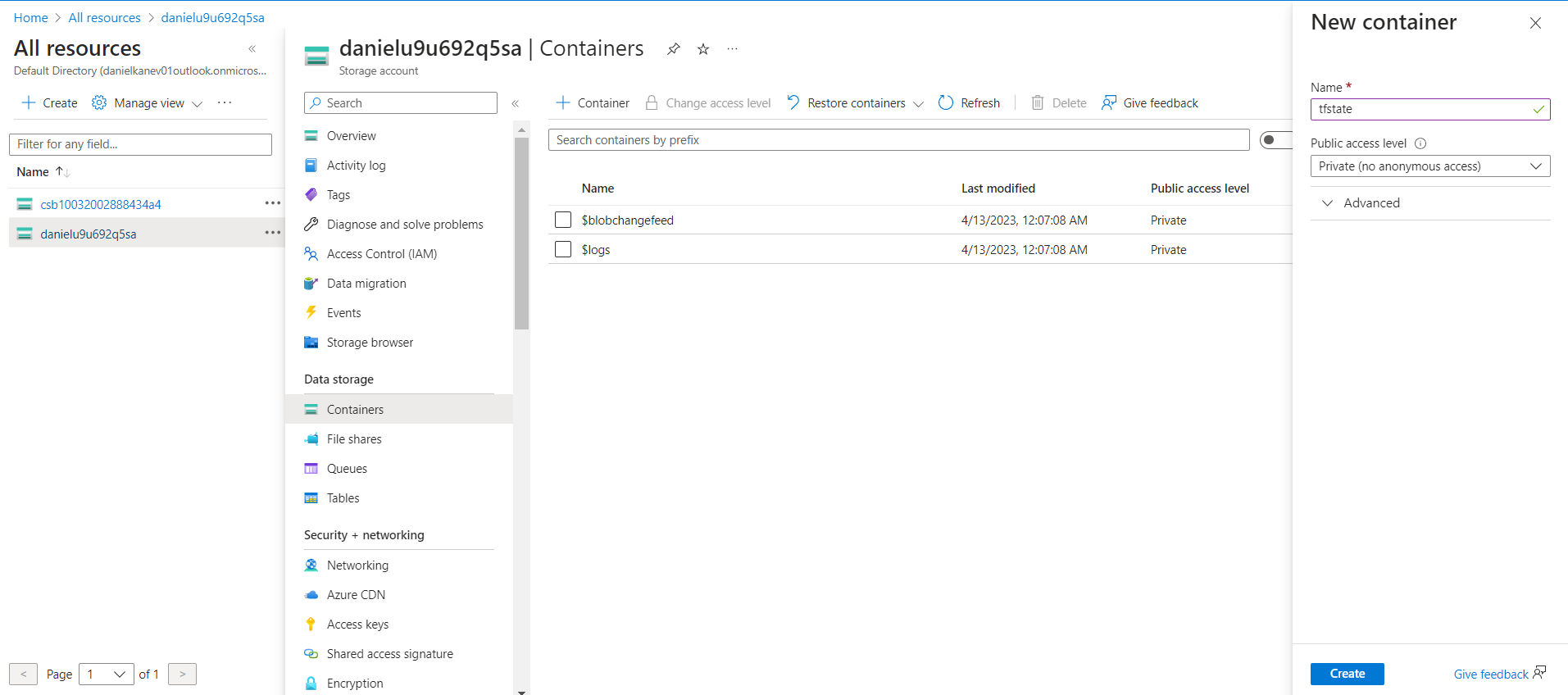
**Terraform Modules.**

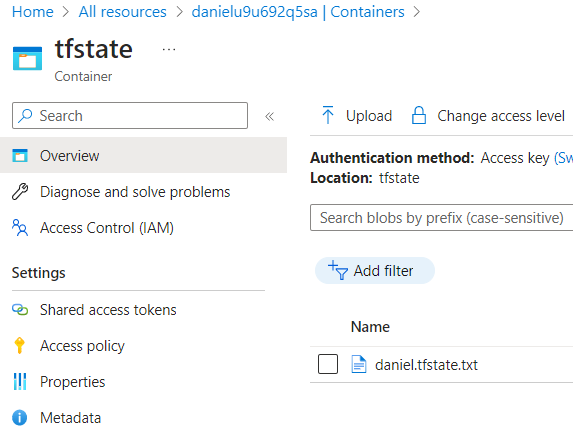
**Task 1: Setup and use remote terraform backend.**

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.)

