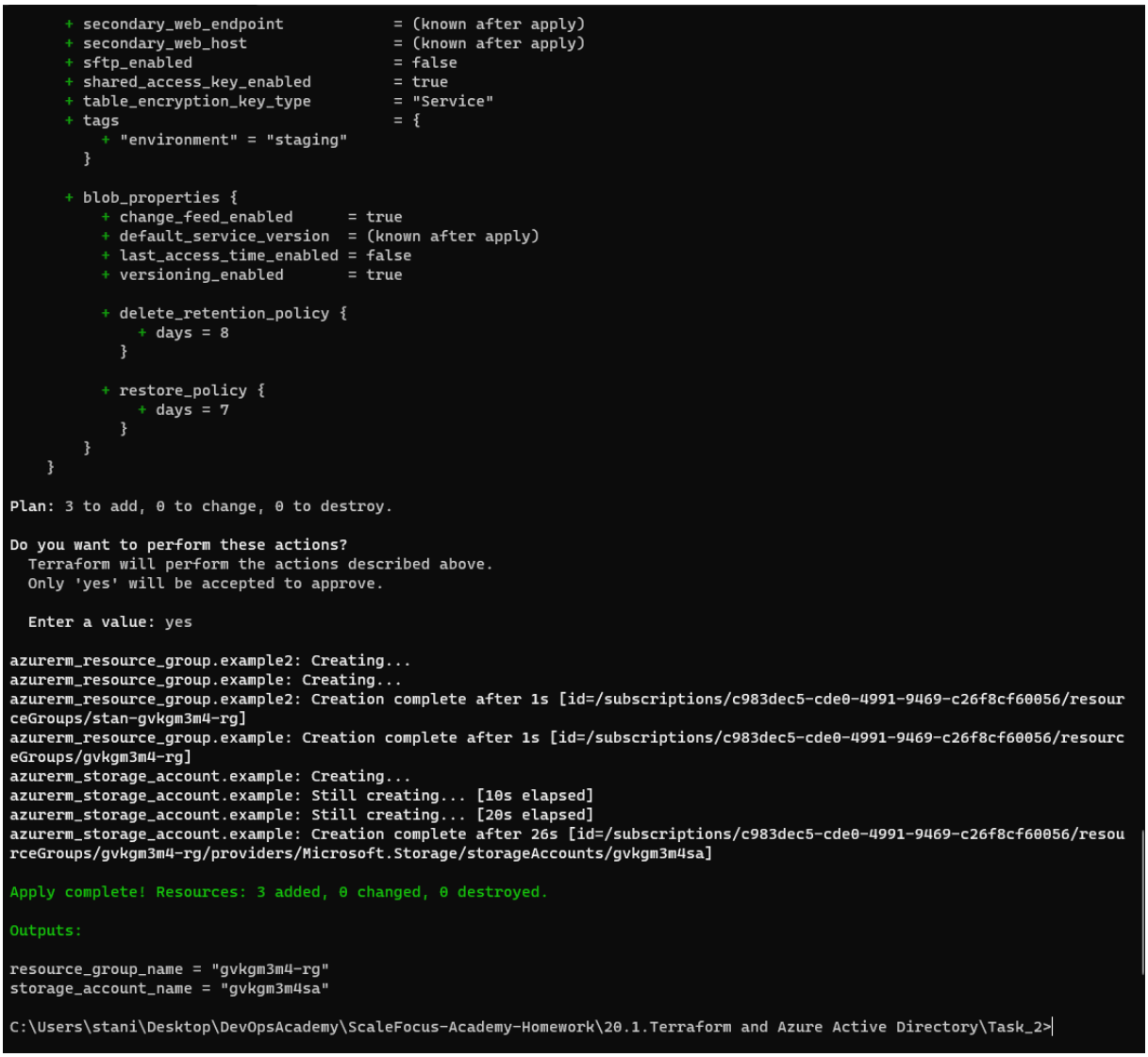
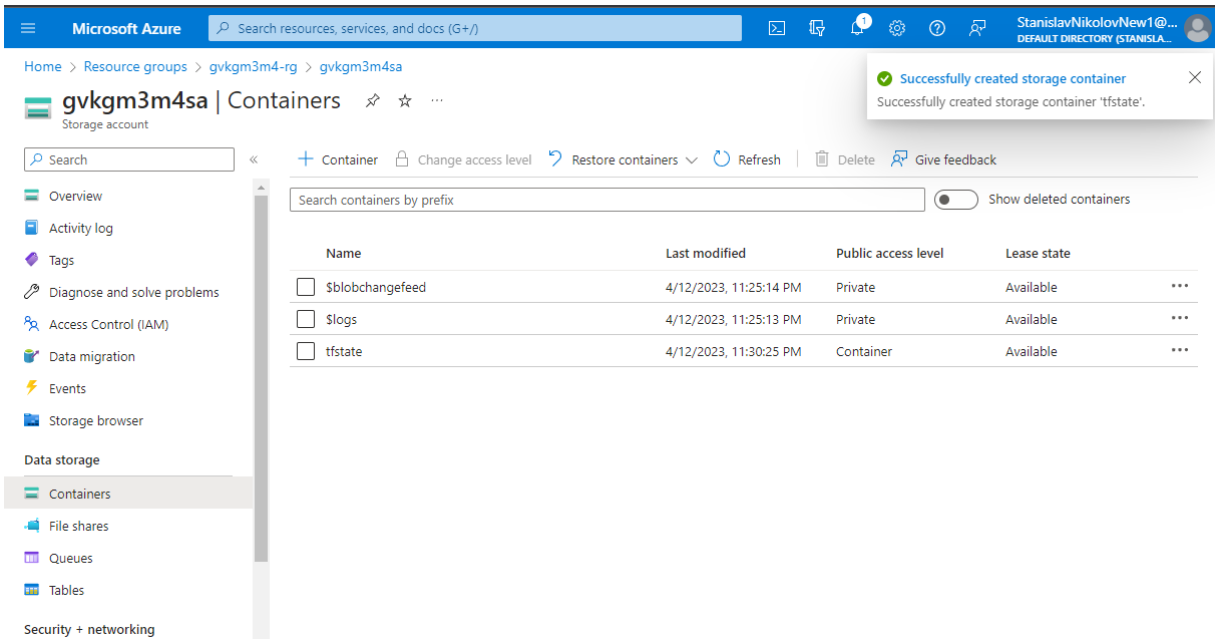
Terraform modules

