Modules-based Terraform resource creation using own modules

TASK:

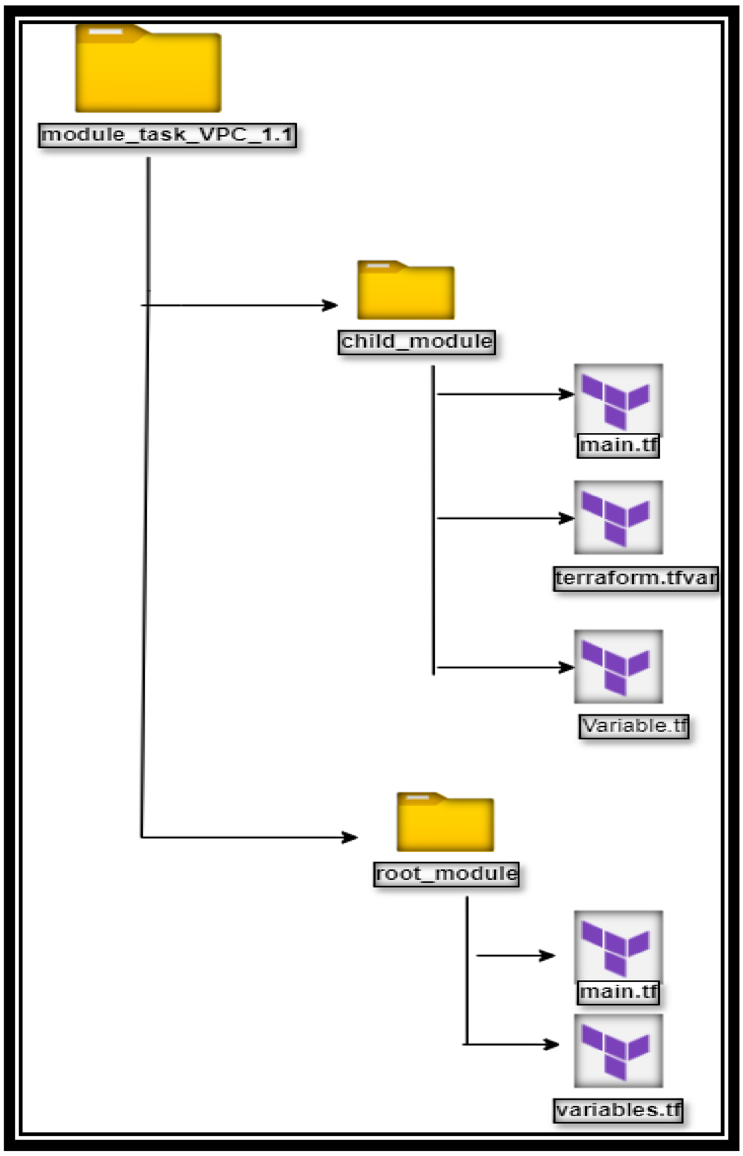
- Module-based Terraform resource creation using own modules.

STEP BY STEP PROCESS:

Directory Layout Setup:

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

Directory layout workflow:



Modules-based Terraform resource creation using own modules

STEP 1:

- To set up the **root_module** directory. In the **root_module** directory, it has **main.tf** you call the **vpc** module by referencing the **child_module** directory.
- Variables such as **vpc_cidr_block, azs, public_subnets, private_subnets**, and **region** are passed into the child module.

root_module/main.tf

```
module "vpc" {
  source = "../root_module"  # path to defined root module.

  vpc_cidr_block = var.vpc_cidr_block
  azs            = var.azs
  public_subnets = var.public_subnets
  private_subnets = var.private_subnets
  region         = var.region
}
```

root_module/variable.tf

```
variable "vpc_cidr_block" {
  description = "CIDR block for the VPC"
  type        = string
  default     = "10.0.0.0/16"
}

variable "azs" {
  description = "List of availability zones"
  type        = list(string)
  default     = ["us-east-1a", "us-east-1b"]
}

variable "public_subnets" {
  description = "CIDR blocks for public subnets"
  type        = list(string)
  default     = ["10.0.1.0/24", "10.0.2.0/24"]
}

variable "private_subnets" {
  description = "CIDR blocks for private subnets"
  type        = list(string)
  default     = ["10.0.101.0/24", "10.0.102.0/24"]
}

variable "region" {
  description = "AWS region"
  type        = string
  default     = "us-east-1"
}
```