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 “.tfstate”.

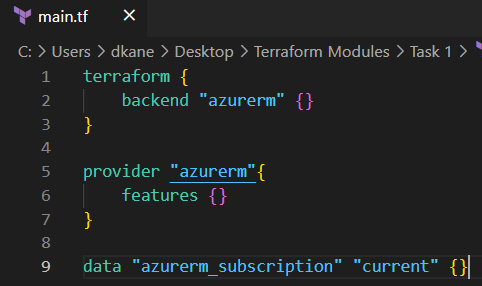


2. Configure your terraform to use the azurerm remote backend.

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

2.2.Add the backend configuration to your terraform code.

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

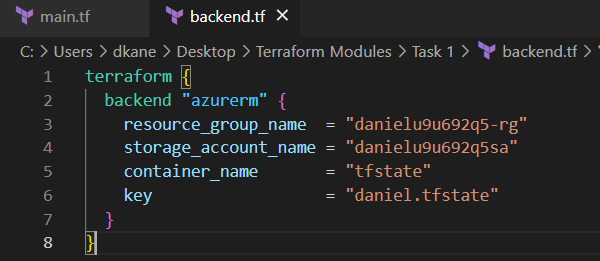


2.2.2. Replace the values accordingly:

• resource\_group\_name – The resource group where the terraform state storage account can be found.  
  
• storage\_account\_name – The storage account name in which the state will be kept.

• container\_name – The name of the container which will hold the blob with terraform state.

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



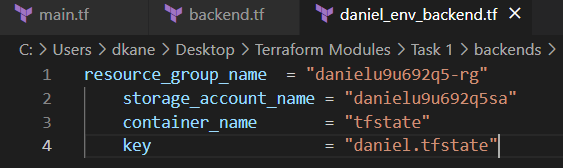
2.3. In case where we have multiple backends configurations for different environments where we configure them in pipelines, we want to be able to switch between different backend configurations for local development also. For this we will use different backend file for each environment.

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

2.3.2. Create a file named <my\_name>\_env\_backend.tf

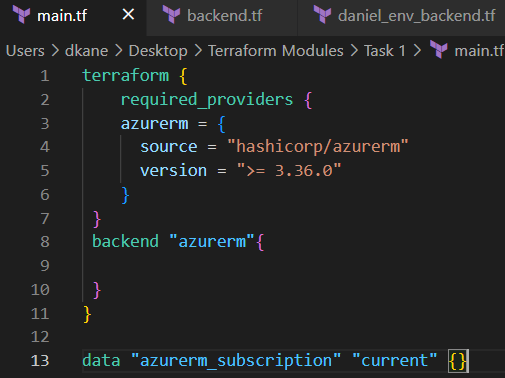
2.3.3. Move the content from the block – backend “azurerm” in main.tf to the \_env\_backend.tf

2.3.4. The files should have the following content:



3. Add the provider configuration for azure without version limitation

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



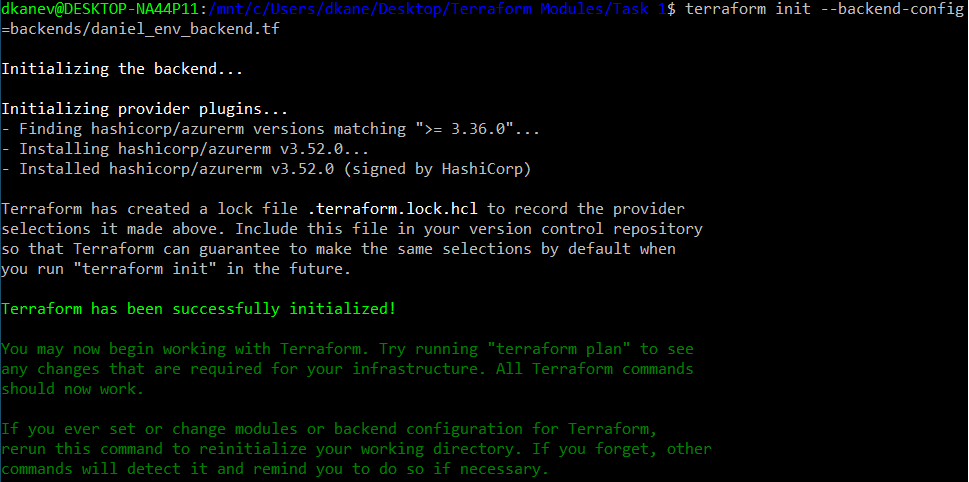
4. Initialize your terraform code with the azurerm\_subscription data source.

