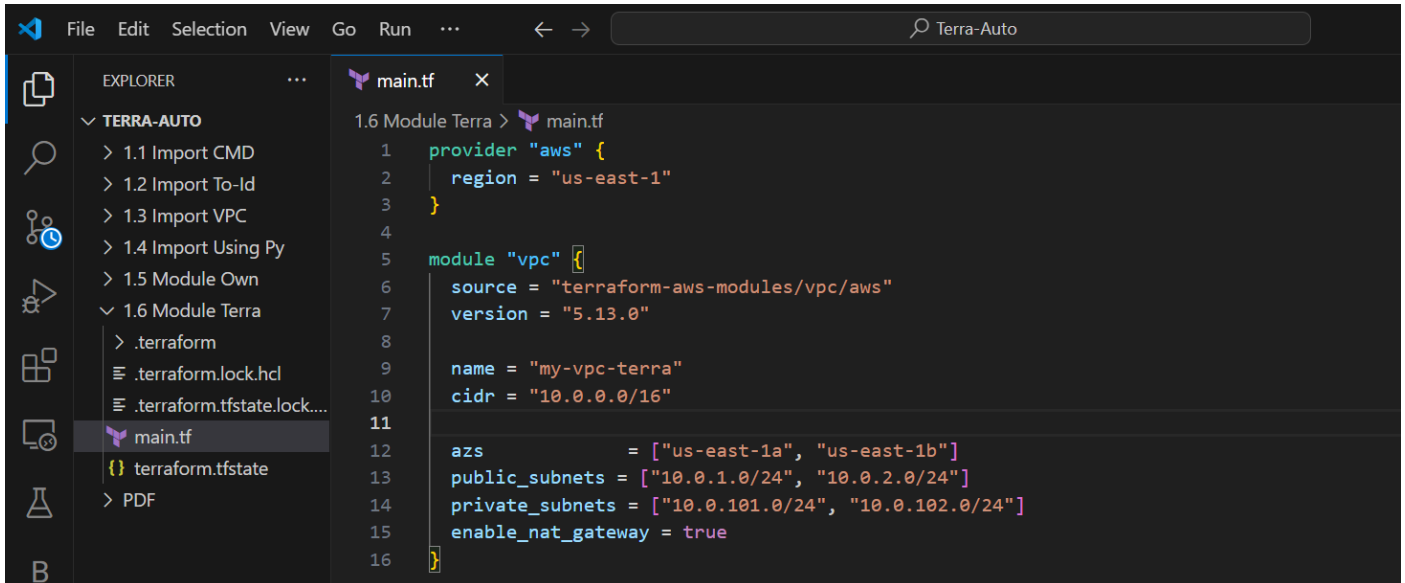


Terraform Module

Task 1.6 → Using terraform module

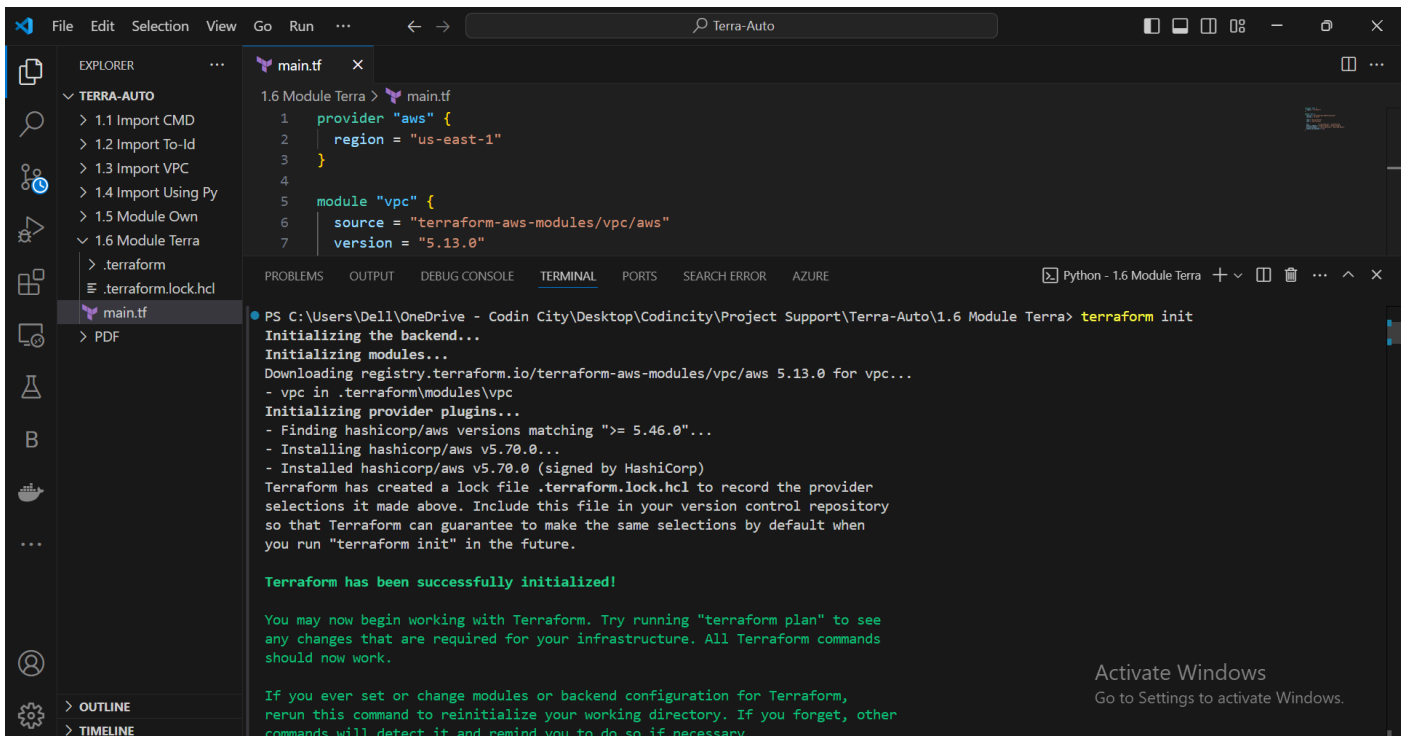
Step:1 I created **main.tf**, Inside it I defined provider & module for creating VPC



The screenshot shows the Visual Studio Code editor with the **main.tf** file open. The Explorer sidebar on the left shows the project structure under **TERRA-AUTO**, including files like **1.1 Import CMD**, **1.2 Import To-Id**, **1.3 Import VPC**, **1.4 Import Using Py**, **1.5 Module Own**, and **1.6 Module Terra**. The **main.tf** file is selected, and its content is displayed in the editor. The code defines an AWS provider and a VPC module.

```
1 provider "aws" {
2   region = "us-east-1"
3 }
4
5 module "vpc" {
6   source = "terraform-aws-modules/vpc/aws"
7   version = "5.13.0"
8
9   name = "my-vpc-terra"
10  cidr = "10.0.0.0/16"
11
12  azs          = ["us-east-1a", "us-east-1b"]
13  public_subnets = ["10.0.1.0/24", "10.0.2.0/24"]
14  private_subnets = ["10.0.101.0/24", "10.0.102.0/24"]
15  enable_nat_gateway = true
16 }
```

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



The screenshot shows the Visual Studio Code editor with the **main.tf** file open. The terminal window at the bottom displays the output of the **terraform init** command. The output shows the initialization of the backend, modules, and provider plugins. The terminal output is as follows:

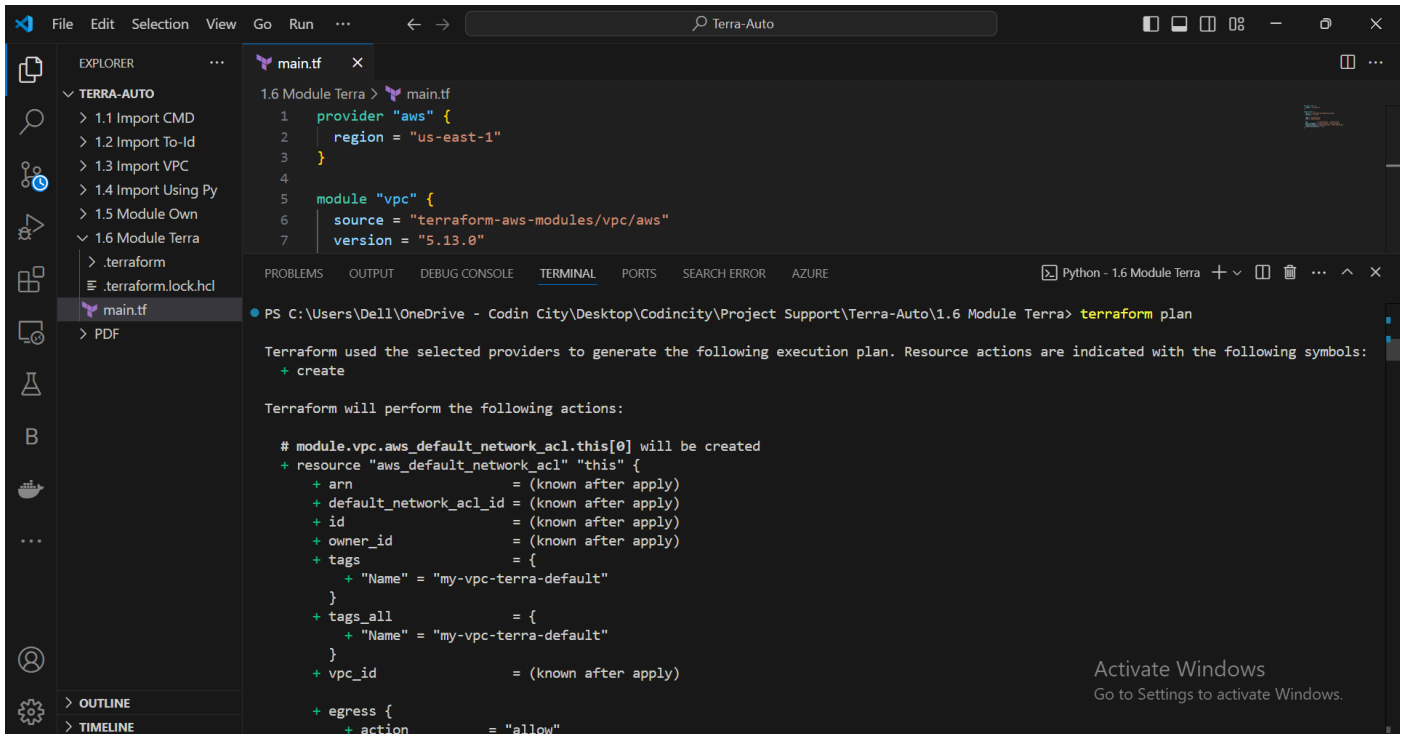
```
PS C:\Users\Dell\OneDrive - Codin City\Desktop\Codincity\Project Support\Terra-Auto\1.6 Module Terra> terraform init
Initializing the backend...
Initializing modules...
Downloading registry.terraform.io/terraform-aws-modules/vpc/aws 5.13.0 for vpc...
- vpc in .terraform\modules\vpc
Initializing provider plugins...
- Finding hashicorp/aws versions matching ">= 5.46.0"...
- 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



The screenshot shows the Visual Studio Code interface with the Explorer sidebar on the left. The Explorer sidebar contains a folder named 'TERRA-AUTO' with subfolders '1.1 Import CMD', '1.2 Import To-Id', '1.3 Import VPC', '1.4 Import Using Py', '1.5 Module Own', and '1.6 Module Terra'. The '1.6 Module Terra' folder is expanded, showing files '.terraform', '.terraform.lock.hcl', 'main.tf', and 'PDF'. The 'main.tf' file is selected and its content is displayed in the editor. The content of 'main.tf' is as follows:

```
1.6 Module Terra > main.tf
1 provider "aws" {
2   region = "us-east-1"
3 }
4
5 module "vpc" {
6   source = "terraform-aws-modules/vpc/aws"
7   version = "5.13.0"
8 }
```

The terminal window at the bottom shows the output of the 'terraform plan' command. The output indicates that Terraform will create a resource 'aws_default_network_acl' and an egress rule. The output is as follows:

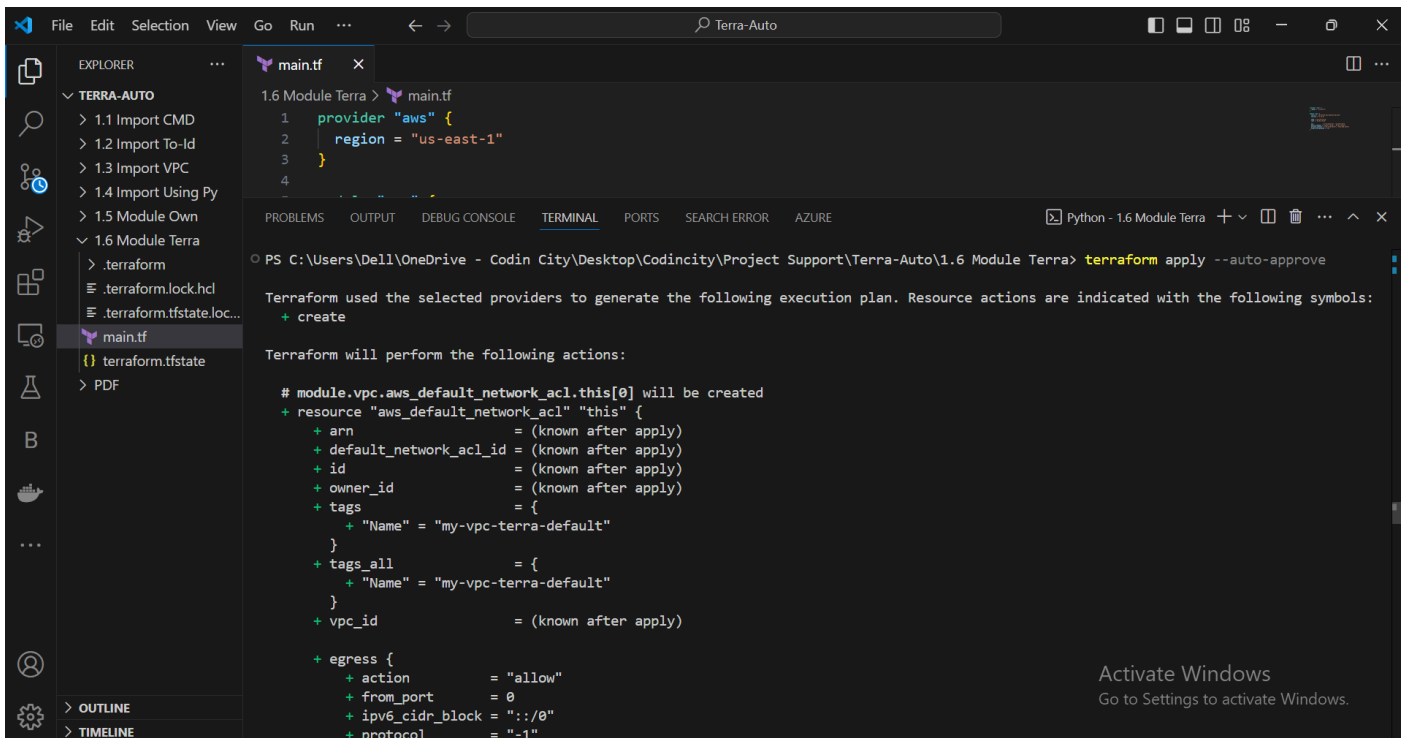
```
PS C:\Users\Dell\OneDrive - Codin City\Desktop\Codincity\Project Support\Terra-Auto\1.6 Module Terra> terraform plan

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.vpc.aws_default_network_acl.this[0] will be created
+ resource "aws_default_network_acl" "this" {
+   arn = (known after apply)
+   default_network_acl_id = (known after apply)
+   id = (known after apply)
+   owner_id = (known after apply)
+   tags = {
+     "Name" = "my-vpc-terra-default"
+   }
+   tags_all = {
+     "Name" = "my-vpc-terra-default"
+   }
+   vpc_id = (known after apply)
+   egress {
+     action = "allow"
  }
```

Step:4 → terraform apply --auto-approve



The screenshot shows the Visual Studio Code interface with the Explorer sidebar on the left. The Explorer sidebar contains a folder named 'TERRA-AUTO' with subfolders '1.1 Import CMD', '1.2 Import To-Id', '1.3 Import VPC', '1.4 Import Using Py', '1.5 Module Own', and '1.6 Module Terra'. The '1.6 Module Terra' folder is expanded, showing files '.terraform', '.terraform.lock.hcl', 'main.tf', and 'PDF'. The 'main.tf' file is selected and its content is displayed in the editor. The content of 'main.tf' is as follows:

