Lab 06: Remote Terraform State Management with Azure Blob Storage

Objectives

- Create a new lab folder to manage Terraform state remotely.
- Use Azure CLI to create a personal container in the existing Azure storage account (`saqeworkshopriskaria`).
- Retrieve and store your Azure storage account access key as an environment variable.
- Configure Terraform backend to store the state remotely in Azure Blob storage.
- Deploy an Azure App Service (Windows Basic Tier with ASP.NET).

Prerequisites

- Azure CLI and Terraform installed.
- Access to Azure Storage Account: `saqeworkshopriskaria`
- Visual Studio Code.

Lab Instructions

✅ Step 1: Prepare your Lab Folder

1. Open **Visual Studio Code**.
2. Open the integrated terminal.
3. Create a new lab folder named `Lab-06-Remote-State`:

   ```
   mkdir Lab-06-Remote-State
   cd Lab-06-Remote-State
   ```

4. In VS Code explorer pane, create a new Terraform configuration file (`main.tf`).

✅ Step 2: Create Azure Blob Container

Use Azure CLI in the VS Code terminal to:

- Create a new container in the existing Azure storage account named `saqeworkshopriskaria`.
- Name the container after yourself (use lowercase letters):

```
az storage container create \
  --name "<yourname>" \
  --account-name "saqeworkshopriskaria"
```

Replace `<yourname>` with your actual name.

✅ Step 3: Retrieve Storage Account Key and Set Environment Variable

Retrieve the storage account key using Azure CLI and set it as an environment variable named `ARM_ACCESS_KEY`.

- **For Linux/macOS (Bash)**:

```
export ARM_ACCESS_KEY=$(az storage account keys list \
  --account-name "saqeworkshopriskaria" \
  --query "[0].value" -o tsv)
```

- **For Windows (PowerShell)**:

```
$env:ARM_ACCESS_KEY = az storage account keys list `
  --account-name "saqeworkshopriskaria" `
  --query "[0].value" -o tsv
```

✅ Step 4: Configure Terraform Backend for Remote State

In your `main.tf` file:

- Configure Terraform backend (`azurerm`) to use Azure Storage.
- Backend configuration details:

Parameter	Value
resource_group_name	`rg-qe-workshop-riskaria`
storage_account_name	`saqeworkshopriskaria`
container_name	`<yourname>`
key	`terraform.tfstate`

Replace `<yourname>` with your actual container name.

Note:
Do NOT hardcode your storage account key in the Terraform files. Terraform will automatically use the environment variable (`ARM_ACCESS_KEY`) you set in Step 3.

✅ Step 5: Configure Azure Provider and Resources in Terraform

- Define Azure provider (`azurerm`) in your Terraform configuration.
- Create Azure resources in `main.tf` as follows:
    - **Resource Group**
        - Name: `rg-app-<yourname>`
        - Location: `East US`
    - **App Service Plan**
        - Windows OS, Basic Tier (`B1`)
    - **Windows Web App**
        - Runtime stack: ASP.NET (e.g., .NET v6.0)

Use Terraform official documentation for reference:

- Azure App Service Plan
- Azure Windows Web App

✅ Step 6: Initialize and Deploy with Terraform

Run these Terraform commands from your VS Code integrated terminal:

```
terraform init
```

Verify the plan (dry-run):

```
terraform plan
```

Deploy the configuration:

```
terraform apply
```

Confirm by typing `yes`.

✅ Step 7: Verify Remote Terraform State Storage

Verify that your Terraform state is stored remotely in Azure Blob Storage:

- Go to the Azure Portal.
- Navigate to your storage account (`saqeworkshopriskaria`).
- Open your personal container (`<yourname>`).
- Confirm `terraform.tfstate` file is present.

✅ (Optional) Cleanup Resources

If you want to remove created resources after verification, run:

```
terraform destroy
```

Type `yes` to confirm.

Lab Completion

You have successfully completed managing Terraform state remotely with Azure Blob storage and deployed an Azure App Service using Terraform.