Terraform import using Module that supports for each concept

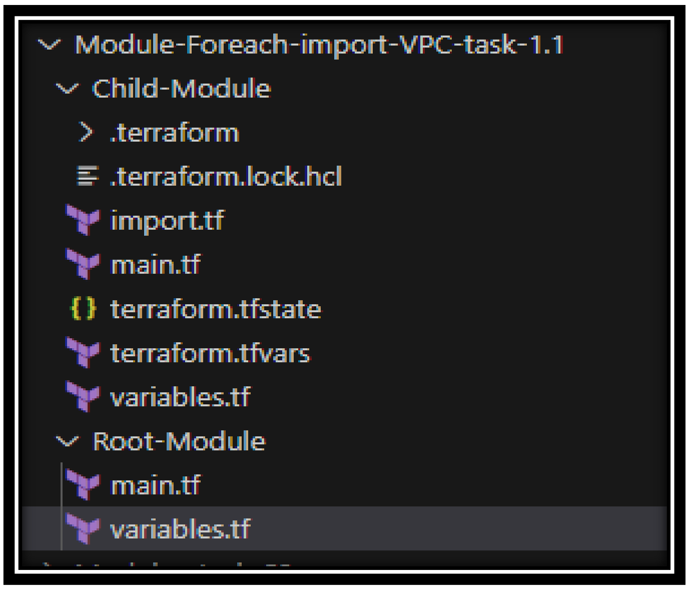
TASK:

- To Implement Terraform import using Module that Support **for_each** concept.

STEP -by-STEP PROCESS:

Directory-layout:

- The AWS VPC creation setup on the directory layout, by configuring root_modules and child_modules structure.



```
Module-Foreach-import-VPC-task-1.1
  Child-Module
    .terraform
    .terraform.lock.hcl
    import.tf
    main.tf
    terraform.tfstate
    terraform.tfvars
    variables.tf
  Root-Module
    main.tf
    variables.tf
```

Root_Module/main.tf

```
resource "aws_vpc" "my_vpc" {
  cidr_block           = var.cidr_block
  enable_dns_support   = true
  enable_dns_hostnames = true
  tags = {
    Name = var.vpc_name
  }
}
```

- This Terraform configuration creates an AWS VPC with a CIDR block (which is specified via a variable), DNS resolution, and DNS hostnames enabled. It also tags the VPC with a name defined by another variable (var.vpc_name).

Terraform import using Module that supports for each concept

Root_Module/variable.tf

```
variable "cidr_block" {

  description = "CIDR block for the VPC"
  type        = string
}

variable "vpc_name" {
  description = "Name for the VPC"
  type        = string
}
```

- This Terraform template is a flexible, reusable configuration for creating an AWS VPC.
- **Customization**: The CIDR block and VPC name can be passed in as variables, allowing the same template to be used across different environments (e.g., production, development) by simply changing the values of the variables.

Child_Modeule Configuration:

Child_Module/import.tf

```
# Importing the exisiting VPC using its ID and NAME.
import {
  to = aws_vpc.arun_vpc["arun_VPC"]
  id = "vpc-0fedf0217cfb11890"
}
```

- The import template was used to import existing resources from the provider. it refers to the Child_Module/**main.tf template.**

Child_Module/main.tf

```
provider "aws" {
  region = var.region
}

# Create  new custome arun_VPC1 and arun_VPC2
module "vpc" {
  source     = "../Root-module"
  for_each   = { for k, v in var.vpcs : k => v if k != "arun_VPC" }
  cidr_block = each.value.cidr_block
  vpc_name   = each.key
}
```

Terraform import using Module that supports for each concept

```
# VPC to Import.
resource "aws_vpc" "default_vpc" {
  for_each = { arun_VPC = var.vpcs["arun_VPC"] }

  cidr_block = each.value.cidr_block

  tags = {
    Name = "arun_VPC"
  }
}
```

Summary of the Template:

- **Provider**: The AWS provider is configured using a region specified by the user.
- **New VPCs (arun_VPC1, arun_VPC2)**: The module block dynamically creates new VPCs (except arun_VPC) using a predefined module. It passes the CIDR block and name from the var.vpcs map.
- **Importing Existing VPC (arun_VPC)**: The configuration explicitly handles the existing arun_VPC using a for_each loop to manage it as a resource. The CIDR block and tags are set dynamically from the same var.vpcs map.

