Challenge-1:

Question: A 3-tier environment is a common setup. Use a tool of your choosing/familiarity create these resources on a cloud environment (Azure/AWS/GCP). Please remember we will not be judged on the outcome but more focusing on the approach, style and reproducibility.

Answer/Output-1: **Infrastructure-as-Code using 3-tier architecture environment using Azure VMs and MS SQL Database using Terraform**

Cloud provider : **Microsoft Azure**
Resource provider : **Azure Resource Manager (ARM)**
Orchestration/IAC : **Terraform**

Brief summary:

From the below example code, we have used the '**azurerm'** Resource provider to provision resources in Azure. The configuration creates a resource group, an Azure App Service Plan and App Service for the **web tier**, an Azure Virtual Network and Virtual Machine Scale Set for the **application tier**, and an Azure SQL Server and SQL Database for the **database tier**.

The App Service and Virtual Machine Scale Set (VMSS) are configured with appropriate settings for Linux-based deployments. The Virtual Machine Scale Set is provisioned with two instances.

The SQL Server is provisioned with an administrator username and password, and SSL enforcement is enabled for secure connections. The SQL Database is created under the SQL Server with Basic edition, a specific collation, and a maximum size of 5GB.

Network traffic (north-south) traffic can also be restricted using NSG rules between 3 tiers (*which is not considered in the code)*

The output section provides the App Service URL, VMSS ID, and SQL Server details for reference.

Terraform Code:

```
# Initialize Terraform with the Azure provider

terraform {

  required_providers {

    azurerm = {

      source  = "hashicorp/azurerm"

      version = ">= 2.0" }}}


# Configure the Azure provider

provider "azurerm" {

  features {}

}
```

```
# Create a resource group
resource "azurerm_resource_group" "example" {
  name     = "glm-rg-prod-01"
  location = "East US"
}

# Create an Azure App Service Plan for the web tier
resource "azurerm_app_service_plan" "web" {
  name                = " glm-asp-prod"
  location            = azurerm_resource_group.example.location
  resource_group_name = azurerm_resource_group.example.name
  kind                = "Linux"

  sku {
    tier = "Standard"
    size = "S1"
  }
}

# Create an Azure App Service for the web tier
resource "azurerm_app_service" "web" {
  name                = " glm-asp-prod-web-01"
  location            = azurerm_resource_group.example.location
  resource_group_name = azurerm_resource_group.example.name
  app_service_plan_id = azurerm_app_service_plan.web.id

  site_config {
    always_on = true
  }
}
```

```
# Create an Azure Virtual Network for the application tier
resource "azurerm_virtual_network" "app" {
  name                = "glm-asp-vnet-01"
  resource_group_name = azurerm_resource_group.example.name
  location            = azurerm_resource_group.example.location
  address_space       = ["10.0.0.0/16"]
}

# Create an Azure Subnet for the application tier
resource "azurerm_subnet" "app" {
  name                 = " glm-asp-vnet-snet-01"
  resource_group_name  = azurerm_resource_group.example.name
  virtual_network_name = azurerm_virtual_network.app.name
  address_prefixes     = ["10.0.1.0/24"]
}

# Create an Azure Virtual Machine Scale Set for the application tier
resource "azurerm_linux_virtual_machine_scale_set" "app" {
  name                = "glm-app-vmss-01"
  resource_group_name = azurerm_resource_group.example.name
  location            = azurerm_resource_group.example.location
  sku                 = "Standard_DS2_v2"

  instance_count = 2

  admin_username = "adminuser"
  admin_password = "password"

  os_disk {
    caching              = "ReadWrite"
    storage_account_type = "Standard_LRS"
```

```
  }

  storage_image_reference {
    publisher = "Canonical"
    offer     = "UbuntuServer"
    sku       = "18.04-LTS"
    version   = "latest"
  }

  network_interface {
    name    = "vmss-app-nic01"
    primary = true

    ip_configuration {
      name                          = "app-ip-config"
      subnet_id                     = azurerm_subnet.app.id
      load_balancer_backend_address_pool_ids = [azurerm_lb_backend_address_pool.app.id]
    }
  }
}
```

 # Create an Azure SQL Server for the database tier

```
resource "azurerm_mssql_server" "database" {
  name                         = "glm-db-prod-sql-server"
  resource_group_name          = azurerm_resource_group.example.name
  location                     = azurerm_resource_group.example.location
  version                      = "12.0"
  administrator_login          = "adminuser"
  administrator_login_password = "password"
  ssl_enforcement              = "Enabled"
}
```

 # Create an Azure SQL Database for the database tier

```
resource "azurerm_mssql_database" "database" {
  name                = " glm-db-sql-prod-db01"
  resource_group_name = azurerm_resource_group.example.name
  server_name         = azurerm_mssql_server.database.name
  edition             = "Basic"
  collation           = "SQL_Latin1_General_CP1_CI_AS"
  max_size_gb         = 5
}
```

