**Scenario1:** Your company is requiring hosting a three-tier application in AWS / AZURE, you are the cloud engineer and are required to accomplish the below activities.

**Solution:**

1. One virtual network tied in three subnets.
2. Each subnet will have one virtual machine.
3. First virtual machine -> allow inbound traffic from internet only.
4. Second virtual machine -> entertain traffic from first virtual machine only and can reply to the same virtual machine again.
5. App can connect to database and database can connect to app, but database cannot connect to web.

├── main.tf // The primary entry point for terraform resources.

├── vars.tf // It contain the declarations for variables.

├── output.tf // It contain the declarations for outputs.

├── terraform.tfvars // The file to pass the terraform variables values.

**Module**

A module is a container for multiple resources that are used together. Modules can be used to create lightweight abstractions, so that you can describe your infrastructure in terms of its architecture, rather than directly in terms of physical objects.

For the solution, we have created and used five modules:

1. resourcegroup - creating resourcegroup
2. networking - creating azure virtual network and required subnets
3. securitygroup - creating network security group, setting desired security rules and associating them to subnets
4. compute - creating availability sets, network interfaces and virtual machines
5. database - creating database server and database

All the stacks are placed in the modules folder and the variable are stored under **terraform.tfvars**

To run the code you need to append the variables in the terraform.tfvars

Each module consists minimum two files: main.tf, vars.tf

resourcegroup and networking modules consists of one extra file named output.tf

## **Deployment Steps**

**Step 0** terraform init

used to initialize a working directory containing Terraform configuration files

**Step 1** terraform plan

used to create an execution plan

**Step 2** terraform validate

validates the configuration files in a directory, referring only to the configuration and not accessing any remote services such as remote state, provider APIs, etc

**Step 3** terraform apply

used to apply the changes required to reach the desired state of the configuration

Visual Code screenshot:

Text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

**Application Virtual Machine**

Graphical user interface, text, application, email

Description automatically generated

**Web-Server**

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Diagram

Description automatically generated

**Network Diagram**