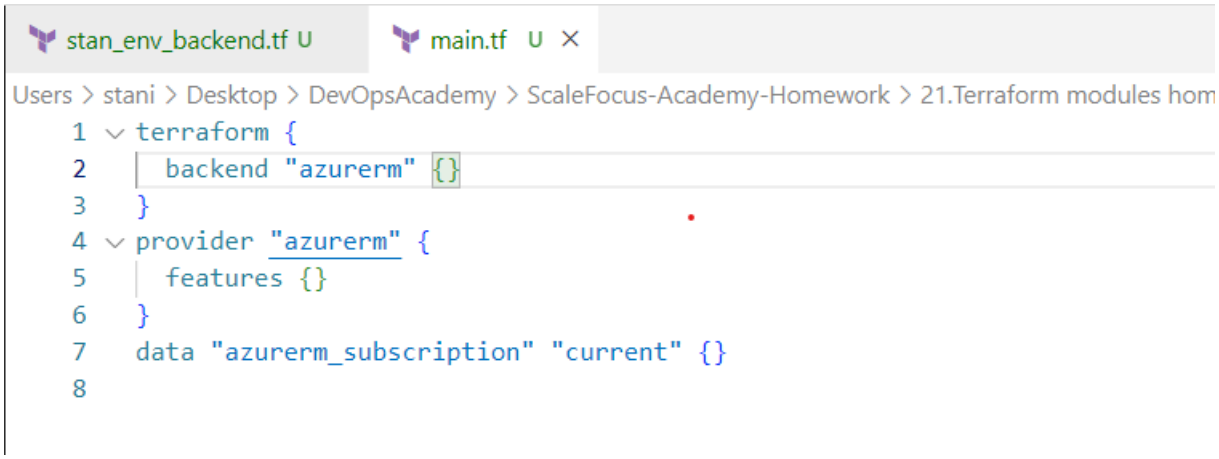
Task 1: Setup and use remote terraform backend.