• Add the following line in your code (already added above).

5. Initialize your terraform code.

5.1. Execute the following command:

terraform init --backend-config=backends/<my\_name>\_env\_backend.tf



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.

**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:

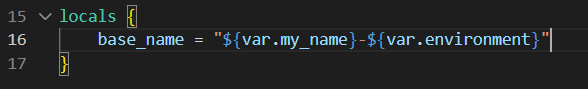
• resource group – base resource used for logical grouping of the resources.  
• virtual network (VNet) – where the virtual machine will be hosted.  
• subnet – the subnet where the virtual machine will be deployed.

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.

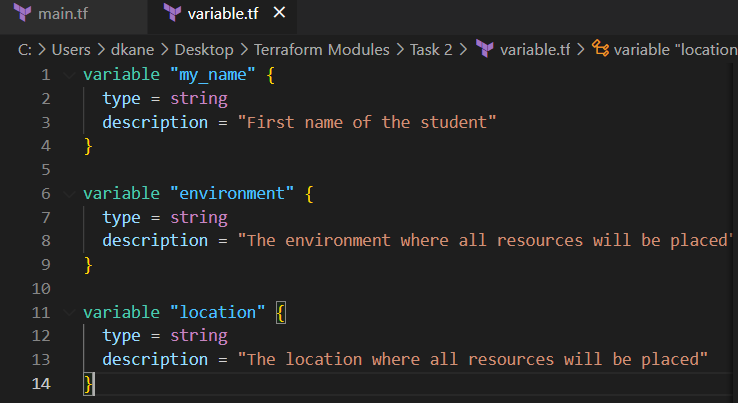
2.1. In the first exercise we have utilized the local vales to give standardized names to our resources. Here we will use the same approach.

2.1.1. Define local value named base\_name with the concatenated values of variables my\_name and environment.

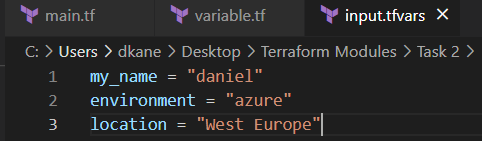
- This is defined in the main.tf file:



2.1.2. Declare the variables my\_name and environment in the variable.tf file:

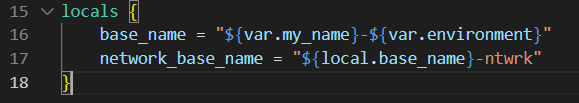


2.1.3. Define values for the variables in your tfvars file:



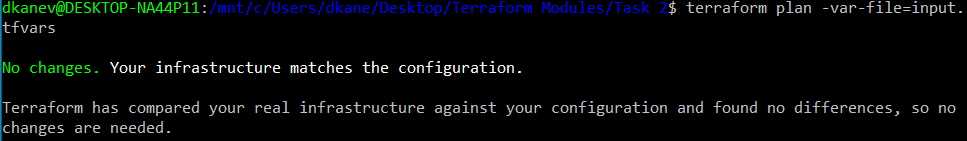
2.1.4. Define network resources network\_base\_name prefix:

- This is also defined in the main.tf file:



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.

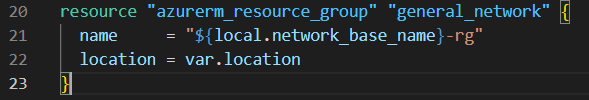


3. Define the general network resources.

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.

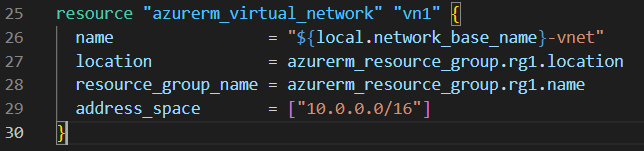
- This is created in the main.tf file again:



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"].