STEP 2:

- To define the **child_module** directory setup. In the child module **child_module/main.tf**, the VPC, subnets, internet gateway, and route tables are created using the variables passed from the root module.
- **Variable definition:** Both the root module and the child module define the necessary variables. The root module can override the default values for the VPC configuration**.**
- **terraform.tfvars :** This file is optional, and you can use it to define variable values. If not used, the defaults from **variables.tf** will be applied.

child_module/main.tf

- The child module will contain the actual implementation of the AWS VPC and its associated resources.

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

resource "aws_vpc" "this" {
  cidr_block = var.vpc_cidr_block

  tags = {
    Name = "My VPC"
  }
}

resource "aws_subnet" "public" {
  count                   = length(var.public_subnets)
  vpc_id                  = aws_vpc.this.id
  cidr_block              = var.public_subnets[count.index]
  availability_zone       = var.azs[count.index]
  map_public_ip_on_launch = true

  tags = {
    Name = "Public Subnet ${count.index + 1}"
  }
}
```

Modules-based Terraform resource creation using own modules

```
resource "aws_subnet" "private" {
  count             = length(var.private_subnets)
  vpc_id            = aws_vpc.this.id
  cidr_block        = var.private_subnets[count.index]
  availability_zone = var.azs[count.index]

  tags = {
    Name = "Private Subnet ${count.index + 1}"
  }
}

resource "aws_internet_gateway" "this" {
  vpc_id = aws_vpc.this.id

  tags = {
    Name = "Internet Gateway"
  }
}

resource "aws_route_table" "public" {
  vpc_id = aws_vpc.this.id

  tags = {
    Name = "Public Route Table"
  }
}

resource "aws_route" "default_route" {
  route_table_id         = aws_route_table.public.id
  destination_cidr_block = "0.0.0.0/0"
  gateway_id             = aws_internet_gateway.this.id
}

resource "aws_route_table_association" "public" {
  count          = length(var.public_subnets)
  subnet_id      = aws_subnet.public[count.index].id
  route_table_id = aws_route_table.public.id
}
```

child_module/variables.tf

```
variable "vpc_cidr_block" {
  description = "CIDR block for the VPC"
  type        = string
}
```

```
variable "azs" {
  description = "List of availability zones"
  type        = list(string)
}

variable "public_subnets" {
  description = "CIDR blocks for public subnets"
  type        = list(string)
}

variable "private_subnets" {
  description = "CIDR blocks for private subnets"
  type        = list(string)
}

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

child_module/terraform.tfvars

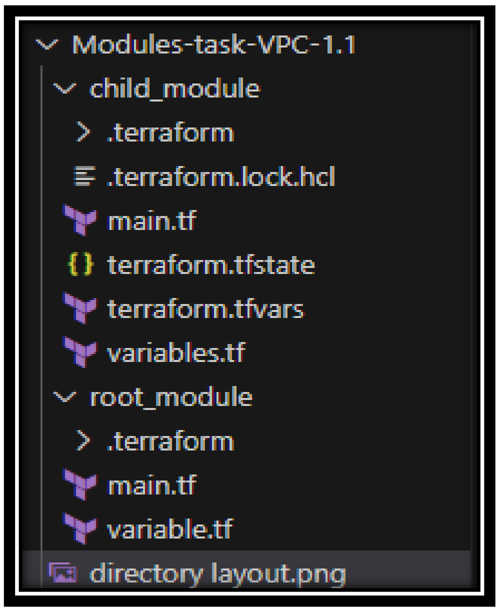
```
vpc_cidr_block = "10.0.0.0/16"
azs            = ["us-east-1a", "us-east-1b"]
public_subnets = ["10.0.1.0/24", "10.0.2.0/24"]
private_subnets = ["10.0.101.0/24", "10.0.102.0/24"]
region         = "us-east-1"
```

USE CASE:

➢ This setup organizes your Terraform code into reusable modules while keeping the root and child configurations clean.

