Task 1: Install terraform and Azure CLI

1. Use official guidelines to install the latest version of terraform and Azure CLI.
2. Authenticate with Azure CLI.

3. Set the exercise subscription as default for Azure CLI.

```
nikki@DESKTOP-I298DGN:~$ az account set --subscription 6ad3de4b-8806-45a3-bdae-c42ab48bc868
```

4. Provide console print screen:

4.1 Time and date when the exercise was worked.

```
nikki@DESKTOP-I298DGN:~$ date
Sat Apr 22 00:51:20 CEST 2023
nikki@DESKTOP-I298DGN:~$
```

4.2 Output of the terraform command that will print out the Terraform version installed.

```
nikki@DESKTOP-I298DGN:~$ terraform --version
Terraform v1.4.4
on linux_amd64

Your version of Terraform is out of date! The latest version
is 1.4.5. You can update by downloading from https://www.terraform.io/downloads.html
nikki@DESKTOP-I298DGN:~$
```

4.3 Azure CLI output of the current subscription.

```
nikki@DESKTOP-I298DGN:~$ az account show
{
  "environmentName": "AzureCloud",
  "homeTenantId": "ed019442-7a00-4fcc-b9f3-8983adb6ea42",
  "id": "6ad3de4b-8806-45a3-bdae-c42ab48bc868",
  "isDefault": true,
  "managedByTenants": [],
  "name": "Azure subscription 1",
  "state": "Enabled",
  "tenantId": "ed019442-7a00-4fcc-b9f3-8983adb6ea42",
  "user": {
    "name": "nikolovska.ana2@outlook.com",
    "type": "user"
  }
}
nikki@DESKTOP-I298DGN:~$
```

Task 2: Define your first terraform infrastructure code

1. Add minimal provider configuration and initialize terraform.

1.1 Create file called "main.tf"

1.2 Inside the file add the minimal configuration needed which is provided on **terraform registry**

site for the Azure Provider (**azurerm**) following the instructions from the **USE PROVIDER** link near the top right corner of the page. (Every provider in terraform registry has instruction on how to use the provider, the configuration Arguments, along with some examples).

1.3 Save the file.

1.4 Initialize terraform. The output of the command should not show any errors.

```
nikki@DESKTOP-I298DGN:~$ nano main.tf
nikki@DESKTOP-I298DGN:~$ terraform init

Initializing the backend...

Initializing provider plugins...
- Finding hashicorp/azurerm versions matching "3.53.0"...
- Installing hashicorp/azurerm v3.53.0...
- Installed hashicorp/azurerm v3.53.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
nikki@DESKTOP-I298DGN:~$
```

1.5 Execute **terraform plan** of the current terraform code. The output should not show any errors and will say that there are no changes (this is expected since we still don't have any infrastructure resources defined).

```
nikki@DESKTOP-I298DGN:~$ terraform plan

No changes. Your infrastructure matches the configuration.

Terraform has compared your real infrastructure against your configuration and found no differences, so no changes are
needed.
nikki@DESKTOP-I298DGN:~$
```

1.6 Beneath the block for the provider add the following code block

```
data "azurerm_subscription" "current" {}
```

1.7 Execute **terraform plan** once again. This time it will throw you an error.

```
nikki@DESKTOP-I298DGN:~$ terraform plan

Planning failed. Terraform encountered an error while generating this plan.


  Error: Insufficient features blocks

    on main.tf line 10, in provider "azurerm":
    10: provider "azurerm" {

  At least 1 "features" blocks are required.

nikki@DESKTOP-I298DGN:~$
```

1.7.1 Read the content of the error and see what you are missing.

- This error tells us that features block is required. We add the features block in provider "azurearm" because we are using the default behaviors of Azure Provider.

