

Exercise 2

1. Task 1: Setup and use remote terraform backend

1. In the previous exercise you have created code that creates a storage account. Now we will configure and use that storage account as our backend for terraform. (if you have deleted that storage account, just reapply your terraform code from your previous exercise)

```
PS C:\Users\filip> az account set --subscription "Azure subscription 1"
PS C:\Users\filip> az account show --output table
EnvironmentName      HomeTenantId          IsDefault      Name                State      TenantId
-----
AzureCloud           7e21abd0-0273-49d9-9c6d-e36814d6f388  True          Azure subscription 1 Enabled     7e21abd0-0273-49d9-9c6d-e36814d6f388
PS C:\Users\filip>
```

```
main.tf
C:\Users\filip\Desktop\SCALEFOCUS\HW24_Terraform_Modules\23try> main.tf
1 locals {
2   resource_prefix = "filip-${random_string.random.result}"
3 }
4
5 terraform {
6   required_providers {
7     azurerm = {
8       source = "hashicorp/azurerm"
9       version = "~> 3.36.0"
10    }
11  }
12 }
13
14 provider "azurerm" {
15   features {}
16 }
17
18 data "azurerm_subscription" "current" {
19   subscription_id = "1c7667de-3367-49d7-8883-06f316cda9b1"
20 }
21
22 resource "random_string" "random" {
23   length      = 8
24   special     = false
25   lower       = true
26   upper       = false
27 }
28
29 resource "azurerm_resource_group" "example" {
30   name       = "${local.resource_prefix}"
31   location   = "West Europe"
32 }
33
34 resource "azurerm_storage_account" "example" {
35   name                = "filipinopalace"
36   resource_group_name = azurerm_resource_group.example.name
37   location             = azurerm_resource_group.example.location
38   account_tier         = "Standard"
39   account_replication_type = "GRS"
40
41   tags = {
42     environment = "staging"
43   }
44 }
45
46 variable "my_name" {
47   type = string
48 }
```

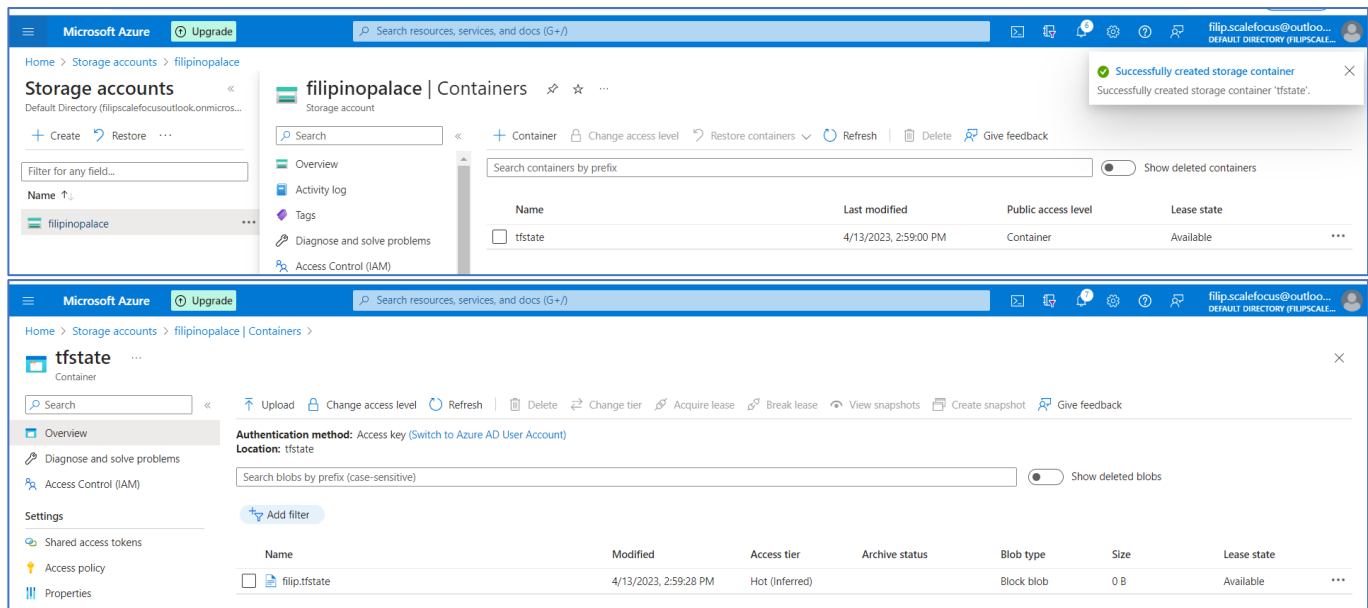
```
Enter a value: yes

azurerm_storage_account.example: Creating...
azurerm_storage_account.example: Still creating... [10s elapsed]
azurerm_storage_account.example: Still creating... [20s elapsed]
azurerm_storage_account.example: Creation complete after 24s [id=/subscriptions/1c7667de-3367-49d7-8883-06f316cda9b1/resourceGroups/filip-ly44thl/providers/Microsoft.Storage/storageAccounts/filipinopalace]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Users\filip\Desktop\scalefocus\HW24_Terraform_Modules\23try>
```

Microsoft Azure Upgrade Search resources, services, and docs (G+/I)					
Home > Storage accounts x ...					
Default Directory (filipscalefocusoutlook.onmicrosoft.com)					
+ Create Restore Manage view Refresh Export to CSV Open query Assign tags Delete					
Filter for any field... Subscription equals all Resource group equals all Location equals all Add filter					
Showing 1 to 1 of 1 records. No grouping List view					
<input type="checkbox"/>	Name ↑	Type ↑	Kind ↑	Resource group ↑	Location ↑
<input checked="" type="checkbox"/>	filipinopalace	Storage account	StorageV2	filip-ly44thl	West Europe

- 1.1. Create a Container in the storage account named "tfstate" with default settings.
- 1.2. Inside the created container, upload an empty file named "<my_name>.tfstate"



Please note that I encountered quite a few problems. So I started over from the beginning. I made a new storage account and continued from there.

The main.tf file code:

```
File Edit Selection View Go Run Terminal Help
main.tf - Visual Studio Code

1 terraform {
2   required_providers {
3     azure = {
4       source = "hashicorp/azure"
5       version = ">= 3.36.0"
6     }
7   }
8 }
9
10 provider "azure" {
11   features {}
12 }
13
14 resource "azurerm_resource_group" "example" {
15   name     = "example-resources"
16   location = "West Europe"
17 }
18
19 resource "azurerm_storage_account" "example" {
20   name                = "prodavnica"
21   resource_group_name = azurerm_resource_group.example.name
22   location             = azurerm_resource_group.example.location
23   account_tier         = "Standard"
24   account_replication_type = "GRS"
25   tags = {
26     environment = "staging"
27   }
28 }
29
30
```

Running the terraform apply command:

```
Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes

azurerm_resource_group.example: Creating...
azurerm_resource_group.example: Creation complete after 2s [id=/subscriptions/1c7667de-3367-49d7-8883-06f316cda9b1/resourceGroups/example-resources]
azurerm_storage_account.example: Creating...
azurerm_storage_account.example: Still creating... [10s elapsed]
azurerm_storage_account.example: Still creating... [20s elapsed]
azurerm_storage_account.example: Creation complete after 23s [id=/subscriptions/1c7667de-3367-49d7-8883-06f316cda9b1/resourceGroups/example-resources/providers/Microsoft.Storage/storageAccounts/prodavnica]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
PS C:\Users\filip\desktop\scalefocus\Terraform>
```

The resource group:

Microsoft Azure | Upgrade | Search resources, services, and docs (G+)

Home > Resource groups

Default Directory (https://scalefocusoutlook.onmicrosoft.com)

+ Create | Manage view | Refresh | Export to CSV | Open query | Assign tags

Filter for any field... | Subscription equals all | Location equals all | Add filter

Showing 1 of 1 records.

