Infrastructure deployment of VM, VM Scale sets and AKS using Terraform

In Capstone project 3,we created an automated provisioned infrastructure using Terraform, EKS cluster, EC2 instances. In this we will create similar in in azure using terraform, AKS and Virtual machine and VM scale sets.

For Virtual machine scale sets configuration, we would need to configure the below:

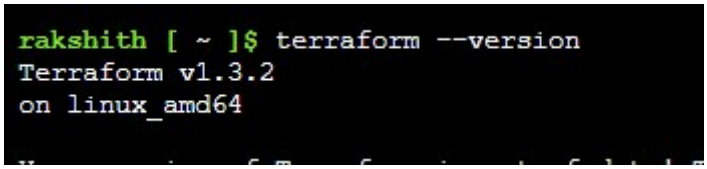
- Resource Group
- Virtual network
- Subnet
- Public ip
- Load balance and backend pools
- Virtual machine scale sets
- Virtual machine
- Public ip and network interface for the Virtual machine

Steps done:

1. **Check terraform setup:**

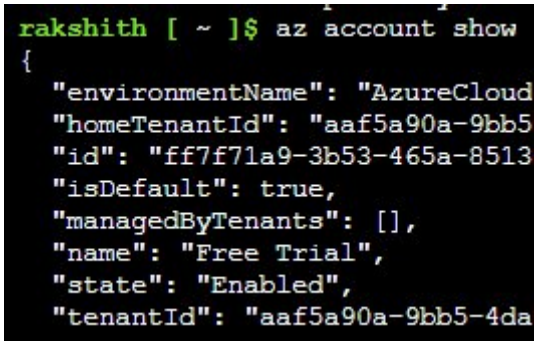
We will use terraform from azure cloud shell.

First we will check if the cloud shell has terraform:



It will already have the terraform.

Now we have to check the subsciption it is using:



It is using the correct subscription.

If you are using from other systems you will have to install azure cli and run ‘az login’ command.

We will now create new directory for terraform and initalize the git because we want to send the files to git once we have completed the task for future use

2. **Write the script for a virtual machine and a Virtual machine scale sets:**

First we wil create providers.tf file which will have provider and version details:

```
terraform {
 required_version = ">=0.12"

 required_providers {
  azurerm = {
   source = "hashicorp/azurerm"
   version = "~>2.0"
  }
 }
}

provider "azurerm" {
 features {}
}
```

```
rakshith [ ~/practice ]$ vim providers.tf
```

```
terraform {
  required_version = ">=0.12"

  required_providers {
    azurerm = {
      source = "hashicorp/azurerm"
      version = "~>2.0"
    }
  }
}

provider "azurerm" {
  features {}
}
```

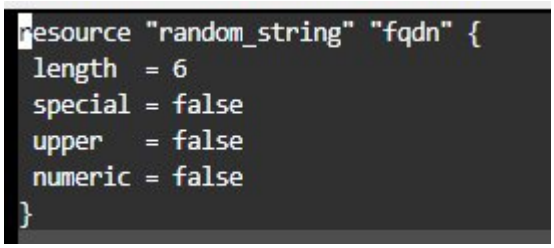
Next we will create resourcegroup.tf :

```
resource "azurerm_resource_group" "vmss" {
 name     = “terraform_rakshith”
 location = "westeurope"
}
```

```
resource "azurerm_resource_group" "vmss" {
 name     = "terraform_rakshith"
 location = "westeurope"
}
~
```

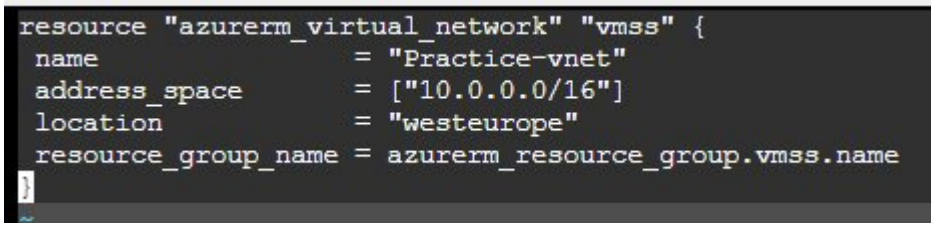
One for randomstring.tf:

```
resource "random_string" "fqdn" {
 length  = 6
 special = false
 upper   = false
 numeric = false
}
```



