Terraform on Azure – Lab 4

You have previously learned that some files, such as .tfvars and .tfstate files, are holding sensitive values and you don't want to have them exposed. Also, keeping a .tfstate file on your own machine poses a risk. In this lab you will learn how to work with the remote .tfstate file

1. You decide first to move the .tfstate file to another folder on your machine. You decide that you will create a new folder and move the .tfstate file using the following command:

```
mkdir state
move terraform.tfstate ./state/terraform.tfstate
```

2. The main.tf expects for .tfstate file to be in the same folder, and that is why you need to set an extra set of instructions for main.tf to be able to find .tfstate. You are going to do so by inserting the code in terraform block in main.tf file on the same level as the required_providers:

```
backend "local" {
  path = "state/terraform.tfstate"
}
```

Your terraform block in main.tf file should look like this:

```
terraform {

 backend "local" {
  path = "state/terraform.tfstate"
 }

 required_providers {
  azurerm = {
   source  = "hashicorp/azurerm"
   version = ">=3.44.0"
  }
  # Add provider for random values
  random = {
   source  = "hashicorp/random"
```

```
    version = "~>3.0"
  }
  # Add provider for Azure Resource Manager
  azapi = {
    source  = "azure/azapi"
    version = "~>1.5"
  }
 }
}
```

3. Since you have changed the location of .tfstate file, it is neccessary to initialize terraform again by following command:

```
terraform init
```

4. To make sure that you are using the proper .tfstate file run the command:

```
terraform state list
```

5. Keeping .tfstate out of obvious places might seem like a good idea, but that is nearly not as good as keeping the .tfstate file remote. You are going to create Azure Storage account (not with Terraform, because you don't want to accidently destroy that storage). You are going to place the storage account into dedicated resource group. Make sure that you set a *UNIQUE* name for the storage account(combination of todays date and your name might be good unique combination), and run the following code in your terminal(console):

```
# Create resource group
az group create --name tfstate --location eastus

# Create storage account
az storage account create --resource-group tfstate --name UNIQUE_STORAGE_ACCOUNT_NAME --
sku Standard_LRS --encryption-services blob

# Create blob container
az storage container create --name state --account-name UNIQUE_STORAGE_ACCOUNT_NAME --
auth-mode login
```

4. To configure the backend state in Terraform, you will need the following Azure storage information:

storage_account_name: The name of the Azure Storage account.
container_name: The name of the blob container.
key: The name of the state store file to be created.

Each of these values can be specified in the Terraform configuration file or on the command line.

5. The most used way of authorization with Storage account is using the access key. You can use access key using the following command:

```
ACCOUNT_KEY=$(az storage account keys list --resource-group tfstate --account-name
UNIQUE__STORAGE_ACCOUNT_NAME --query '[0].value' -o tsv)
export ARM_ACCESS_KEY=$ACCOUNT_KEY
```

6. Since you want to continue to work on the same infrastructure, you are going to move your terraform.tfstate file from *state* folder to state container in your storage account. The most convenient way to do so is by using Azure portal.
7. Next thing you want to do is to replace the local backend configuration to point to Azure using the following code:

```
backend "azurerm" {
    resource_group_name  = "tfstate"
    storage_account_name = "UNIQUE__STORAGE_ACCOUNT_NAME"
    container_name       = "tfstate"
    key                  = "terraform.tfstate"
}
```

That being said and done, your main.tf should look somewhat like this:

```
terraform {

 backend "azurerm" {
  resource_group_name = "tfstate"
  storage_account_name = "ivan20240222"
  container_name       = "state"
  key                  = "terraform.tfstate"
 }

 required_providers {
  azurerm = {
   ...
```

7. After you've configured your files to use remote .tfstate file it is time to give it a go by reintializing Terraform, planning and applying the configuration.

```
terraform init -reconfigure

terraform plan

terraform apply
```

Congratulations, you have completed lab 4!