**Terraform Exercises**

**Task 1: Install terraform and Azure CLI.**

**To install Terraform, you can follow this:**

* Visit the Terraform website at https://www.terraform.io/downloads.html and download the appropriate package for your operating system. Terraform is available for Windows, Mac, and Linux.
* Once the download is complete, extract the package to a directory of your choice.
* Add the directory to your system's PATH environment variable. This will allow you to run the terraform command from any directory.
* To test your installation, open a command prompt or terminal window and type terraform version. This should display the installed version of Terraform.

**To install Azure CLI on your system, you can follow this:**

* Go to the official Azure CLI installation page at https://docs.microsoft.com/en-us/cli/azure/install-azure-cli and choose your operating system.
* Follow the instructions provided for your operating system to install Azure CLI.
* Once installed, open a command prompt or terminal window and type: “az login” to authenticate with your Azure account. This will open a browser window and prompt you to log in to your Azure account.
* After successfully logging in, you can start using Azure CLI to manage your Azure resources.

**To connect my azure account to terraform I did this:**

* First, you need to create an Azure service principal. This is a type of identity that Terraform can use to authenticate with Azure. To create a service principal, you can follow the instructions in the official Azure documentation.
* Once you have created a service principal, you will need to obtain the following information:
* Subscription ID: This is the unique identifier for your Azure subscription. You can find this in the Azure portal under the "Overview" section of your subscription.
* Tenant ID: This is the identifier for your Azure Active Directory tenant. You can find this in the Azure portal under the "Properties" section of your Azure Active Directory.
* Client ID: This is the identifier for your service principal. You can find this in the Azure portal under the "App registrations" section of your Azure Active Directory.
* Client secret: This is the password for your service principal. You should have created this when you created the service principal.
* Next, you need to create a Terraform configuration file that includes the Azure provider and the information you obtained in the previous step.
* Here is my configuration file:

provider "azurerm" {

subscription\_id = "subscription\_id"

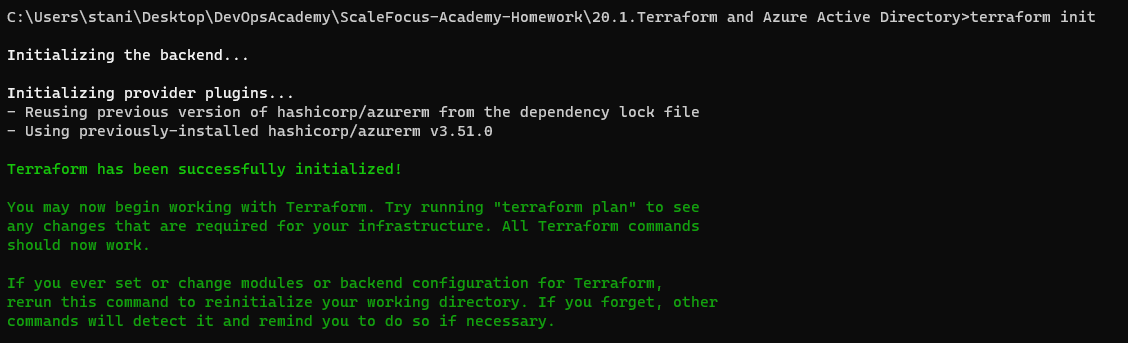
tenant\_id = "tenant\_id"

client\_id = "client\_id"

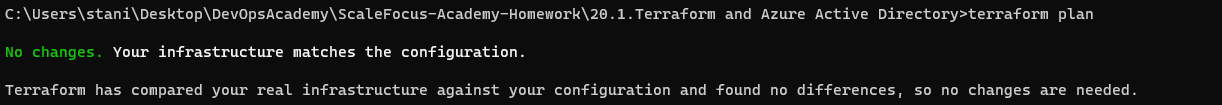
client\_secret = "client\_secret"