1.7.2 Go back to the azurerm terraform registry page and see the Example Usage code block.

1.7.3 Compare the problematic block with the one in the example.

1.7.4 Scroll down to **Argument reference** part of the page and see the required arguments for the provider block. (The Argument reference part of the provider or resource description gives us information about the arguments that we can configure on one provider or resource and most importantly the mandatory ones marked as **Required**)

- The **features** argument is the only argument that is mandatory and marked as **Required**, all others are **Optional**.

1.8 Make the corrections and execute another terraform plan command. This time you should not see any errors and changes.

```
nikki@DESKTOP-I298DGN:~$ terraform plan
data.azurerm_subscription.current: Reading...
data.azurerm_subscription.current: Read complete after 0s [id=/subscriptions/6ad3de4b-8806-45a3-bdae-c42
ab48bc868]

No changes. Your infrastructure matches the configuration.

Terraform has compared your real infrastructure against your configuration and found no differences, so
no changes are needed.
nikki@DESKTOP-I298DGN:~$
```

2. Usage of static provider version

2.1 Currently our version of azurerm provider is set to fixed value, which is not always a good practice because we rarely think about upgrading the provider version in normal work.

2.2 Set the azurerm provider to version 3.35.0

2.3 Add the code block bellow to your code:

```
resource "random_string" "random" {
  length  = 8
  special = false
  lower   = true
  upper   = false
}

resource "azurerm_resource_group" "example" {
  name     = "${random_string.random.result}-rg"
  location = "West Europe"
}

resource "azurerm_storage_account" "example" {
  name                     = "${random_string.random.result}sa"
  resource_group_name      = azurerm_resource_group.example.name
  location                 =
azurerm_resource_group.example.location
  account_tier             = "Standard"
  account_replication_type = "GRS"
  blob_properties {
    restore_policy {
      days = 7
    }
  }
  tags = {
    environment = "staging"
  }
}
```

2.4 Let's see the plan of our code.

2.4.1 Execute **terraform plan**.

```
nikki@DESKTOP-I298DGN:~$ terraform plan

Error: Inconsistent dependency lock file

The following dependency selections recorded in the lock file are inconsistent with the current
configuration:
  - provider registry.terraform.io/hashicorp/random: required by this configuration but no version is
selected

To update the locked dependency selections to match a changed configuration, run:
  terraform init -upgrade

nikki@DESKTOP-I298DGN:~$
```

2.4.2 Now we have a problem that terraform is initialized with different provider version and we must reinitialize with "upgrade" terraform to get the assigned version.

- Execute **terraform init –upgrade** command to accomplish that. (This is required whenever we work with already initialized terraform working directory.)

```
nikki@DESKTOP-I298DGN:~$ terraform init -upgrade

Initializing the backend...

Initializing provider plugins...
- Finding hashicorp/azurerm versions matching "3.53.0"...
- Finding latest version of hashicorp/random...
- Using previously-installed hashicorp/azurerm v3.53.0
- Installing hashicorp/random v3.5.1...
- Installed hashicorp/random v3.5.1 (signed by HashiCorp)

Terraform has made some changes to the provider dependency selections recorded
in the .terraform.lock.hcl file. Review those changes and commit them to your
version control system if they represent changes you intended to make.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
nikki@DESKTOP-I298DGN:~$
```

2.4.3 Since we have reinitialized and updated our working directory, we can proceed again with the **terraform plan** command.

```
nikki@DESKTOP-I298DGN:~$ terraform plan

Error: Missing required argument

  with azurerm_storage_account.example,
  on main.tf line 37, in resource "azurerm_storage_account" "example":
  37:     restore_policy {

"blob_properties.0.restore_policy": all of
`blob_properties.0.delete_retention_policy,blob_properties.0.restore_policy` must be specified

nikki@DESKTOP-I298DGN:~$
```

2.4.4 Now the output gives us another error which says that the block ‘restore_policy’ is not expected in our “azurem_storage_account” resource. But why is that?

- Go to the terraform registry and read the provided documentation for the “azurem_storage_account” resource. Check the blob properties block and check if it has description for field named restore_policy.
- The field is present, then why do we have the issue?