```
1.6 Module Terra > main.tf
1 provider "aws" {
2   region = "us-east-1"
3 }
4
```

The terminal window at the bottom shows the output of the 'terraform apply --auto-approve' command. The output indicates that Terraform will create a resource 'aws_default_network_acl' and an egress rule. The output is as follows:

```
PS C:\Users\Dell\OneDrive - Codin City\Desktop\Codincity\Project Support\Terra-Auto\1.6 Module Terra> 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:

# module.vpc.aws_default_network_acl.this[0] will be created
+ resource "aws_default_network_acl" "this" {
+   arn = (known after apply)
+   default_network_acl_id = (known after apply)
+   id = (known after apply)
+   owner_id = (known after apply)
+   tags = {
+     "Name" = "my-vpc-terra-default"
+   }
+   tags_all = {
+     "Name" = "my-vpc-terra-default"
+   }
+   vpc_id = (known after apply)
+   egress {
+     action = "allow"
+     from_port = 0
+     ipv6_cidr_block = ":::/0"
+     protocol = "-1"
  }
```

Step:5 The VPC has been created named as my-vpc-terra

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 VPCs. The VPC 'my-vpc-terra' (vpc-098f9e048772666e4) is selected, and its details are shown below.

Name	VPC ID	State	IPv4 CIDR	IPv6 CIDR	DHCP option set
my-vpc-terra	vpc-098f9e048772666e4	Available	10.0.0.0/16	-	dopt-09cd5f3382
Default_VPC	vpc-0ac3883de5bde45b6	Available	172.31.0.0/16	-	dopt-09cd5f3382

vpc-098f9e048772666e4 / my-vpc-terra

Details

VPC ID	State	DNS hostnames	DNS resolution
vpc-098f9e048772666e4	Available	Enabled	Enabled
Tenancy	DHCP option set	Main route table	Main network ACL
Default	dopt-09cd5f3382	rtb-04ce098fde192b8ed	acl-00a623eeef38bba8
Default VPC	IPv4 CIDR	IPv6 pool	IPv6 CIDR (Network border group)
No	10.0.0.0/16	-	-
Network Address Usage metrics	Route 53 Resolver DNS Firewall rule groups	Owner ID	
Disabled	-	339713187727	

Step:6 Created two public subnets & two private subnets

The screenshot shows the AWS Subnets dashboard. On the left, the 'Virtual private cloud' section is expanded, showing 'Subnets'. The main panel displays 'Subnets (10/10)' with a table listing subnets. Four subnets are selected, all associated with the 'my-vpc-terra' VPC.

Name	Subnet ID	State	VPC	IPv4 CIDR
my-vpc-terra-private-us-east-1a	subnet-0664a4b892d462102	Available	vpc-098f9e048772666e4 my-vpc-terra	10.0.101.0/24
my-vpc-terra-private-us-east-1b	subnet-0d7c6bfa498623225	Available	vpc-098f9e048772666e4 my-vpc-terra	10.0.102.0/24
my-vpc-terra-public-us-east-1a	subnet-067a8610591eae23e	Available	vpc-098f9e048772666e4 my-vpc-terra	10.0.1.0/24
my-vpc-terra-public-us-east-1b	subnet-03addbd620ed0125e	Available	vpc-098f9e048772666e4 my-vpc-terra	10.0.2.0/24

Subnets: subnet-0664a4b892d462102, subnet-0d7c6bfa498623225, subnet-067a8610591eae23e, subnet-03addbd620ed0125e, subnet-037ca5150d86bfa9c, subnet-0b8c870cd8cf2b6e8, subnet-025fcd0dc02a182c1, subnet-01a1b0471903cf7ca, subnet-0d5170b8337358f5c, subnet-05716b00f5ce2ae37

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

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 (4/5)' with a table listing route tables. Four route tables are selected, all associated with the 'my-vpc-terra' VPC.

Name	Route table ID	Explicit subnet associations	Edge associations	Main	VPC
my-vpc-terra-public	rtb-0b0440f65af9c8c22	2 subnets	-	No	vpc-098f9e048772666e4 my-vpc-terra