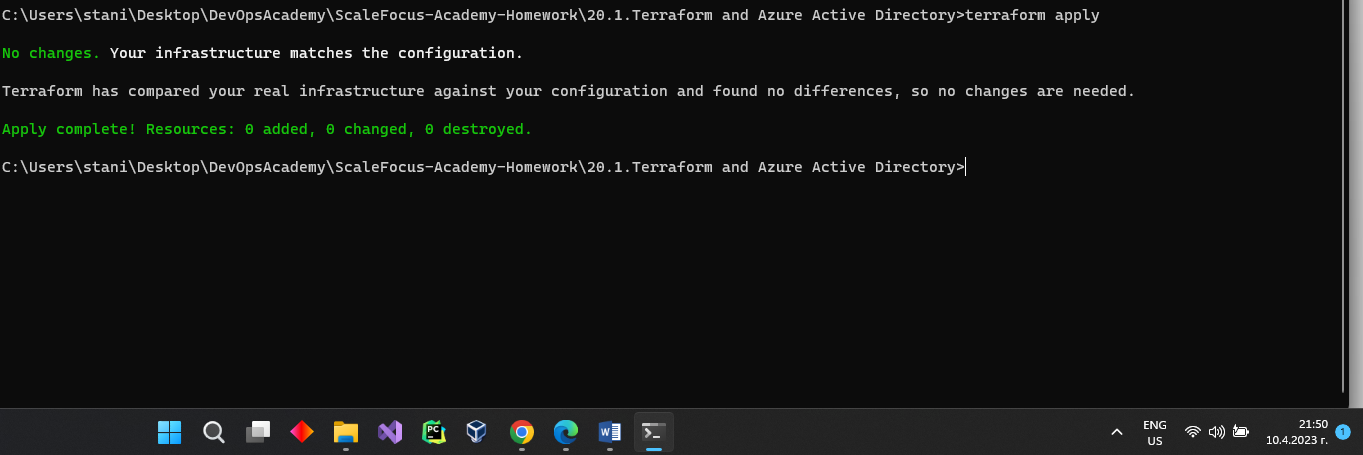
}

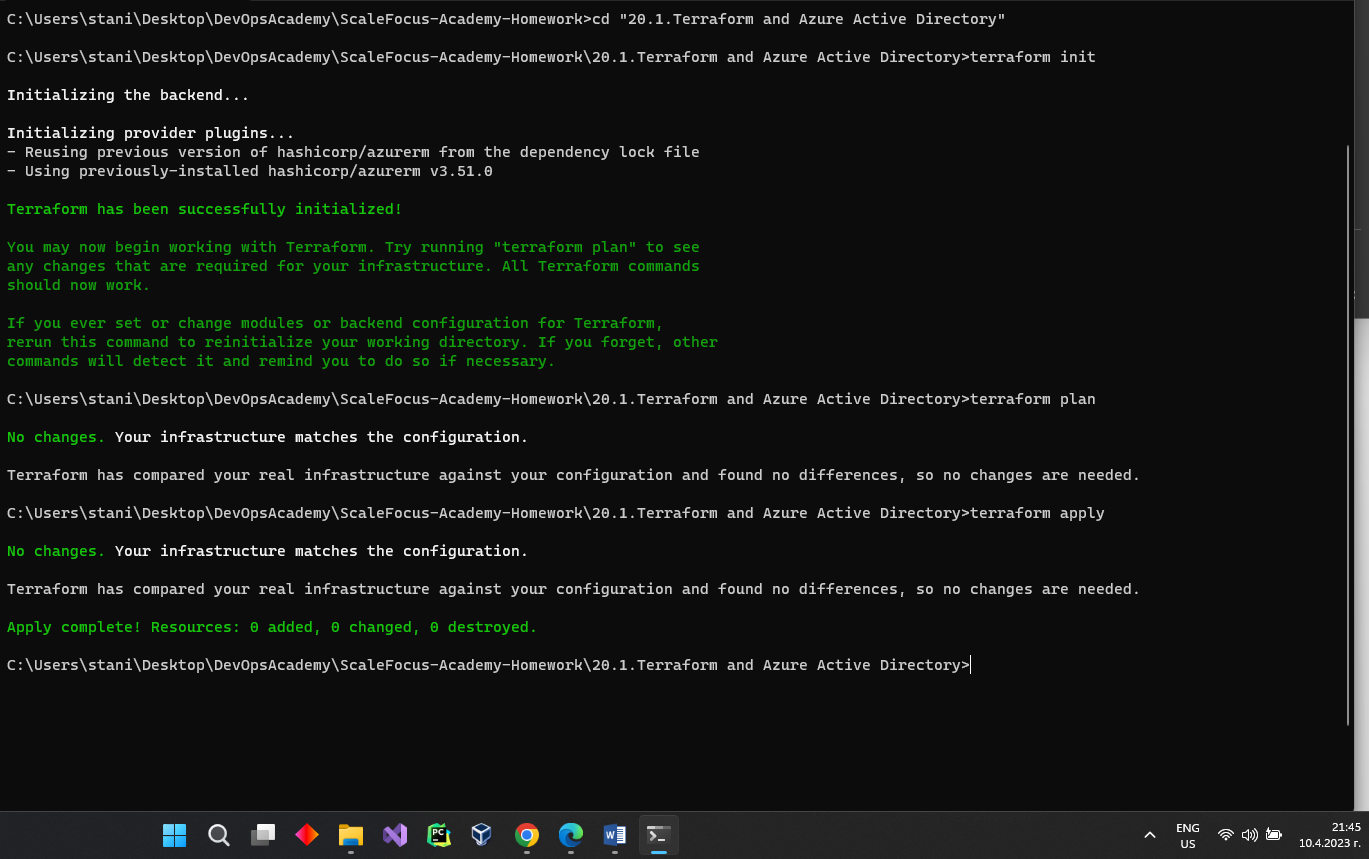
* Replace subscription\_id, tenant\_id, client\_id, and client\_secret with the corresponding values you obtained in the previous step.
* Save the configuration file with a .tf extension, such as main.tf.
* Open a command prompt or terminal window and navigate to the directory where you saved the configuration file.
* Run the “terraform init” command to initialize the Terraform working directory and download the Azure provider.



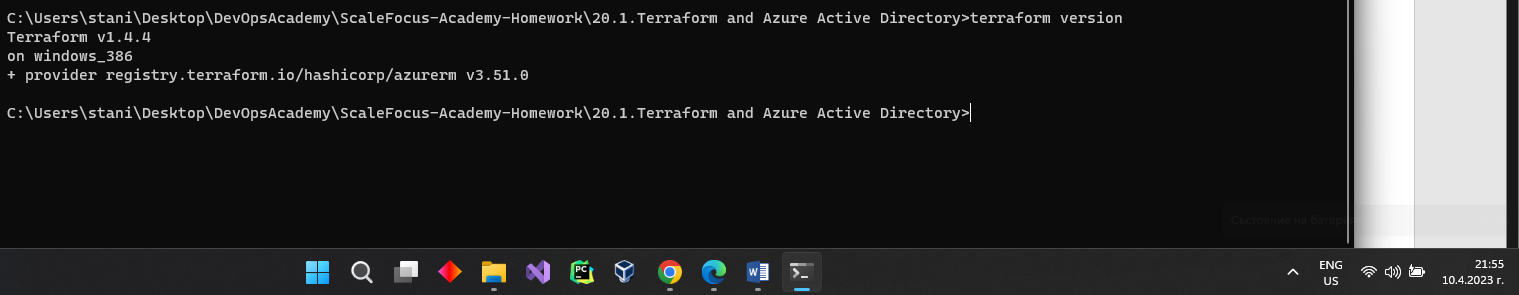
* Once the initialization is complete, you can run other terraform commands, such as terraform plan or terraform apply.



