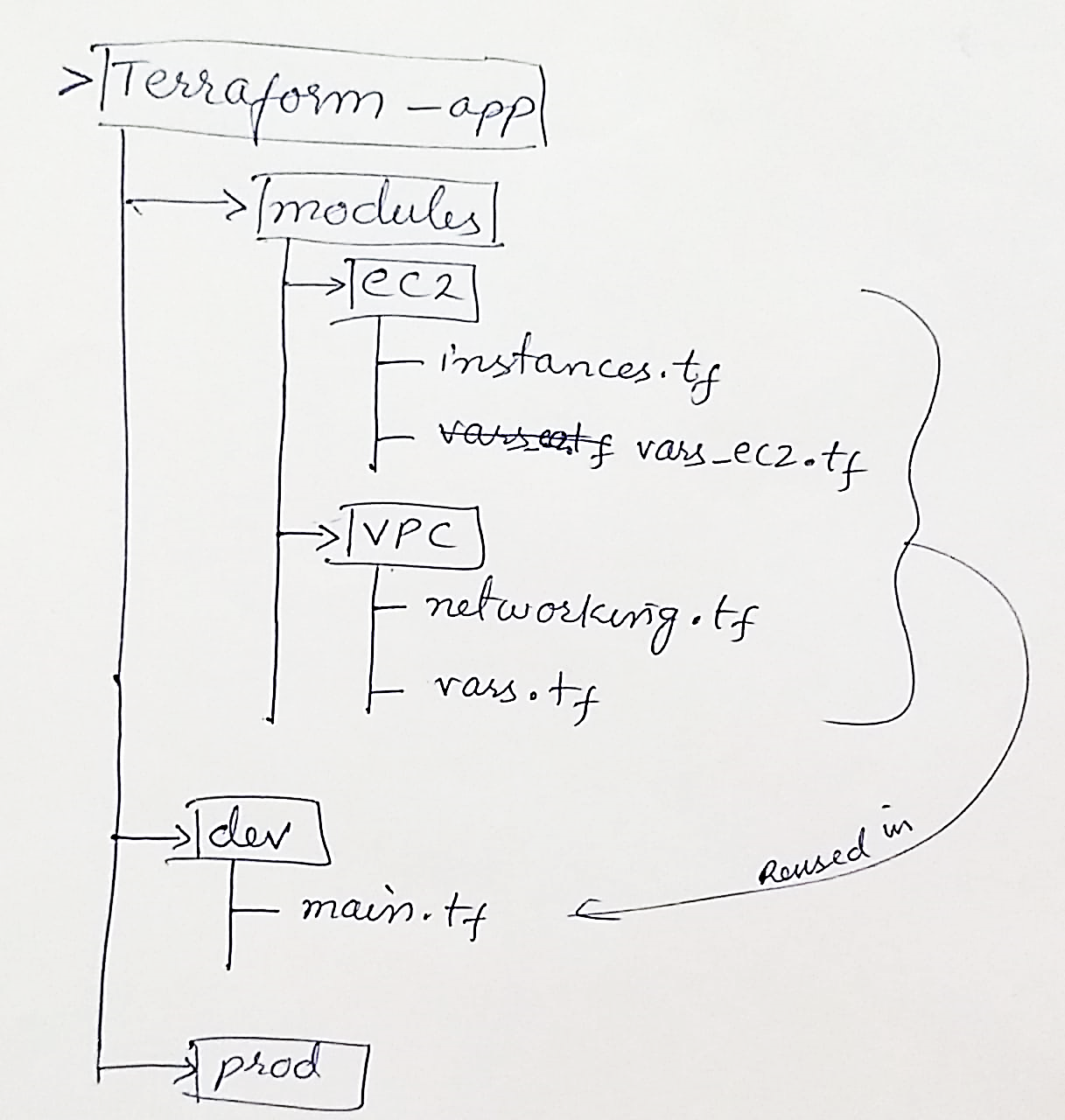
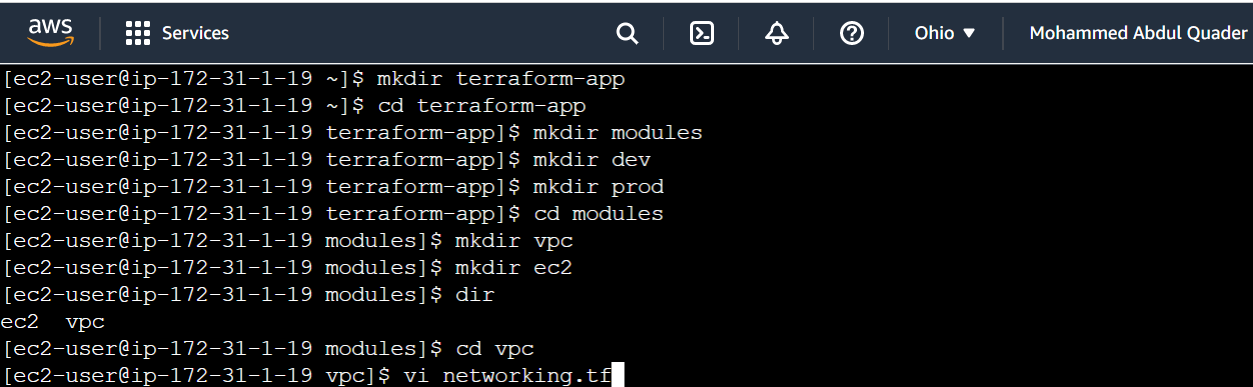
**HASHICORP MODULES**

Modules helps us manage code in reusable form. Module is a directory containing set of related terraform files/templates in organized fashion.