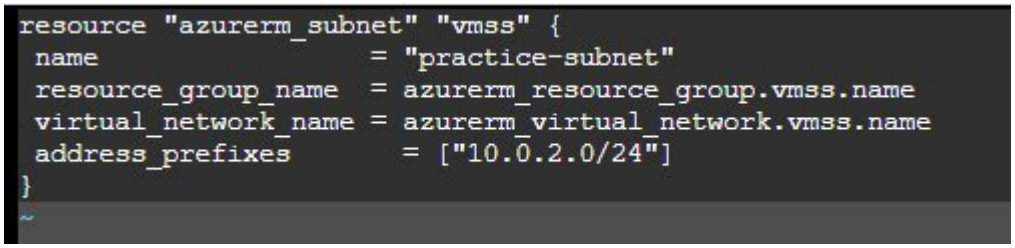
Next we will write for virtual network: vn.tf

```
resource "azurerm_virtual_network" "vmss" {
 name                = "Practice-vnet"
 address_space       = ["10.0.0.0/16"]
 location            = "westeurope"
 resource_group_name = azurerm_resource_group.vmss.name
}
```



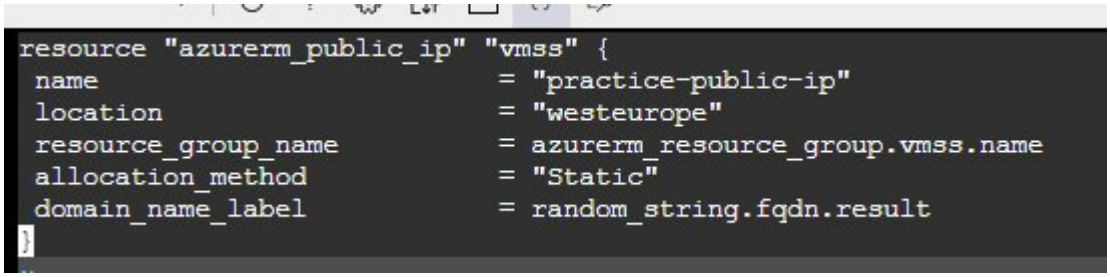
Next we will write for subnet: subnet.tf

```
resource "azurerm_subnet" "vmss" {
 name                 = "practice-subnet"
 resource_group_name  = azurerm_resource_group.vmss.name
 virtual_network_name = azurerm_virtual_network.vmss.name
 address_prefixes     = ["10.0.2.0/24"]
}
```



Next we will write for publicip: publicip.tf

```
resource "azurerm_public_ip" "vmss" {
 name                         = "practice-public-ip"
 location                     = "westeurope"
 resource_group_name          = azurerm_resource_group.vmss.name
 allocation_method            = "Static"
 domain_name_label            = random_string.fqdn.result
}
```



```
resource "azurerm_public_ip" "vmss" {
 name                         = "practice-public-ip"
 location                     = "westeurope"
 resource_group_name          = azurerm_resource_group.vmss.name
 allocation_method            = "Static"
 domain_name_label            = random_string.fqdn.result
}
```