Modules-based Terraform resource creation using own modules

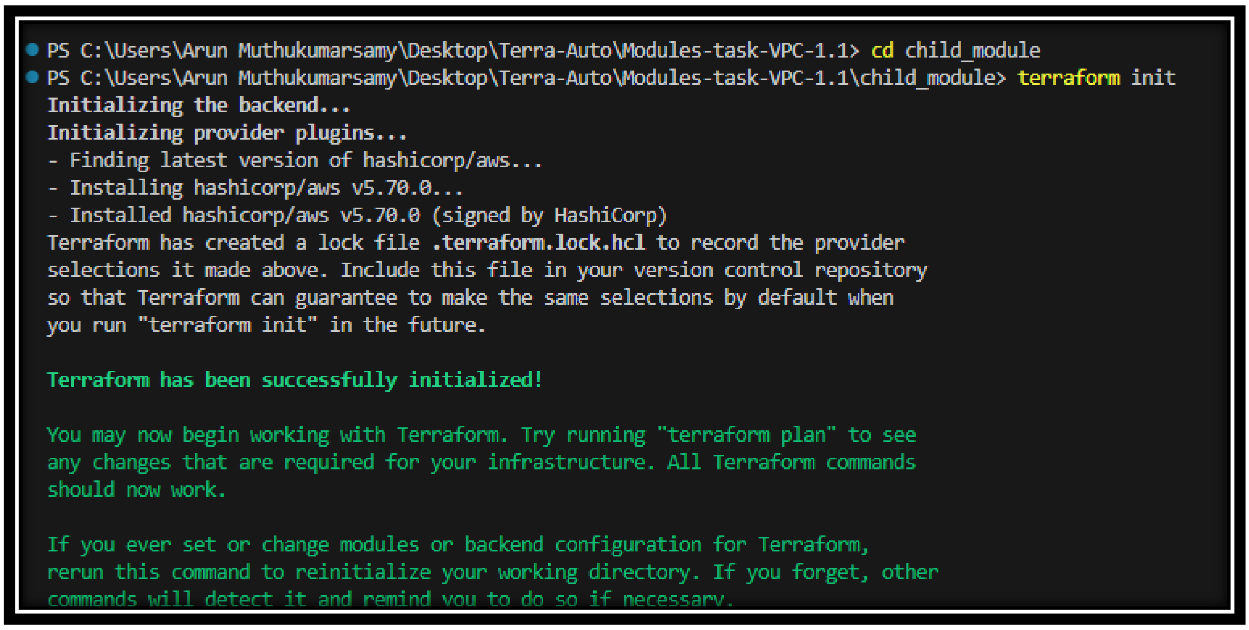
IMPLEMENTATION & VERIFICATION SCREENSHOT:



```
Modules-task-VPC-1.1
  child_module
    .terraform
    .terraform.lock.hcl
    main.tf
    terraform.tfstate
    terraform.tfvars
    variables.tf
  root_module
    .terraform
    main.tf
    variable.tf
  directory layout.png
```

- Directory allocation of child_module and root_module.