- When we browse terraform registry we are always forwarded to the latest version of the provider. What version of the provider do we have? Isn't it 3.35.0.
- Let's see the documentation for version 3.35.0 of azurerm provider for the "azurem_storage_account" resource. Check the blob properties block and check if it has description for field named restore_policy.

- The field is not present in version 3.35.0, and this is causing our problem. The change for the restore_policy on the blob_properties is introduced in version 3.36 of the azurerm provider, and we have set fixed version of 3.35.0 for the provider.
- In order to fix this, we would need to allow our terraform code to be able to automatically upgrade to the latest version of the provider by default. But, at the same time we would need to setup the minimum allowed version for which our code works, and that is 3.36.0

2.5 Allow automatic updating of azurerm provider to the latest version, but with minimum version of 3.36.0

2.5.1 Go to your provider configuration and replace the current version with ">= 3.36.0" (allow azurerm version that is greater or equal to 3.36.0)

2.6 Check the result of terraform plan now. Don't forget to reinitialize with an upgrade.

2.6.1 Again, do we have issues with our code?

- Well in the latest provider documentation for "azurem_storage_account" for the restore policy it says that must be used together with delete_retention_policy set, versioning_enabled and change_feed_enabled set to true. This is the reason why we must consult terraform registry constantly and read the documentation for the resource configuration more carefully.

2.6.2 Add the following code bellow the restore_policy block:

```
    delete_retention_policy {
      days = 8
    }
    versioning_enabled = true
    change_feed_enabled = true
```

2.7 Check the results from your terraform plan now. There should be no issues and it will show what it will manage. Take the time to answer the following questions:

```
nikki@DESKTOP-I298DGN:~$ terraform plan
data.azurerm_subscription.current: Reading...
data.azurerm_subscription.current: Read complete after 0s [id=/subscriptions/6ad3de4b-8806-45a3-bdae-c42ab48bc868]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # azurerm_resource_group.example will be created
  + resource "azurerm_resource_group" "example" {
      + id       = (known after apply)
      + location = "westeurope"
      + name     = (known after apply)
    }

  # azurerm_storage_account.example will be created
  + resource "azurerm_storage_account" "example" {
      + access_tier                     = (known after apply)
      + account_kind                    = "StorageV2"
      + account_replication_type        = "GRS"
      + account_tier                    = "Standard"
      + allow_nested_items_to_be_public = true
      + cross_tenant_replication_enabled  = true
      + default_to_oauth_authentication   = false
      + enable_https_traffic_only       = true
      + id                              = (known after apply)
      + infrastructure_encryption_enabled = false
      + is_hns_enabled                  = false
      + large_file_share_enabled        = (known after apply)
      + location                        = "westeurope"
      + min_tls_version                 = "TLS1_2"
      + name                            = (known after apply)
      + nfsv3_enabled                   = false
      + primary_access_key              = (sensitive value)
      + primary_blob_connection_string  = (sensitive value)
      + primary_blob_endpoint           = (known after apply)
      + primary_blob_host               = (known after apply)
      + primary_connection_string       = (sensitive value)
      + primary_dfs_endpoint            = (known after apply)
      + primary_dfs_host                = (known after apply)
      + primary_file_endpoint           = (known after apply)
      + primary_file_host               = (known after apply)
      + primary_location                = (known after apply)
      + primary_queue_endpoint          = (known after apply)
      + primary_queue_host              = (known after apply)
      + primary_table_endpoint          = (known after apply)
      + primary_table_host              = (known after apply)
      + primary_web_endpoint            = (known after apply)
      + primary_web_host                = (known after apply)
```

```
    + primary_web_endpoint             = (known after apply)
    + primary_web_host                 = (known after apply)
    + public_network_access_enabled    = true
    + queue_encryption_key_type        = "Service"
    + resource_group_name              = (known after apply)
    + secondary_access_key             = (sensitive value)
    + secondary_blob_connection_string = (sensitive value)
    + secondary_blob_endpoint          = (known after apply)
    + secondary_blob_host              = (known after apply)
    + secondary_connection_string      = (sensitive value)
    + secondary_dfs_endpoint           = (known after apply)
    + secondary_dfs_host               = (known after apply)
    + secondary_file_endpoint          = (known after apply)
    + secondary_file_host              = (known after apply)
    + secondary_location               = (known after apply)
    + secondary_queue_endpoint         = (known after apply)
    + secondary_queue_host             = (known after apply)
    + secondary_table_endpoint         = (known after apply)
    + secondary_table_host             = (known after apply)
    + secondary_web_endpoint           = (known after apply)
    + secondary_web_host               = (known after apply)
    + sftp_enabled                     = false
    + shared_access_key_enabled        = true
    + table_encryption_key_type        = "Service"
    + tags                             = {
        + "environment" = "staging"
      }

    + blob_properties {
        + change_feed_enabled      = true
        + default_service_version  = (known after apply)
        + last_access_time_enabled = false
        + versioning_enabled       = true

        + delete_retention_policy {
            + days = 8
          }

        + restore_policy {
            + days = 7
          }
      }
  }
```

```
# random_string.random will be created
+ resource "random_string" "random" {
    + id          = (known after apply)
    + length      = 8
    + lower       = true
    + min_lower   = 0
    + min_numeric = 0
    + min_special = 0
    + min_upper   = 0
    + number      = true
    + numeric     = true
    + result      = (known after apply)
    + special     = false
    + upper       = false
  }

Plan: 3 to add, 0 to change, 0 to destroy.

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if
you run "terraform apply" now.
```