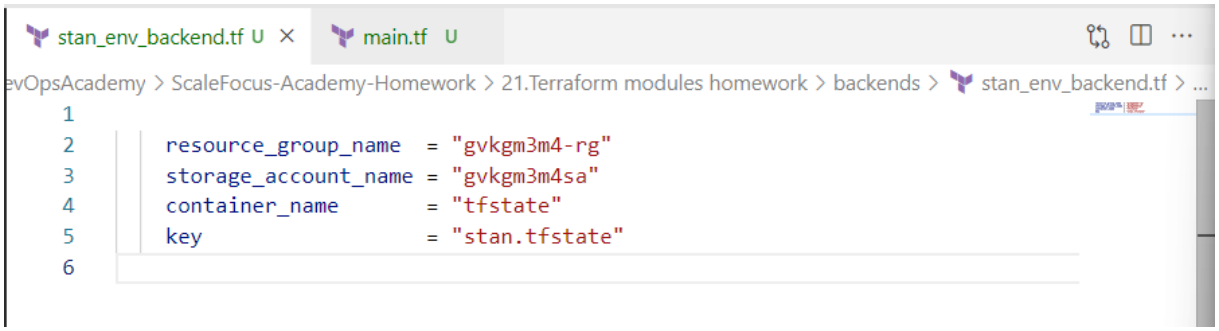
- Reapplying my terraform configurational file from the previous lab, because I destroyed the resources.



- Going to Azure portal and creating a container in the storage account named “tfstate”, where we are uploading an empty file, named “stan.tfstate”.



- Here we are adding the backend configuration file called “main.tf”.
- We also have another file, called “stan_env_backend.tf”, where we have the variables for the resource group, storage account, container name and key, which is the empty file (stan.tfstate), that I uploaded.



- As I said, here are the values for the backend configuration.
- Also worth mentioning that this file is in subdirectory “backends”, because we can have multiple modules for backend.

```
C:\Users\stani\Desktop\DevOpsAcademy\ScaleFocus-Academy-Homework\21.Terraform modules homework>terrafo
rm init --backend-config=backends/stan_env_backend.tf

Initializing the backend...

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

Initializing provider plugins...
- Reusing previous version of hashicorp/azurerm from the dependency lock file
- 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.

C:\Users\stani\Desktop\DevOpsAcademy\ScaleFocus-Academy-Homework\21.Terraform modules homework>
```

- terraform init --backend-config=backends/_env_backend.tf

- 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.

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

main.tf C:\...\21.Terraform modules homework U | input.tfvars U | variables.tf U

rs > stani > Desktop > DevOpsAcademy > ScaleFocus-Academy-Homework > 21.Terraform modules home

```
1  variable "my_name" {
2    type = string
3    description = "First name of the student"
4  }
5
6  variable "environment" {
7    type        = string
8    description = "The environment, where all resources will be placed."
9  }
10
11 variable "location" {
12   type        = string
13   description = "The location, where the resources will be."
14 }
```

- Creating a variable file (variable.tf), in which I am declaring 3 variables for “location”, “environment” and “my_name”.

main.tf C:\...\21.Terraform modules homework U | input.tfvars U

esktop > DevOpsAcademy > ScaleFocus-Academy-Homework > 21.Terraform m

```
1  my_name = "stan"
2  environment = "azure"
3  location = "West Europe"
```

- We also need a file (input.tfvars), in which we are defining the values of the variables, so that we will build a skeleton, which we can use multiple times.

main.tf U | input.tfvars U | variables.tf U

ni > Desktop > DevOpsAcademy > ScaleFocus-Academy-Homework > 21.Terraform modules homework > main.tf > locals

```
1  terraform {
2    backend "azurerm" {}
3  }
4  provider "azurerm" {
5    features {}
6  }
7  data "azurerm_subscription" "current" {}
8  locals {
9    base_name = "${var.my_name}-${var.environment}"
10   network_base_name = "${local.base_name}-ntwrk"
11 }
12
13
14
```

