

## Task 1: Install terraform and Azure CLI

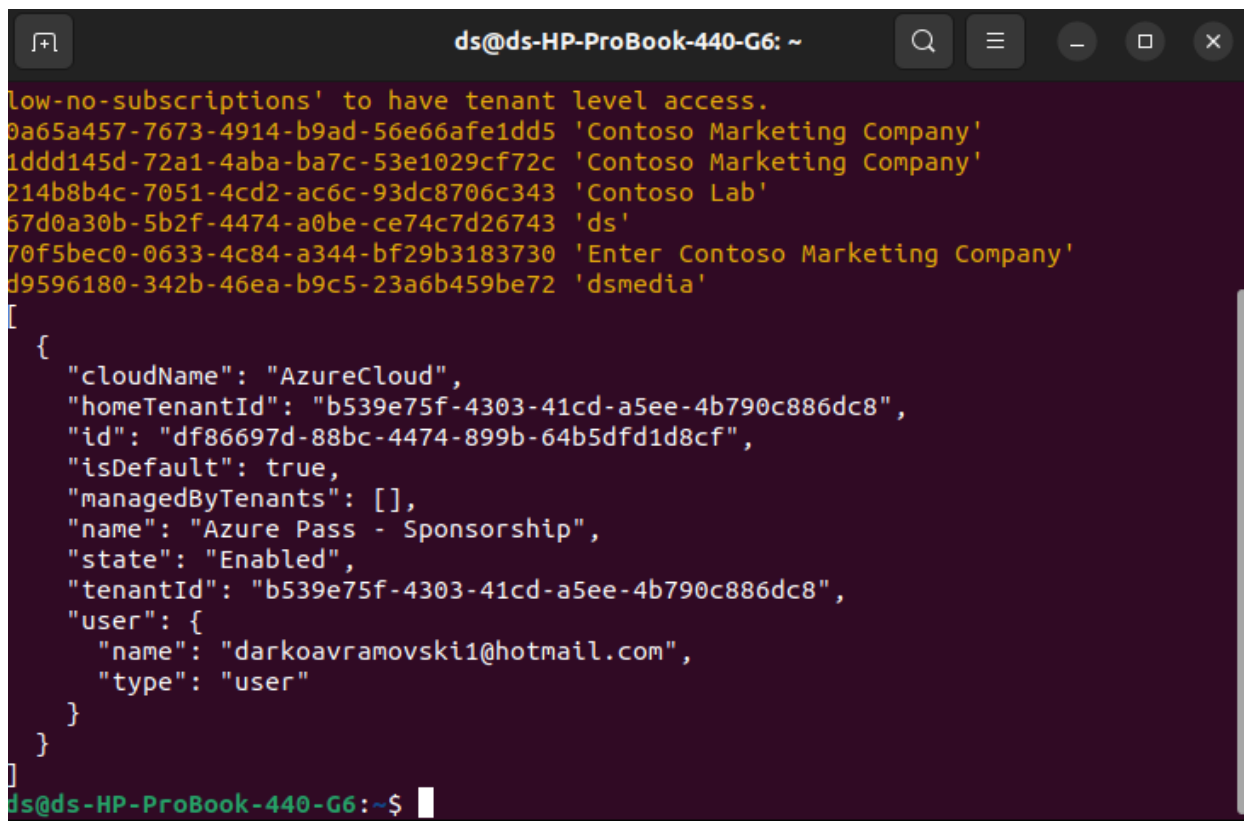
1. Use official guidelines to install the latest version of terraform and Azure CLI