Next write for loadbalancer: lb.tf

This will include the load balancer, backend address pool, load balancer probe, load balancer rule:

```
resource "azurerm_lb" "vmss" {
 name                = "practice-lb"
 location            = "westeurope"
 resource_group_name = azurerm_resource_group.vmss.name

 frontend_ip_configuration {
   name                 = "PublicIPAddress"
   public_ip_address_id = azurerm_public_ip.vmss.id
 }
}

resource "azurerm_lb_backend_address_pool" "bpepool" {
 loadbalancer_id     = azurerm_lb.vmss.id
 name                = "BackEndAddressPool"
}

resource "azurerm_lb_probe" "vmss" {
```

```
 resource_group_name = azurerm_resource_group.vmss.name
 loadbalancer_id     = azurerm_lb.vmss.id
 name                = "ssh-running-probe"
 port                = 80
}

resource "azurerm_lb_rule" "lbnatrule" {
  resource_group_name          = azurerm_resource_group.vmss.name
  loadbalancer_id              = azurerm_lb.vmss.id
  name                         = "http"
  protocol                     = "Tcp"
  frontend_port                = 80
  backend_port                 = 80
  backend_address_pool_ids     = [azurerm_lb_backend_address_pool.bpepool.id]
  frontend_ip_configuration_name = "PublicIPAddress"
  probe_id                     = azurerm_lb_probe.vmss.id
}
```

```
resource "azurerm_lb" "vmss" {
 name                = "practice-lb"
 location            = "westeurope"
 resource_group_name = azurerm_resource_group.vmss.name

 frontend_ip_configuration {
   name                 = "PublicIPAddress"
   public_ip_address_id = azurerm_public_ip.vmss.id
 }
}

resource "azurerm_lb_backend_address_pool" "bpepool" {
 loadbalancer_id     = azurerm_lb.vmss.id
 name                = "BackEndAddressPool"
}

resource "azurerm_lb_probe" "vmss" {
 resource_group_name = azurerm_resource_group.vmss.name
 loadbalancer_id     = azurerm_lb.vmss.id
 name                = "ssh-running-probe"
 port                = 80
}

resource "azurerm_lb_rule" "lbnatrule" {
   resource_group_name            = azurerm_resource_group.vmss.name
   loadbalancer_id                = azurerm_lb.vmss.id
   name                           = "http"
   protocol                       = "Tcp"
   frontend_port                  = 80
   backend_port                   = 80
   backend_address_pool_ids       = [azurerm_lb_backend_address_pool.bpepool
```

Now you can write for virtual machine scale sets: vmss.tf

```
resource "azurerm_linux_virtual_machine_scale_set" "vmss" {
 name                = "practicevmscaleset"
 location            = "westeurope"
 resource_group_name = azurerm_resource_group.vmss.name
 sku                 = "Standard_DS1_v2"
 instances           = 2
 admin_username      = "rakshith"
 admin_password      = "temporary@54321"
 disable_password_authentication = false



 source_image_reference {
   publisher = "Canonical"
   offer     = "0001-com-ubuntu-server-focal"
   sku       = "20_04-lts"
   version   = "latest"
}

 os_disk {
   storage_account_type = "Standard_LRS"
   caching              = "ReadWrite"
  }
 network_interface {
  name    = "terraformnetworkprofile"
  primary = true

  ip_configuration {
    name                                   = "IPConfiguration"
    subnet_id                              = azurerm_subnet.vmss.id
    load_balancer_backend_address_pool_ids = [azurerm_lb_backend_address_pool.bpepool.id]
    primary = true
  }
 }
}
```

```
resource "azurerm_linux_virtual_machine_scale_set" "vmss" {
 name                 = "practicevmscaleset"
 location             = "westeurope"
 resource_group_name = azurerm_resource_group.vmss.name
 sku                  = "Standard_DS1_v2"
 instances            = 2
 admin_username       = "rakshith"
 admin_password       = "temporary@54321"
 disable_password_authentication = false


 source_image_reference {
    publisher = "Canonical"
    offer     = "0001-com-ubuntu-server-focal"
    sku       = "20_04-lts"
    version   = "latest"
}

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


 network_interface {
   name    = "terraformnetworkprofile"
   primary = true

   ip_configuration {
     name                                  = "IPConfiguration"
     subnet_id                             = azurerm_subnet.vmss.id
"vmss.tf" 36L, 991B
```