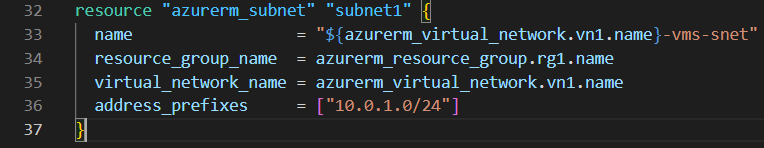
- This is created in the main.tf file again:



3.3. Create a subnet with following parameters (azurerm\_subnet):

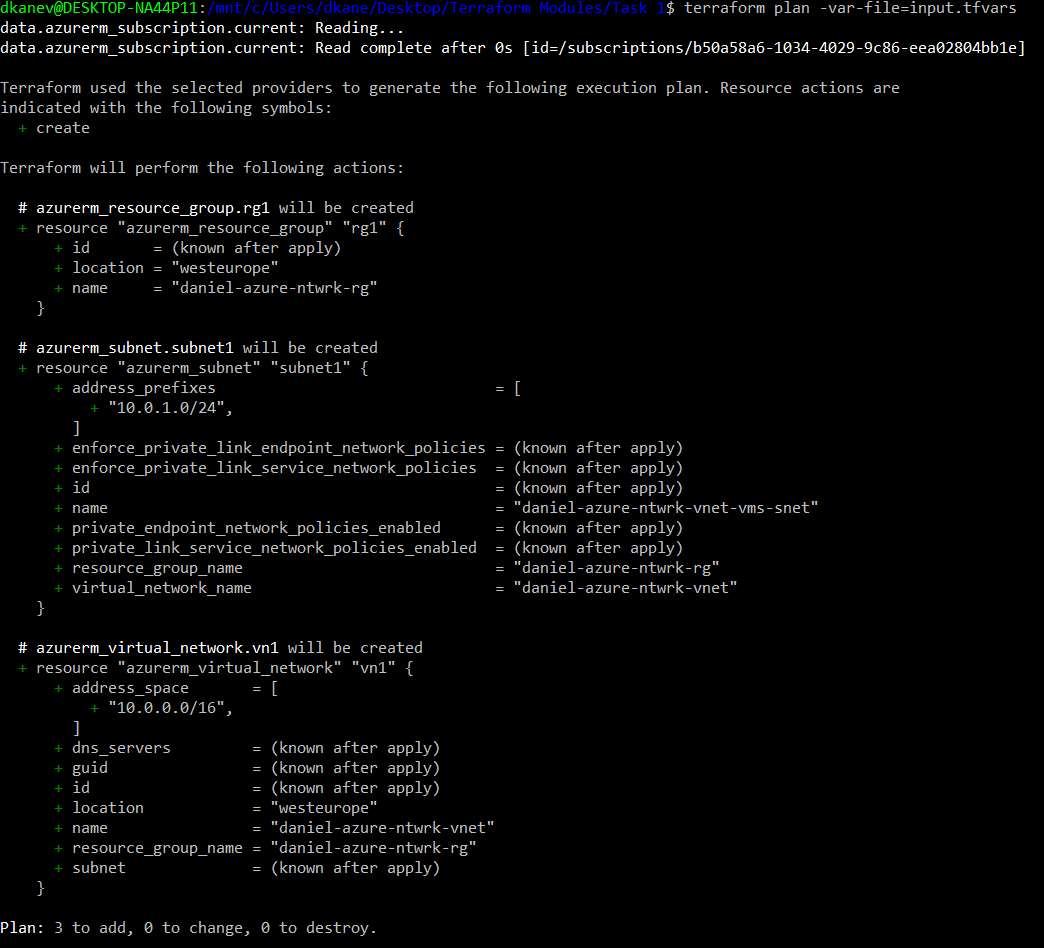
• Terraform resource name – general\_network\_vms.  
• name – ${azurerm\_virtual\_network.general\_network.name}-vms-snet.  
• resource\_group\_name – reference the general\_network resource group name attribute.  
• virtual\_network\_name – reference the general\_network virtual network name attribute.  
• address\_prefixes - ["10.0.1.0/24"].

- This is created in the main.tf file again:



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

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



**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

To be finished later today.