Bringing up the Infrastructure using terraform:

1. Open the Azure cloud shell.
2. Create a new directory for terraform using the command:

**md terraform**

1. Now we need to move to the newly created directory. Use the following command to move to the created directory:

**cd terraform**

1. Once we are in the terraform directory, we need to get the main.tf, terraform.tfvars, variables.tf files from the git repository.

**foreach ($file in @('main.tf','terraform.tfvars','variables.tf'))**

**{**

**$uri = "https://raw.githubusercontent.com/Shubham-Girdhar/Infrastructure-As-Code/main/terraform/$file"**

**Invoke-WebRequest -Uri $uri -OutFile $file**

**}**

The syntax above will download all the files from the git repository to the terraform directory which was created in step 2.

1. By this time, we have the requisites files available in Azure cloud shell, its time to run the terraform configuration. We initialize terraform using the command:

**terraform init**

1. After the initialization of terraform we should bring up the plan using the command:

**terraform plan**

1. Find the public IP address of Azure cloud shell session by running the below command. We need to pass this IP to the terraform configuration.

**curl http://ifconfig.me**

1. Find the local public IP address. This is because terraform needs to open up a port in the network security group to eventually deploy the web app with Visual Studio which is shown later in deploying the application part. We can find the local IP address by opening up PowerShell on the local machine and running the following with PowerShell:

**curl http://ifconfig.me**

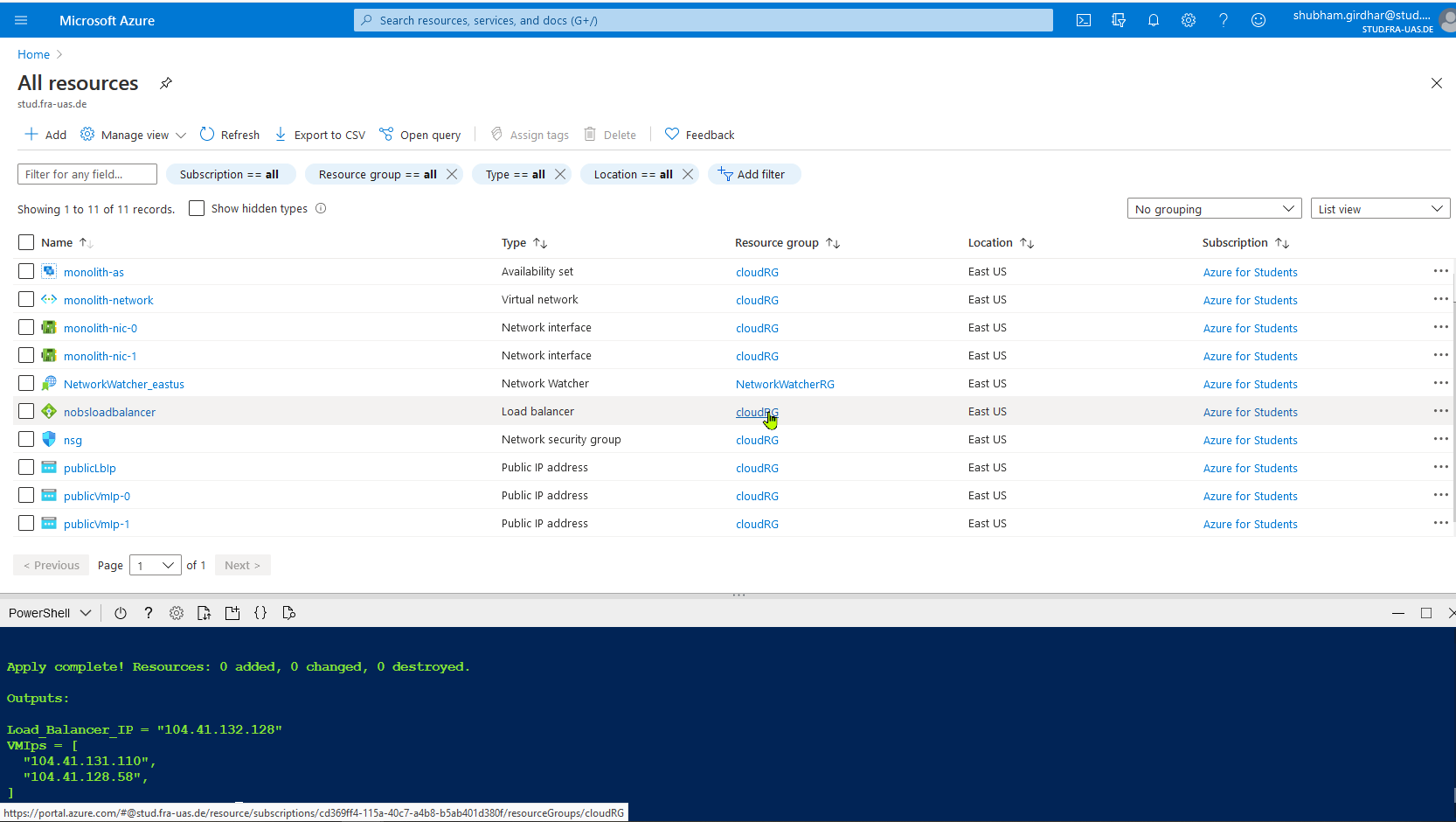
1. Run the terraform plan to build all the Infrastructure using terraform apply using the syntax below. This process might take some time.

**terraform apply -var="location=eastus" -var="resource\_group=cloudRG" -var="cloud\_shell\_source=14.72.123.66" -var="domain\_name\_prefix=cloud-project" -var="management\_ip=141.5.143.17" -auto-approve**

Be sure to fill in the appropriate variable values with the -var parameters.

* location: The location where the Azure resource group is in.
* resource\_group: The name of the resource group to deploy resources into. This variable defaults to NoBSDevopsCloud.
* cloud\_shell\_source: The IP address of the Azure cloud shell which is obtained in \ step 7.
* domain\_name\_prefix: The DNS prefix that will be applied to each VM. This variable name must be unique.
* management\_ip: The public IP address of the local machine which is obtained in step 8.
* auto\_approve: This removes the usual prompt to ask if we really like to do this or not.

1. When terraform is completed, we should have a resource group created with everything that is required for Ansible to take over in the later stages. After the apply syntax is executed successfully terraform will return the public IP address of the load balancer which will placed in front of the web servers. It will also return both of the public Ips for the VMs. We need to make sure we note these IPs as we need it later. If it does not return the IP address on the first run, then run the apply command again.
2. The below image shows the resource groups that are created along with the IP addresses of load balancer and the VMs.



1. By the end of step 11, the Infrastructure is successfully built.