Flexible and Modular Setup:

- This template is **modular**: It uses a Terraform module to manage multiple VPCs, and the same logic can be reused in different environments or for different VPCs.
- It is **dynamic**: By using the for_each loop and filtering logic, the configuration allows selective creation and management of VPCs.
- **Separation of concerns**: The configuration separates the logic for creating new VPCs (arun_VPC1, arun_VPC2, etc.) and importing an existing VPC (arun_VPC), keeping the template clean and manageable.

Child_Module/terraform.tfvars

```
} region = "us-east-1"


vpcs = {
 arun_VPC1 = {
    cidr_block = "10.0.0.0/16"
  },
 arun_VPC2 = {
    cidr_block = "10.1.0.0/16"
  },
  arun_VPC = {
    cidr_block = "10.0.0.0/24"
  }
}
```

- **Terraform.tfvars** it contains all the necessary attributes of the resources.

Child_Module/Variables.tf

```
variable "region" {
  description = "AWS region"
  type        = string
}

variable "vpcs" {
  description = "Map of VPCs to create or import"
  type = map(object({
    cidr_block = string
  }))
}
```

Description: This variable defines a map of VPCs, where each key represents the VPC name (e.g., "arun_VPC1", "arun_VPC2", etc.), and the value contains the CIDR block for that VPC.

Type: It is defined as a map of objects.

- map(object({})): The variable is a map where each key points to an object.
- Object: Each object contains at least one property:
  - cidr_block (string): A string value that holds the CIDR block for that particular VPC.

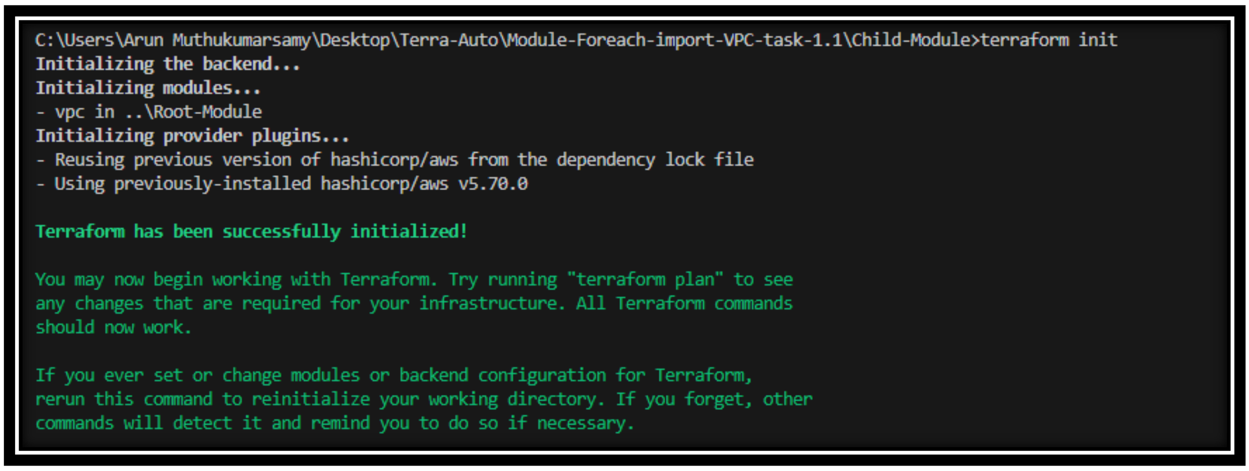
Key Advantages of This Approach:

- **Scalability**: You can easily add or remove VPCs by updating the vpcs map, without needing to modify the core template.
- **Flexibility**: By using dynamic variables, the template can handle both VPC creation and importing of existing ones seamlessly.
- **Modularity**: It separates the logic for defining regions and VPC configurations, making the template cleaner and easier to maintain.

Terraform import using Module that supports for each concept

Execution of the Modules:

Initialize Terraform: Navigate to the child_module directory and run by using terraform init**.**