### Install Terraform and Azure CLI

2. Authenticate with Azure CLI

az login

3. Set the exercise subscription as default for Azure CLI
4. Provide console print screen:



```
ds@ds-HP-ProBook-440-G6: ~  
low-no-subscriptions' to have tenant level access.  
9a65a457-7673-4914-b9ad-56e66afe1dd5 'Contoso Marketing Company'  
1ddd145d-72a1-4aba-ba7c-53e1029cf72c 'Contoso Marketing Company'  
214b8b4c-7051-4cd2-ac6c-93dc8706c343 'Contoso Lab'  
67d0a30b-5b2f-4474-a0be-ce74c7d26743 'ds'  
70f5bec0-0633-4c84-a344-bf29b3183730 'Enter Contoso Marketing Company'  
d9596180-342b-46ea-b9c5-23a6b459be72 'dsmedia'  
[  
  {  
    "cloudName": "AzureCloud",  
    "homeTenantId": "b539e75f-4303-41cd-a5ee-4b790c886dc8",  
    "id": "df86697d-88bc-4474-899b-64b5dfd1d8cf",  
    "isDefault": true,  
    "managedByTenants": [],  
    "name": "Azure Pass - Sponsorship",  
    "state": "Enabled",  
    "tenantId": "b539e75f-4303-41cd-a5ee-4b790c886dc8",  
    "user": {  
      "name": "darkoavramovski1@hotmail.com",  
      "type": "user"  
    }  
  }  
]  
ds@ds-HP-ProBook-440-G6:~$
```

PROF

- 4.1 Time and date when the exercise was worked

n the console type **date**



```
ds@ds-HP-ProBook-440-G6:~$ date  
Fri, 11 Aug 09:43:50 CEST 2023  
ds@ds-HP-ProBook-440-G6:~$
```

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

To see terraform version type

```
terraform --version
```

```
Terraform v1.4.4 on linux_amd64
```

```
ds@ds-HP-ProBook-440-G6:~$ terraform --version
Terraform v1.4.4
on linux_amd64
```

### 4.3 Azure CLI output of the current subscription

```
az version
```

```
{
  "environmentName": "AzureCloud",
  "homeTenantId": "b539e75f-4303-41cd-a5ee-4b790c886dc8",
  "id": "df86697d-88bc-4474-899b-64b5dfd1d8cf",
  "isDefault": true,
  "managedByTenants": [],
  "name": "Azure Pass - Sponsorship",
  "state": "Enabled",
  "tenantId": "b539e75f-4303-41cd-a5ee-4b790c886dc8",
  "user": {
    "name": "darkoavramovski1@hotmail.com",
    "type": "user"
  }
}
```

```
ds@ds-HP-ProBook-440-G6:~$ az account show
{
  "environmentName": "AzureCloud",
  "homeTenantId": "b539e75f-4303-41cd-a5ee-4b790c886dc8",
  "id": "df86697d-88bc-4474-899b-64b5dfd1d8cf",
  "isDefault": true,
  "managedByTenants": [],
  "name": "Azure Pass - Sponsorship",
  "state": "Enabled",
  "tenantId": "b539e75f-4303-41cd-a5ee-4b790c886dc8",
  "user": {
    "name": "darkoavramovski1@hotmail.com",
    "type": "user"
  }
}
```

PROF

## Task 2: Define your first terraform infrastructure code

### 1. Add minimal provider configuration and initialize terraform

#### 1.1 Create file called "main.tf"

create new 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).

```
terraform {
  required_providers {
```

```
azurerm = {  
  source = "hashicorp/azurerm"  
  version = "3.51.0"  
}  
}  
}  
  
provider "azurerm" {
```

## Configuration options

---

```
}
```

1.3 Save the file

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

Use this command to init the terraform.

```
terraform init
```

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)

```
ds@ds-HP-ProBook-440-G6: ~/Documents/homework/DevOps La...
Mid-Term
README.md
README.pdf
ds@ds-HP-ProBook-440-G6:~/Documents/homework$ cd DevOps\ Lab\ 24\ \ -\ Terraform/
ds@ds-HP-ProBook-440-G6:~/Documents/homework/DevOps Lab 24 - Terraform$ ls
images main.tf README.md README.pdf
ds@ds-HP-ProBook-440-G6:~/Documents/homework/DevOps Lab 24 - Terraform$ terraform
init

Initializing the backend...

Initializing provider plugins...
- Finding hashicorp/azurerm versions matching "3.51.0"...
- Installing hashicorp/azurerm v3.51.0...
- Installed hashicorp/azurerm v3.51.0 (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.
ds@ds-HP-ProBook-440-G6:~/Documents/homework/DevOps Lab 24 - Terraform$
```

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

Execute this command, and terraform it will throw you an error

```
terraform plan
```

PROF

```
ds@ds-HP-ProBook-440-G6:~/Documents/homework/DevOps Lab 24 - Terraform$ 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.
```

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

Error:

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.
```

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

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

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)

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

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**.

