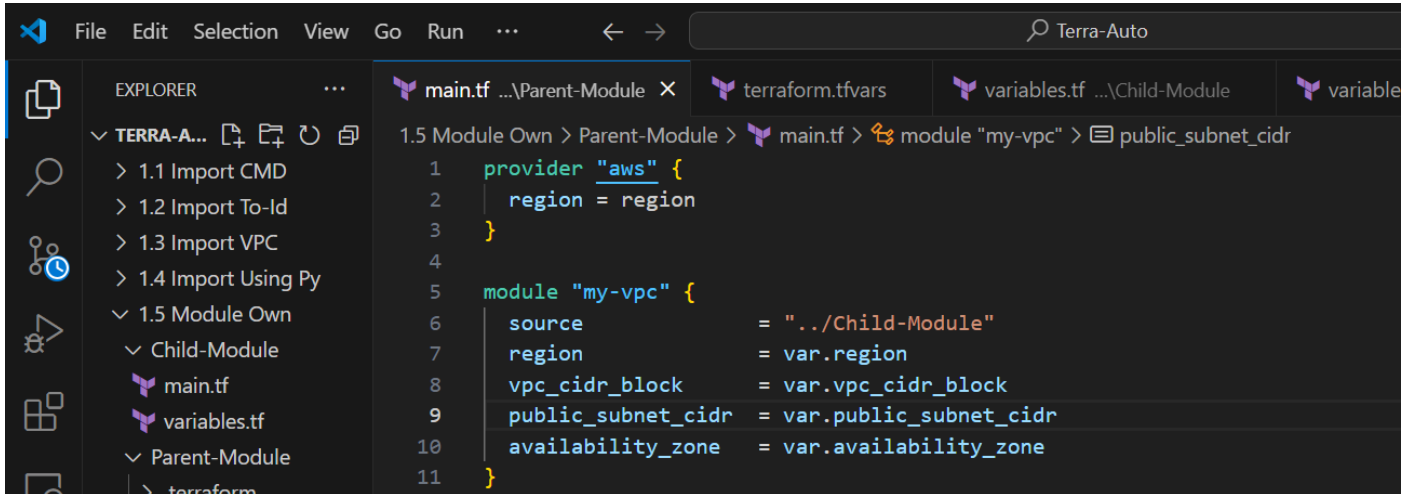


Terraform Own Module

Task 1.5 → Creating terraform own module

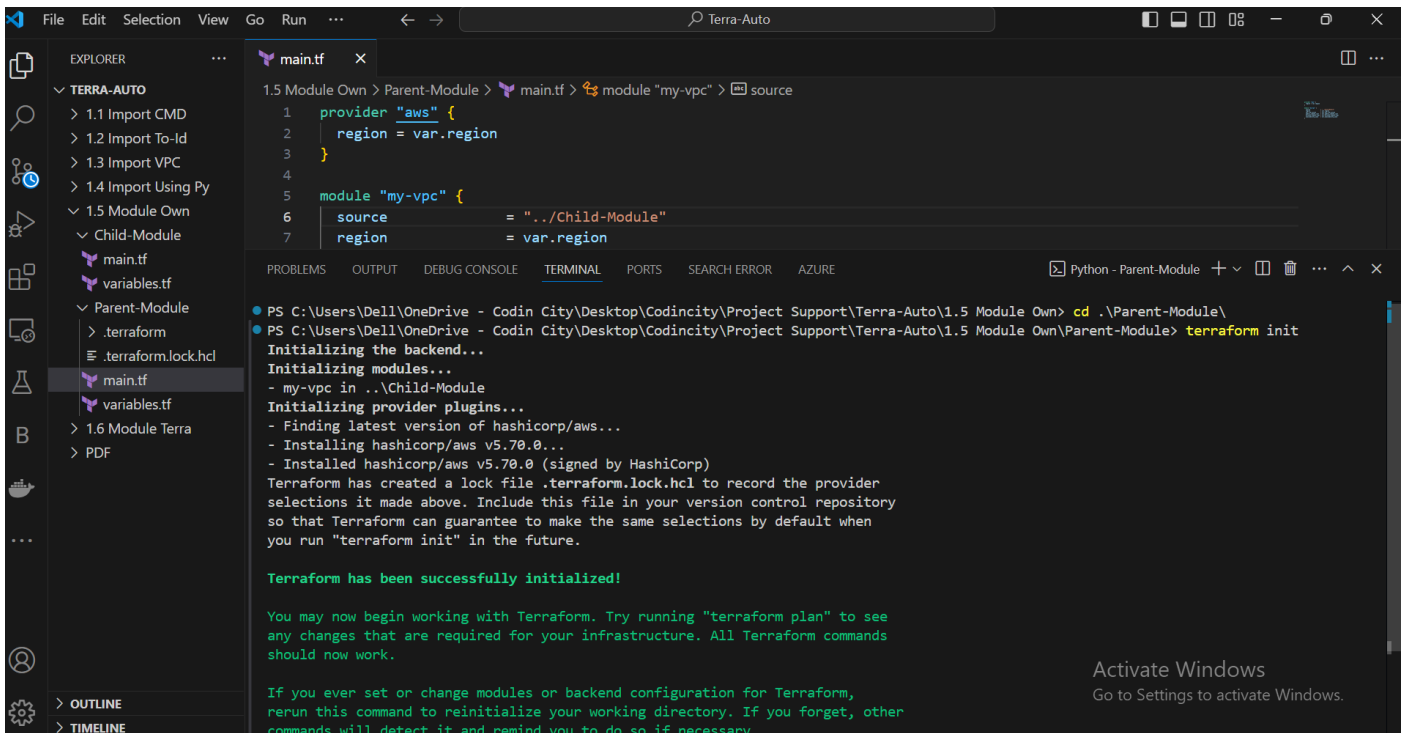
Step:1 I created two folders for two modules **Child-Module**, **Parent-Module**, Inside the module I created **main.tf** & **variables.tf**



The screenshot shows the Visual Studio Code interface with the Explorer view on the left and the main editor on the right. The Explorer view shows a project structure with a folder named 'TERRA-AUTO' containing subfolders 'Child-Module' and 'Parent-Module'. Inside 'Parent-Module', there are files 'main.tf' and 'variables.tf'. The main editor displays the content of 'main.tf' for the 'Parent-Module'.

```
1 provider "aws" {
2     region = region
3 }
4
5 module "my-vpc" {
6     source           = "../Child-Module"
7     region           = var.region
8     vpc_cidr_block   = var.vpc_cidr_block
9     public_subnet_cidr = var.public_subnet_cidr
10    availability_zone = var.availability_zone
11 }
```

Step:2 After creating module I will initializing the terraform file → **terraform init**