 # Output the App Service URL, VMSS ID, and SQL Server details

```
output "app_service_url" {
  value = azurerm_app_service.web.default_site_hostname
}

output "vmss_id" {
  value = azurerm_linux_virtual_machine_scale_set.app.id
}

output "sql_server_details" {
  value = {
    server_name     = azurerm_mssql_server.database.fully_qualified_domain_name
    admin_username = azurerm_mssql_server.database.administrator_login
    admin_password = azurerm_mssql_server.database.administrator_login_password
    database_name  = azurerm_mssql_database.database.name
  }
}
```

#Initialize Terraform using **terraform init,** then run **terraform plan** to see the planned changes and **terraform apply** to provision the resources.

Challenge-1:

Question: A 3-tier environment is a common setup. Use a tool of your choosing/familiarity create these resources on a cloud environment (Azure/AWS/GCP). Please remember we will not be judged on the outcome but more focusing on the approach, style and reproducibility.

Answer/Output-2: Infrastructure-as-Code using 2-tier architecture environment using Azure App service and Azure managed SQL server using Terraform

Cloud provider : **Microsoft Azure**
Resource provider : **Azure Resource Manager (ARM)**
Orchestration/IAC : **Terraform**

Brief summary:

From the below example code, we are using the **'azurerm'** provider to provision resources in Azure. The configuration creates a resource group, an App Service Plan, an App Service, an Azure SQL Server, and an Azure SQL Database.

The App Service is configured with a Basic Linux plan and the always_on setting enabled.

The Azure SQL Server is provisioned with an administrator username and password, and SSL enforcement is enabled for secure connections.

The Azure SQL Database is created under the SQL Server with Basic edition, a specific collation, and a maximum size of 2GB.

Remember to customize the values and adjust the configuration to match your requirements, such as region, names, and additional settings.

To use this Terraform configuration, make sure you have Terraform installed and configured with the appropriate credentials for Azure. Initialize Terraform using terraform init, then run terraform plan to see the planned changes and terraform apply to provision the resources.

Network traffic (north-south) traffic can also be restricted using NSG rules between 3 tiers (*which is not considered in the code)*

The output section provides the App Service URL, VMSS ID, and SQL Server details for reference.

Terraform Code:

```
# Initialize Terraform with the Azure provider

terraform {

  required_providers {

    azurerm = {

      source  = "hashicorp/azurerm"

      version = ">= 2.0"

    }
```

```
  }
}

# Configure the Azure provider
provider "azurerm" {
  features {}
}

# Create a resource group
resource "azurerm_resource_group" "example" {
  name     = " glm-rg-prod-01"
  location = "East US"
}

# Create an Azure App Service Plan
resource "azurerm_app_service_plan" "example" {
  name                = " glm-asp-prod"
  location            = azurerm_resource_group.example.location
  resource_group_name = azurerm_resource_group.example.name
  kind                = "Linux"

  sku {
    tier = "Basic"
    size = "B1"
  }
}

# Create an Azure App Service
resource "azurerm_app_service" "example" {
  name                = " glm-asp-prod-01"
  location            = azurerm_resource_group.example.location
```

```
  resource_group_name = azurerm_resource_group.example.name
  app_service_plan_id = azurerm_app_service_plan.example.id

  site_config {
    always_on = true
  }
}
```

 # Create an Azure SQL Server

```
resource "azurerm_mssql_server" "example" {
  name                         = " glm-db-prod-sql-server"
  resource_group_name          = azurerm_resource_group.example.name
  location                     = azurerm_resource_group.example.location
  version                      = "12.0"
  administrator_login          = "adminuser"
  administrator_login_password = "password"
  ssl_enforcement              = "Enabled"

  tags = {
    environment = "prod"
  }
}
```

 # Create an Azure SQL Database

```
resource "azurerm_mssql_database" "example" {
  name                = " glm-db-sql-prod-db01"
  resource_group_name = azurerm_resource_group.example.name
  server_name         = azurerm_mssql_server.example.name
  edition             = "Basic"
  collation           = "SQL_Latin1_General_CP1_CI_AS"
  max_size_gb         = 2
```

```
}

# Output the App Service URL and SQL Server details
output "app_service_url" {
  value = azurerm_app_service.example.default_site_hostname
}

output "sql_server_details" {
  value = {
    server_name     = azurerm_mssql_server.example.fully_qualified_domain_name
    admin_username = azurerm_mssql_server.example.administrator_login
    admin_password = azurerm_mssql_server.example.administrator_login_password
    database_name  = azurerm_mssql_database.example.name
  }
}
```

#Initialize Terraform using **terraform init,** then run **terraform plan** to see the planned changes and **terraform apply** to provision the resources.
