# Terraform Module for Creation & Import Using Py

## Task 1.8 → Terraform import & creating VPC using Python

**Step:1** I created two folders for two modules **Child-Module, Parent-Module,** Inside the module I created **main.tf, variables.tf, terraform.tfvars, imp.py** file

imp.py | import.tf

Child_Module > imp.py > ...

```python
import os
import subprocess
import boto3
import json

def get_module_path():
    return os.path.dirname(os.path.abspath(__file__))

def run_terraform_command(command, cwd):
    try:
        result = subprocess.run(command, cwd=cwd, check=True, capture_output=True, text=True)
        print(result.stdout)
    except subprocess.CalledProcessError as e:
        print(f"Error executing Terraform command: {e}")
        print(f"Stderr: {e.stderr}")
        raise

def create_vpc_resources(module_path):
    print("Creating new VPC and associated resources...")
    run_terraform_command(["terraform", "init"], module_path)
    run_terraform_command(["terraform", "apply", "-auto-approve"], module_path)
    print("New VPC and associated resources created successfully.")

def import_existing_vpc(module_path, vpc_id):
    print(f"Importing existing VPC {vpc_id}...")
    import_command = ["terraform", "import", f"aws_vpc.imported_vpc", vpc_id]
    run_terraform_command(import_command, module_path)
    print(f"Existing VPC {vpc_id} imported successfully.")
```



**Step:2** After creating module, Python file I run it → **python imp.py**

PROBLEMS 13 OUTPUT DEBUG CONSOLE TERMINAL PORTS AZURE

```
ld_Module> python imp.py
Creating new VPC and associated resources...
Initializing the backend...
Initializing modules...
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,
```

**Step:3** It automatically run the process init, plan, apply

```
module.vpc.aws_internet_gateway.my_igw["igw"]: Creation complete after 2s [id=igw-028f78dd16dc4eca7]
module.vpc.aws_subnet.my_subnet["sub_pri"]: Creation complete after 2s [id=subnet-02a2656d752ce7dbb]
module.vpc.aws_route_table.my_route_table["rt_pub"]: Creation complete after 2s [id=rtb-0b540f9792c09608d]
module.vpc.aws_route.my_route["rt_pub"]: Creating...
module.vpc.aws_route.my_route["rt_pub"]: Creation complete after 2s [id=r-rtb-0b540f9792c09608d1080289494]
module.vpc.aws_security_group.my_sg["sg"]: Creation complete after 4s [id=sg-075bdd9d27a616a4f]
module.vpc.aws_subnet.my_subnet["sub_pub"]: Still creating... [10s elapsed]
module.vpc.aws_subnet.my_subnet["sub_pub"]: Creation complete after 12s [id=subnet-024eac73baab8691d]
module.vpc.aws_nat_gateway.my_nat_gw["nat"]: Creating...
module.vpc.aws_nat_gateway.my_nat_gw["nat"]: Still creating... [10s elapsed]
module.vpc.aws_nat_gateway.my_nat_gw["nat"]: Still creating... [20s elapsed]
module.vpc.aws_nat_gateway.my_nat_gw["nat"]: Still creating... [30s elapsed]
module.vpc.aws_nat_gateway.my_nat_gw["nat"]: Still creating... [40s elapsed]
module.vpc.aws_nat_gateway.my_nat_gw["nat"]: Still creating... [50s elapsed]
module.vpc.aws_nat_gateway.my_nat_gw["nat"]: Still creating... [1m0s elapsed]
module.vpc.aws_nat_gateway.my_nat_gw["nat"]: Still creating... [1m10s elapsed]
module.vpc.aws_nat_gateway.my_nat_gw["nat"]: Still creating... [1m20s elapsed]
module.vpc.aws_nat_gateway.my_nat_gw["nat"]: Still creating... [1m31s elapsed]
module.vpc.aws_nat_gateway.my_nat_gw["nat"]: Creation complete after 1m39s [id=nat-0c2389b232c4c38e8]

Apply complete! Resources: 9 added, 0 changed, 0 destroyed.
```