The screenshot shows the Visual Studio Code interface with the Explorer view on the left and the main editor on the right. The Explorer view shows the project structure with a folder named 'TERRA-AUTO' containing subfolders 'Child-Module' and 'Parent-Module'. Inside 'Parent-Module', there are files 'main.tf' and 'variables.tf'. The main editor displays the content of 'main.tf' for the 'Parent-Module'. The terminal view at the bottom shows the output of the 'terraform init' command.

```
PS C:\Users\Dell\OneDrive - Codin City\Desktop\Codincity\Project Support\Terra-Auto\1.5 Module Own> cd .\Parent-Module\
PS C:\Users\Dell\OneDrive - Codin City\Desktop\Codincity\Project Support\Terra-Auto\1.5 Module Own\Parent-Module> terraform init

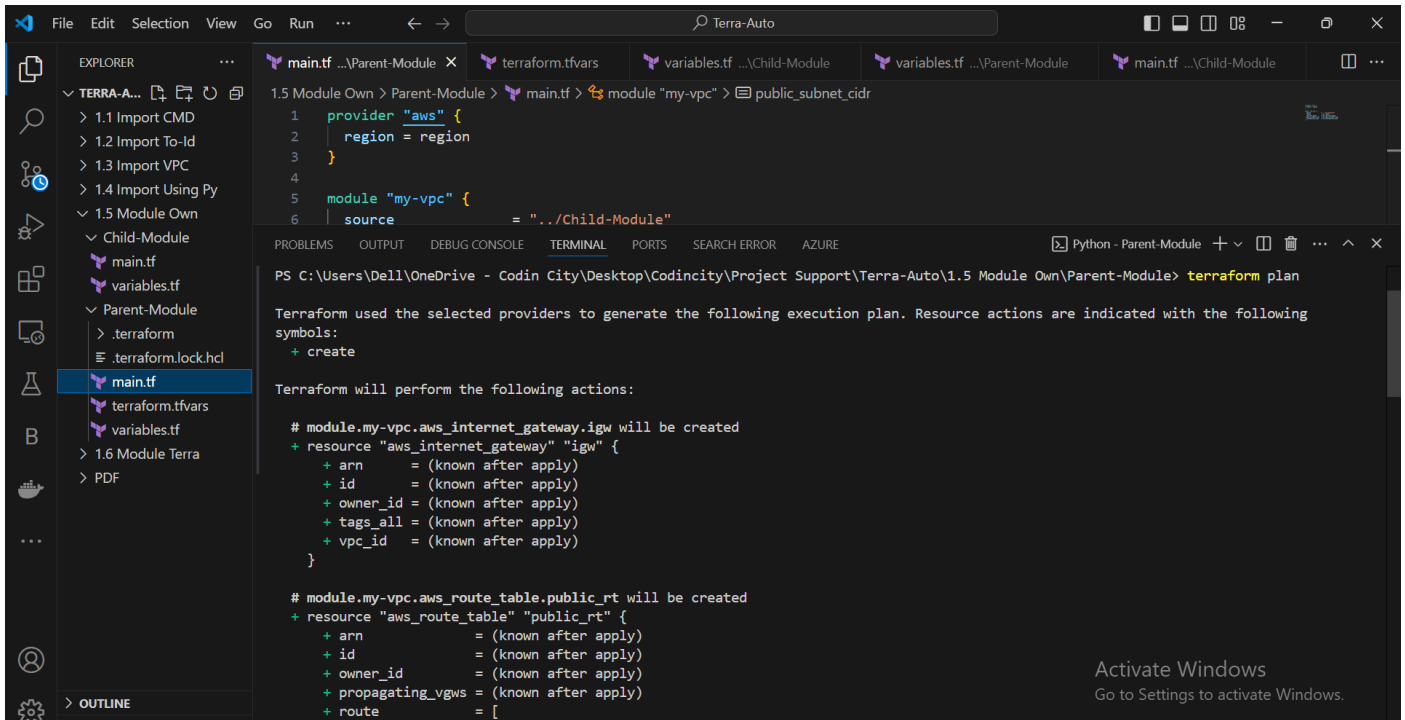
Initializing the backend...
Initializing modules...
- my-vpc in ../Child-Module
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.70.0...
- Installed hashicorp/aws v5.70.0 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl 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.
```

Step:3 → terraform plan



```
File Edit Selection View Go Run ... Terra-Auto
```

EXPLORER

- TERRA-A...
- > 1.1 Import CMD
- > 1.2 Import To-Id
- > 1.3 Import VPC
- > 1.4 Import Using Py
- > 1.5 Module Own
 - Child-Module
 - main.tf
 - variables.tf
 - Parent-Module
 - .terraform
 - terraform.lock.hcl
 - main.tf
 - terraform.tfvars
 - variables.tf
 - 1.6 Module Terra
 - PDF

1.5 Module Own > Parent-Module > main.tf > module "my-vpc" > public_subnet_cidr

```
1 provider "aws" {
2   region = region
3 }
4
5 module "my-vpc" {
6   source      = "../Child-Module"
7 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR AZURE

Python - Parent-Module + - - - - -