Now write the configuration for virtual machine: vm.tf

```
resource "azurerm_public_ip" "practice" {
 name                 = "practice-vm-public-ip"
 location             = "westeurope"
 resource_group_name        = azurerm_resource_group.vmss.name
 allocation_method          = "Static"
 domain_name_label          = "${random_string.fqdn.result}-ssh"
}
resource "azurerm_network_interface" "practice" {
 name              = "practice-nic"
 location          = "westeurope"
 resource_group_name = azurerm_resource_group.vmss.name
 ip_configuration {
   name                          = "IPConfiguration"
   subnet_id                     = azurerm_subnet.vmss.id
   private_ip_address_allocation = "dynamic"
   public_ip_address_id          = azurerm_public_ip.practice.id
```

```
 }
}
resource "azurerm_linux_virtual_machine" "practice" {
 name                = "practice"
 location            = "westeurope"
 resource_group_name   = azurerm_resource_group.vmss.name
 network_interface_ids = [azurerm_network_interface.practice.id]
 size              = "Standard_DS1_v2"
 source_image_reference {
    publisher = "Canonical"
    offer     = "0001-com-ubuntu-server-focal"
    sku       = "20_04-lts"
    version   = "latest"
}
 os_disk {
    caching              = "ReadWrite"
    storage_account_type = "Standard_LRS"
 }
 computer_name                  = "practicevm"
 admin_username                 = "rakshith"
 admin_password                 = "temporary@54321"
 disable_password_authentication = false
}
```

```
resource "azurerm_public_ip" "practice" {
  name                         = "practice-vm-public-ip"
  location                     = "westeurope"
  resource_group_name          = azurerm_resource_group.vmss.name
  allocation_method            = "Static"
  domain_name_label            = "${random_string.fqdn.result}-ssh"
}

resource "azurerm_network_interface" "practice" {
  name                = "practice-nic"
  location            = "westeurope"
  resource_group_name = azurerm_resource_group.vmss.name

  ip_configuration {
    name                          = "IPConfiguration"
    subnet_id                     = azurerm_subnet.vmss.id
    private_ip_address_allocation = "dynamic"
    public_ip_address_id          = azurerm_public_ip.practice.id
  }

}

resource "azurerm_linux_virtual_machine" "practice" {
  name                  = "practice"
  location              = "westeurope"
  resource_group_name   = azurerm_resource_group.vmss.name
  network_interface_ids = [azurerm_network_interface.practice.id]
  size               = "Standard_DS1_v2"

  source_image_reference {
     publisher = "Canonical"
```

We have created required files for the virtual machine scale sets:

```
rakshith [ ~/practice ]$ ls
lb.tf  providers.tf  publicip.tf  randomstring.tf  resourcegroup.tf  subnet.tf  vmss.tf  vm.tf  vn.tf
```