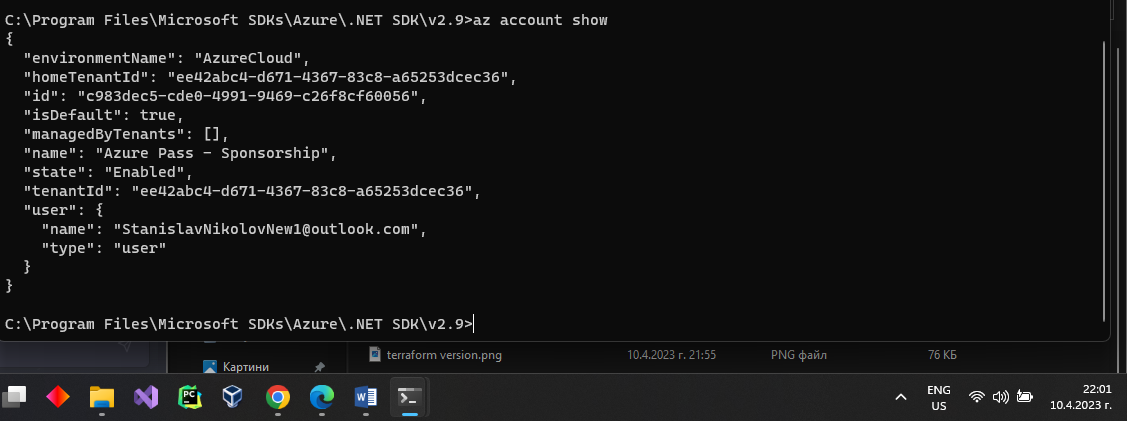




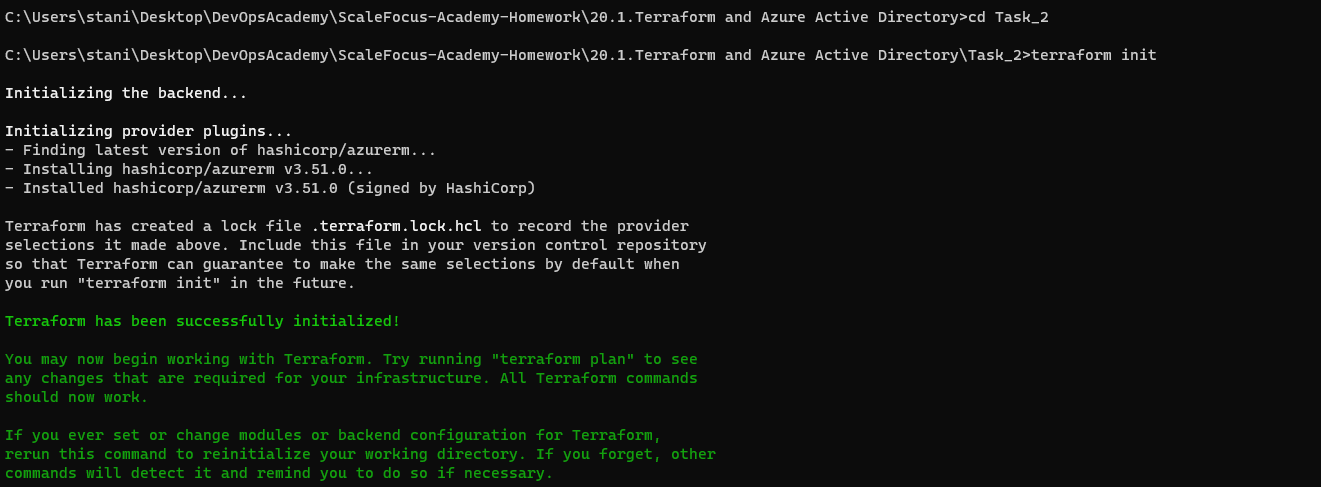
* The output from the commands shows that there are no changes, because I already connected my Azure account with Terraform, and I didn’t change my configuration file(\*.tf)
* Here I am providing a screenshot for the version of terraform and azurerm:



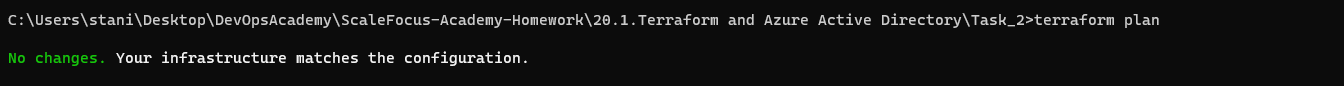
* Here is Azure CLI output of the current subscription:

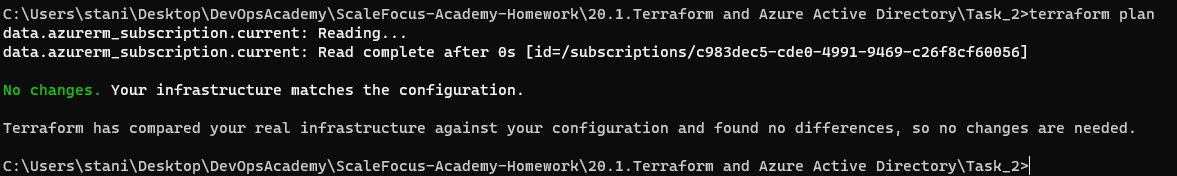


**Task 2: Define your first terraform infrastructure code.**

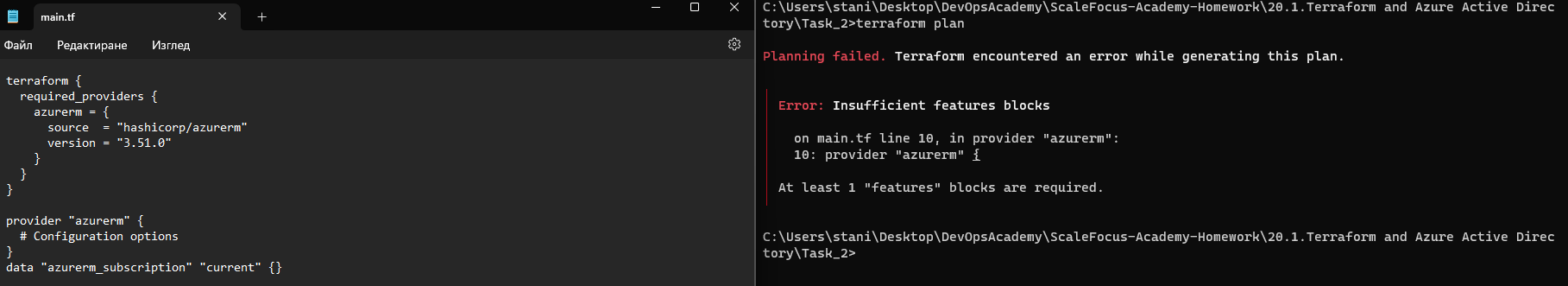


* Initializing terraform.