Name ↑↓	Subscription ↑↓	Location ↑↓
example-resources	Azure subscription 1	West Europe

The storage account:

Microsoft Azure | Upgrade | Search resources, services, and docs (G+)

Home > Resource groups > example-resources > prodavnica

Storage account

Search

Overview | Activity log | Tags | Diagnose and solve problems | Access Control (IAM) | Data migration | Events | Storage browser

Upload | Open in Explorer | Delete | Move | Refresh | Open in mobile | CLI / PS | Feedback

JSON View

Essentials

Resource group (move)	: example-resources	Performance	: Standard
Location	: West Europe	Replication	: Geo-redundant storage (GRS)
Primary/Secondary Location	: Primary: West Europe, Secondary: North Europe	Account kind	: StorageV2 (general purpose v2)
Subscription (move)	: Azure subscription 1	Provisioning state	: Succeeded
Subscription ID	: 1c7667de-3367-49d7-8883-06f316cda9b1	Created	: 13/04/2023, 15:17:15
Disk state	: Primary: Available, Secondary: Available		
Tags (edit)	: environment : staging		

Creating a storage container:

```
resource "azurerm_storage_container" "tfstate" {
  name = "tfstate"
  storage_account_name = azurerm_storage_account.example.name
  container_access_type = "private"
}
```

Terraform will perform the following actions:

```
# azurerm_storage_container.tfstate will be created
+ resource "azurerm_storage_container" "tfstate" {
+   container_access_type = "private"
+   has_immutability_policy = (known after apply)
+   has_legal_hold         = (known after apply)
+   id                     = (known after apply)
+   metadata               = (known after apply)
+   name                   = "tfstate"
+   resource_manager_id    = (known after apply)
+   storage_account_name  = "prodavnica"
}
```

Enter a value: yes

azurerm_storage_container.tfstate: Creating...

azurerm_storage_container.tfstate: Creation complete after 1s [id=https://prodavnica.blob.core.windows.net/tfstate]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.

PS C:\Users\filip\desktop\scalefocus\Terraform> |

Uploading an empty file:

Desktop > SCALEFOCUS > Terraform				
Name	Date modified	Type	Size	
.terraform	4/13/2023 3:15 PM	File folder		
.terraform.lock.hcl	4/13/2023 3:15 PM	HCL File	2 KB	
filip.tfstate	4/13/2023 3:34 PM	TFSTATE File	0 KB	
main.tf	4/13/2023 3:20 PM	TF File	1 KB	
terraform.tfstate	4/13/2023 3:23 PM	TFSTATE File	9 KB	
terraform.tfstate.backup	4/13/2023 3:23 PM	BACKUP File	8 KB	

```
PS C:\Users\filip\desktop\scalefocus\Terraform> az storage blob upload --account-name prodavnica --account-key vgoT9Tlt002Dga0sAWxCD5g9BsoaaTJ5A4c7ov6nEoxgB
UQs9yLHNXswhu+vg/uA5zyzw9wLo33o+AStdtIeGQ== --container-name tfstate --name filip.tfstate --type block --content-type application/json --content-encoding UT
F-8 --no-progress --verbose --auth-mode login --file C:\Users\filip\Desktop\SCALEFOCUS\Terraform\filip.tfstate
In "Login" auth mode, the following arguments are ignored: --account-key
Request URL: 'https://prodavnica.blob.core.windows.net/tfstate/filip.tfstate'
Request method: 'PUT'
Request headers:
  'x-ms-blob-type': 'REDACTED'
  'Content-Length': '0'
  'x-ms-blob-content-type': 'REDACTED'
  'x-ms-blob-content-encoding': 'REDACTED'
  'If-None-Match': '*'
  'x-ms-version': 'REDACTED'
  'Content-Type': 'application/octet-stream'
  'Accept': 'application/xml'
  'User-Agent': 'AZURECLI/2.46.0 (MSI) azsdk-python-storage-blob/12.12.0 Python/3.10.10 (Windows-10-10.0.22621-SP0)'
  'x-ms-client-request-id': 'f81a66e2-d9ff-11ed-8ce2-dc8b284d38e7'
  'CommandName': 'REDACTED'
  'ParameterSetName': 'REDACTED'
  'x-ms-date': 'REDACTED'
  'Authorization': 'REDACTED'
No body was attached to the request
Response status: 201
Response headers:
  'Content-Length': '0'
  'Content-MD5': 'REDACTED'
  'Last-Modified': 'Thu, 13 Apr 2023 13:34:52 GMT'
  'ETag': '"0x8DB3C23DC4833B0"'
  'Server': 'Windows-Azure-Blob/1.0 Microsoft-HTTPAPI/2.0'
  'x-ms-request-id': 'd90ecf0d-f01e-0040-4e0c-6e743e000000'
  'x-ms-client-request-id': 'f81a66e2-d9ff-11ed-8ce2-dc8b284d38e7'
  'x-ms-version': 'REDACTED'
  'x-ms-content-crc64': 'REDACTED'
  'x-ms-request-server-encrypted': 'REDACTED'
  'Date': 'Thu, 13 Apr 2023 13:34:51 GMT'
{
  "client_request_id": "f81a66e2-d9ff-11ed-8ce2-dc8b284d38e7",
  "content_md5": "1B2M2Y8AsgTpgAmY7PhCfG==",

```

Upload was successful:

```
{
  "Date": "Thu, 13 Apr 2023 13:34:51 GMT"
}
{
  "client_request_id": "f81a66e2-d9ff-11ed-8ce2-dc8b284d38e7",
  "content_md5": "1B2M2Y8AsgTpgAmY7PhCfG==",
  "date": "2023-04-13T13:34:51+00:00",
  "encryption_key_sha256": null,
  "encryption_scope": null,
  "etag": "\"0x8DB3C23DC4833B0\"",
  "lastModified": "2023-04-13T13:34:52+00:00",
  "request_id": "d90ecf0d-f01e-0040-4e0c-6e743e000000",
  "request_server_encrypted": true,
  "version": "2021-06-08",
  "version_id": null
}
Command ran in 1.821 seconds (init: 0.495, invoke: 1.326)
PS C:\Users\filip\desktop\scalefocus\Terraform>
```

2. Configure your terraform to use the azurearm remote backend

2.1. Read the terraform documentation for azurearm terraform backend configuration (use authentication using Azure CLI).

2.2. Add the backend configuration to your terraform code.

```
9   backend "azurerm" {  
10     storage_account_name = "prodavnica"  
11     container_name       = "tfstate"  
12     key                   = "terraform.tfstate"  
13     resource_group_name  = "example-resources"  
14   }  
15 }
```

2.2.1. Create a main.tf file and paste the code snippet from the documentation for backend configuration.




The screenshot shows the Visual Studio Code editor with a file named `main.tf`. The file path in the breadcrumb is `Users > filip > Desktop > SCALEFOCUS > Terraform > main.tf`. The code in the editor is as follows:


```
1 terraform {  
2   required_providers {  
3     azurerm = {  
4       source = "hashicorp/azurerm"  
5       version = ">= 3.36.0"  
6     }  
7   }  
8  
9   backend "azurerm" {  
10     storage_account_name = "prodavnica"  
11     container_name       = "tfstate"  
12     key                   = "terraform.tfstate"  
13     resource_group_name  = "example-resources"  
14   }  
15 }  
16  
17 provider "azurerm" {  
18   features {}  
19 }  
20  
21 resource "azurerm_resource_group" "example" {  
22   name       = "example-resources"  
23   location   = "West Europe"  
24 }  
25  
26 resource "azurerm_storage_container" "tfstate" {  
27   name                       = "tfstate"  
28   storage_account_name      = azurerm_storage_account.example.name  
29   container_access_type     = "private"  
30 }  
31  
32 resource "azurerm_storage_account" "example" {  
33   name                         = "prodavnica"  
34   resource_group_name         = azurerm_resource_group.example.name  
35   location                    = azurerm_resource_group.example.location  
36   account_tier                = "Standard"  
37   account_replication_type    = "GRS"  
38  
39   tags = {  
40     environment = "staging"  
41   }  
42 }  
43
```

2.2.2. Replace the values accordingly:

- **resource_group_name** – The resource group where the terraform state storage account can be found

Name ↑↓	Subscription ↑↓	Location ↑↓
 example-resources	Azure subscription 1	West Europe

- **storage_account_name** – The storage account name in which the state will be kept

Storage accounts					
Default Directory (filipscalefocusoutlook.onmicrosoft.com)					
+ Create Restore Manage view Refresh Export to CSV Open query Assign tags Delete					
Filter for any field... Subscription equals all Resource group equals all Location equals all Add filter					
Showing 1 to 1 of 1 records.					
No grouping	List view				
Name ↑↓	Type ↑↓	Kind ↑↓	Resource group ↑↓	Location ↑↓	Subscription ↑↓
 prodavnica	Storage account	StorageV2	example-resources	West Europe	Azure subscription 1

- **container_name** – The name of the container which will hold the blob with terraform state

prodavnica Containers			
Storage account			
Search			
+ Container Change access level Restore containers Refresh Delete Give feedback			
Search containers by prefix			
Show deleted containers			
Name	Last modified	Public access level	Lease state
<input type="checkbox"/> tfstate	4/13/2023, 3:23:03 PM	Private	Available

- **key** – the name of the empty file that you uploaded in step 1.2.