my-vpc-terra-private-us-east-1b	rtb-0d7582d2a91dae71d	subnet-0d7c6bfa498623225 / ...	-	No	vpc-098f9e048772666e4 my-vpc-terra
my-vpc-terra-private-us-east-1a	rtb-0188bccc372128bd	subnet-0664a4b892d462102 / ...	-	No	vpc-098f9e048772666e4 my-vpc-terra
my-vpc-terra-default	rtb-04ce098fde192b8ed	-	-	Yes	vpc-098f9e048772666e4 my-vpc-terra

Route tables: rtb-0b0440f65af9c8c22, rtb-0d7582d2a91dae71d, rtb-0188bccc372128bd, rtb-04ce098fde192b8ed

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

The screenshot shows the AWS Internet Gateways dashboard. On the left, the 'Virtual private cloud' section is expanded, showing 'Internet gateways'. The main panel displays 'Internet gateways (1/2)' with a table listing internet gateways. One internet gateway is selected, associated with the 'my-vpc-terra' VPC.

Name	Internet gateway ID	State	VPC ID	Owner
my-vpc-terra	igw-0cc08f5b08f8b9aa8	Attached	vpc-098f9e048772666e4 my-vpc-terra	339713187727
-	igw-0759257f25ef27e69	Attached	vpc-0ac3883de5bde45b6 Default_VPC	339713187727

igw-0cc08f5b08f8b9aa8 / my-vpc-terra

Details

Internet gateway ID	State	VPC ID	Owner
igw-0cc08f5b08f8b9aa8	Attached	vpc-098f9e048772666e4 my-vpc-terra	339713187727

Step:9 NAT Gateway has been created in public subnets & Associated with Private subnets

The screenshot shows the AWS VPC console with the 'NAT gateways' page selected. A table lists two NAT gateways: 'my-vpc-terra-us-east-1a' and 'my-vpc-terra-us-east-1b'. The first gateway is selected, and its details are shown below. The details include the NAT gateway ID, connectivity type (Public), state (Available), and primary public and private IP addresses.

Name	NAT gateway ID	Connectivity...	State	State message	Primary public I...	Primary private I...
my-vpc-terra-us-east-1a	nat-069f8a53446203b1c	Public	Available	-	107.20.150.152	10.0.1.165
my-vpc-terra-us-east-1b	nat-03ee0eb0a2e9983c8	Public	Available	-	50.16.24.7	10.0.2.205

Details for nat-069f8a53446203b1c / my-vpc-terra-us-east-1a

Details			
NAT gateway ID	Connectivity type	State	State message
nat-069f8a53446203b1c	Public	Available	-
NAT gateway ARN	Primary public IPv4 address	Primary private IPv4 address	Primary network interface ID
arn:aws:ec2:us-east-1:339713187727:natgateway/nat-	107.20.150.152	10.0.1.165	eni-0e6335a062a1f202f

Step:10 Default Security Group has been created

The screenshot shows the AWS VPC console with the 'Security Groups' page selected. A table lists one security group: 'my-vpc-terra-default'. The security group is selected, and its details are shown below. The details include the security group ID, name (default), VPC ID, and description.

Name	Security group ID	Security group name	VPC ID	Description
my-vpc-terra-default	sg-07e182dddc017f13	default	vpc-098f9e048772666e4	default VPC security group

sg-07e182dddc017f13 - default

Details			
Security group name	Security group ID	Description	VPC ID
default	sg-07e182dddc017f13	default VPC security group	vpc-098f9e048772666e4

Step:11 This is the flow of using modules

```
✓ 1.6 Module Terra
  ✓ .terraform
    ✓ modules
      > vpc
      {} modules.json
  ✓ providers \ registry.terraform.io \ hashicorp \ aws \ 5.70.0 \ windows_amd64
    LICENSE.txt
    terraform-provider-aws_v5.70.0_x5.exe
    .terraform.lock.hcl
    main.tf
    {} terraform.tfstate
```