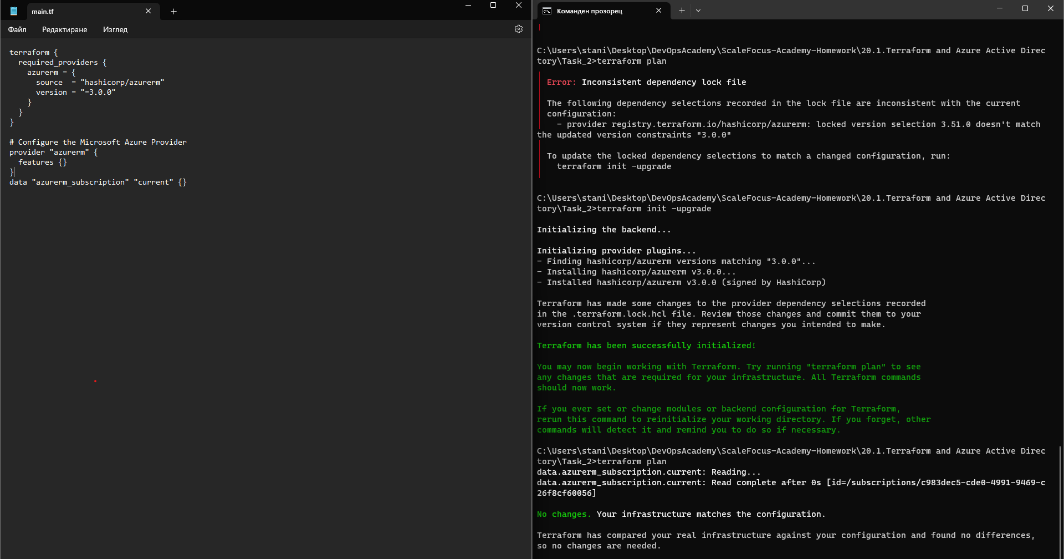




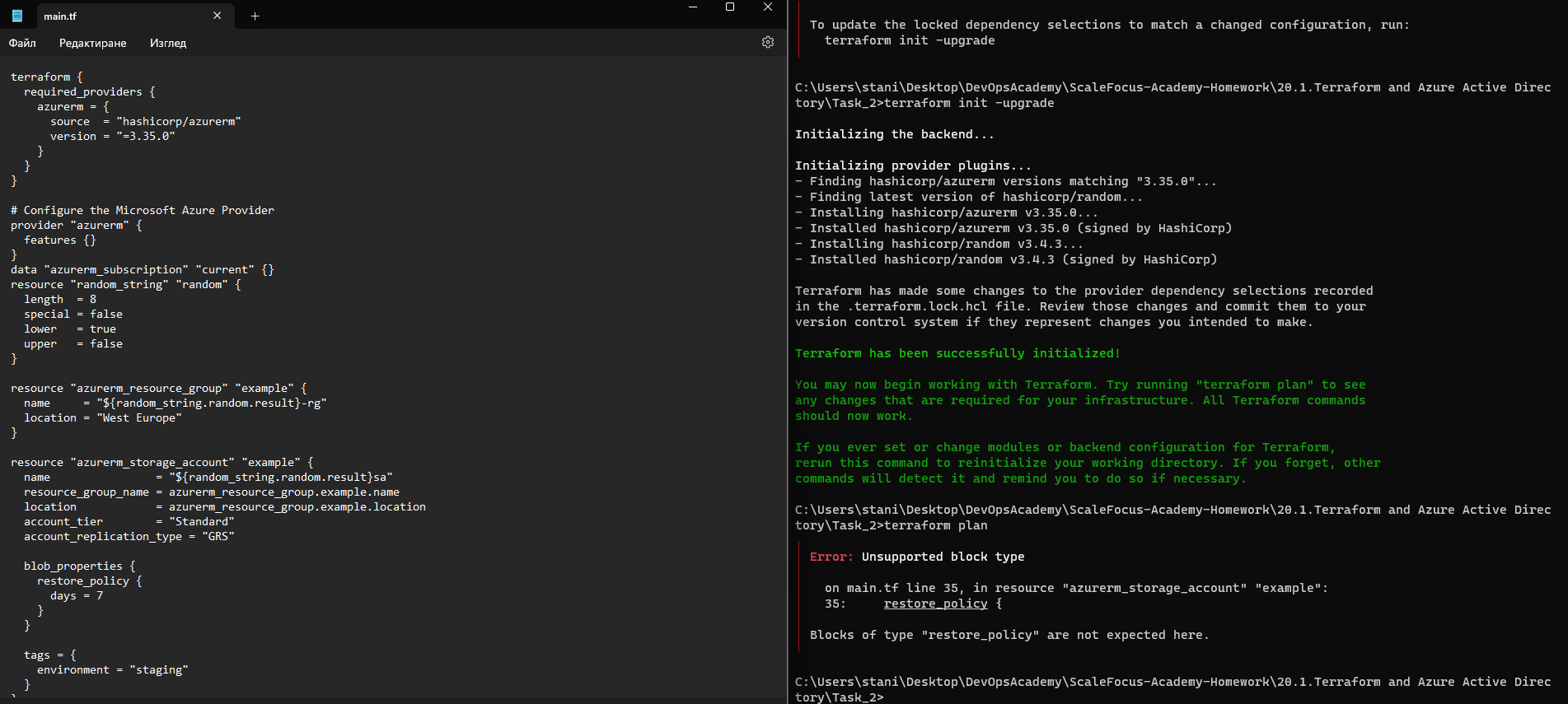
* Terraform plan, which is successful.



* This time it will throw you an error, because azurerm needs a required arguments in order to run properly.



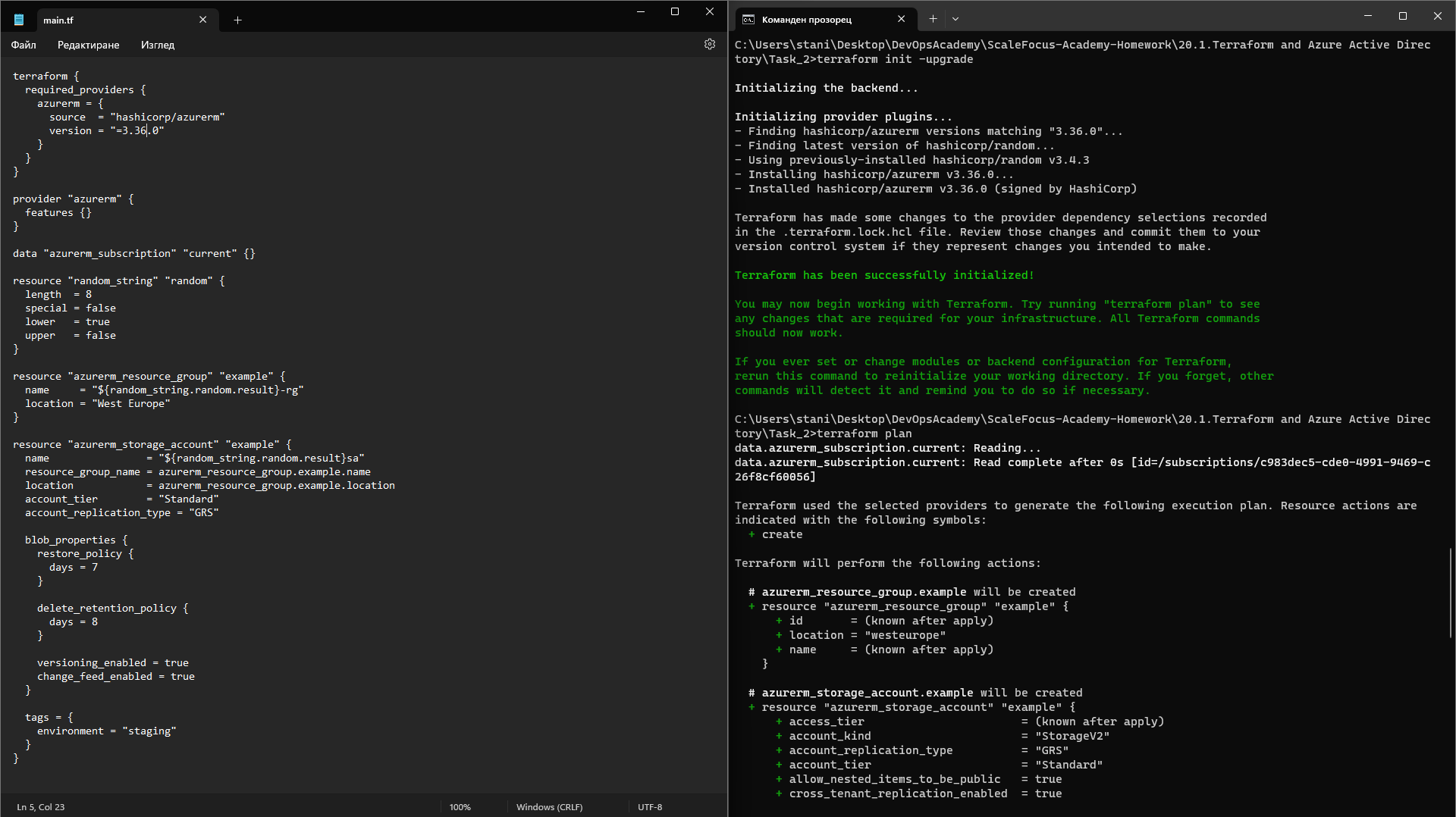
* When we provide the needed arguments, the result will show the subscription\_id without any errors.



* Now we will set the azurerm provider version to 3.35.0, and will add the following code to the configuration file.

When we do it, we will receive an error for the restore\_policy. In order to fix this error we can go to the terraform registry and read the provided documentation.