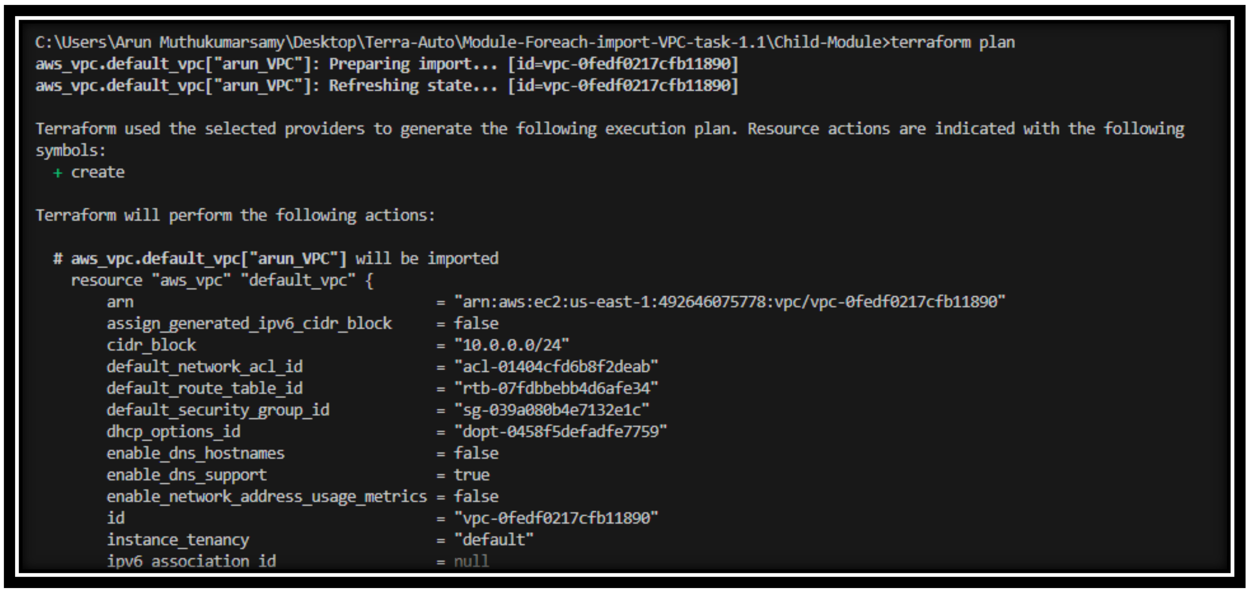
```
C:\Users\Arun Muthukumarsamy\Desktop\Terra-Auto\Module-Foreach-import-VPC-task-1.1\Child-Module>terraform init
Initializing the backend...
Initializing modules...
- vpc in ..\Root-Module
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.70.0

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

Plan: To see what resources will be created & imported, run by using the **terraform plan**.

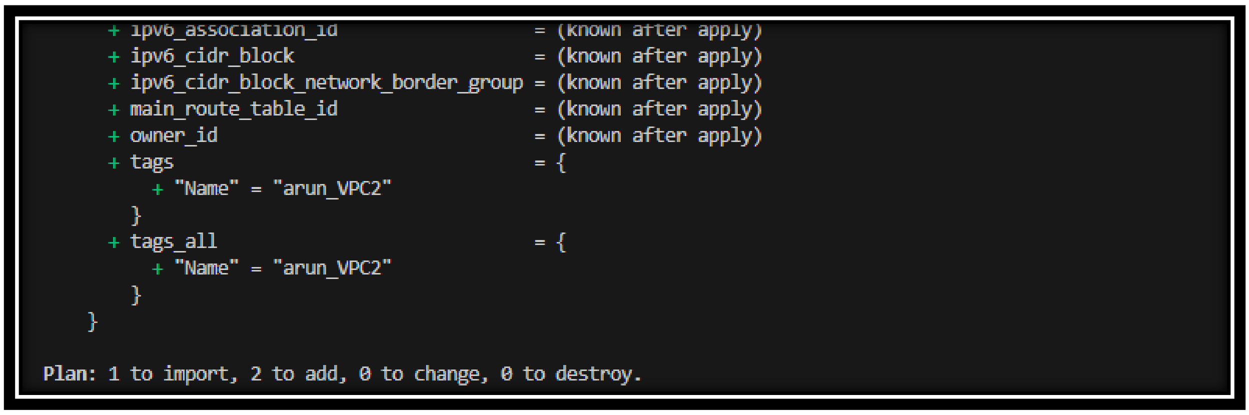


```
C:\Users\Arun Muthukumarsamy\Desktop\Terra-Auto\Module-Foreach-import-VPC-task-1.1\Child-Module>terraform plan
aws_vpc.default_vpc["arun_VPC"]: Preparing import... [id=vpc-0fedf0217cfb11890]
aws_vpc.default_vpc["arun_VPC"]: Refreshing state... [id=vpc-0fedf0217cfb11890]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following
symbols:
  + create