Upload Change access level Refresh Delete Change tier Acquire lease Break lease View snapshots Create snapshot Give feedback						
Authentication method: Access key (Switch to Azure AD User Account)						
Location: tfstate						
Search blobs by prefix (case-sensitive)						
Show deleted blobs						
Add filter						
Name	Modified	Access tier	Archive status	Blob type	Size	Lease state
<input type="checkbox"/> filip.tfstate	4/13/2023, 3:34:52 PM	Hot (Inferred)		Block blob	0 B	Available

2.3. In case where we have multiple backends configurations for different environments where we configure them in pipelines, we want to be able to switch between different backend configurations for local development also. For this we will use different backend file for each environment.

2.3.1. Create a subdirectory inside your current directory named “backends”

```
PS C:\Users\filip\desktop\scalefocus\Terraform> mkdir backends
```

```
Directory: C:\Users\filip\desktop\scalefocus\Terraform
```

Mode	LastWriteTime	Length	Name
d----	4/13/2023 3:48 PM		backends

2.3.2. Create a file named <my_name>_env_backend.tf

```
Directory: C:\Users\filip\desktop\scalefocus\Terraform\backends
```

Mode	LastWriteTime	Length	Name
-a----	4/13/2023 3:50 PM	0	filip_env_backend.tf

2.3.3. Move the content from the block – backend “azurerm” in main.tf to the <my_name>_env_backend.tf

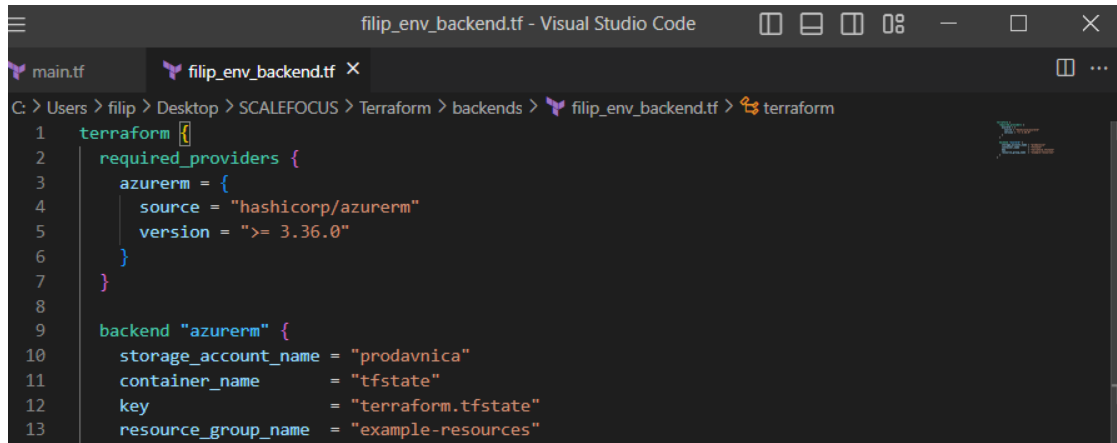
2.3.4. The files should have the following content:

- main.tf

```
terraform {
  backend "azurerm" {}
}
```

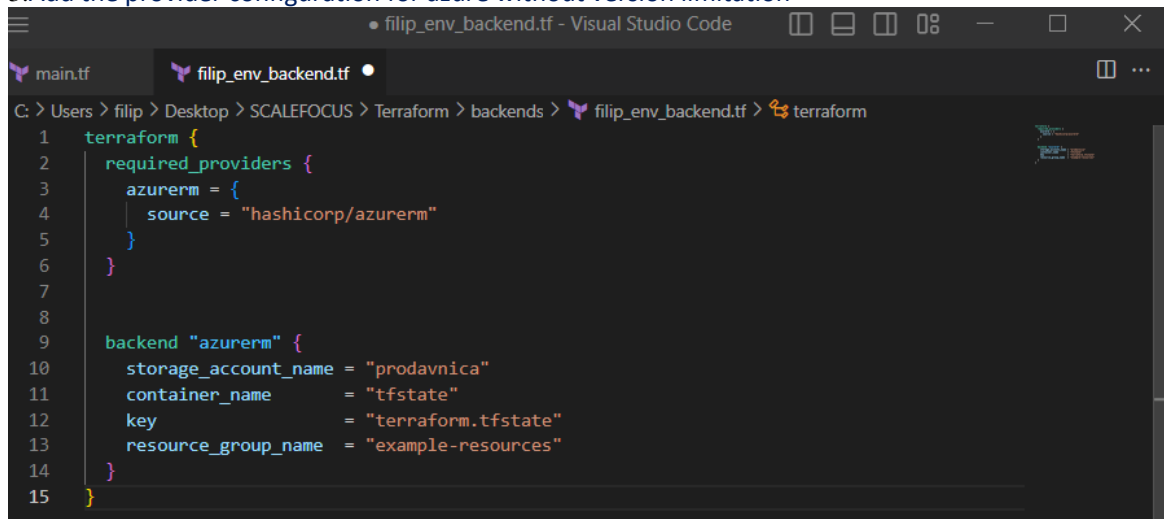
- <my_name>_env_backend.tf

```
resource_group_name = "boris0jbjtlfq-rg"
storage_account_name = "boris0jbjtlfqsa"
container_name      = "tfstate"
key                 = "boris.tfstate"
```



```
1 terraform {
2   required_providers {
3     azurerm = {
4       source = "hashicorp/azurerm"
5       version = ">= 3.36.0"
6     }
7   }
8 }
9 backend "azurerm" {
10  storage_account_name = "prodnvica"
11  container_name       = "tfstate"
12  key                  = "terraform.tfstate"
13  resource_group_name = "example-resources"
```

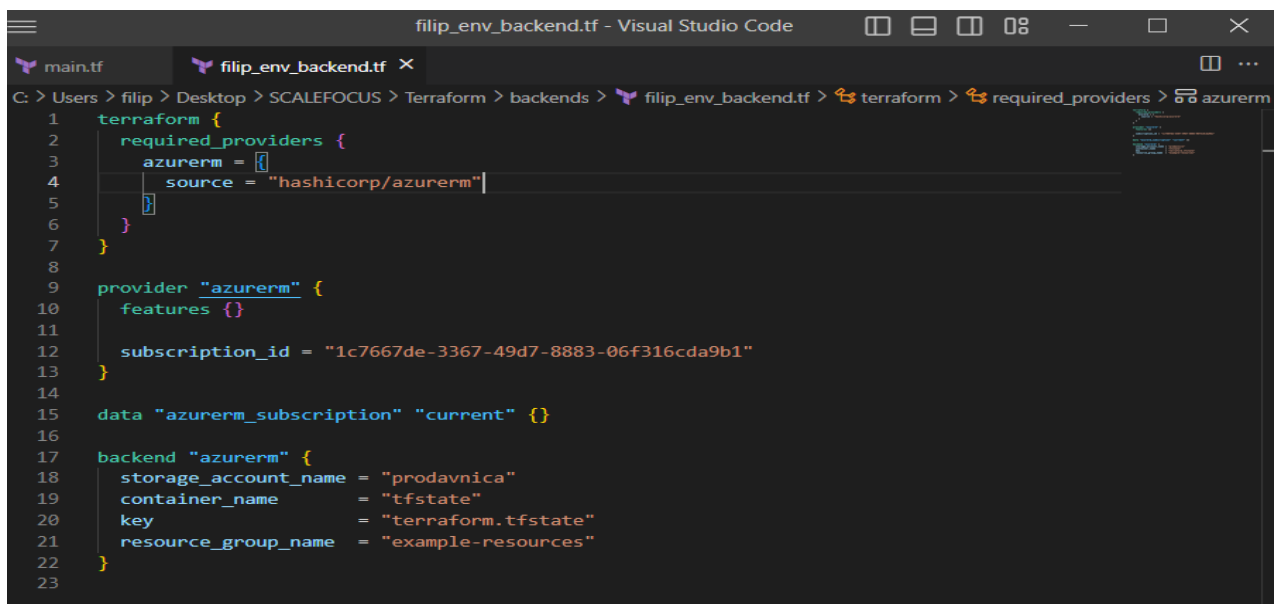
3. Add the provider configuration for azure without version limitation



```
1 terraform {
2   required_providers {
3     azurerm = {
4       source = "hashicorp/azurerm"
5     }
6   }
7 }
8
9 backend "azurerm" {
10  storage_account_name = "prodnvica"
11  container_name       = "tfstate"
12  key                  = "terraform.tfstate"
13  resource_group_name = "example-resources"
14 }
15 }
```