* 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 3.36.0 of the azurerm provider and we have set fixed version of 3.35.0 for the provider.



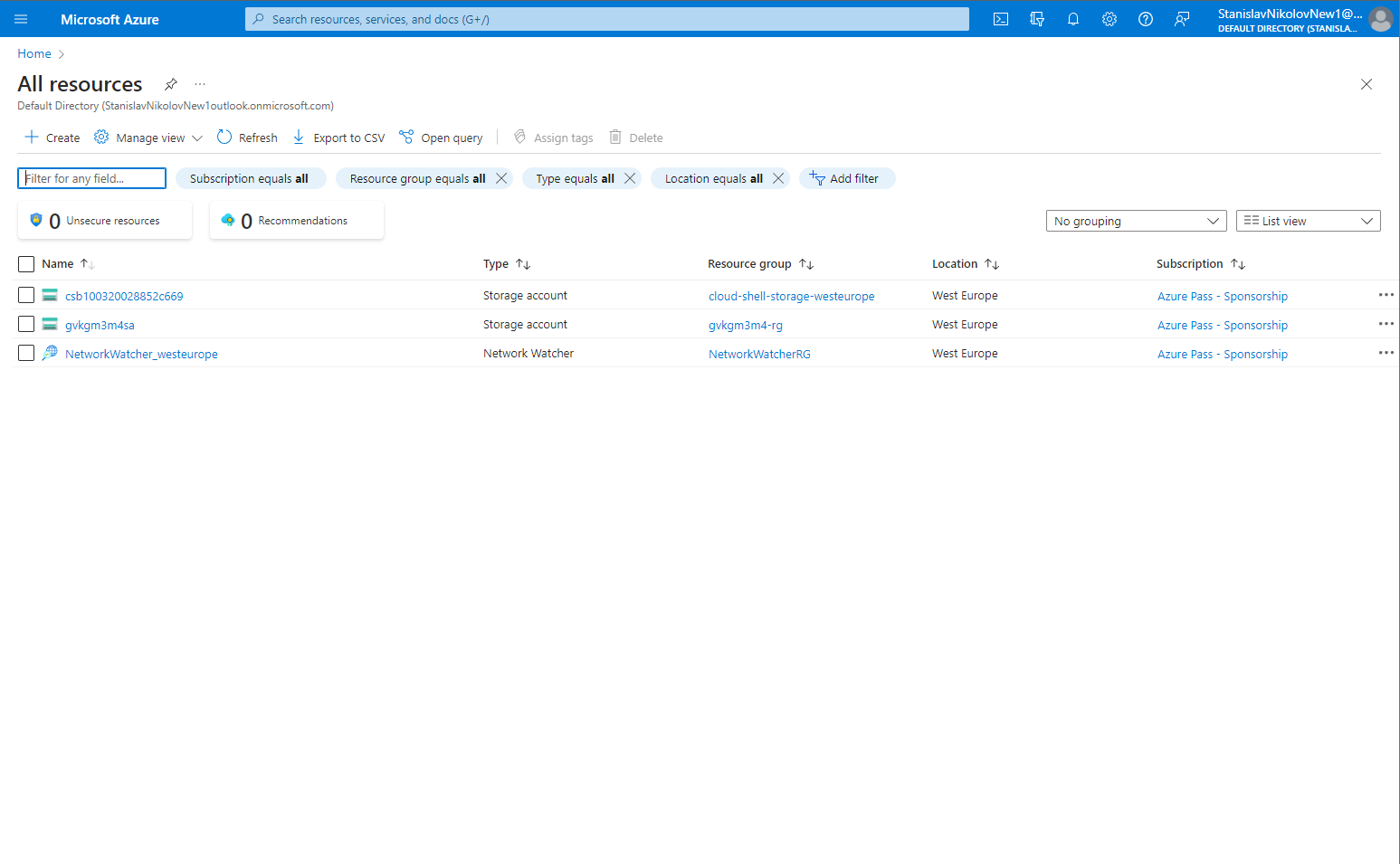
* Now when we changed the version to 3.36.0 and we also need to add delete\_retention\_policy under the restore\_policy block.

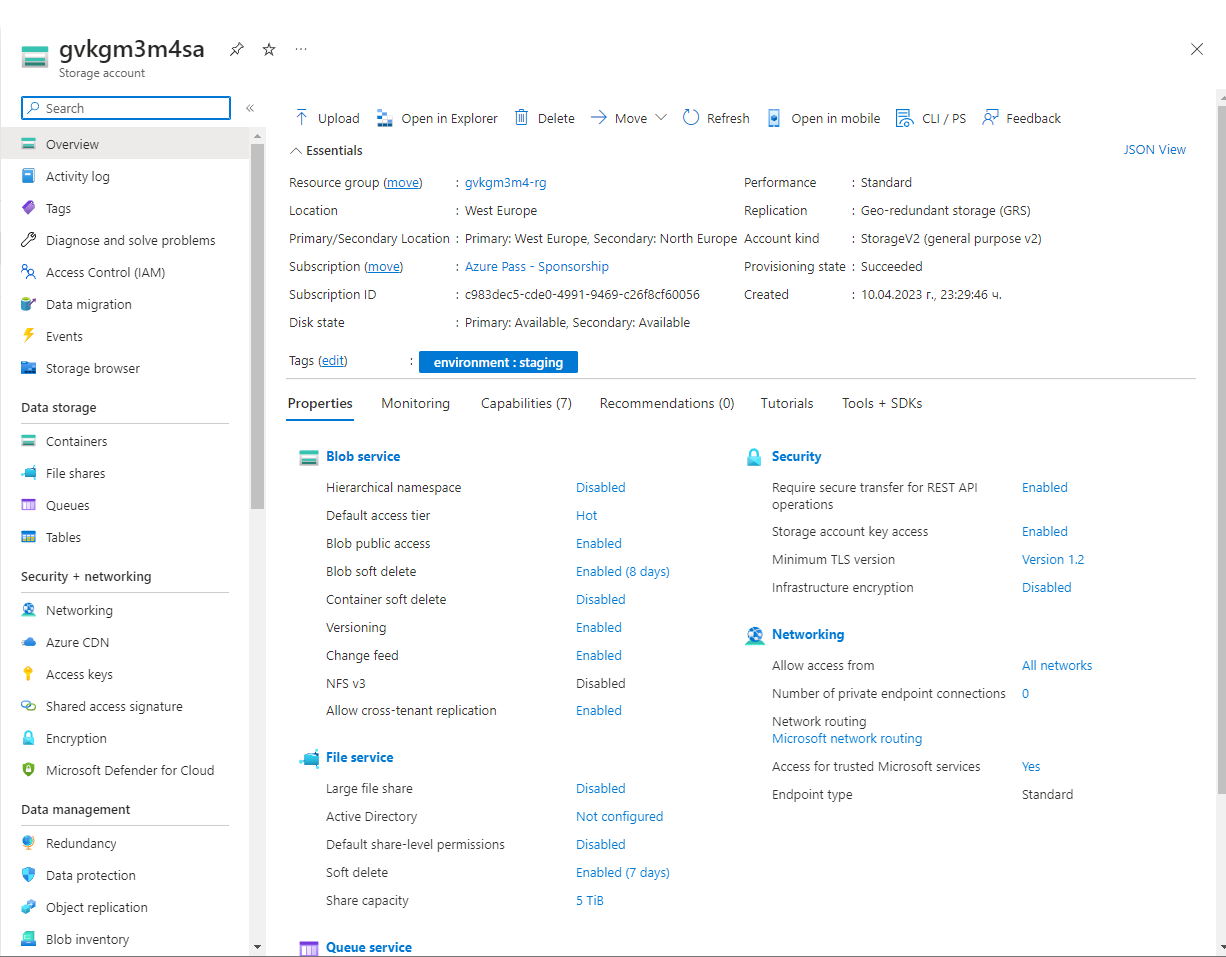


* Part 2 of the output.