Terraform will perform the following actions:

  # aws_vpc.default_vpc["arun_VPC"] will be imported
    resource "aws_vpc" "default_vpc" {
        arn                                  = "arn:aws:ec2:us-east-1:492646075778:vpc/vpc-0fedf0217cfb11890"
        assign_generated_ipv6_cidr_block     = false
        cidr_block                           = "10.0.0.0/24"
        default_network_acl_id               = "acl-01404cfd6b8f2deab"
        default_route_table_id               = "rtb-07fdbbebb4d6afe34"
        default_security_group_id            = "sg-039a080b4e7132e1c"
        dhcp_options_id                      = "dopt-0458f5defadfe7759"
        enable_dns_hostnames                 = false
        enable_dns_support                   = true
        enable_network_address_usage_metrics = false
        id                                   = "vpc-0fedf0217cfb11890"
        instance_tenancy                     = "default"
        ipv6_association_id                  = null
```

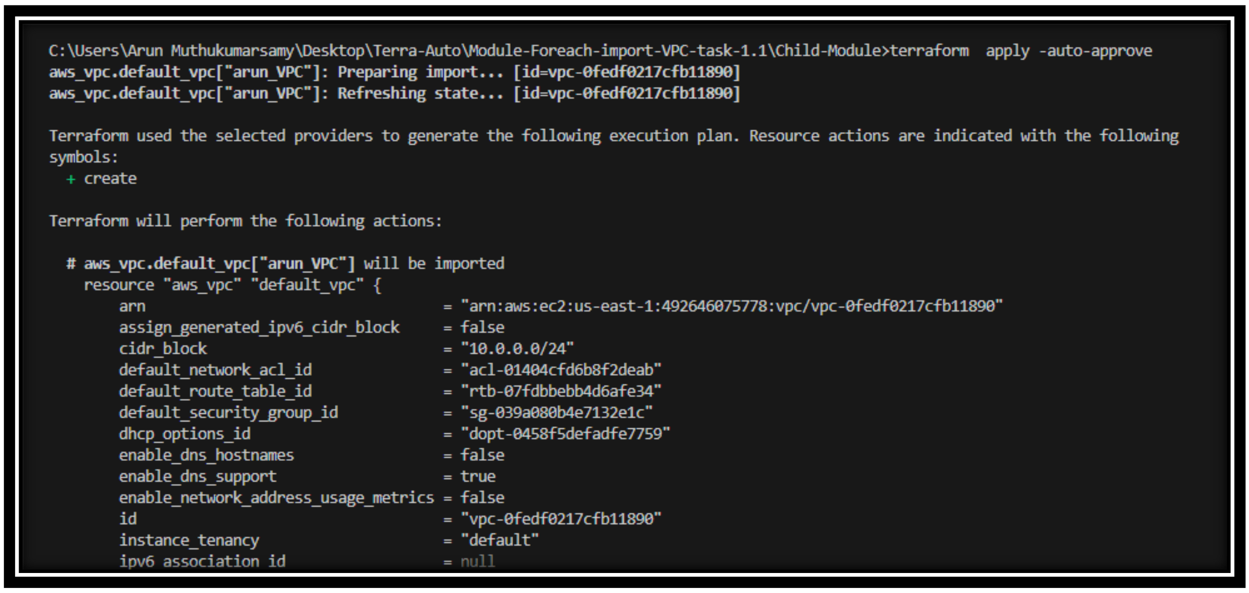


```
      + ipv6_association_id                  = (known after apply)
      + ipv6_cidr_block                      = (known after apply)
      + ipv6_cidr_block_network_border_group = (known after apply)
      + main_route_table_id                  = (known after apply)
      + owner_id                             = (known after apply)
      + tags                                 = {
          + "Name" = "arun_VPC2"
        }
      + tags_all                             = {
          + "Name" = "arun_VPC2"
        }
    }