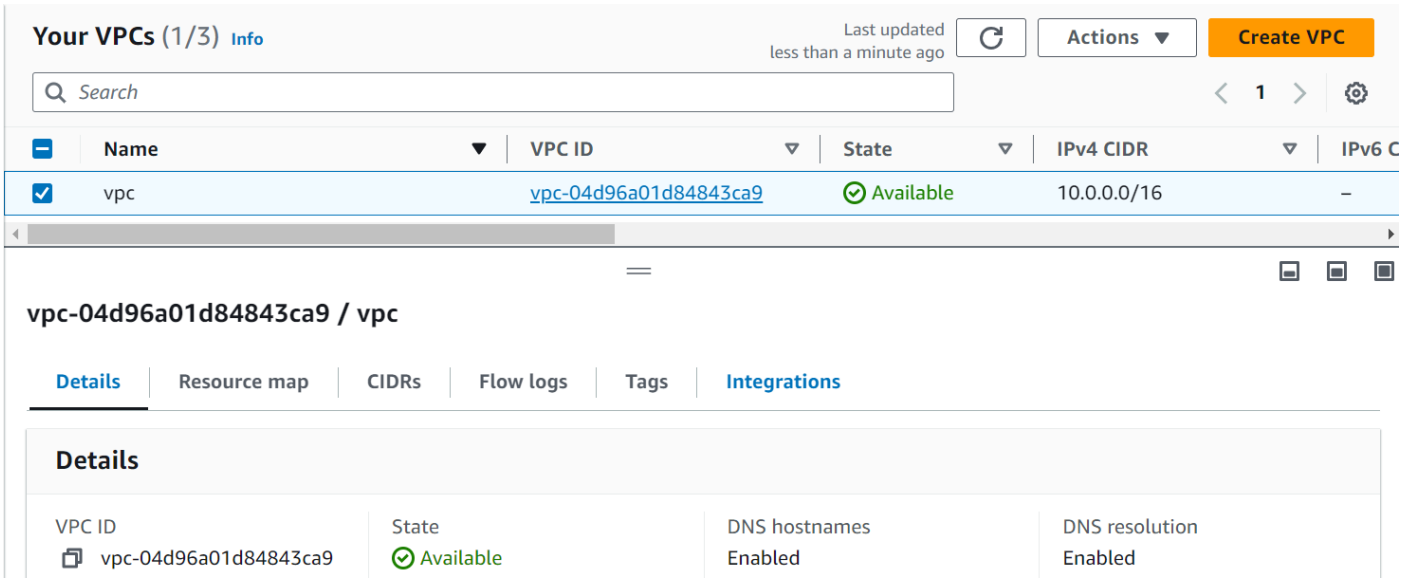
**Step:4** After that it will goes to the import function

```
New VPC and associated resources created successfully.

Existing VPCs:
1. VPC ID: vpc-04f15e6f778a69fe3, CIDR: 192.0.0.0/24, Tags: Name:test_vpc
2. VPC ID: vpc-0ac3883de5bde45b6, CIDR: 172.31.0.0/16, Tags: Name:Default_VPC
3. VPC ID: vpc-04d96a01d84843ca9, CIDR: 10.0.0.0/16, Tags: Environment:production, Name:vpc

Enter the number of the VPC you want to import: 1
Created/Updated import.tf with imported VPC vpc-04f15e6f778a69fe3 configuration
Importing existing VPC vpc-04f15e6f778a69fe3...
aws_vpc.imported_vpc: Importing from ID "vpc-04f15e6f778a69fe3"...
aws_vpc.imported_vpc: Import prepared!
  Prepared aws_vpc for import
aws_vpc.imported_vpc: Refreshing state... [id=vpc-04f15e6f778a69fe3]

Import successful!

The resources that were imported are shown above. These resources are now in
your Terraform state and will henceforth be managed by Terraform.
```