* How many resources have you defined in your code and how many resources does the plan output show? Are they the same and why?
* The code defines 3 resources: a resource group, a storage account, and a random string. The plan output shows that 3 resources will be added, which matches the number of resources defined in the code.
* What is the location of your resource group and what is the location of the storage account?
* The location of the resource group and storage account is "westeurope", as specified in the code.
* After Deploying the code:
* 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. 2 storage accounts(one for the cloud shell and one created now with terraform)

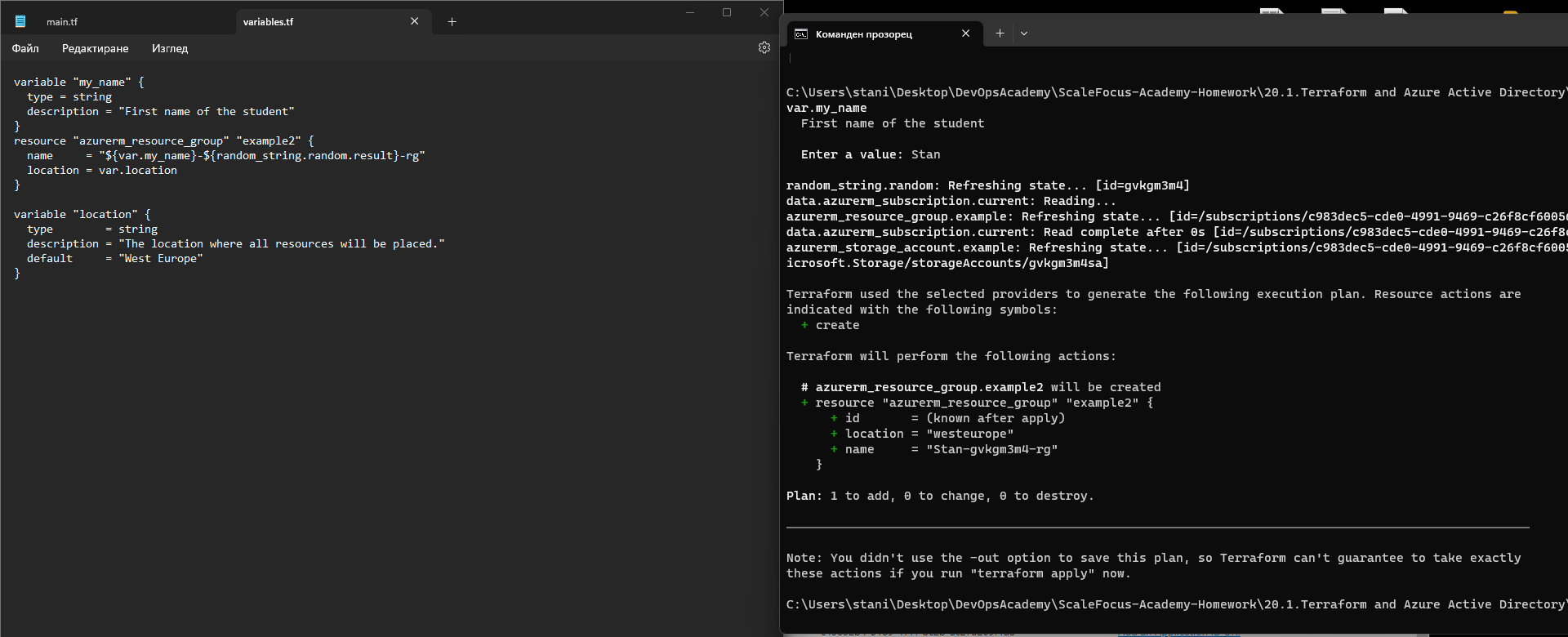
* Are the number of resources shown in the All resources portal window the same with the ones from your plan?

The one storage account, that we created is shown in here, so I think this is the name, that’s assigned in the variable.

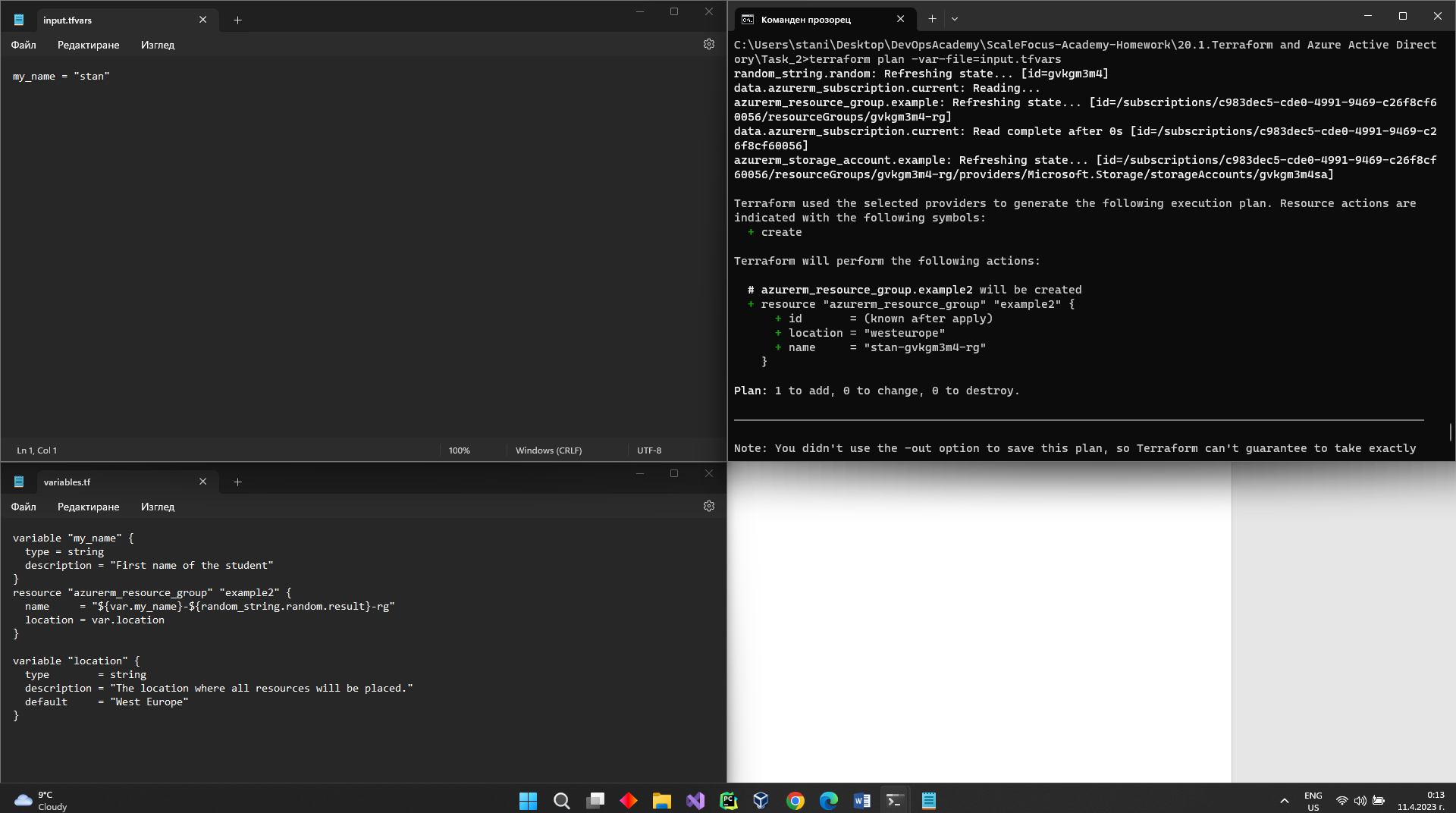
* Give short explanation about the resources that are not shown?

