**Setting Up Azure Storage to Be Used for Terraform Remote State**

**Introduction**

In this lab, we will cover 4 objectives:

1. First, we will log into the Azure portal and then configure the Cloud Shell to use bash.
2. Second, we will download and run a script to setup the lab environment.
3. Third, we will create an Azure Storage account with a storage container to use for remote storage of our Terraform state file.
4. And for our fourth objective, we will configure Terraform to use Azure Storage to remotely store our state.

It is considered best practice to remotely store your state since it helps keep the file secure as it may contain sensitive data in plain text. It also allows for collaboration when managing and working with resources in a team environment.

**Solution**

**Create the Azure Storage Account Configuration**

1. Edit the remote\_state\_storage.tf file:

vim remote\_state\_storage.tf

1. In the Azure Portal, copy the **Resource group** name.
2. In the Cloud Shell, on line 10 of the remote\_state\_storage.tf file, uncomment the line by deleting the # and replace <RESOURCE\_GROUP\_NAME> with the **Resource group** name that was just copied.
3. On line 11, uncomment the line by deleting the # and replace <LOCATION> with the location for the lab environment, using eastus for East US, centralus for Central US, and so on.
4. Uncomment the tags by deleting the remaining # symbols.
5. Starting on line 17, add a new azurerm\_storage\_account resource using the following configuration:

resource "azurerm\_storage\_account" "tfstate" {

name = "tfstate${random\_string.resource\_code.result}"

resource\_group\_name = azurerm\_resource\_group.guru.name

location = azurerm\_resource\_group.guru.location

account\_tier = "Standard"

account\_replication\_type = "LRS"

allow\_blob\_public\_access = true

tags = {

environment = "dev"

}

}

resource "azurerm\_storage\_container" "tfstate" {

name = "tfstate"

storage\_account\_name = azurerm\_storage\_account.tfstate.name

container\_access\_type = "blob"

}

# Outputs

output "storage\_account\_name" {

value = azurerm\_storage\_account.tfstate.name

}

output "storage\_container\_name" {

value = azurerm\_storage\_container.tfstate.name

}

1. Save and exit the file:

ESC :wq!

1. Format the file:

terraform fmt

1. Validate the syntax:

terraform validate

1. Do a dry run of the configuration:

terraform plan

1. Apply the configuration:

terraform apply

1. When prompted, type yes to proceed.
2. In the Outputs section, copy the storage\_account\_name and storage\_container\_name for later use.

**Deploy the Azure Storage Account with Terraform**

1. Create a new file named backend.tf:

vim backend.tf

1. In the file, paste the following, replacing the RESOURCE\_GROUP\_NAME, STORAGE\_ACCOUNT\_NAME, and CONTAINER\_NAME with the information copied earlier in the lab:

terraform {

backend "azurerm" {

resource\_group\_name = "<RESOURCE\_GROUP\_NAME>"

storage\_account\_name = "<STORAGE\_ACCOUNT\_NAME>"

container\_name = "<CONTAINER\_NAME>"

key = "terraform.tfstate"

}

}

1. Save and exit the file:

ESC :wq!

1. If needed, run terraform fmt again to correct any formatting issues with the backend.tf file.
2. Using terraform init initialize the backend:

terraform init

1. When prompted, enter yes to copy the existing state to the new backend.
2. Once completed, in the Azure Portal, use the left-hand menu to navigate to **Storage accounts**.
3. Open the newly created tfstate storage account.
4. Under **Data storage**, click **Containers**.
5. Click the **tfstate** container and verify that the **terraform.tfstate** file has been stored successfully.
6. Now that the state file is being stored remotely, in the Cloud Shell, remove the local terraform.tfstate file:

rm -rf terraform.tfstate

1. Using terraform show, read the state file and verify that it is correctly using the remotely stored file.
2. Using terraform state list, confirm that the resources are still being managed by Terraform.