```
PS C:\Users\Dell\OneDrive - Codin City\Desktop\Codincity\Project Support\Terra-Auto\1.5 Module Own\Parent-Module> terraform plan

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

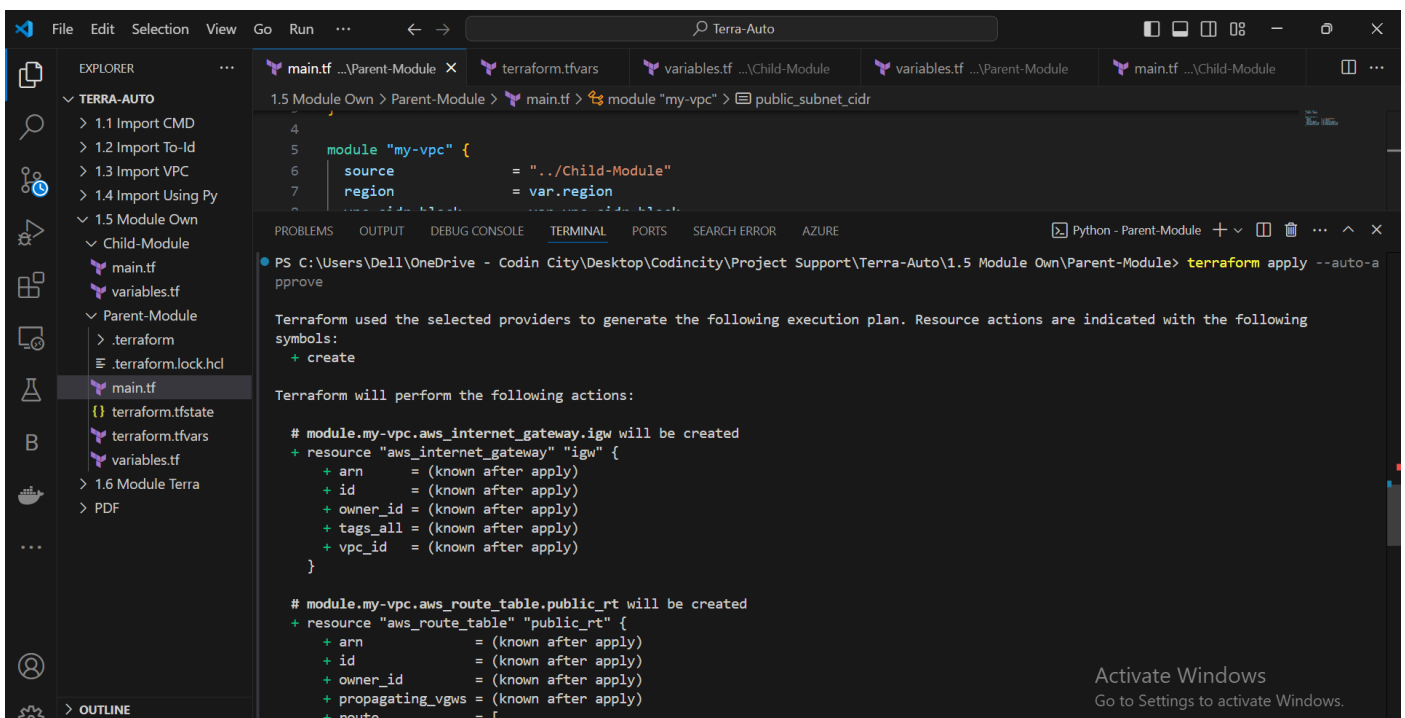
Terraform will perform the following actions:

# module.my-vpc.aws_internet_gateway.igw will be created
+ resource "aws_internet_gateway" "igw" {
+   arn          = (known after apply)
+   id           = (known after apply)
+   owner_id     = (known after apply)
+   tags_all     = (known after apply)
+   vpc_id       = (known after apply)
}