3.1. Follow the guidelines from previous exercise about the provider configuration

4. Initialize your terraform code with the azurerm_subscription data source



```
1 terraform {
2   required_providers {
3     azurerm = {
4       source = "hashicorp/azurerm"
5     }
6   }
7 }
8
9 provider "azurerm" {
10  features {}
11
12  subscription_id = "1c7667de-3367-49d7-8883-06f316cda9b1"
13 }
14
15 data "azurerm_subscription" "current" {}
16
17 backend "azurerm" {
18  storage_account_name = "prodnvica"
19  container_name       = "tfstate"
20  key                  = "terraform.tfstate"
21  resource_group_name = "example-resources"
22 }
23 }
```

- Add the following line in your code

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

5. Initialize your terraform code

```
PS C:\Users\filip\desktop\scalefocus\Terraform\backends> terraform init

Initializing the backend...

Successfully configured the backend "azurerm"! Terraform will automatically
use this backend unless the backend configuration changes.

Initializing provider plugins...
- Finding hashicorp/azurerm versions matching ">= 3.36.0"...
- 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.
PS C:\Users\filip\desktop\scalefocus\Terraform\backends>
```

The command was ran successfully. The screenshot above that I provided is the same as the one bellow.

5.1. Execute the following command

```
terraform init --backend-config=backends/<my_name>_env_backend.tf
```

5.2. The code should initialize successfully and should give you the following output

```
Initializing the backend...

Successfully configured the backend "azurerm"! Terraform will automatically
use this backend unless the backend configuration changes.

Initializing provider plugins...
- Finding latest version of hashicorp/azurerm...
- 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.
```

6. With this we have finalized our remote backend setup and we can define different backends and switch between them using the command option --backend-config

7.

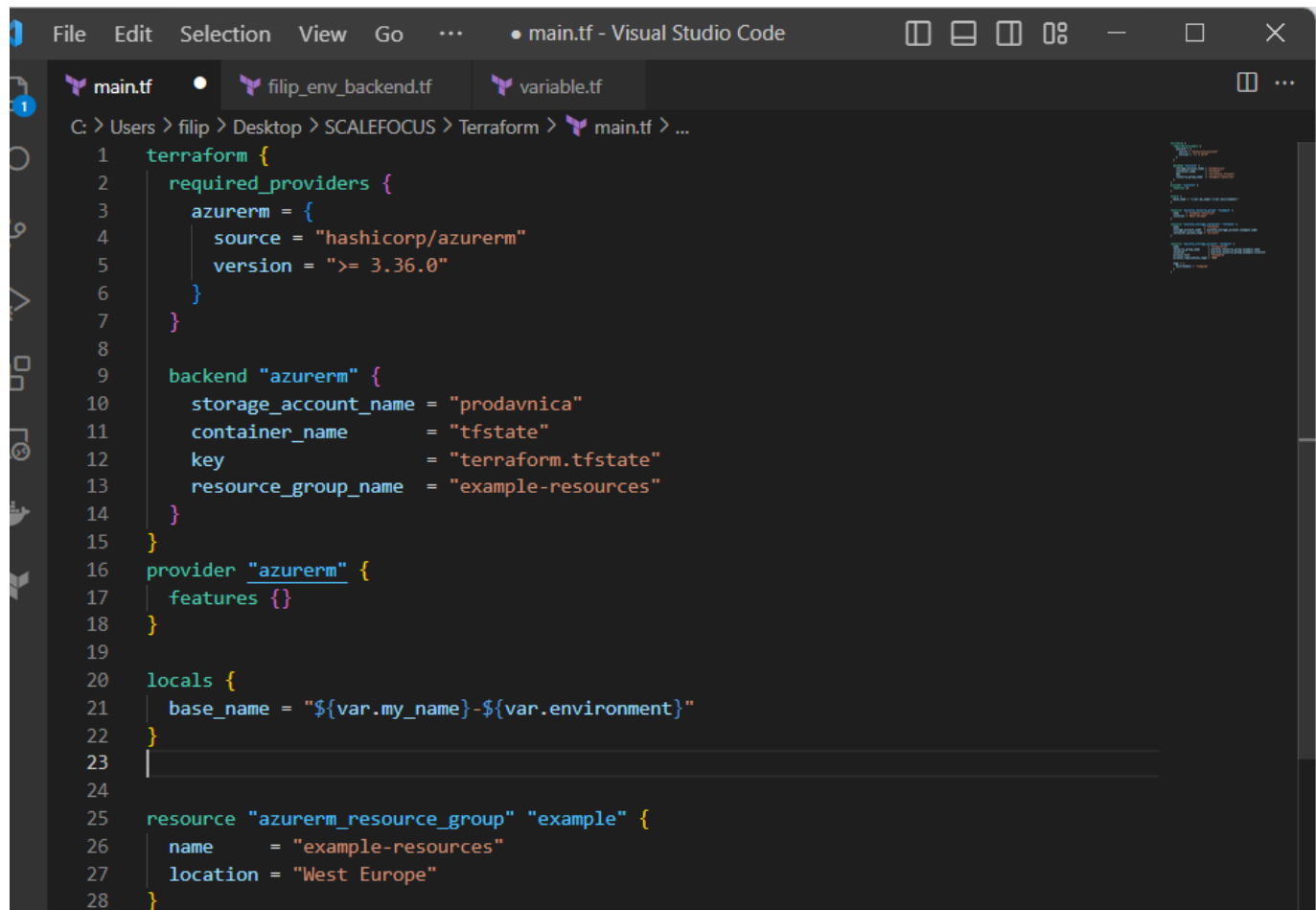
2. Task 2: Define the network resources from your second midterm assignment

1. During your midterm assignment we have defined few network resources that were shared and needed to be created before the virtual machine can be deployed and configured and were not owned by the VM itself. Those resources are:
 - resource group – base resource used for logical grouping of the resources
 - virtual network (VNet) – where the virtual machine will be hosted
 - subnet – the subnet where the virtual machine will be deployed
2. Since the resources that will be created are being managed from one terraform code, the simplest way of recognizing that is to use standardized naming for your resources and the simplest grouping of the resources in the code is to use same terraform resource names.

2.1. In the first exercise we have utilized the local vales to give standardized names to our resources. Here we will use the same approach.