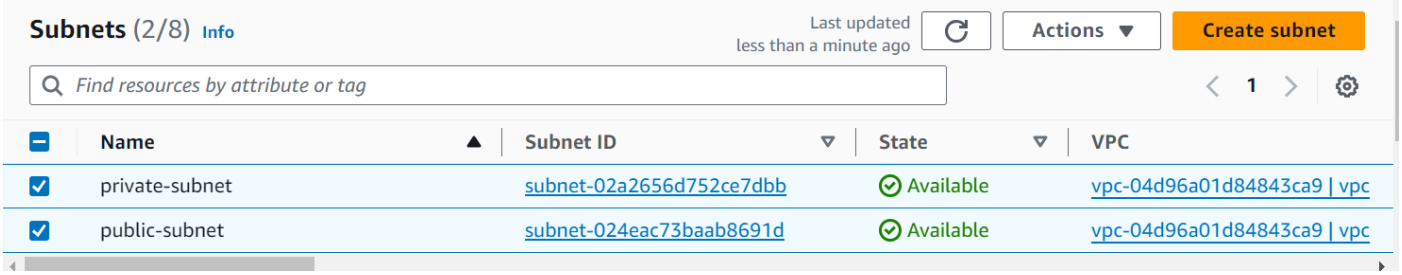
**Step:5** It will be fetch the state file and give the result like there will be **No Changes**

```
module.vpc.aws_eip.nat_eip["nat"]: Refreshing state... [id=eipalloc-0575300b99cb7ed59]
module.vpc.aws_vpc.my_vpc["vpc"]: Refreshing state... [id=vpc-04d96a01d84843ca9]
aws_vpc.imported_vpc: Refreshing state... [id=vpc-04f15e6f778a69fe3]
module.vpc.aws_internet_gateway.my_igw["igw"]: Refreshing state... [id=igw-028f78dd16dc4eca7]
module.vpc.aws_security_group.my_sg["sg"]: Refreshing state... [id=sg-075bdd9d27a616a4f]
module.vpc.aws_route_table.my_route_table["rt_pub"]: Refreshing state... [id=rtb-0b540f9792c09608d]
module.vpc.aws_subnet.my_subnet["sub_pri"]: Refreshing state... [id=subnet-02a2656d752ce7dbb]
module.vpc.aws_subnet.my_subnet["sub_pub"]: Refreshing state... [id=subnet-024eac73baab8691d]
module.vpc.aws_route.my_route["rt_pub"]: Refreshing state... [id=r-rtb-0b540f9792c09608d1080289494]
module.vpc.aws_nat_gateway.my_nat_gw["nat"]: Refreshing state... [id=nat-0c2389b232c4c38e8]

No changes. Your infrastructure matches the configuration.

Terraform has compared your real infrastructure against your configuration
and found no differences, so no changes are needed.

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

VPC creation and import process completed successfully.
```

