Objective 5: Interact with Terraform modules

▼ Contrast module source options

- **Module Overview**
  - Definition - a set of configuration files in a single directory. A container for multiple resources that are used together.
  - A module that is called by another configuration is sometimes referred to as a "child module" of that configuration.
- **Applications**
  - Organize configuration - easier to navigate, understand, and update our configuration by keeping all related parts together.
  - Encapsulate configuration - put configuration into distinct logical components. Reduces chance of error. Ex/naming two diff resources the same thing.
  - Re-use configuration - share and re-use modules with the public and teams
  - Provide consistency and ensure best practices
- **Module source options:**
  - we reference a **Public Registry Module** by declaring the source.

```
module "consul" {
#<NAMESPACE>/<NAME>/<PROVIDER>
source = "hashicorp/consul/aws"
version = "0.1.0"
}
```

  - **Private Registry Module** Sources follow this syntax

```
module "vpc" {
  #<HOSTNAME>/<NAMESPACE>/<NAME>/  <PROVIDER>
  source = "app.terraform.io/example_corp/vpc/aws"
  version = "0.9.3"
}
```

▼ Interact with module inputs and outputs

Modules

▼ Describe variable scope within modules/child modules

- variables are parameters for modules
- variables allow us to customize modules without changing the source code and they allow for modules to be shared between different configurations.
- root module variables can be set with CLI and environment variables.
- When declaring variables in child modules, the calling module should pass values in the module block.
- **Declaring a variable:**
- variable names have to be unique per module
- any name can be used except for :source, version, providers, count,for_each,lifecycle,depends_on,locals
- Note: if type and default are used, default must be convertible to the type

```
variable "image_id" {
type = string
#defines what value types are accepted for the variable, if not explicit any type
#Types: string,number,bool, any(to allow for any type) |  Complex Type: list(<TYPE
validation {
  condition     = length(var.image_id) > 4 && substr(var.image_id, 0, 4) == "ami-"
  error_message = "The image_id value must be a valid AMI id, starting with \"ami-
   }
#validation rules are experimental - uses value of variable to return true or fals
}

variable "availability_zone_names" {
type    = list(string)
default = ["us-west-1a"]
#default means the variable is considered optional, used if no other value is set
description = "variable description, purpose and value expected"
 }

variable "docker_ports" {
type = list(object({
  internal = number
  external = number
  protocol = string
  }))

default = [
  {
    internal = 8300
    external = 8300
    protocol = "tcp"
```

```
  }
 ]
}
#---------------------
#To use validation we need to opt in
terraform {
experiments = [variable_validation]
}
```

- Using variable values

```
resource "aws_instance" "example" {
instance_type = "t2.micro"
ami  = var.image_id #expression reads var.<NAME> name is the label declared on t
}
```

- Set root module variables 1) In Terraform Cloud Workspace 2) Individual CLI with `-var` 3) In `.tfvars` file 4) As environment variable
- child modules have variables set in the configuration of the parent module

▼ Discover modules from the public Terraform Module Registry

- Finding and Using Modules
  - Terraform Registry

▼ Defining module version

- Use the version attribute in the module block to specify versions:

```
module "consul" {
  source  = "hashicorp/consul/aws"
  version = "0.0.5"  #single explicit version
  #or
  version = >= 1.2.0  #version constraint expression
  servers = 3
 }
```

⏮️Objective 4 || Objective 6 ⏩

⬅️BACK README