2.7.1 How many resources have you defined in your code and how many resources does the plan output show? Are they the same and why?

- We have 3 resources defined and the output shows 3 resources. They are Resource Group, Storage Account and Random String.

2.7.2 What is the location of your resource group and what is the location of the storage account?

- The location for the Resource Group and the Storage Account is West Europe.

2.8 Deploy your code to on the subscription and answer the questions bellow:

2.8.1 How many resources do you have on your subscription? (To list all resources, type “All resources” in the search bar on the top in Azure Portal)

- I have 3 resources total in my All resources. First is for the newly created storage with the help of Terraform, the second is for the cloud shell storage, and the third is for the NetworkWathcers.

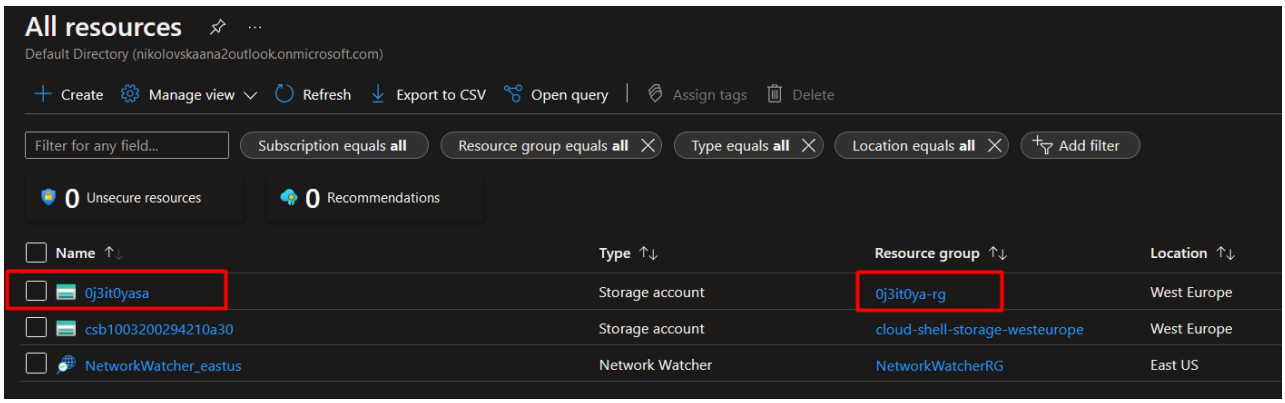
2.8.2 Are the number of resources shown in the All-resources portal window the same as the ones from your plan?