Plan: 1 to import, 2 to add, 0 to change, 0 to destroy.
```

Terraform import using Module that supports for each concept

Apply: To apply the configuration and create the resources, run by using terraform apply -auto-approve.

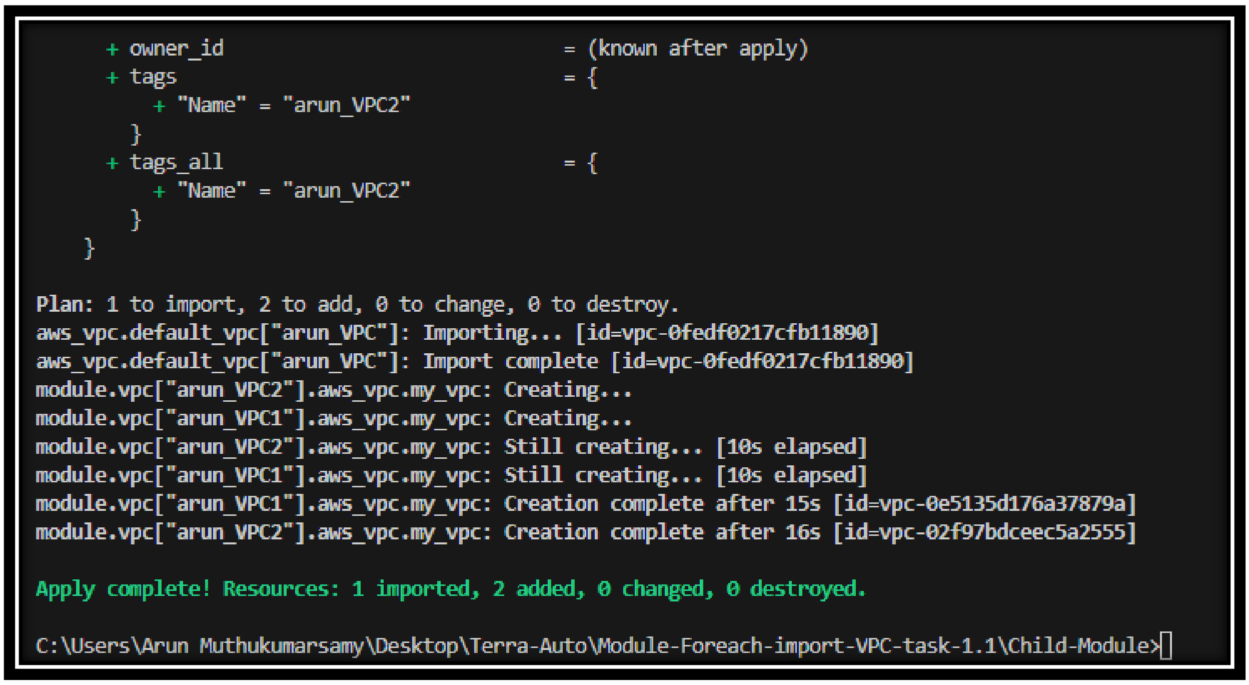


```
C:\Users\Arun Muthukumarsamy\Desktop\Terra-Auto\Module-Foreach-import-VPC-task-1.1\Child-Module>terraform  apply -auto-approve
aws_vpc.default_vpc["arun_VPC"]: Preparing import... [id=vpc-0fedf0217cfb11890]
aws_vpc.default_vpc["arun_VPC"]: Refreshing state... [id=vpc-0fedf0217cfb11890]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following
symbols:
  + create

Terraform will perform the following actions:

  # aws_vpc.default_vpc["arun_VPC"] will be imported
    resource "aws_vpc" "default_vpc" {
        arn                                  = "arn:aws:ec2:us-east-1:492646075778:vpc/vpc-0fedf0217cfb11890"
        assign_generated_ipv6_cidr_block     = false
        cidr_block                           = "10.0.0.0/24"
        default_network_acl_id               = "acl-01404cfd6b8f2deab"
        default_route_table_id               = "rtb-07fdbbebb4d6afe34"
        default_security_group_id            = "sg-039a080b4e7132e1c"
        dhcp_options_id                      = "dopt-0458f5defadfe7759"
        enable_dns_hostnames                 = false
        enable_dns_support                   = true
        enable_network_address_usage_metrics = false
        id                                   = "vpc-0fedf0217cfb11890"
        instance_tenancy                     = "default"
        ipv6_association_id                  = null
```



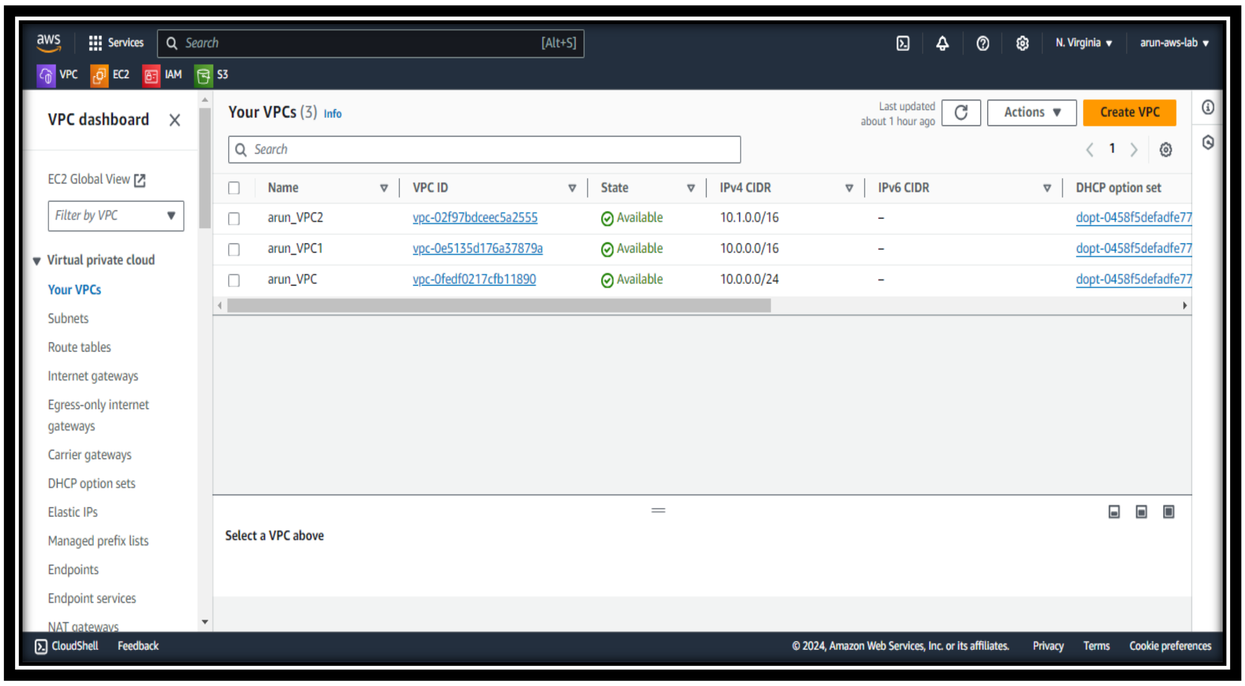
```
      + owner_id                             = (known after apply)
      + tags                                 = {
          + "Name" = "arun_VPC2"
        }
      + tags_all                             = {
          + "Name" = "arun_VPC2"
        }
    }

Plan: 1 to import, 2 to add, 0 to change, 0 to destroy.
aws_vpc.default_vpc["arun_VPC"]: Importing... [id=vpc-0fedf0217cfb11890]
aws_vpc.default_vpc["arun_VPC"]: Import complete [id=vpc-0fedf0217cfb11890]
module.vpc["arun_VPC2"].aws_vpc.my_vpc: Creating...
module.vpc["arun_VPC1"].aws_vpc.my_vpc: Creating...
module.vpc["arun_VPC2"].aws_vpc.my_vpc: Still creating... [10s elapsed]
module.vpc["arun_VPC1"].aws_vpc.my_vpc: Still creating... [10s elapsed]
module.vpc["arun_VPC1"].aws_vpc.my_vpc: Creation complete after 15s [id=vpc-0e5135d176a37879a]
module.vpc["arun_VPC2"].aws_vpc.my_vpc: Creation complete after 16s [id=vpc-02f97bdceec5a2555]

Apply complete! Resources: 1 imported, 2 added, 0 changed, 0 destroyed.

C:\Users\Arun Muthukumarsamy\Desktop\Terra-Auto\Module-Foreach-import-VPC-task-1.1\Child-Module>
```

Terraform import using Module that supports for each concept



VPC dashboard

Your VPCs (3)

Name	VPC ID	State	IPv4 CIDR	IPv6 CIDR	DHCP option set
arun_VPC2	vpc-02f97bdceec5a2555	Available	10.1.0.0/16	–	dopt-0458f5defadfe77
arun_VPC1	vpc-0e5135d176a37879a	Available	10.0.0.0/16	–	dopt-0458f5defadfe77
arun_VPC	vpc-0fedf0217cfb11890	Available	10.0.0.0/24	–	dopt-0458f5defadfe77

Select a VPC above

© 2024, Amazon Web Services, Inc. or its affiliates.

- ➢ In the AWS console 2 new VPCs were created and 1 was imported.