2.1.1. Define local value named `base_name` with the concatenated values of variables `my_name` and `environment`

```
base_name = "${var.my_name}-${var.environment}"
```



```
File Edit Selection View Go ... main.tf - Visual Studio Code
main.tf filip_env_backend.tf variable.tf
C:\Users\filip\Desktop\SCALEFOCUS\Terraform> main.tf > ...
1 terraform {
2   required_providers {
3     azurerm = {
4       source = "hashicorp/azurerm"
5       version = ">= 3.36.0"
6     }
7   }
8
9   backend "azurerm" {
10    storage_account_name = "prodavnica"
11    container_name       = "tfstate"
12    key                  = "terraform.tfstate"
13    resource_group_name = "example-resources"
14  }
15 }
16 provider "azurerm" {
17   features {}
18 }
19
20 locals {
21   base_name = "${var.my_name}-${var.environment}"
22 }
23
24
25 resource "azurerm_resource_group" "example" {
26   name       = "example-resources"
27   location   = "West Europe"
28 }
```

2.1.2. Declare the variables my_name and environment in the variable.tf file

```
main.tf  filip_env_backend.tf  variable.tf X
C: > Users > filip > Desktop > SCALEFOCUS > Terraform > variable.tf > ...
1  variable "my_name" {
2    description = "The name"
3    type        = string
4  }
5
6  variable "environment" {
7    description = "The environment desc"
8    type        = string
9  }
10
```

2.1.3. Define values for the variables in your tfvars file

```
terraform.tfvars - Visual Studio Code
main.tf  filip_env_backend.tf  variable.tf  terraform.tfvars X
C: > Users > filip > Desktop > SCALEFOCUS > Terraform > terraform.tfvars > environment
1  my_name = "filip"
2  environment = "homework"
3
```

2.1.4. Define network resources network_base_name prefix

```
network_base_name = "${local.base_name}-ntwrk"
```

```
47 }
48
49 resource "azurerm_virtual_network" "example" {
50   name            = "${local.base_name}-vnet"
51   address_space   = ["10.0.0.0/16"]
52   location        = azurerm_resource_group.example.location
53   resource_group_name = azurerm_resource_group.example.name
54 }
55
56
```

2.2. Execute terraform plan with the input from your tfvars file

2.2.1. You should not see any errors and your output should be like below. If you have any errors, then you have written something incorrectly in your code. Try to resolve it by reading the error message.

```
data.azurerm_subscription.current: Reading...
data.azurerm_subscription.current: Read complete after 1s [id=/subscriptions/f65065a1-4bb6-48b4-bc72-e71cccfcfcc6]

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.
```

```
PS C:\Users\filip\desktop\scalefocus\Terraform> terraform plan
azurerm_resource_group.example: Refreshing state... [id=/subscriptions/1c7667de-3367-49d7-8883-06f316cda9b1/resourceGroups/example-resources]
azurerm_storage_account.example: Refreshing state... [id=/subscriptions/1c7667de-3367-49d7-8883-06f316cda9b1/resourceGroups/example-resources/providers/Microsoft.Storage/storageAccounts/prodavnica]
azurerm_storage_container.tfstate: Refreshing state... [id=https://prodavnica.blob.core.windows.net/tfstate]

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.
PS C:\Users\filip\desktop\scalefocus\Terraform>
```

```
PS C:\Users\filip\desktop\scalefocus\Terraform> terraform plan -var-file="C:\Users\filip\Desktop\SCALEFOCUS\Terraform\terraform.tfvars"
azurerm_resource_group.example: Refreshing state... [id=/subscriptions/1c7667de-3367-49d7-8883-06f316cda9b1/resourceGroups/example-resources]
azurerm_storage_account.example: Refreshing state... [id=/subscriptions/1c7667de-3367-49d7-8883-06f316cda9b1/resourceGroups/example-resources/providers/Microsoft.Storage/storageAccounts/prodavnica]
azurerm_storage_container.tfstate: Refreshing state... [id=https://prodavnica.blob.core.windows.net/tfstate]

No changes. Your infrastructure matches the configuration.

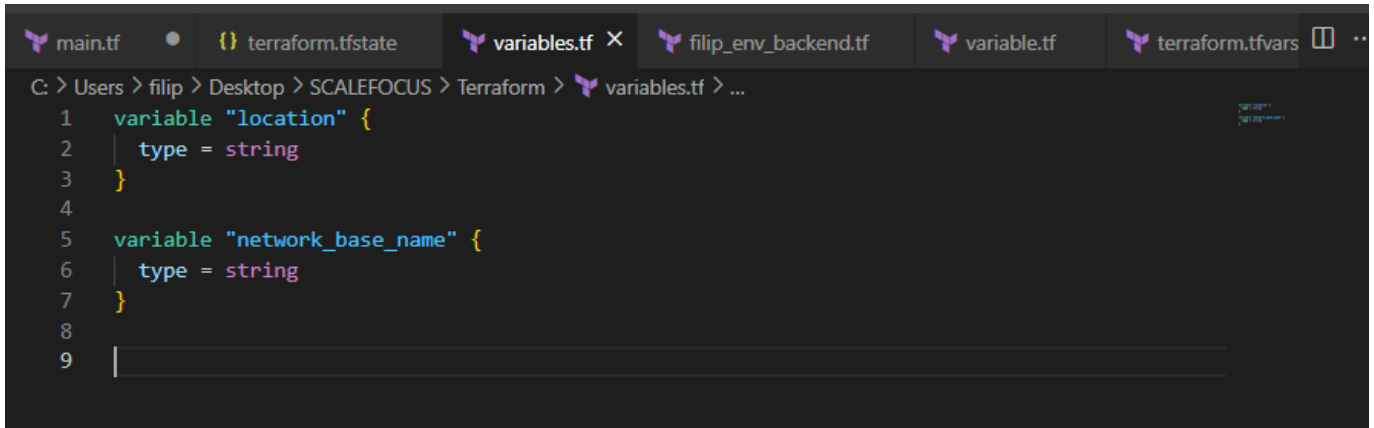
Terraform has compared your real infrastructure against your configuration and found no differences, so no changes are needed.
PS C:\Users\filip\desktop\scalefocus\Terraform>
```

3. Define the general network resources

3.1. Create a resource group with following parameters (see terraform registry documentation for `azurerm_resource_group`):

- Terraform resource name – `general_network`
- name – `${local.network_base_name}-rg`
- location – `var.location`

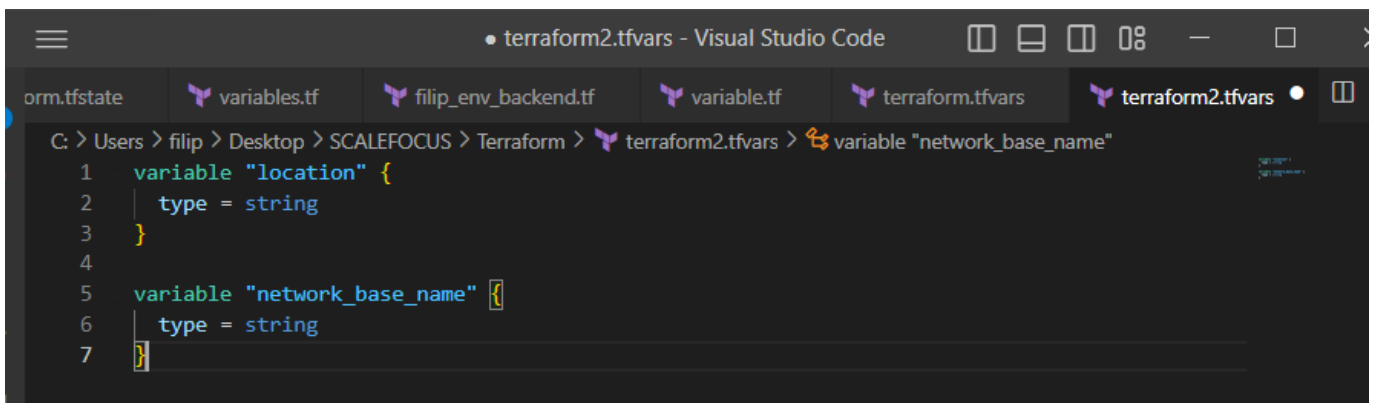
Defining variables. Note that this file is named `variables`, the previous one was `variable`.



The screenshot shows the Visual Studio Code editor with the `variables.tf` file open. The file contains two variable definitions:

```
1 variable "location" {
2   type = string
3 }
4
5 variable "network_base_name" {
6   type = string
7 }
8
9
```

Creating a `tfvars` file, it is named `terraform2`. So there is no confusion.