- The number of resources is the same as the ones from the Terraform plan we created.

2.8.3 Give a short explanation about the resources that are not shown?

- The only resource not shown here is the random string. But the random string resource in Terraform is used to create the names of the account storage and the resource group in Azure.

2.8.4 Provide a print screen of your portal with all resources.



IMPORTANT!!! Make sure you have the correct subscription selected if you already have multiple.

Task 3: Using variables and outputs

1. Using input variable

1.1. While you're in the same directory from the previous task, create a file named variables.tf.

1.2. Define the variable from the code below in the variables.tf file.

```
variable "my_name" {
  type = string
  description = "First name of the student"
}
```

1.3. Reference the input variable value in your code in the beginning of the name of the resource group

```
resource "azurerm_resource_group" "example" {
  name     = "${var.my_name}-${random_string.random.result}-rg"
  location = "West Europe"
}
```

1.4. Define another variable of type string named "location" with default value of "West Europe" and description "The location where all resources will be placed.". Use the definition of step 1.2 as reference and search in terraform documentation for defining a default value in a variable.

1.5. Reference the location variable as the location for the azurerm_resource_group resource. Use the example in step 1.2 of referencing the value but this time without the use of interpolation.

1.6. Execute terraform plan.

1.6.1. You are seeing that the code is asking you to insert an input value. Type your first name in lowercase and press enter.

```
nikki@DESKTOP-I298DGN:~/terraform-aure$ terraform plan
var.my_name
  First name of the student

  Enter a value: ana

random_string.random: Refreshing state... [id=0j3it0ya]
data.azurerm_subscription.current: Reading...
azurerm_resource_group.example: Refreshing state... [id=/subscriptions/6ad3de4b-8806-45a3-bdae-c42ab48bc
868/resourceGroups/0j3it0ya-rg]
data.azurerm_subscription.current: Still reading... [10s elapsed]
data.azurerm_subscription.current: Read complete after 10s [id=/subscriptions/6ad3de4b-8806-45a3-bdae-c4
2ab48bc868]
azurerm_storage_account.example: Refreshing state... [id=/subscriptions/6ad3de4b-8806-45a3-bdae-c42ab48b
c868/resourceGroups/0j3it0ya-rg/providers/Microsoft.Storage/storageAccounts/0j3it0yasa]

Terraform used the selected providers to generate the following execution plan. Resource actions are
indicated with the following symbols:
-/+ destroy and then create replacement

Terraform will perform the following actions:

  # azurerm_resource_group.example must be replaced
-/+ resource "azurerm_resource_group" "example" {
      ~ id       = "/subscriptions/6ad3de4b-8806-45a3-bdae-c42ab48bc868/resourceGroups/0j3it0ya-rg" -> (
known after apply)
      ~ name     = "0j3it0ya-rg" -> "ana-0j3it0ya-rg" # forces replacement
      - tags     = {} -> null
        # (1 unchanged attribute hidden)
    }

  # azurerm_storage_account.example must be replaced
-/+ resource "azurerm_storage_account" "example" {
      ~ access_tier                     = "Hot" -> (known after apply)
      ~ id                              = "/subscriptions/6ad3de4b-8806-45a3-bdae-c42ab48bc868/resourc
eGroups/0j3it0ya-rg/providers/Microsoft.Storage/storageAccounts/0j3it0yasa" -> (known after apply)
      + large_file_share_enabled        = (known after apply)
        name                            = "0j3it0yasa"
      ~ primary_access_key              = (sensitive value)
      ~ primary_blob_connection_string  = (sensitive value)
      ~ primary_blob_endpoint           = "https://0j3it0yasa.blob.core.windows.net/" -> (known after
apply)
      ~ primary_blob_host               = "0j3it0yasa.blob.core.windows.net" -> (known after apply)
      ~ primary_connection_string       = (sensitive value)
      ~ primary_dfs_endpoint            = "https://0j3it0yasa.dfs.core.windows.net/" -> (known after a
pply)
      ~ primary_dfs_host                = "0j3it0yasa.dfs.core.windows.net" -> (known after apply)
      ~ primary_file_endpoint           = "https://0j3it0yasa.file.core.windows.net/" -> (known after
```

```
      ~ primary_file_endpoint             = "https://0j3it0yasa.file.core.windows.net/" -> (known after
apply)