```
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/azurerm: locked version selection 3.
51.0 doesn't match the updated version constraints "3.35.0"
- provider registry.terraform.io/hashicorp/random: required by this configurati
on but no version is selected

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

now in the terminal execute

2.4.2 Now we have 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.)

```
terraform init --upgrade
```

```
ds@ds-HP-ProBook-440-G6:~/Documents/homework/DevOps Lab 24 - Terraform$ terraform init -upgrade
Initializing the backend...

Initializing provider plugins...
- Finding hashicorp/azurerm versions matching "> 3.35.0"...
- Finding latest version of hashicorp/random...
- Using previously-installed hashicorp/azurerm v3.51.0
- Installing hashicorp/random v3.4.3...
- Installed hashicorp/random v3.4.3 (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.
ds@ds-HP-ProBook-440-G6:~/Documents/homework/DevOps Lab 24 - Terraform$ terraform init -upgrade
```

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

```
terraform plan
```

we get an error

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?
  - o 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.
  - o 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

```
Error: Missing required argument
|
|   with azurerm_storage_account.example,
|   on main.tf line 36, in resource "azurerm_storage_account" "example":
|   36:         restore_policy {
|
|   "blob_properties.0.restore_policy": all of
|   `blob_properties.0.delete_retention_policy,blob_properties.0.restore_pol
|   icy`
|   must be specified
```

PROF

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)

```
terraform {
  required_providers {
    azurerm = {
      source = "hashicorp/azurerm"
      version = ">= 3.36.0"
    }
  }
}
```

now run terraform init -upgrade

```
Initializing the backend...
```

```
Initializing provider plugins...
```

- Finding latest version of hashicorp/random...
- Finding hashicorp/azurerm versions matching "> 3.35.0"...
- Using previously-installed hashicorp/random v3.4.3
- Using previously-installed hashicorp/azurerm v3.51.0

```
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.

Then run terraform plan

2.6.1 Again, we have issues with our code?

- Well in the latest provider documentation for "azurerm\_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.

PROF

```
ds@ds-HP-ProBook-440-G6:~/Documents/homework/DevOps Lab 24 - Terraform$ terraform plan
Error: Missing required argument

  with azurerm_storage_account.example,
  on main.tf line 36, in resource "azurerm_storage_account" "example":
   36:     restore_policy {

"blob_properties.0.restore_policy": all of `blob_properties.0.delete_retention_policy,blob_properties.0.restore_policy` must be specified
ds@ds-HP-ProBook-440-G6:~/Documents/homework/DevOps Lab 24 - Terraform$
```

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:

```
terraform plan
```

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?

Total output that are we getting are 3 as our plan in main.tf file

- random\_string
- resource\_group
- storage\_account

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

Location isn + **location = "westeurope"**

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)

Created one **Storage account**

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

one storage account

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

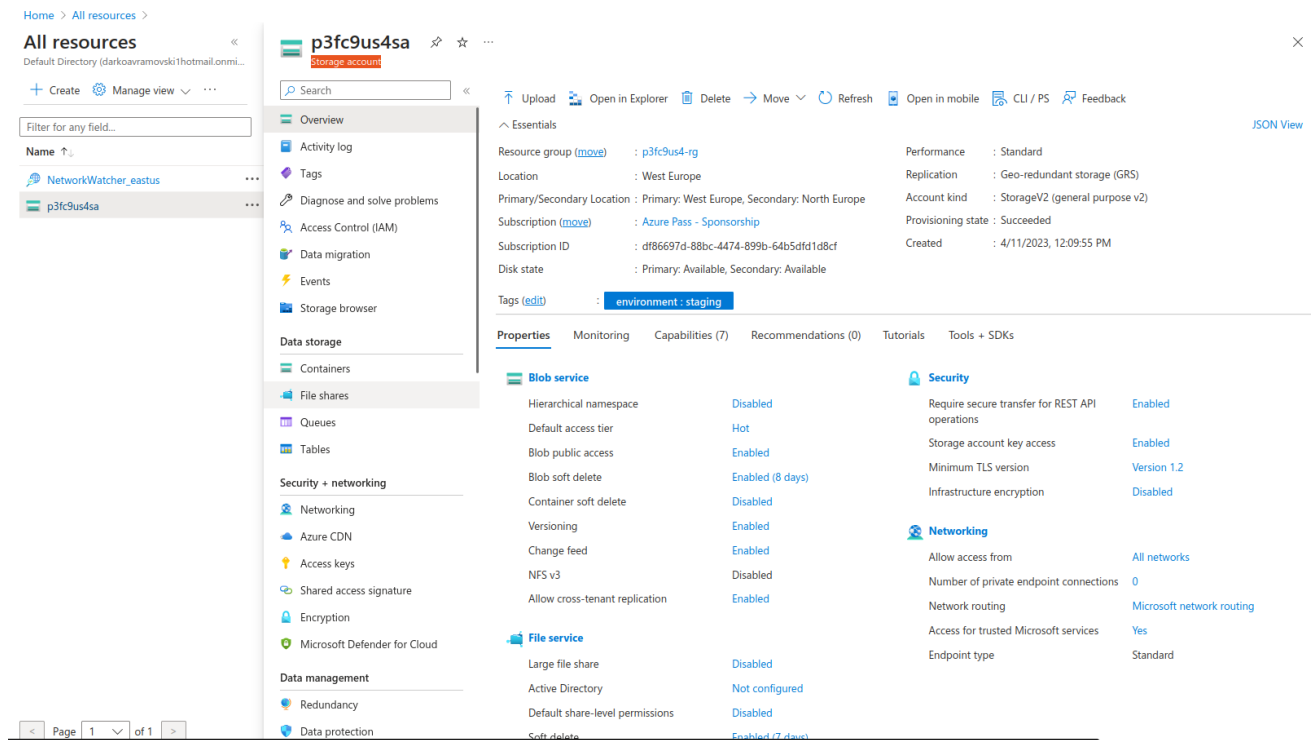
random\_string is not visible

Random String

—  
PROF

This is probably the resource in this provider that I make use of the most. It provides a lot of configurability to tailor to your needs. I use this to generate a random character set to append to resource names to make them unique. A big one of these is in Azure, the App Service name becomes part of the URL. When I have people running examples that I produce, this keeps the name unique enough to not have conflicts. Let's create a random string and an output variable.

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



### 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

create file variables.tf

1.2. Define the variable from the code bellow 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"  
}
```

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.

Create new file inputs.tfvars, and assign variable name

```
name = "darko"
```

## 1.6. Execute terraform plan.

### 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.

```
create mode 100644 DevOps Lab 24 - Terraform/terraform.tfstate
ds@ds-HP-ProBook-440-G6:~/Documents/homework/DevOps Lab 24 - Terraform$ terraform plan
var.my_name
  First name of the student
Enter a value: Darko

random_string.random: Refreshing state... [id=p3fc9us4]
data.azurerm_subscription.current: Reading...
azurerm_resource_group.example: Refreshing state... [id=/subscriptions/df86697d-88bc-4474-899b-64b5dfd1d8cf/resourceGroups/p3fc9us4-rg]
data.azurerm_subscription.current: Read complete after 1s [id=/subscriptions/df86697d-88bc-4474-899b-64b5dfd1d8cf]
azurerm_storage_account.example: Refreshing state... [id=/subscriptions/df86697d-88bc-4474-899b-64b5dfd1d8cf/resourceGroups/p3fc9us4-rg/providers/Microsoft.Storage/storageAccounts/p3fc9us4sa]

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.
```

### 1.6.2. Please answer the following questions:

- How many variables do we have defined, and which are they?

Answer: we need to assign one variable with **name**

- Why did terraform asked us to input a value only for the my\_name variable?

because we create file who accepts input values and input variable

### 1.7.2. Define the value of the my\_name variable inside the inputs.tfvars file like bellow:

```
name = "darko"
```

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

## 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 azure\_rm\_resource\_group and azure\_rm\_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.

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 for destroy and 2 resources to create. You will also see at the bottom that there will be outputs.

#### 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 "~".

```
Changes to Outputs:
+ resource_group_name = "p3fc9us4-rg"
+ storage_account_name = "p3fc9us4sa"

You can apply this plan to save these new output values to the Terraform state, without changing any real infrastructure.
```

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

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

```
Outputs:
resource_group_name = "p3fc9us4-rg"
storage_account_name = "p3fc9us4sa"
ds@ds-rf-Prodbook-440-66:~/documents/homework/DevOps_Lab_24 - Terraform$
```