The screenshot shows the Visual Studio Code editor with the `terraform2.tfvars` file open. The file contains two variable definitions:

```
1 variable "location" {
2   type = string
3 }
4
5 variable "network_base_name" {
6   type = string
7 }
```

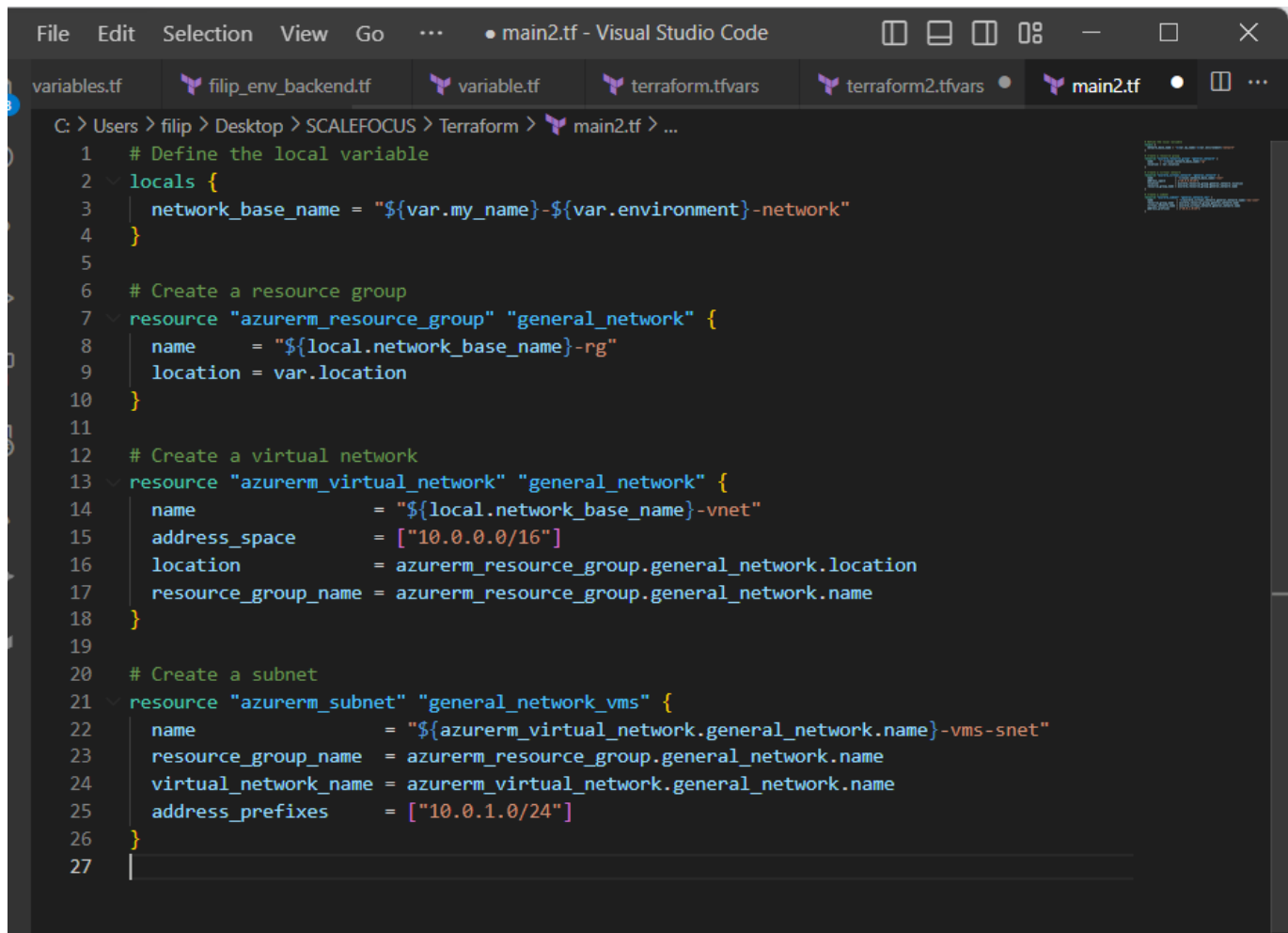
3.2. Create a virtual network with following parameters (`azurerm_virtual_network`):

- Terraform resource name – `general_network`
- name – `${local.network_base_name}-vnet`
- location – reference the `general_network` resource group location attribute
- resource_group_name – reference the `general_network` resource group name attribute
- address_space - `["10.0.0.0/16"]`

3.3. Create a subnet with following parameters (`azurerm_subnet`):

- Terraform resource name – `general_network_vms`
- name – `${azurerm_virtual_network.general_network.name}-vms-snet`
- resource_group_name – reference the `general_network` resource group name attribute
- virtual_network_name – reference the `general_network` virtual network name attribute
- address_prefixes - `["10.0.1.0/24"]`

3.4. Execute terraform plan with the input from your `tfvars` file



```
File Edit Selection View Go ... • main2.tf - Visual Studio Code
variables.tf filip_env_backend.tf variable.tf terraform.tfvars terraform2.tfvars • main2.tf ...

C:\Users\filip\Desktop\SCALEFOCUS\Terraform> main2.tf > ...
1 # Define the local variable
2 locals {
3     network_base_name = "${var.my_name}-${var.environment}-network"
4 }
5
6 # Create a resource group
7 resource "azurerm_resource_group" "general_network" {
8     name      = "${local.network_base_name}-rg"
9     location = var.location
10 }
11
12 # Create a virtual network
13 resource "azurerm_virtual_network" "general_network" {
14     name            = "${local.network_base_name}-vnet"
15     address_space   = ["10.0.0.0/16"]
16     location        = azurerm_resource_group.general_network.location
17     resource_group_name = azurerm_resource_group.general_network.name
18 }
19
20 # Create a subnet
21 resource "azurerm_subnet" "general_network_vms" {
22     name                = "${azurerm_virtual_network.general_network.name}-vms-snet"
23     resource_group_name = azurerm_resource_group.general_network.name
24     virtual_network_name = azurerm_virtual_network.general_network.name
25     address_prefixes     = ["10.0.1.0/24"]
26 }
27
```

3.4.1. Your plan should not throw any errors. If any errors found troubleshoot your code from the error information.

```
Terraform has compared your real infrastructure against your configuration and found no differences, so no changes are needed.
```

```
PS C:\Users\filip\desktop\scalefocus\Terraform> terraform plan
```

```
var.location
```

```
Enter a value: Skopje
```

```
var.network_base_name
```

```
Enter a value: Domasna
```

```
Terraform has compared your real infrastructure against your configuration and found no differences, so no changes are needed.
```

```
PS C:\Users\filip\desktop\scalefocus\Terraform> terraform plan
```

```
var.location
```

```
Enter a value: Skopje
```

```
var.network_base_name
```

```
Enter a value: Domasna
```

```
azurerm_resource_group.example: Refreshing state... [id=/subscriptions/1c7667de-3367-49d7-8883-06f316cda9b1/resourceGroups/example-resources]
```

```
azurerm_storage_account.example: Refreshing state... [id=/subscriptions/1c7667de-3367-49d7-8883-06f316cda9b1/resourceGroups/example-resources/providers/Microsoft.Storage/storageAccounts/prodavnica]
```

```
azurerm_storage_container.tfstate: Refreshing state... [id=https://prodavnica.blob.core.windows.net/tfstate]
```

```
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.
```

```
Releasing state lock. This may take a few moments...
```

```
PS C:\Users\filip\desktop\scalefocus\Terraform>
```

1. Task 3: Define and group the virtual machine and its resources into a module

1. Since we have already defined the base resources that are general and not directly related to the virtual machine, now we can start working on the linux virtual machine declaration. If we look at the resource `azurerm_linux_virtual_machine` in terraform registry, there are some other resources (components) of the virtual machine that need to be declared first and assigned to the virtual machine. Those resources are:
 - resource group – the group where the virtual machine will be placed. We will use this resource group for the resto of the VM components also, so we can have clear understanding what belongs and where
 - public IP – this is not related directly to our virtual machine but to the network interface that is used by the virtual machine
 - network interface – we need to define a network interface before we create a virtual machine
 - network security group (NSG) – which will be configured for management and service public access by the virtual machine
 - assignment of the NSG to the network interface – this is a separate resource in terraform because of the API functionality of the cloud provider
2. Because here we will also use a module to group the resources configuration, the first step for the module creation is the creation of the directory and the basic module configurations.
 - 2.1. In your current directory create a subdirectory named “vm_module”.