      ~ primary_file_host                 = "0j3it0yasa.file.core.windows.net" -> (known after apply)
      ~ primary_location                  = "westeurope" -> (known after apply)
      ~ primary_queue_endpoint            = "https://0j3it0yasa.queue.core.windows.net/" -> (known after
 apply)
      ~ primary_queue_host                = "0j3it0yasa.queue.core.windows.net" -> (known after apply)
      ~ primary_table_endpoint            = "https://0j3it0yasa.table.core.windows.net/" -> (known after
 apply)
      ~ primary_table_host                = "0j3it0yasa.table.core.windows.net" -> (known after apply)
      ~ primary_web_endpoint              = "https://0j3it0yasa.z6.web.core.windows.net/" -> (known afte
r apply)
      ~ primary_web_host                  = "0j3it0yasa.z6.web.core.windows.net" -> (known after apply)
      ~ resource_group_name               = "0j3it0ya-rg" -> "ana-0j3it0ya-rg" # forces replacement
      ~ secondary_access_key              = (sensitive value)
      + secondary_blob_connection_string  = (sensitive value)
      + secondary_blob_endpoint           = (known after apply)
      + secondary_blob_host               = (known after apply)
      ~ secondary_connection_string       = (sensitive value)
      + secondary_dfs_endpoint            = (known after apply)
      + secondary_dfs_host                = (known after apply)
      + secondary_file_endpoint           = (known after apply)
      + secondary_file_host               = (known after apply)
      ~ secondary_location                = "northeurope" -> (known after apply)
      + secondary_queue_endpoint          = (known after apply)
      + secondary_queue_host              = (known after apply)
      + secondary_table_endpoint          = (known after apply)
      + secondary_table_host              = (known after apply)
      + secondary_web_endpoint            = (known after apply)
      + secondary_web_host                = (known after apply)
        tags                              = {
            "environment" = "staging"
        }
        # (17 unchanged attributes hidden)

      ~ blob_properties {
          - change_feed_retention_in_days = 0 -> null
          + default_service_version       = (known after apply)
            # (3 unchanged attributes hidden)

            # (2 unchanged blocks hidden)
        }

      - network_rules {
          - bypass                   = [
              - "AzureServices",
            ] -> null
          - default_action           = "Allow" -> null
          - ip_rules                 = [] -> null
```

```
        - default_action             = "Allow" -> null
        - ip_rules                   = [] -> null
        - virtual_network_subnet_ids = [] -> null
      }

    - queue_properties {
        - hour_metrics {
            - enabled               = true -> null
            - include_apis          = true -> null
            - retention_policy_days = 7 -> null
            - version               = "1.0" -> null
          }
        - logging {
            - delete                = false -> null
            - read                  = false -> null
            - retention_policy_days = 0 -> null
            - version               = "1.0" -> null
            - write                 = false -> null
          }
        - minute_metrics {
            - enabled               = false -> null
            - include_apis          = false -> null
            - retention_policy_days = 0 -> null
            - version               = "1.0" -> null
          }
      }

    - share_properties {
        - retention_policy {
            - days = 7 -> null
          }
      }
  }

Plan: 2 to add, 0 to change, 2 to destroy.

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly
these actions if you run "terraform apply" now.
nikki@DESKTOP-I298DGN:~/terraform-aure$
```

1.6.2. Please answer the following questions:

- How many variables do we have defined, and which are they?
  - We defined 2 variables. They are “my_name” and “location”.
- Why did terraform asked us to input a value only for the my_name variable?
  - Terraform asked us to input a value only for the “my_name” variable because we don’t have default argument set. If default argument is present, the variable is considered to be optional and the default value will be used if no value is set when calling the module or running Terraform.