A project: Creating a workspace **terraform-app**





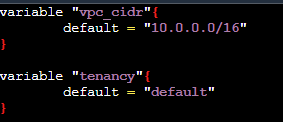
Create a file **networking.tf**

Go to terraform vpc registry to copy the following code.

|  |  |
| --- | --- |
|  |  |

Create a file **networking.tf** as follows and replace the hard-coded values of cidr\_block and instance\_type with variables as follows.

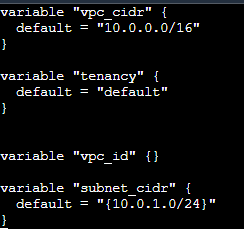
Let us create a file/template **vars.tf** for variables in the same vpc module.



Create subnet

|  |  |
| --- | --- |
|  |  |

Append **vars.tf**

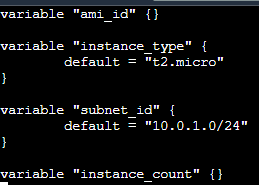


Keep the vpc\_id undeclared to make it dynamically allocated.

Now go to **ec2** directory and create a file **instances.tf.** Copy the following code from terraform registry.

|  |  |
| --- | --- |
|  |  |

Create **vars\_ec2.tf**



Now two modules **vpc** and **ec2** are ready with templates **networking.tf** and **instances.tf** respectively.

Each module has a variables file.

Let us now reuse them.

Creating a file **main.tf** in **dev** directory.

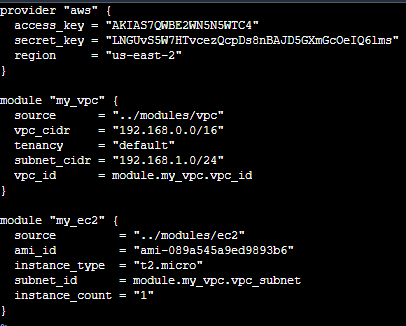
Here we test the reusability.

Modify the **networking.tf** and add **output** block to return the **vpc\_id** and **vpc\_subnet**.

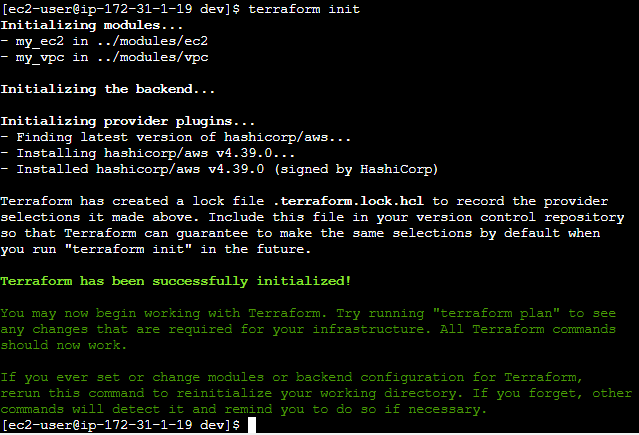




Now open **main.tf** from **dev** directory create module blocks for vpc and ec2. And initialize all the variables from **vars.tf** as shown below.

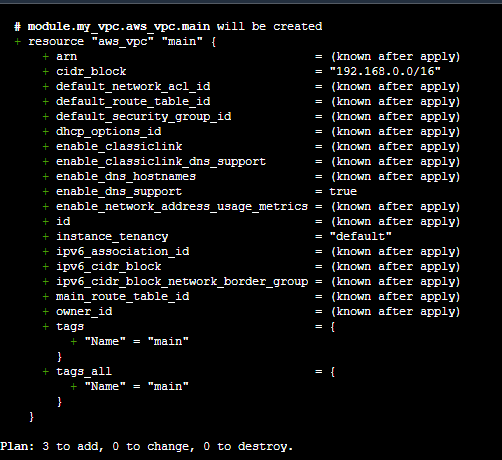
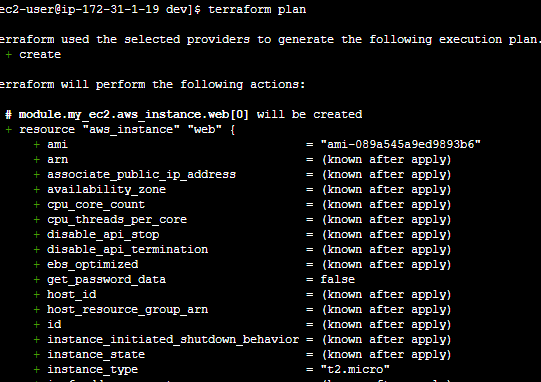


Now run the **terraform init** command.



You can see the modules has been successfully initialized.

Now run **terraform plan** command.



Similarly we can reuse the modules in **prod.**