- Defining a locals block, in which we have some variables for “base_name” and “network_base_name”.

```
C:\Users\stani\Desktop\DevOpsAcademy\ScaleFocus-Academy-Homework\21.Terraform modules homework>terraform plan -var-file=in
put.tfvars
data.azurerm_subscription.current: Reading...
data.azurerm_subscription.current: Read complete after 0s [id=/subscriptions/c983dec5-cde0-4991-9469-c26f8cf60056]

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.

C:\Users\stani\Desktop\DevOpsAcademy\ScaleFocus-Academy-Homework\21.Terraform modules homework>
```

- terraform plan -var-file=input.tfvars

- As expected, we shouldn’t have any errors, and the output should be like this.
- We are executing here a “terraform plan” with a specific “var-file” flag for the variables definitions.

main.tf U × input.tfvars U variables.tf U

emy > ScaleFocus-Academy-Homework > 21.Terraform modules homework > main.tf > resource "azurerm_su

```
terraform {
  backend "azurerm" {}
}
provider "azurerm" {
  features {}
}
data "azurerm_subscription" "current" {}
locals {
  base_name = "${var.my_name}-${var.environment}"
  network_base_name = "${local.base_name}-ntwrk"
}

resource "azurerm_resource_group" "rg1" {
  name     = "${local.network_base_name}-rg"
  location = var.location
}

resource "azurerm_virtual_network" "vn1" {
  name                = "${local.network_base_name}-vnet"
  address_space       = ["10.0.0.0/16"]
  location            = azurerm_resource_group.rg1.location
  resource_group_name = azurerm_resource_group.rg1.name
}

resource "azurerm_subnet" "subnet" {
  name                 = "${azurerm_virtual_network.vn1.name}-vms-snet"
  resource_group_name  = azurerm_resource_group.rg1.name
  virtual_network_name = azurerm_virtual_network.vn1.name
  address_prefixes     = ["10.0.2.0/24"]
}
```

- In the end of the second task: How the configurational file looks like:
  - I created a resource group “rg1”, virtual network “vn1” and subnet “subnet”.

```
C:\Users\stani\Desktop\DevOpsAcademy\ScaleFocus-Academy-Homework\21.Terraform modules homework>terraform plan -var-file=input.tfvars
data.azurerm_subscription.current: Reading...
data.azurerm_subscription.current: Read complete after 1s [id=/subscriptions/c983dec5-cde0-4991-9469-c26f8cf60056]

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.rg1 will be created
  + resource "azurerm_resource_group" "rg1" {
      + id       = (known after apply)
      + location = "westeurope"
      + name     = "stan-azure-ntwrk-rg"
    }

  # azurerm_subnet.subnet will be created
  + resource "azurerm_subnet" "subnet" {
      + address_prefixes                               = [
          + "10.0.2.0/24",
        ]
      + enforce_private_link_endpoint_network_policies = (known after apply)
      + enforce_private_link_service_network_policies  = (known after apply)
      + id                                             = (known after apply)
      + name                                           = "stan-azure-ntwrk-vnet-vms-snet"
      + private_endpoint_network_policies_enabled      = (known after apply)
      + private_link_service_network_policies_enabled  = (known after apply)
      + resource_group_name                            = "stan-azure-ntwrk-rg"
      + virtual_network_name                           = "stan-azure-ntwrk-vnet"
    }

  # azurerm_virtual_network.vn1 will be created
  + resource "azurerm_virtual_network" "vn1" {
      + address_space       = [
          + "10.0.0.0/16",
        ]
      + dns_servers         = (known after apply)
      + guid                = (known after apply)
      + id                  = (known after apply)
      + location            = "westeurope"
      + name                = "stan-azure-ntwrk-vnet"
      + resource_group_name = "stan-azure-ntwrk-rg"
      + subnet              = (known after apply)
    }

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

- Execute “terraform plan” with the input from my “input.tfvars” file, here we can see everything that will be created.
- Our plan should not throw any errors.

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