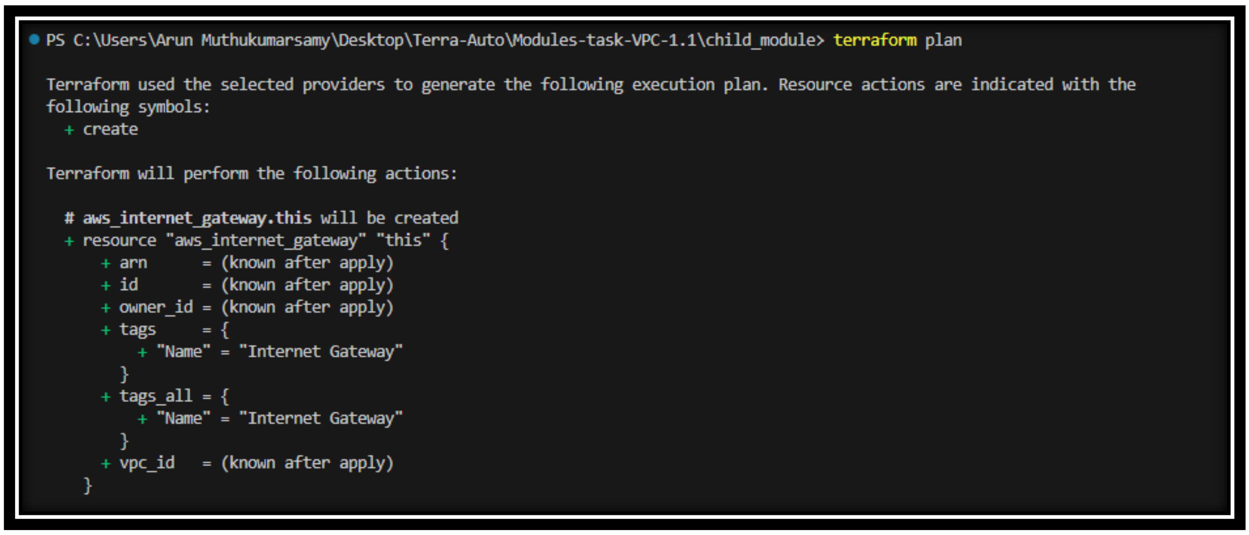
Initialize Terraform: Navigate to the child_module directory and run by using **terraform init.**



```
PS C:\Users\Arun Muthukumarsamy\Desktop\Terra-Auto\Modules-task-VPC-1.1> cd child_module
PS C:\Users\Arun Muthukumarsamy\Desktop\Terra-Auto\Modules-task-VPC-1.1\child_module> terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.70.0...
- Installed hashicorp/aws v5.70.0 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

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.
```

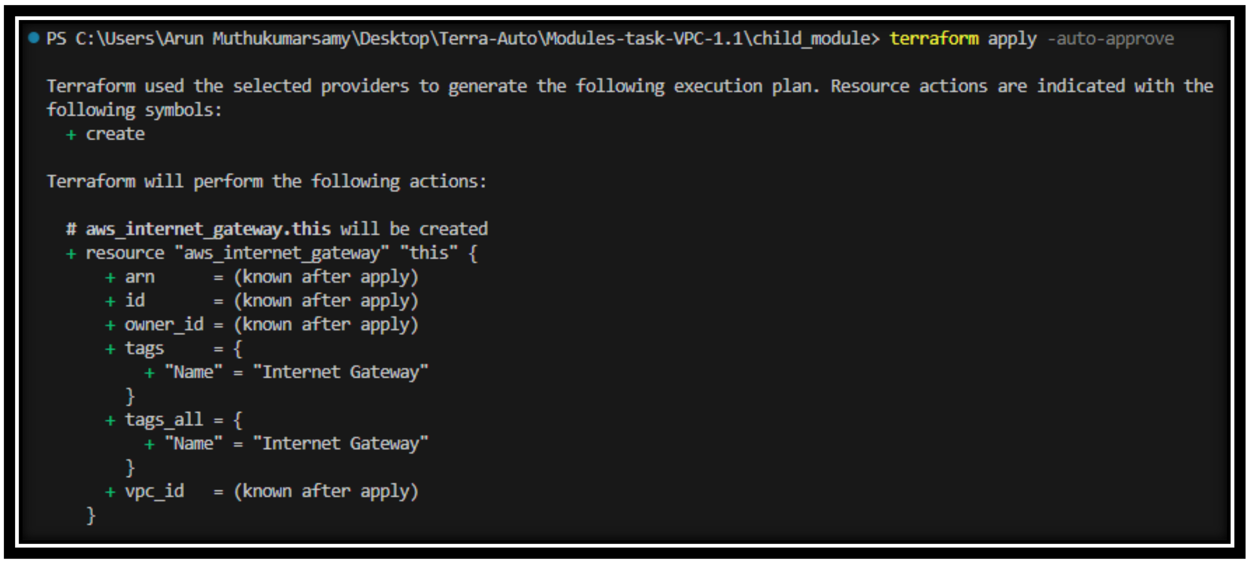
Modules-based Terraform resource creation using own modules