1.7. Define input variable value in file.

1.7.1. To allow automation we would need to have the non secret variables defined in a tfvars file named “inputs.tfvars” that you will create.

1.7.2. Define the value of the my_name variable inside the inputs.tfvars file like bellow:

```
my_name = <add_your_name_with_lower_letter_in_quotes>
```

1.7.3. Execute terraform plan command with the option –var-file=inputs.tfvars

1.7.4.The plan should try to destroy 2 resources and create 2 resources.

```
      - share_properties {
          - retention_policy {
              - days = 7 -> null
            }
        }
    }

Plan: 2 to add, 0 to change, 2 to destroy.
```

2. Using local values

2.1. In main.tf before the data block create locals block where we will define a local value named resource_prefix where we will concatenate the input variable my_name with the generated value from the random string resource like shown below:

```
locals {
  resource_prefix = "${var.my_name}${random_string.random.result}"
}
```

2.2. Add this resource_prefix as prefix of the name of the azurerm_resource_group and azurerm_storage_account resources. (This is very useful for standardizing and differentiating resources when deployed on portal.) Here is an example for the resources group and you should apply the same concept for the storage account.

```
resource "azurerm_resource_group" "example" {
  name     = "${local.resource_prefix}-rg"
  location = var.location
}
```

2.3. Execute the terraform plan with the input variable file switch. It should show you again 2 resources for destroy and 2 resources to create.

```
            - version                = "1.0" -> null
            - write                  = false -> null
          }
        - minute_metrics {
            - enabled                = false -> null
            - include_apis           = false -> null
            - retention_policy_days = 0 -> null
            - version                = "1.0" -> null
          }
      }

    - share_properties {
        - retention_policy {
            - days = 7 -> null
          }
      }
  }

Plan: 2 to add, 0 to change, 2 to destroy.
```

3. Using output values

3.1. To display some values from our resources we need to define the output values. For better visibility create an output.tf file where we will place all output values that we want to display after applying.

3.2. Inside the outputs.tf file define an output value named resource_group_name with the value of the name of the resource group that we create, like shown below:

```
output "resource_group_name" {
  value = azurerm_resource_group.example.name
  description = "The name of the resource group we deployed"
}
```

3.3. Do the same for the output value named storage_account_name where the value will be the name of the storage account by using the example from step 3.2.

3.4. Execute the terraform plan with the input variable file switch. It should show you again 2 resources to destroy and 2 resources to create. You will also see at the bottom that there will be outputs.

```
Plan: 2 to add, 0 to change, 2 to destroy.

Changes to Outputs:
  + resource_group_name  = "ana0j3it0ya-rg"
  + storage_account_name = "ana0j3it0yasa"
```

4. Understanding the reason why our resources are being destroyed.

4.1. When you execute terraform plan it will give you information about the resources and parameters that are being created with “+”, destroyed and recreated with “-/+”, the ones destroyed with “–“ and the ones that will be modified with “~”.

4.2. In this task we will need to go over our terraform plan and identify the reasons why our resources are being replaced.

4.2.1. Search for the term “forces replacement” and note the resource name and the parameter that forces replacement. Describe the reason behind it.

- The reason behind it is that we change the name of the resource group and the name of the storage account with the help of the input variables.

5. Apply the terraform code and write down the outputs.

```
Apply complete! Resources: 2 added, 0 changed, 2 destroyed.

Outputs:

resource_group_name = "ana0j3it0ya-rg"
storage_account_name = "ana0j3it0yasa"
nikki@DESKTOP-I298DGN:~/terraform-aure$
```