Lab 07: Deploy an Azure Windows Web App Using Terraform Data Blocks

Objectives

- Create a new lab folder for the deployment.
- Use Terraform data blocks to reference previously created resources.
- Deploy a Windows Web App (.NET) using existing Azure resources.
- Output the Web App URL after deployment.
- Reinforce concepts: variables, terraform.tfvars, remote state, data blocks, and outputs.

Prerequisites

- Completion of Lab 06 (Remote State configured).
- Terraform and Azure CLI installed.
- Access to Azure subscription with previously created resources:
  - **Resource Group**
  - **Windows App Service Plan**
- Visual Studio Code.

Lab Instructions

✅ Step 1: Set Up Lab Workspace

1. Open **Visual Studio Code**.
2. Open the integrated terminal and create a new lab directory:

   ```
   mkdir Lab-07-Data-Blocks
   cd Lab-07-Data-Blocks
   ```

3. In VS Code explorer pane, create a new Terraform configuration file named `main.tf`.

✅ Step 2: Configure Remote State Backend

- In your Terraform configuration file (`main.tf`), configure Terraform backend to use the new state file called `terraform-lab07.tfstate`
    - Use the same backend configuration as Lab 06 (storage account and container).

Tip: Ensure your environment variable (`ARM_ACCESS_KEY`) from Lab 06 is correctly set.

✅ Step 3: Define Input Variables

In your Terraform configuration:

- Define variables for dynamic inputs that are new and unique for this lab:
    - **Web App Name**
    - **Resource Group Name** *(existing, to help identify it via data block easily)*
    - **App Service Plan Name** *(existing, helps retrieve via data block)*
- Create a new file `terraform.tfvars` in VS Code and specify the actual values for these variables.

Note:
We define variables for Resource Group and App Service Plan names to clearly indicate what resources to look up via the data block. Terraform data blocks require these identifiers as input to fetch existing resource details.

✅ Step 4: Using Data Blocks to Retrieve Existing Resources

In your `main.tf`:

- Add Terraform **data blocks** to retrieve details about:
    - Existing Azure **Resource Group** (for obtaining its location).
    - Existing Azure **App Service Plan** (for its ID and configuration).
- Use the variable values defined previously to identify and fetch these resources accurately.

✅ Step 5: Configure Azure Web App

- In your Terraform configuration:
    - Create an Azure **Windows Web App** resource.

    - Reference the existing App Service Plan and Resource Group details retrieved via the Terraform data blocks.
    - Set the runtime stack to **ASP.NET** (recommended version: .NET v6.0).

Important:
Do not hardcode any existing resource details; always reference using Terraform data block attributes.

✅ Step 6: Define Terraform Outputs

- After configuring your resources, define a Terraform output to display the Azure Web App URL.
- This will help verify successful deployment easily.

✅ Step 7: Deploy Using Terraform

Execute the following commands in the VS Code terminal:

- Initialize Terraform:

- Plan and verify the deployment:

- Apply to create the resources:

Type `yes` to confirm the deployment when prompted.

✅ Step 8: Validate Deployment

- Confirm that Terraform displays the Web App URL as an output.
- Access the Azure Portal and verify your Web App is successfully deployed and running.

✅ Optional Cleanup

- After completing validation, destroy resources if desired:

```
terraform destroy
```

Confirm with `yes`.

Lab Completion

You've successfully deployed an Azure Windows Web App using existing Azure resources via Terraform data blocks, effectively reinforcing previous concepts of variables, tfvars, remote state, and outputs.