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



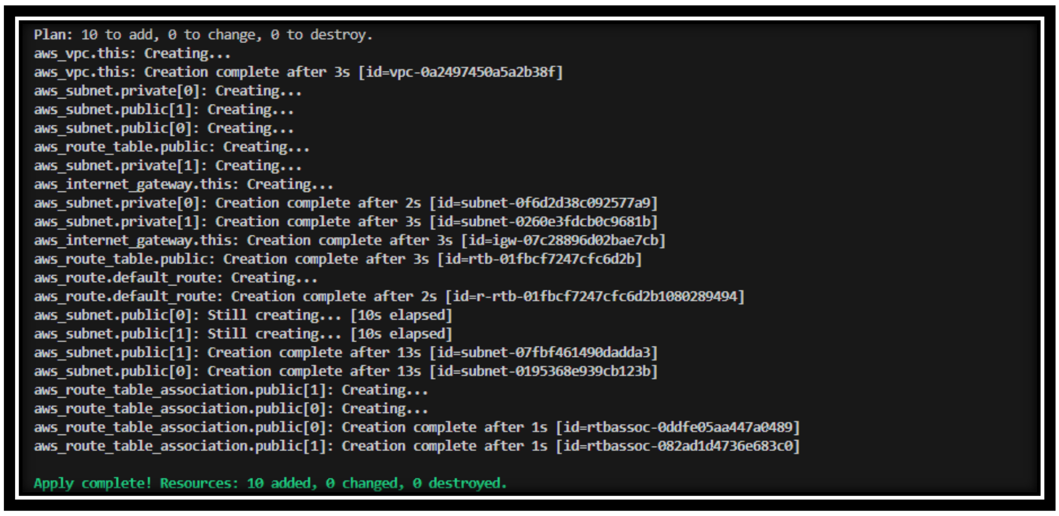
```
PS C:\Users\Arun Muthukumarsamy\Desktop\Terra-Auto\Modules-task-VPC-1.1\child_module> terraform plan

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_internet_gateway.this will be created
  + resource "aws_internet_gateway" "this" {
      + arn      = (known after apply)
      + id       = (known after apply)
      + owner_id = (known after apply)
      + tags     = {
          + "Name" = "Internet Gateway"
        }
      + tags_all = {
          + "Name" = "Internet Gateway"
        }
      + vpc_id   = (known after apply)
    }
```