The only resource, that I can’t see is random\_string with name random.

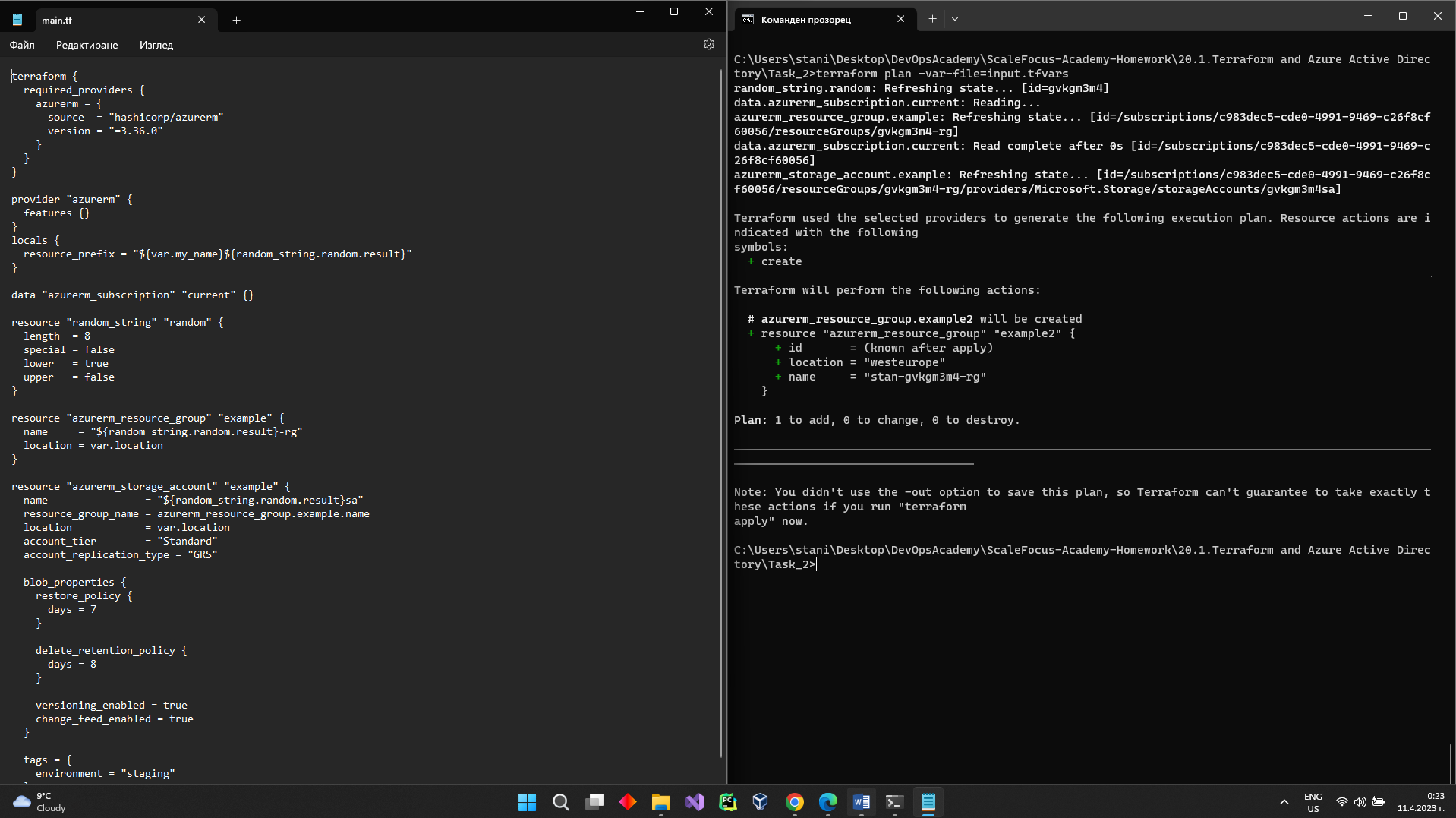
**Task 3: Using variables and outputs.**