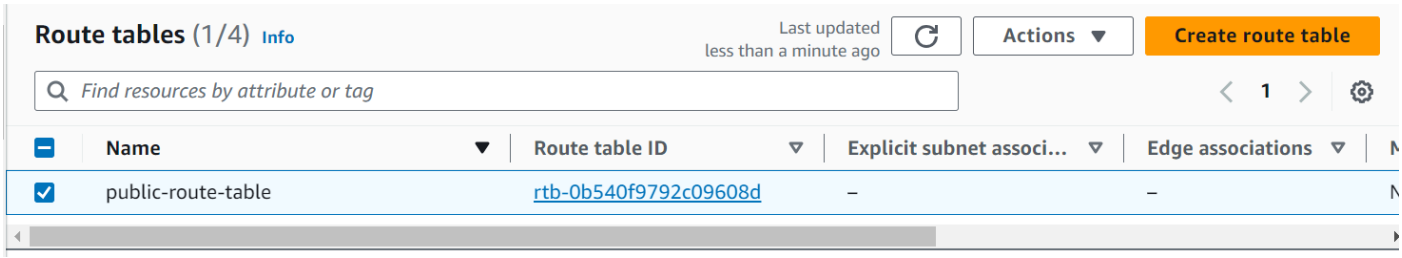
**Step:6** The VPC has been created



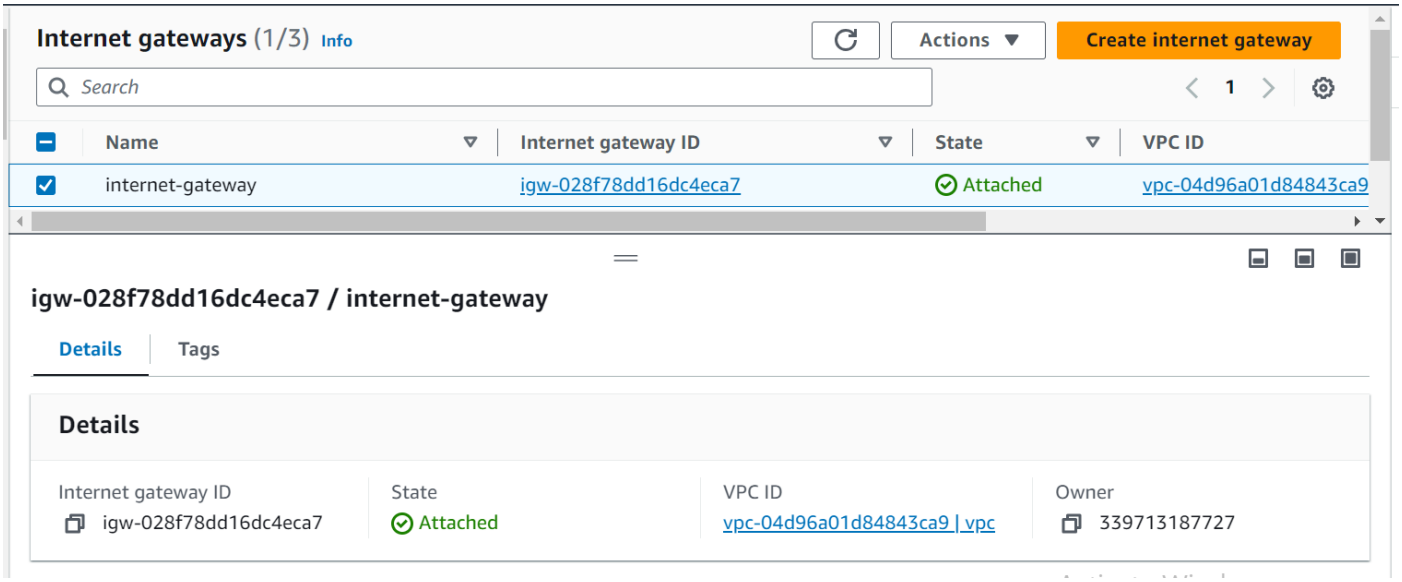

**Step:7** Created one public