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

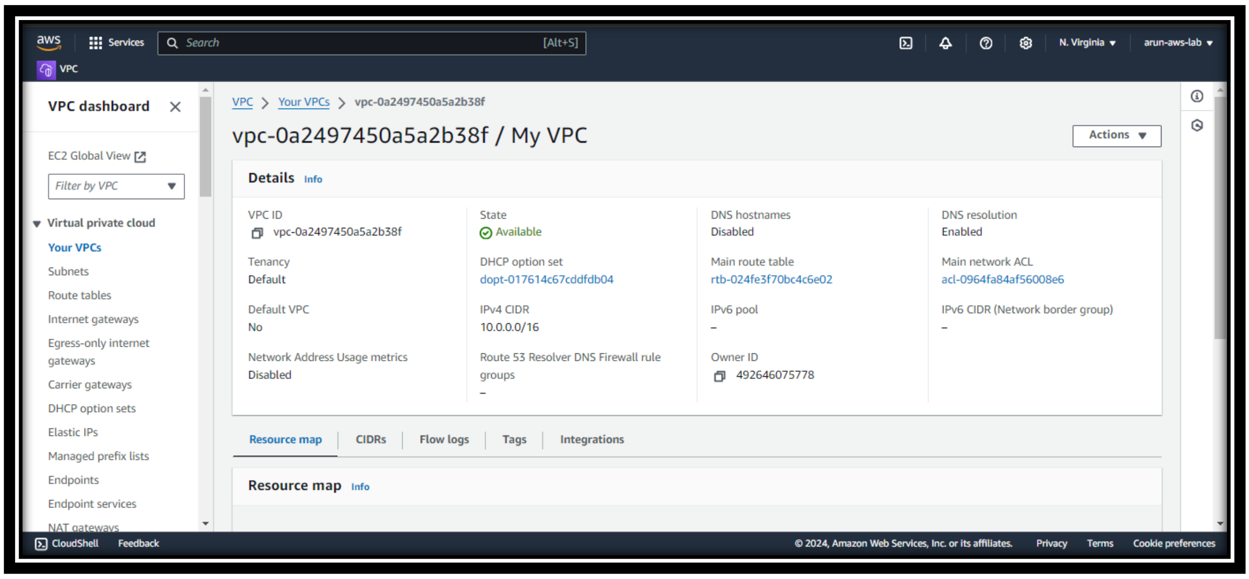


```
PS C:\Users\Arun Muthukumarsamy\Desktop\Terra-Auto\Modules-task-VPC-1.1\child_module> terraform apply -auto-approve

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_internet_gateway.this will be created
  + resource "aws_internet_gateway" "this" {
      + arn      = (known after apply)
      + id       = (known after apply)
      + owner_id = (known after apply)
      + tags     = {
          + "Name" = "Internet Gateway"
        }
      + tags_all = {
          + "Name" = "Internet Gateway"
        }
      + vpc_id   = (known after apply)
    }
```

Modules-based Terraform resource creation using own modules



```
Plan: 10 to add, 0 to change, 0 to destroy.
aws_vpc.this: Creating...
aws_vpc.this: Creation complete after 3s [id=vpc-0a2497450a5a2b38f]
aws_subnet.private[0]: Creating...
aws_subnet.public[1]: Creating...
aws_subnet.public[0]: Creating...
aws_route_table.public: Creating...
aws_subnet.private[1]: Creating...
aws_internet_gateway.this: Creating...
aws_subnet.private[0]: Creation complete after 2s [id=subnet-0f6d2d38c092577a9]
aws_subnet.private[1]: Creation complete after 3s [id=subnet-0260e3fdcb0c9681b]
aws_internet_gateway.this: Creation complete after 3s [id=igw-07c28896d02bae7cb]
aws_route_table.public: Creation complete after 3s [id=rtb-01fbcf7247cfc6d2b]
aws_route.default_route: Creating...
aws_route.default_route: Creation complete after 2s [id=r-rtb-01fbcf7247cfc6d2b1080289494]
aws_subnet.public[0]: Still creating... [10s elapsed]
aws_subnet.public[1]: Still creating... [10s elapsed]
aws_subnet.public[1]: Creation complete after 13s [id=subnet-07fbf461490dadda3]
aws_subnet.public[0]: Creation complete after 13s [id=subnet-0195368e939cb123b]
aws_route_table_association.public[1]: Creating...
aws_route_table_association.public[0]: Creating...
aws_route_table_association.public[0]: Creation complete after 1s [id=rtbassoc-0ddfe05aa447a0489]
aws_route_table_association.public[1]: Creation complete after 1s [id=rtbassoc-082ad1d4736e683c0]

Apply complete! Resources: 10 added, 0 changed, 0 destroyed.
```



vpc-0a2497450a5a2b38f / My VPC

Details Info

VPC ID
vpc-0a2497450a5a2b38f

State
Available

DNS hostnames
Disabled

DNS resolution
Enabled

Tenancy
Default

DHCP option set
dopt-017614c67cddfdb04

Main route table
rtb-024fe3f70bc4c6e02

Main network ACL
acl-0964fa84af56008e6

Default VPC
No

IPv4 CIDR
10.0.0.0/16

IPv6 pool
–

IPv6 CIDR (Network border group)
–

Network Address Usage metrics
Disabled

Route 53 Resolver DNS Firewall rule groups
–

Owner ID
492646075778

Resource map Info