn` attribute exposed by an `aws_lambda_function` resource.