```
resource "azurerm_resource_group" "vm" {
  name     = "${local.vm_name}-rg"
  location = var.location
}


resource "azurerm_public_ip" "vm" {
  name                = "${local.vm_name}-pip"
  location            = azurerm_resource_group.vm.location
  resource_group_name = azurerm_resource_group.vm.name
  allocation_method   = "Static"

  tags = {
    environment = "dev"
  }
}

resource "azurerm_network_interface" "vm" {
  name                = "${local.vm_name}-nic"
  location            = azurerm_resource_group.vm.location
  resource_group_name = azurerm_resource_group.vm.name

  ip_configuration {
    name                          = "external"
    private_ip_address_allocation = "Dynamic"
    public_ip_address_id = azurerm_public_ip.vm.id
  }
}
```

- As we can see here, we are creating in a subdirectory called “vm_module”:
  - 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.

```
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      = "10.0.2.0/24"
    destination_address_prefix = "*"
    source_port_range          = "*"
  }

  security_rule {
    name                       = "allow_http_from_my_ip"
    priority                   = 100
    direction                  = "Inbound"
    access                     = "Allow"
    protocol                   = "Tcp"
    destination_port_range     = "80"
    source_address_prefix      = "10.0.2.0/24"
    destination_address_prefix = "*"
    source_port_range          = "*"
  }
}

resource "azurerm_network_interface_security_group_association" "vm_nsg_to_vm_nic" {
  network_interface_id = azurerm_network_interface.vm.id
  network_security_group_id = azurerm_network_security_group.vm.id
}
```

- 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

```
resource "azurerm_linux_virtual_machine" "web_srv" {
  name                            = local.vm_name
  resource_group_name             = azurerm_resource_group.vm.name
  location                        = azurerm_resource_group.vm.location
  size                            = "Standard_B2s"
  admin_username                  = "adminuser"
  network_interface_ids           = [azurerm_network_interface.vm.id]
  disable_password_authentication = false
  admin_password                  = "Password123456!"

  os_disk {
    caching              = "ReadWrite"
    storage_account_type = "Standard_LRS"
  }

  source_image_reference {
    publisher = "Canonical"
    offer     = "UbuntuServer"
    sku       = "18.04-LTS"
    version   = "latest"
  }
}
```

- And finally the Linux virtual machine.

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

- Also in a local block we are adding local variable called vm_name.

main.tf C:\...\21.Terraform modules homework U × | main.tf C:\...\vm_module U | output.tf U

C: > Users > stani > Desktop > DevOpsAcademy > ScaleFocus-Academy-Homework > 21.Terraform modul

```
terraform {
  backend "azurerm" {}
}
provider "azurerm" {
  features {}
}
data "azurerm_subscription" "current" {}
locals {
  base_name = "${var.my_name}-${var.environment}"
  network_base_name = "${local.base_name}-ntwrk"
}

resource "azurerm_resource_group" "rg1" {
  name     = "${local.network_base_name}-rg"
  location = var.location
}

resource "azurerm_virtual_network" "vn1" {
  name                = "${local.network_base_name}-vnet"
  address_space       = ["10.0.0.0/16"]
  location            = azurerm_resource_group.rg1.location
  resource_group_name = azurerm_resource_group.rg1.name
}

resource "azurerm_subnet" "subnet" {
  name                 = "${azurerm_virtual_network.vn1.name}-vms-snet"
  resource_group_name  = azurerm_resource_group.rg1.name
  virtual_network_name = azurerm_virtual_network.vn1.name
  address_prefixes     = ["10.0.2.0/24"]
}