# module.my-vpc.aws_route_table.public_rt will be created
+ resource "aws_route_table" "public_rt" {
+   arn          = (known after apply)
+   id           = (known after apply)
+   owner_id     = (known after apply)
+   propagating_vgws = (known after apply)
+   route        = [
```

Activate Windows
Go to Settings to activate Windows.

Step:4 → terraform apply –auto-approve



```
File Edit Selection View Go Run ... Terra-Auto
```

EXPLORER

- TERRA-AUTO
- > 1.1 Import CMD
- > 1.2 Import To-Id
- > 1.3 Import VPC
- > 1.4 Import Using Py
- > 1.5 Module Own
 - Child-Module
 - main.tf
 - variables.tf
 - Parent-Module
 - .terraform
 - terraform.lock.hcl
 - main.tf
 - terraform.tfstate
 - terraform.tfvars
 - variables.tf
 - 1.6 Module Terra
 - PDF

1.5 Module Own > Parent-Module > main.tf > module "my-vpc" > public_subnet_cidr

```
4
5 module "my-vpc" {
6   source      = "../Child-Module"
7   region      = var.region
8 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR AZURE

Python - Parent-Module + - - - - -

```
PS C:\Users\Dell\OneDrive - Codin City\Desktop\Codincity\Project Support\Terra-Auto\1.5 Module Own\Parent-Module> terraform apply --auto-a
pprove

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:

# module.my-vpc.aws_internet_gateway.igw will be created
+ resource "aws_internet_gateway" "igw" {
+   arn          = (known after apply)
+   id           = (known after apply)
+   owner_id     = (known after apply)
+   tags_all     = (known after apply)
+   vpc_id       = (known after apply)
}

# module.my-vpc.aws_route_table.public_rt will be created
+ resource "aws_route_table" "public_rt" {
+   arn          = (known after apply)
+   id           = (known after apply)
+   owner_id     = (known after apply)
+   propagating_vgws = (known after apply)
+   route        = [
```

Activate Windows
Go to Settings to activate Windows.

Step:5 The VPC has been created

The screenshot shows the AWS VPC dashboard. On the left, the 'Virtual private cloud' section is expanded, showing 'Your VPCs'. The main panel displays 'Your VPCs (1/2)' with a table listing two VPCs:

Name	VPC ID	State	IPv4 CIDR	IPv6 CIDR	DHCP option set
-	vpc-06d8cfcadc14dae6a	Available	10.0.0.0/16	-	dopt-09cd5f3382f696b6f
Default_VPC	vpc-0ac3883de5bde45b6	Available	172.31.0.0/16	-	dopt-09cd5f3382f696b6f

The details for VPC 'vpc-06d8cfcadc14dae6a' are shown below:

- VPC ID:** vpc-06d8cfcadc14dae6a
- State:** Available
- Tenancy:** Default
- Default VPC:** No
- Network Address Usage metrics:** Disabled
- DHCP option set:** dopt-09cd5f3382f696b6f
- IPv4 CIDR:** 10.0.0.0/16
- Route 53 Resolver DNS Firewall rule groups:** -
- DNS hostnames:** Enabled
- Main route table:** rtb-08e0d4 added30445977
- IPv6 pool:** -
- DNS resolution:** Enabled
- Main network ACL:** acl-060bee1239ff633
- IPv6 CIDR (Network border group):** -
- Owner ID:** 339713187727

Step:6 Created one public subnet

The screenshot shows the AWS Subnets dashboard. On the left, the 'Virtual private cloud' section is expanded, showing 'Subnets'. The main panel displays 'Subnets (1/7)' with a table listing two subnets:

Name	Subnet ID	State	VPC	IPv4 CIDR
-	subnet-0280a9360ea9e80e5	Available	vpc-06d8cfcadc14dae6a	10.0.1.0/24
Sub_us-east-1a	subnet-037ca5150d86bfa9c	Available	vpc-0ac3883de5bde45b6 Default_VPC	172.31.80.0/20

The details for subnet 'subnet-0280a9360ea9e80e5' are shown below:

- Subnet ID:** subnet-0280a9360ea9e80e5
- Subnet ARN:** arn:aws:ec2:us-east-1:339713187727:subnet/subnet-0280a9360ea9e80e5
- State:** Available
- Available IPv4 addresses:** 251
- Availability Zone ID:** use1-az2
- Network ACL:** acl-060bee1239ff633
- IPv6 CIDR:** -
- Network border group:** us-east-1
- IPv4 CIDR:** 10.0.1.0/24
- IPv6 CIDR association ID:** -
- VPC:** vpc-06d8cfcadc14dae6a
- Route table:** rtb-07ab397ec31dd6578
- Auto-assign public IPv4 address:** No
- Auto-assign IPv6 address:** No

Step:7 Route table has been created & associated with corresponding subnet

The screenshot shows the AWS Route Tables dashboard. On the left, the 'Virtual private cloud' section is expanded, showing 'Route tables'. The main panel displays 'Route tables (1/3)' with a table listing two route tables:

Name	Route table ID	Explicit subnet associations	Edge associations	Main	VPC
-	rtb-07ab397ec31dd6578	subnet-0280a9360ea9e80e5	-	No	vpc-06d8cfcadc14dae6a
-	rtb-08e0d4 added30445977	-	-	Yes	vpc-06d8cfcadc14dae6a

The details for route table 'rtb-07ab397ec31dd6578' are shown below:

- Route table ID:** rtb-07ab397ec31dd6578
- Main:** No
- VPC:** vpc-06d8cfcadc14dae6a
- Owner ID:** 339713187727
- Explicit subnet associations:** subnet-0280a9360ea9e80e5
- Edge associations:** -

Step:8 Internet Gateway has been created & associated with public subnet

The screenshot shows the AWS Management Console interface for Internet Gateways. The left sidebar displays the 'VPC dashboard' with a search bar and a list of services including EC2 Global View, Virtual private cloud, Subnets, Route tables, Internet gateways, Egress-only internet gateways, Carrier gateways, DHCP option sets, and Elastic IPs. The main content area is titled 'Internet gateways (1/2)' and features a table with columns: Name, Internet gateway ID, State, VPC ID, and Owner. Two gateways are listed: 'igw-016c2689b2e9f24da' (Attached, VPC ID: vpc-06d8cfcadc14dae6a) and 'igw-0759257f25ef27e69' (Attached, VPC ID: vpc-0ac3883de5bde45b6 | Default_VPC). Below the table, the details for 'igw-016c2689b2e9f24da' are shown, including its ID, state (Attached), VPC ID (vpc-06d8cfcadc14dae6a), and owner (339713187727).

Step:9 Default Security Group has been created

The screenshot shows the AWS Management Console interface for Security Groups. The left sidebar displays the 'Security' section with a search bar and a list of services including Network ACLs, Security groups, DNS firewall, Rule groups, Domain lists, Network Firewall, Firewalls, Firewall policies, Network Firewall rule groups, TLS inspection configurations, Network Firewall resource groups, and Virtual private network. The main content area is titled 'Security Groups (1/2)' and features a table with columns: Name, Security group ID, Security group name, VPC ID, and Description. One security group is listed: 'sg-0e511363d302badb9' (default, VPC ID: vpc-06d8cfcadc14dae6a, Description: default VPC security group). Below the table, the details for 'sg-0e511363d302badb9 - default' are shown, including its name, ID, description, VPC ID, owner, inbound rules count (1 Permission entry), and outbound rules count (1 Permission entry).

Step:10 This is the flow of using modules

The screenshot shows a file explorer view of a Terraform project structure. The root directory is '1.5 Module Own'. It contains two subdirectories: 'Child-Module' and 'Parent-Module', both highlighted with red boxes. The 'Child-Module' directory contains files 'main.tf' and 'variables.tf'. The 'Parent-Module' directory contains a subdirectory '.terraform' (which includes 'modules' and 'providers'), a file '.terraform.lock.hcl', and files 'main.tf', 'terraform.tfstate', 'terraform.tfvars', and 'variables.tf'.