Provisioning Azure database for MySQL

- Azure Database for MySQL is an enterprise-ready, fully managed community MySQL, delivered as a PaaS (Platform as a Service).

The variables.tf file

```
variable "mysql-admin-login" {
  type = string
  description = "Login to authenticate to MySQL Server"
}
variable "mysql-admin-password" {
  type = string
  description = "Password to authenticate to MySQL Server"
}
variable "mysql-version" {
  type = string
  description = "MySQL Server version to deploy"
  default = "8.0"
}
variable "mysql-sku-name" {
  type = string
  description = "MySQL SKU Name"
  default = "8.0"
}
variable "mysql-storage" {
  type = string
  description = "MySQL Storage in MB"
  default = "5120"
}
```

Variables definition:

mysql-sku-name

- Specifies the SKU Name for our MySQL Server. The name of the SKU, follows the tier + family + cores pattern. For example: B_Gen4_1, GP_Gen5_8.

Sku Tier: The tier of the particular SKU.

- Basic (B)
- GeneralPurpose (GP)
- MemoryOptimized (MO)

Family: The generation of MySQL Service to deploy.

- Generation 4 (Gen4)
- Generation 5 (Gen5)

mysql-version

- The version of a MySQL server to deploy. Current options are 5.6, 5.7 and 8.0

mysql-storage

- Max storage allowed for a server. Possible values are between 5120 MB(5GB) and 1048576 MB(1TB) for the Basic SKU and between 5120 MB(5GB) and 4194304 MB(4TB) for General Purpose/Memory Optimized SKUs

The main.tf file

- Then, we create our main.tf and we add the following code:
- The first step, as usual, is to create a Resource Group to store our MySQL service.

```
resource "azurerm_resource_group" "mysql-rg" {
  name     = "kopi-mysql-rg"
  location = "North Europe"
}
```

- And then, we deploy the MySQL server.

```
resource "azurerm_mysql_server" "mysql-server" {
  name = "kopi-mysql-server1"
  location = azurerm_resource_group.mysql-rg.location
  resource_group_name = azurerm_resource_group.mysql-rg.name

  administrator_login = var.admin_login
  administrator_login_password = var.admin_password

  sku_name = var.mysql-sku-name
  version = var.mysql-version

  storage_mb = var.mysql-storage
  auto_grow_enabled = true

  backup_retention_days = 7
  geo_redundant_backup_enabled = false
  public_network_access_enabled = true
  ssl_enforcement_enabled = true
  ssl_minimal_tls_version_enforced = "TLS1_2"
}
```

- After that, we create a MySQL database.

```
resource "azurerm_mysql_database" "mysql-db" {
  name                = "kopidb"
  resource_group_name = azurerm_resource_group.mysql-rg.name
  server_name         = azurerm_mysql_server.mysql-server.name
  charset             = "utf8"
  collation           = "utf8_unicode_ci"
}
```

- Finally, we configure the firewall to restrict access to our server.

```
resource "azurerm_mysql_firewall_rule" "mysql-fw-rule" {
  name                = "MySQL Office Access"
  resource_group_name = azurerm_resource_group.mysql-rg.name
  server_name         = azurerm_mysql_server.mysql-server.name
  start_ip_address    = "210.170.94.100"
  end_ip_address      = "210.170.94.110"
}
```

- The output.tf file
- We created the output.tf file and add the following content.

```
output "mysql_server" {
  value = azurerm_mysql_server.mysql-server
}
```

Creating the Input Definition Variables File

- In the last step, we are going to create input definition variables file terraform.tfvars and add the following code to the file:

```
company     = "kopicloud"
prefix      = "kopi"
environment = "dev"
location    = "northeurope"
description = "Deploy a MySQL Server"
owner       = "Guillermo Musumeci"
azure-subscription-id = "complete-me"
azure-client-id       = "complete-me"
azure-client-secret   = "complete-me"
azure-tenant-id       = "complete-me"
mysql-admin-login    = "kopiadmin"
mysql-admin-password = "Th1sIsAP@ssw0rd"
mysql-version        = "11"
mysql-sku-name       = "B_Gen5_1"
mysql-storage        = "5120"
```

Building the Azure Database for MySQL with Terraform

- We open our command line and type the following command to initialize the providers.

```
terraform init
```

- Then, to build our infrastructure we type:

```
terraform apply -auto-approve
```