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

}
```

- Now we are back to the network configuration file, where we are using the module “vm”, that we created.

> Users > stani > Desktop > DevOpsAcademy > ScaleFocus-Academy-Homework > 21.Terraform modules homework > variables.tf > variable "environment" > {

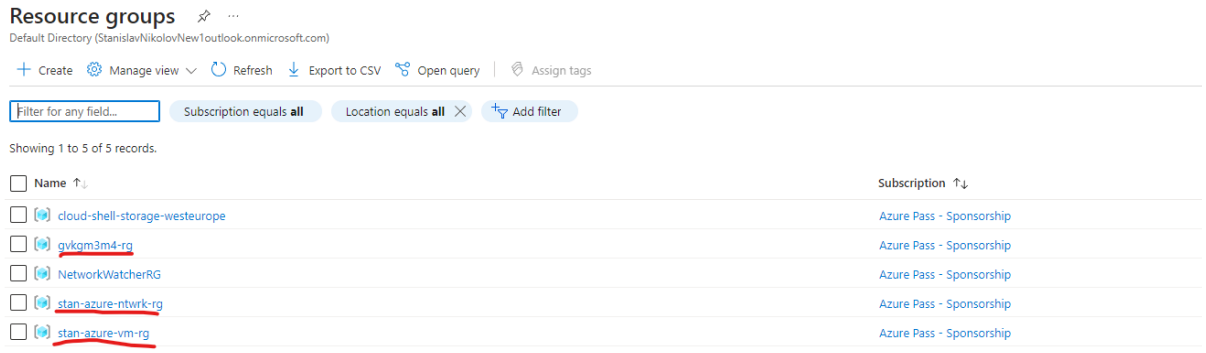
```
variable "my_name" {
  type = string
  description = "First name of the student"
}

variable "environment" {
  type        = string
  description = "The environment, where all resources will be placed."
}

variable "location" {
  type        = string
  description = "The location, where the resources will be."
}
variable "vms_subnet_id" {
  type        = string
  description = "The subnet."
  default = azurerm_subnet.general_network_vms.id
}

variable "my_public_ip" {
  type        = string
  description = "What the public IP will be."
}
```

- The variable file should look like this, I didn't use a variable for the password, and just hardcoded the password.

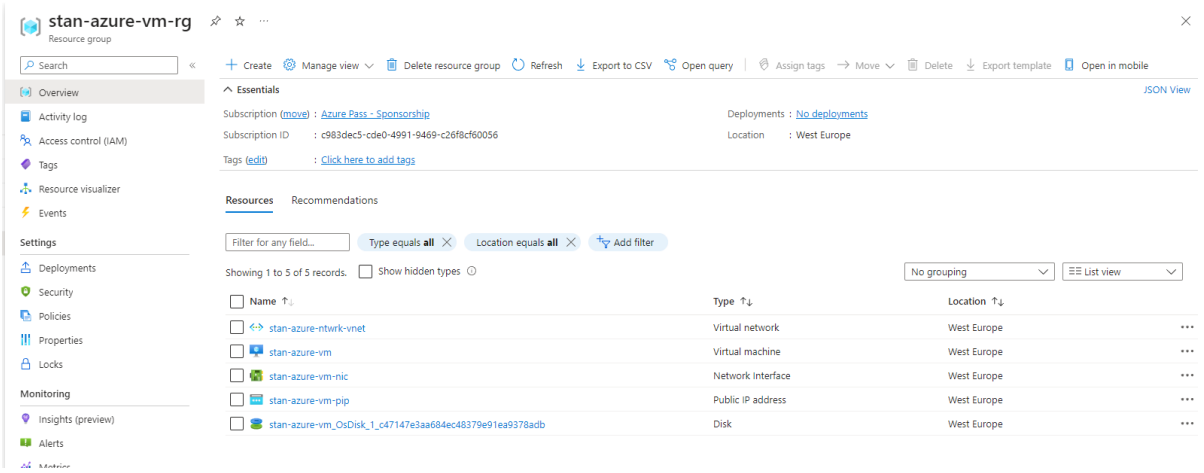
C: > Users > stani > Desktop > DevOpsAcademy > ScaleFocus-Academy-Homework > 21.Terraform modules homework > input.tfvars >