subnet, one private subnet



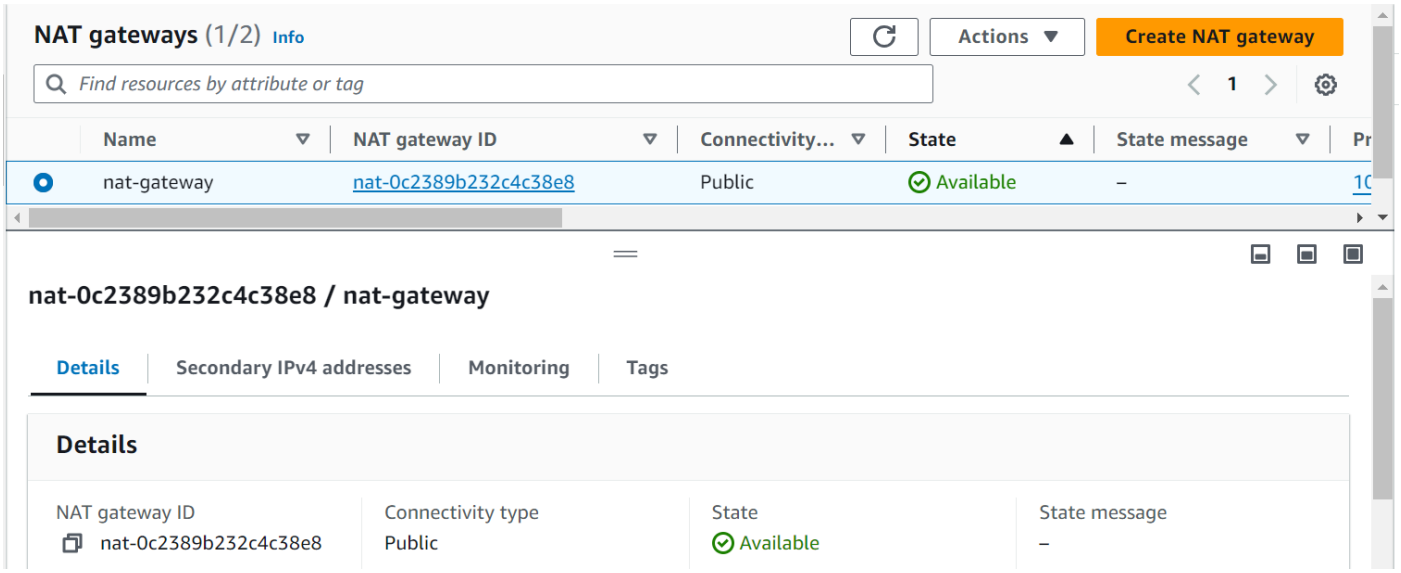
**Step:8** Route table has been created & associated with corresponding subnet