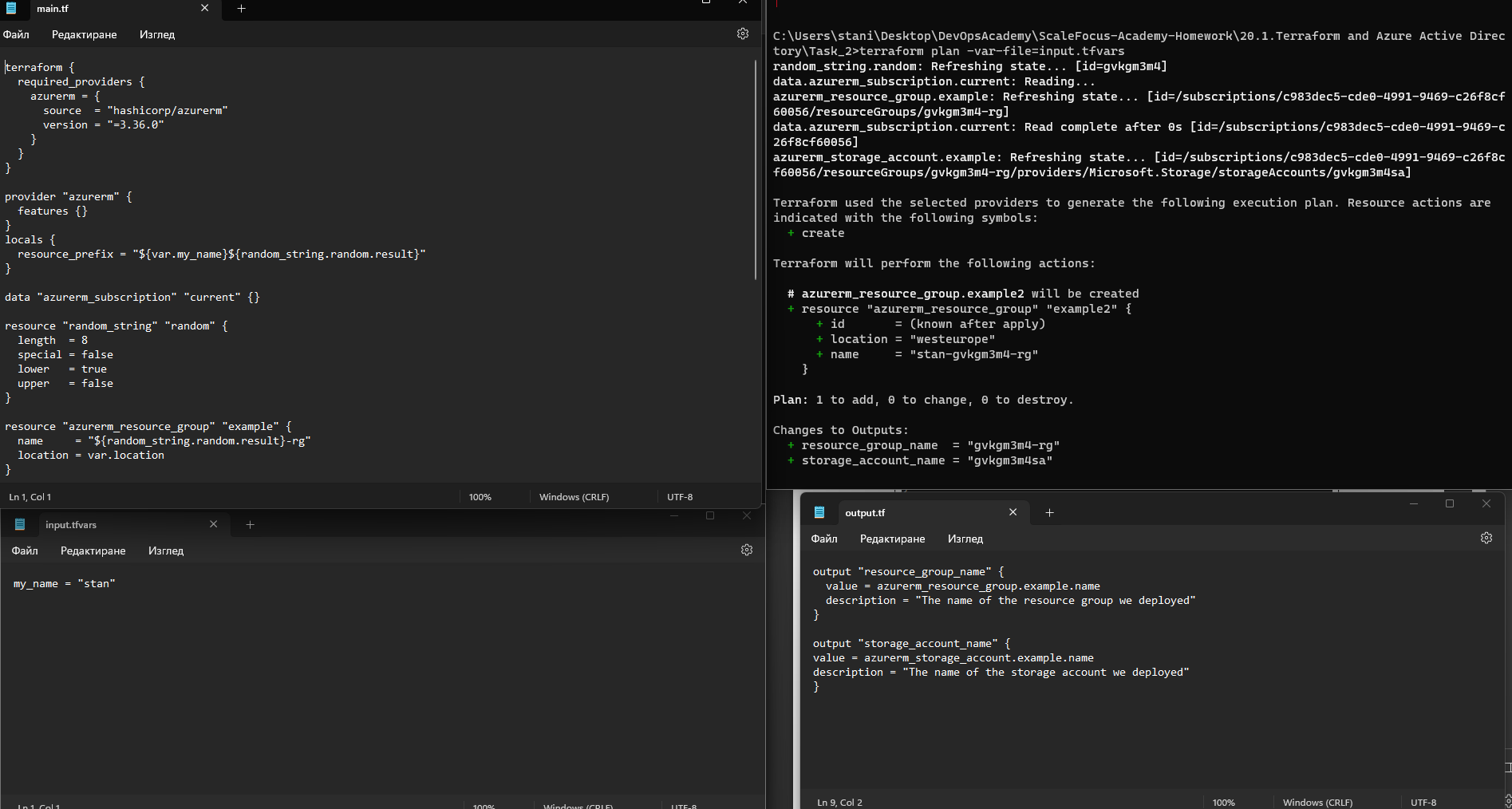
* How many variables do we have defined, and which are they?
* We have defined 2 variables, “my\_name” variable and “location” variable, which has a default value of “West Europe”
* Why did terraform asked us to input a value only for the my\_name variable?
* Terraform asked for input only for the my\_name variable because it was not defined with a default value in the variables.tf file, while the location variable was defined with a default value of "West Europe".
* When executing terraform plan, Terraform checks for any input variables that are not defined with a default value or specified in a variable file and prompts the user to input their values. In this case, since my\_name did not have a default value, Terraform prompted for input.



* Creating a file for the input variables(input.tfvars), also exectuting the terraform plan with variable file.



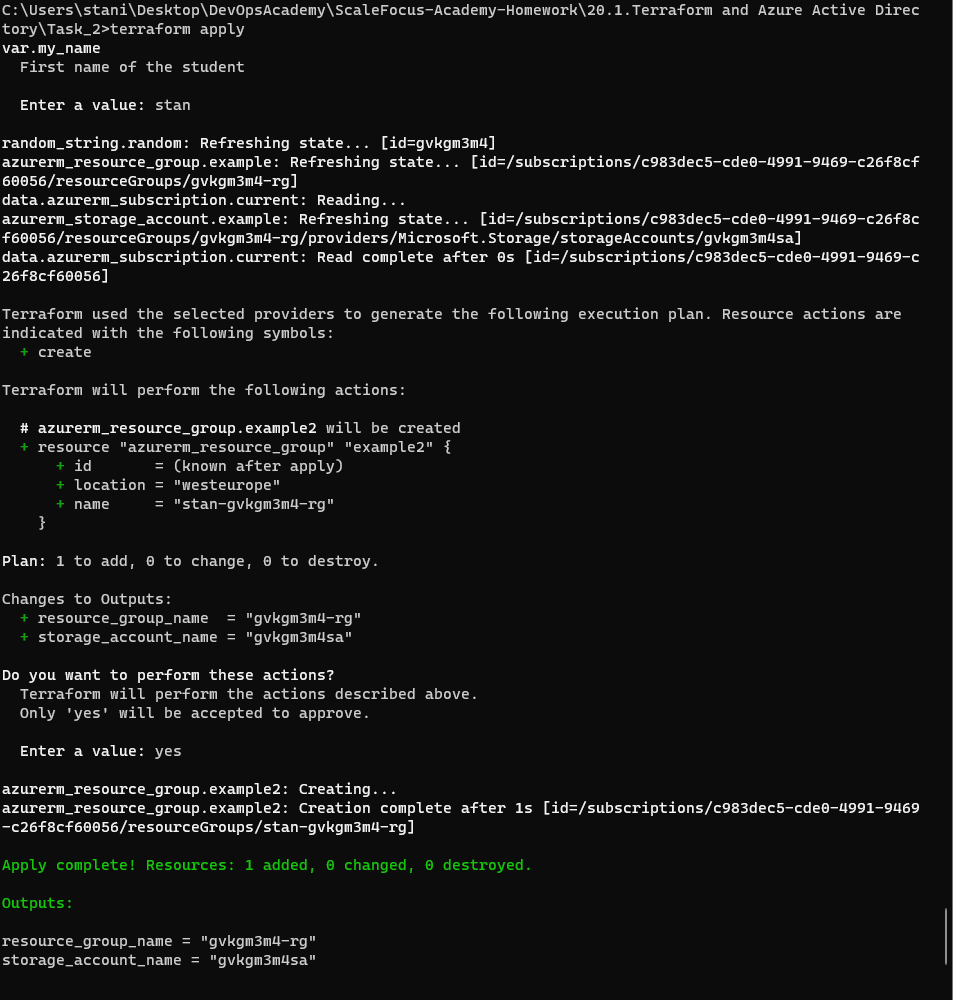
* Editing main.tf by adding locals{} and changing location to the variable location.



* Executing the terraform plan with the input variable file switch.

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 “~”.

* In Terraform, "force replacement" occurs when a resource needs to be destroyed and recreated in order to apply changes to its configuration. This happens when a change is made to a property that is not updatable, or when a change is made to a property that is part of the resource's identity (such as its name).
* When Terraform detects a change that requires force replacement, it will mark the resource as "-/+" in the plan output, indicating that it will be destroyed and recreated. This can have implications for the availability of the resource during the update process, so it's important to understand why a resource is being replaced and what the impact will be.
* In general, it's a good idea to avoid making changes to properties that require force replacement unless it's absolutely necessary. If a resource must be replaced, Terraform will first attempt to gracefully destroy it and create a new one in its place, but this can still result in downtime or data loss depending on the resource and the change being made.



* The outputs of the terraform apply.