```
my_name = "stan"
environment = "azure"
location = "West Europe"
my_public_ip = "123.45.67.89"

```

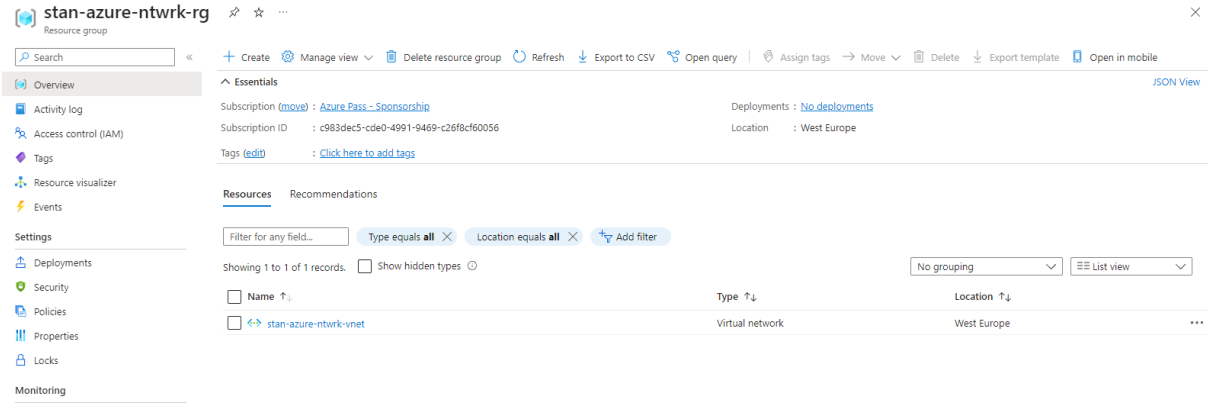
- And that's how the input.tfvars should look like.



- After terraform apply, we should have 2 resources, and the first resource was for the first task.



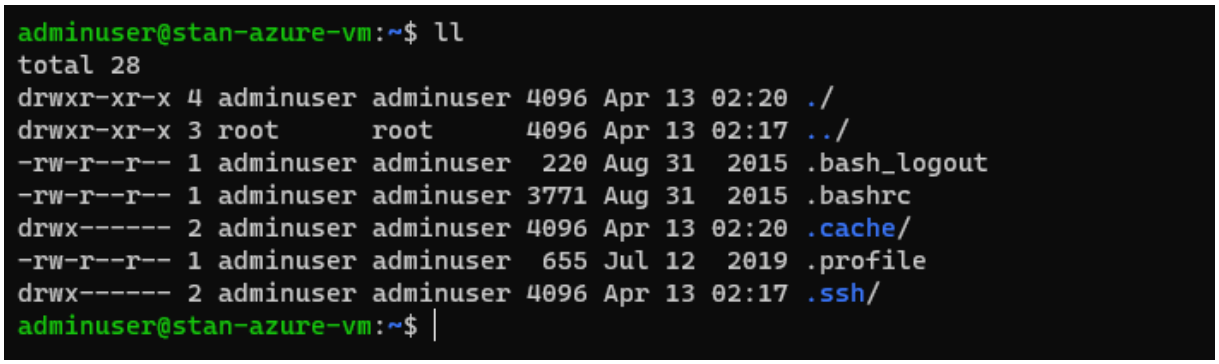
- The resource group of the virtual machine.



- The resource group of the network.

```
C:\Users\stani\Desktop\DevOpsAcademy\ScaleFocus-Academy-Homework\21.Terraform modules homework>ssh adminuser@98.71.194.74
The authenticity of host '98.71.194.74 (98.71.194.74)' can't be established.
ED25519 key fingerprint is SHA256:ztpU4JoKkuyJw2Ep3eyBR9iIGj6PmOfkeY9QHny+zsg.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '98.71.194.74' (ED25519) to the list of known hosts.
adminuser@98.71.194.74's password:
Welcome to Ubuntu 16.04.7 LTS (GNU/Linux 4.15.0-1113-azure x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

UA Infra: Extended Security Maintenance (ESM) is not enabled.

0 updates can be applied immediately.

52 additional security updates can be applied with UA Infra: ESM
Learn more about enabling UA Infra: ESM service for Ubuntu 16.04 at
https://ubuntu.com/16-04



The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

adminuser@stan-azure-vm:~$
```

- Connecting to the virtual machine, that we just created.



- Listing the files of the virtual machine.