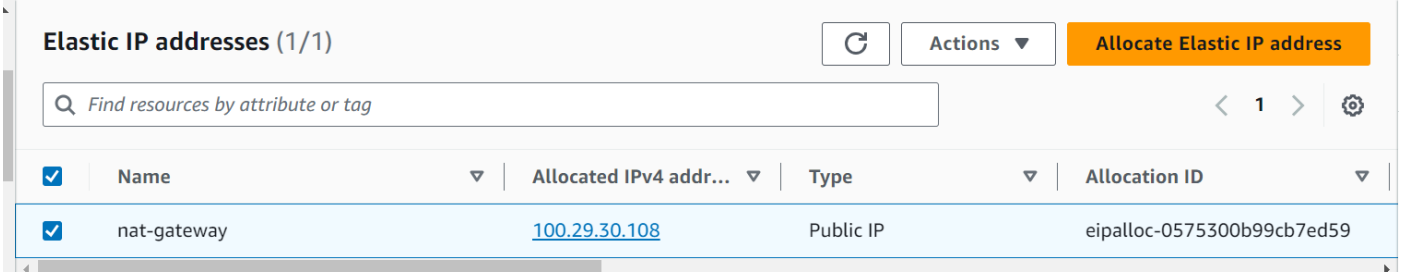

**Step:9** Internet Gateway has been created & associated with public subnet



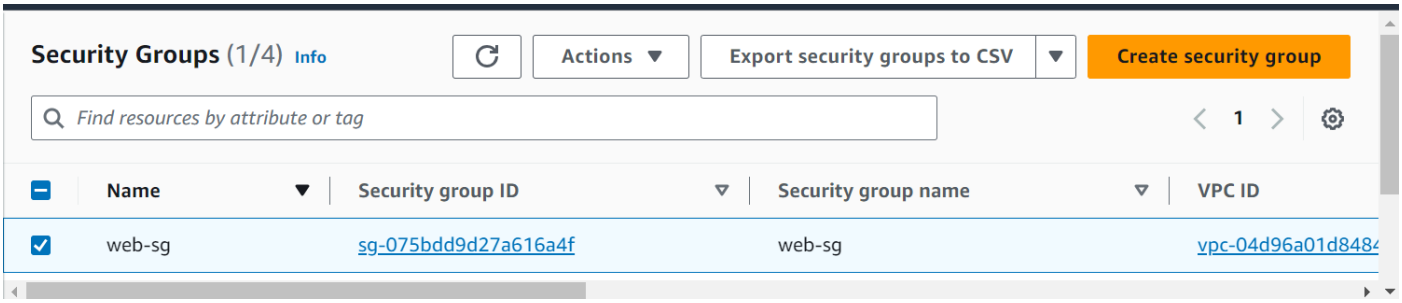

**Step:10** NAT Gateway has been created & associated with private subnet




**Step:11** Elastic IP has been created for NAT Gateway



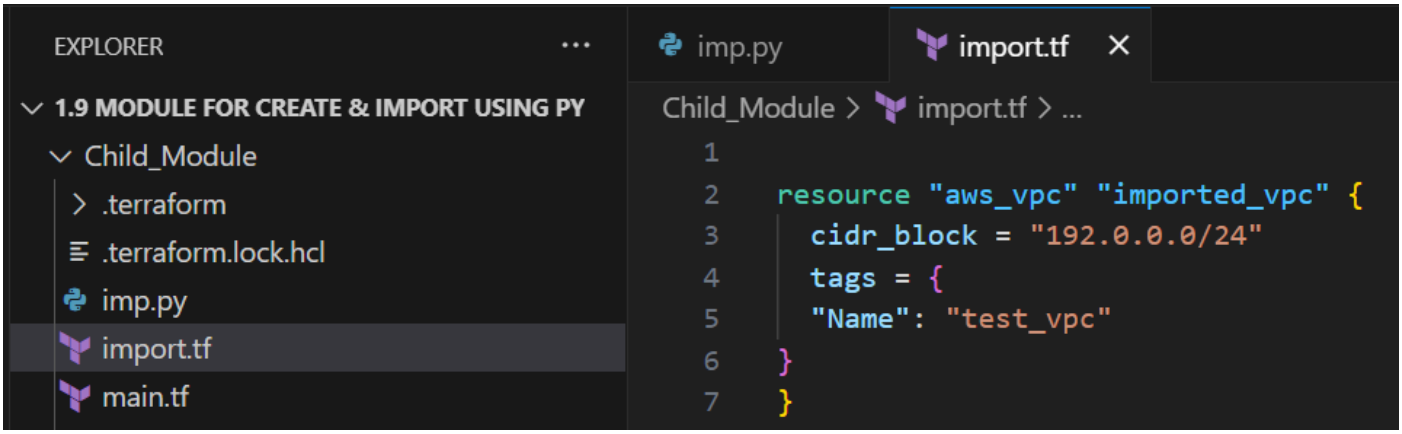
**Step:12** Security Group has been created




**Step:13** Now We can see that state file has been imported the test_vpc & Others also created


```
{
  "version": 4,
  "terraform_version": "1.9.7",
  "serial": 12,
  "lineage": "e85c81ec-2fc9-ddcf-4064-a1f378a94ee7",
  "outputs": {},
  "resources": [
    {
      "mode": "managed",
      "type": "aws_vpc",
      "name": "imported_vpc",
      "provider": "provider[\"registry.terraform.io/hashicorp/aws\"]",
      "instances": [
        {
          "schema_version": 1,
          "attributes": {
            "arn": "arn:aws:ec2:us-east-1:339713187727:vpc/vpc-04f15e6f778a69fe3",
            "assign_generated_ipv6_cidr_block": false,
            "cidr_block": "192.0.0.0/24",
            "default_network_acl_id": "acl-07cbce156f612019f",
            "default_route_table_id": "rtb-0c588ef0805f6d9ac",
            "default_security_group_id": "sg-0f3798844093d7bac",
            "dhcp_options_id": "dopt-09cd5f3382f696b6f",
            "enable_dns_hostnames": false,
            "enable_dns_support": true,
```


**Step:14** My python file has been created import.tf file for that configuration



**Step:15** This is the flow of using modules