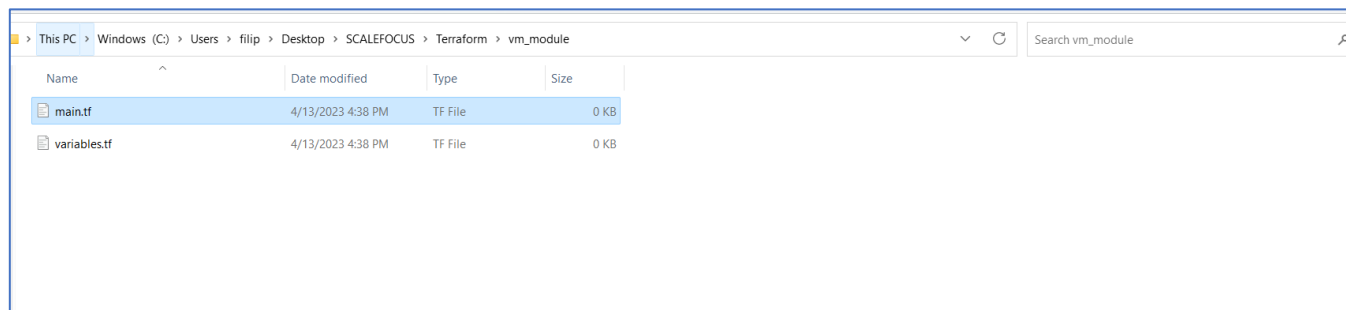
```
PS C:\Users\filip\desktop\scalefocus\Terraform> mkdir vm_module

Directory: C:\Users\filip\desktop\scalefocus\Terraform

Mode                LastWriteTime         Length Name
----                -
d-----          4/13/2023   4:35 PM              vm_module

PS C:\Users\filip\desktop\scalefocus\Terraform>
```

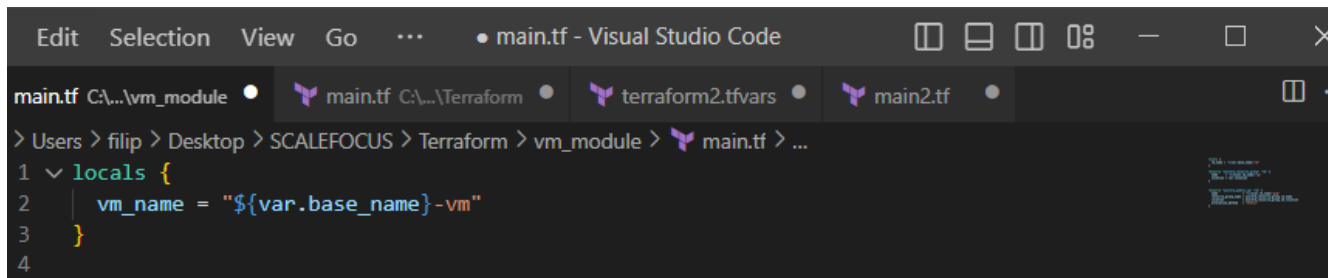
- 2.2. Next, inside the directory we will create our required files for terraform code and variables. Inside the directory create files named `main.tf` and `variables.tf`



3. Since the components here are being defined to be used by the VM as resource, we will create one common name which will be the base for the resources.

- 3.1. Follow the instructions from previous task and define local value name "vm_name" in main.tf, that will have the concatenated value of the base name variable and abbreviated with vm (don't forget to declare the variable in variable.tf):

```
vm_name = "${var.base_name}-vm"
```



```
main.tf C:\...\vm_module • main.tf C:\...\Terraform • terraform2.tfvars • main2.tf •
> Users > filip > Desktop > SCALEFOCUS > Terraform > vm_module > main.tf > ...
1  locals {
2      vm_name = "${var.base_name}-vm"
3  }
4
```

4. Define the resources

- 4.1. Define the azurerm_resource_group resource with following parameters:

- Terraform resource name – vm
- name - \${local.vm_name}-rg
- location – var.location

```
4
5  resource "azurerm_resource_group" "vm" {
6      name      = "${local.vm_name}-rg"
7      location = var.location
8  }
9
```

- 4.2. Define the azurerm_public_ip with the following parameters:

- Terraform resource name – vm
- name - \${local.vm_name}-pip
- resource_group_name – reference the name attribute of the vm resource group
- location – reference the location attribute of the vm resource group
- allocation_method – Static

```
10
11 resource "azurerm_public_ip" "vm" {
12     name                = "${local.vm_name}-pip"
13     resource_group_name = azurerm_resource_group.vm.name
14     location             = azurerm_resource_group.vm.location
15     allocation_method    = "Static"
16 }
17 |
```

4.3. Define the azurerm_network_interface resource with following parameters:

- Terraform resource name – vm
- name - \${local.vm_name}-nic
- resource_group_name – reference the name attribute of the vm resource group
- location – reference the location attribute of the vm resource group
- under the ip_configuration block define the following parameters:
 - name – external
 - subnet_id – var.vms_subnet_id (define the variable in the variables.tf file)
 - private_ip_address_allocation – Dynamic
- public_ip_address_id – reference the attribute id from the vm public ip resource

```

17
18 resource "azurerm_network_interface" "vm" {
19     name                = "${local.vm_name}-nic"
20     location            = azurerm_resource_group.vm.location
21     resource_group_name = azurerm_resource_group.vm.name
22
23     ip_configuration {
24         name                = "external"
25         subnet_id          = var.vms_subnet_id
26         private_ip_address_allocation = "Dynamic"
27         public_ip_address_id = azurerm_public_ip.vm.id
28     }
29 }
30

```

4.4. Define the azurerm_network_security_group resource with the following parameters

- Terraform resource name – vm
- name - \${azurerm_network_interface.vm.name}-nsg
- resource_group_name – reference the name attribute of the vm resource group
- location – reference the location attribute of the vm resource group
- define the first security_rule block with following parameters
 - name – allow_ssh_from_my_ip
 - priority – 110
 - direction – Inbound
 - access – Allow
 - protocol – Tcp
 - destination_port_range – 22
 - source_address_prefix – var.my_public_ip (define the variable in the variables file)
 - destination_address_prefix - ""
 - source_port_range - ""

```

resource "azurerm_network_security_group" "vm" {
  name                       = "${azurerm_network_interface.vm.name}-nsg"
  resource_group_name       = azurerm_resource_group.vm.name
  location                  = azurerm_resource_group.vm.location

  security_rule {
    name                     = "allow_ssh_from_my_ip"
    priority                 = 110
    direction               = "Inbound"
    access                   = "Allow"
    protocol                 = "Tcp"
    destination_port_range  = "22"
    source_address_prefix   = var.my_public_ip
    destination_address_prefix = "*"
    source_port_range       = "*"
  }
}

```

- define the second security_rule block with following parameters

- name – allow_http_from_my_ip
- priority – 100
- direction – Inbound
- access – Allow
- protocol – Tcp
- destination_port_range – 80
- source_address_prefix – var.my_public_ip
- destination_address_prefix - "*"
 - source_port_range - "*"
 - 48

```

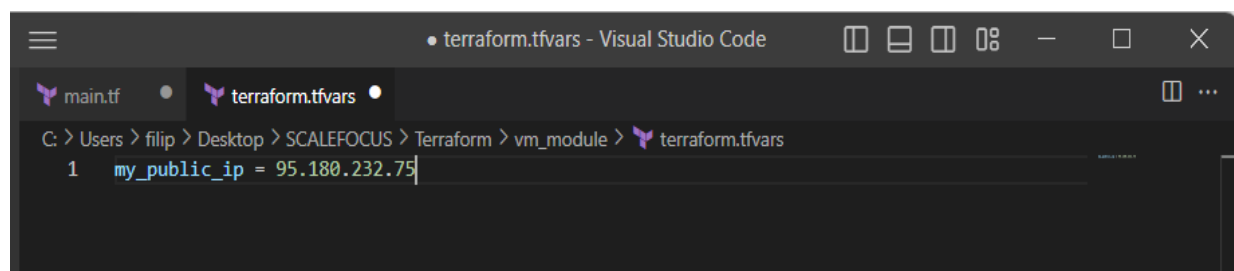
48
49  resource "azurerm_network_security_rule" "http_inbound" {
50      name                     = "allow_http_from_my_ip"
51      priority                 = 100
52      direction               = "Inbound"
53      access                   = "Allow"
54      protocol                 = "Tcp"
55      source_port_range        = "*"
56      destination_port_range   = 80
57      source_address_prefix    = var.my_public_ip
58      destination_address_prefix = "*"
59      resource_group_name      = azurerm_network_interface.vm.resource_group_name
60      network_interface_id     = azurerm_network_interface.vm.id
61  }
62

```


4.5. Now we need to associate the network interface and the network security group using the resource called `network_interface_security_group_association` with following parameters:

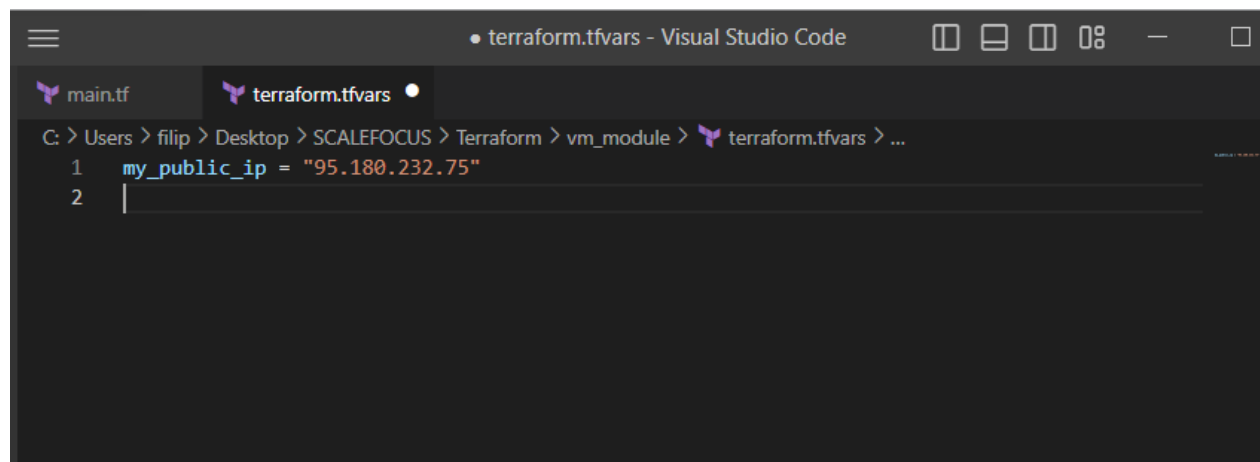
- Terraform resource name - `vm_nsg_to_vm_nic`
- `network_interface_id` – reference the id attribute of the vm nic resource
- `network_security_group_id` – reference the id attribute of the vm nsg resource

```
3
4 resource "azurerm_network_interface_security_group_association" "vm_nsg_to_vm_nic" {
5     network_interface_id      = azurerm_network_interface.vm.id
6     network_security_group_id = azurerm_network_security_group.vm.id
7 }
8
```



```
PS C:\Users\filip\desktop\scalefocus\Terraform\vm_module> terraform plan
Error: Invalid number literal
on terraform.tfvars line 1:
1: my_public_ip = 95.180.232.75
Failed to recognize the value of this number literal.
PS C:\Users\filip\desktop\scalefocus\Terraform\vm_module>
```

Got an error because I did not enter the ip in “”.



4.6. And at the end we came to the point where we need to define the virtual machine resource `azurerm_linux_virtual_machine` with following parameters:

- Terraform resource name – `web_srv`
- `name` – the local value `vm_name`
- `resource_group_name` – reference the name attribute of the vm resource group
- `location` – reference the location attribute of the vm resource group

- size – Standard_ B2s
- admin_username – adminuser
- network_interface_ids – [azure_rm_network_interface.vm.id]
- admin_password – var.my_password
- disable_password_authentication – false

The screenshot shows the Visual Studio Code editor with the file 'variables.tf' open. The file contains six variable declarations, each with a name in quotes and an empty object as a default value. The line numbers 1 through 7 are visible on the left margin.

```

1  variable "base_name" {}
2  variable "location" {}
3  variable "vnet_name" {}
4  variable "subnet_name" {}
5  variable "my_public_ip" {}
6  variable "my_password" {}
7  |

```

The screenshot shows the Visual Studio Code editor with the file 'terraform.tfvars' open. The file contains six assignments of values to the variables defined in the previous file. The line numbers 1 through 6 are visible on the left margin.

```

1  base_name = "my-vm"
2  location = "West Europe"
3  vnet_name = "my-vnet"
4  subnet_name = "my-subnet"
5  my_public_ip = "95.180.232.75"
6  my_password = "filiposkopje"

```

- in the os_disk block define the following parameters:
 - caching – ReadWrite
 - storage_account_type - Standard_LRS
- In the source_image_reference block define the following parameters:
 - publisher – Canonical
 - offer – UbuntuServer
 - sku - 18.04-LTS
 - version – latest

```
File Edit Selection View Go ... main.tf - Visual Studio Code
C:\...vm_module variables.tf C:\...vm_module terraform.tfvars main.tf C:\...VM variables.tf C:\...VM ...
C:\Users> filip > Desktop > SCALEFOCUS > Terraform > VM > main.tf > resource "azurerm_linux_virtual_machine" "web_srv"
50     name = local.vm_name
51     subnet_id = azurerm_subnet.vm.id
52     private_ip_address_allocation = "Dynamic"
53     public_ip_address_id = azurerm_public_ip.vm.id
54 }
55 }
56
57 resource "azurerm_network_security_rule" "http_inbound" {
58     name = "http_inbound"
59     priority = 100
60     direction = "Inbound"
61     access = "Allow"
62     protocol = "Tcp"
63     source_port_range = "*"
64     destination_port_range = "80"
65     source_address_prefix = "*"
66     destination_address_prefix = "*"
67     network_security_group_name = azurerm_network_security_group.vm.name
68 }
69
70 resource "azurerm_network_interface_security_group_association" "vm" {
71     network_interface_id = azurerm_network_interface.vm.id
72     network_security_group_id = azurerm_network_security_group.vm.id
73 }
74
75 resource "azurerm_linux_virtual_machine" "web_srv" {
76     name = local.vm_name
77     resource_group_name = azurerm_resource_group.vm.name
78     location = azurerm_resource_group.vm.location
79     size = "Standard_B2s"
80     admin_username = "adminuser"
81     network_interface_ids = [azurerm_network_interface.vm.id]
82
83     os_disk {
84         caching = "ReadWrite"
85         storage_account_type = "Standard_LRS"
86     }
87
88     source_image_reference {
89         publisher = "Canonical"
90         offer = "UbuntuServer"
91         sku = "18.04-LTS"
92         version = "latest"
93     }
94 }
```

```
69
70 resource "azurerm_network_interface_security_group_association" "vm" {
71     network_interface_id = azurerm_network_interface.vm.id
72     network_security_group_id = azurerm_network_security_group.vm.id
73 }
74
```

```
PS C:\Users\filip\desktop\scalefocus\Terraform\VM> terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/azurerm...
- 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.
PS C:\Users\filip\desktop\scalefocus\Terraform\VM>
```

- Since we defined all module resources, now we need to call the module from our root main.tf file
 - Declare the module named vm with source in the directory vm_module:

```
module "vm" {
  source      = "../vm_module"
  base_name   = local.base_name
  location    = var.location
}
```

4.7. Add the rest of the variables that you have defined with their respective values:

- vms_subnet_id – reference the id attribute from the general_network_vms subnet resource
- my_public_ip – define it as input variable in the variables file and assign your public ip in the tfvars file
- my_password – define it as input variable in the variables file and assign value in the tfvars file (not recommended in real use case)

4.8. When we add a module to our code, we need to reinitialize our terraform code

4.9. Now we can execute terraform plan with input from tfvars file

4.10. Since we have declared the public ip in the code, we would need to get it as an output from our code

4.10.1. In the vm_module directory create file named outputs.tf

4.10.2. Inside the file define an output value named "vm_public_ip" for the attribute ip_address from the vm public ip resource

4.10.3. Reference the output value as your code output value like bellow:

```
output "vm_public_ip" {
  value = module.vm.vm_public_ip
}
```

4.11. As last execute another terraform plan with input variables from file and it should be without any errors. If you have errors correct them.

5. Apply the terraform code
6. Get the public IP from the output
7. Connect to the vm using ssh with user adminuser. Usually takes around 2 minutes for the VM to be ready.
8. When successfully connected you should see the prompt from the VM. Provide print screen from it.
9. Review all of your resources on the subscription and provide print screen of the resource groups.
10. Take the time to observe the resources that you created and how the code reflects to the resources

11. Once completed, destroy the resources that you created using terraform destroy command with the parameter from input tfvars file (similar to plan and apply)
12. Once the destroy command has completed go with your terminal context to the directory from exercise 1 and execute terraform destroy over there also using the tfvars file input parameter
13. With this you have successfully completed the exercise and cleaned your subscription from the resources created from your code. (There will be a network watcher resource that is being created by azure automatically when we add vnet, and you can delete it manually)