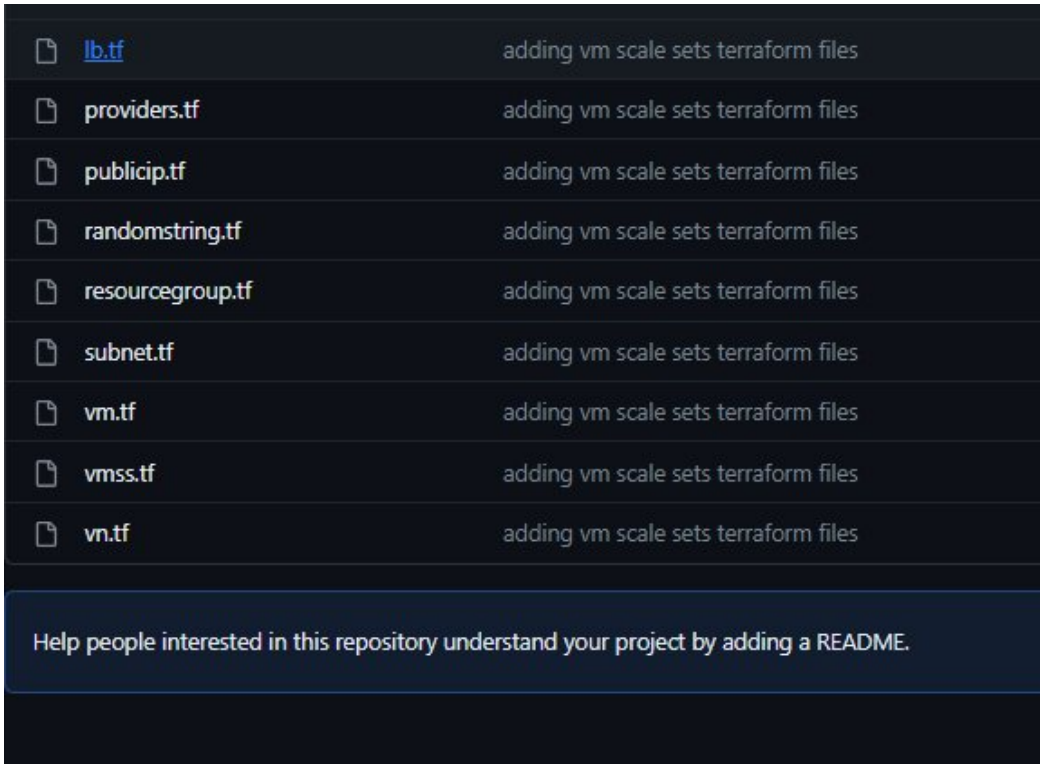
We will push it to github our porgress till here.

```
rakshith [ ~/practice ]$ git remote add origin git@github.com:kotianrakshith/AzureProj4.git
rakshith [ ~/practice ]$ git branch -M main
```

We have added the files and comitted:

```
rakshith [ ~/practice ]$ git add .
rakshith [ ~/practice ]$ git commit -m "adding vm scale sets terraform files"
```

Then we can push using “git push -u origin main”



We will have all the required files in github

3. **Write terraform configuration file for AKS**

First we will create a new resource group for kubernetes cluser:

aksrg.tf

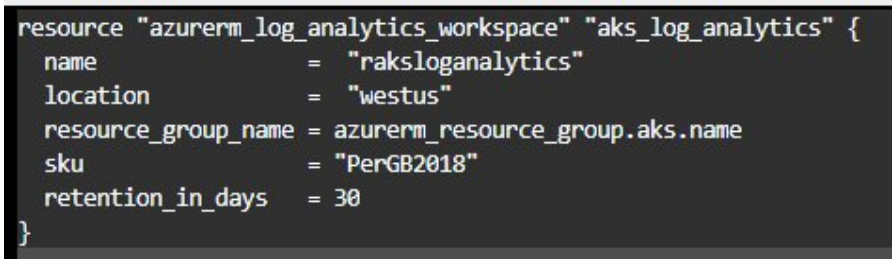
```
resource "azurerm_resource_group" "aks" {
 name     = "aks_rakshith"
 location = "westus"
}
```

```
resource "azurerm_resource_group" "aks" {
 name     = "aks_rakshith"
 location = "westus"
}
```

First we will write a configuration for log analytics workspace (optional), we can assign this to our aks cluser

loganalyticsworkspace.tf

```
resource "azurerm_log_analytics_workspace" "aks_log_analytics" {
 name                = "raksloganalytics"
 location            = "westus"
 resource_group_name = azurerm_resource_group.aks.name
 sku                 = "PerGB2018"
 retention_in_days   = 30
}
```



Now we will write a configuration file for AKS:

aks.tf

```
resource "azurerm_kubernetes_cluster" "cluster" {
 name                = "rakscluster123"
 location            = "westus"
 resource_group_name = azurerm_resource_group.aks.name
 dns_prefix          = "aksrakshith-cluster"

 default_node_pool {
  name       = "default"
  node_count = "1"
  vm_size    = "standard_d2_v2"
 }
 identity {
  type = "SystemAssigned"
 }
 addon_profile {
  azure_policy {enabled =  true}
  oms_agent {
   enabled =  true
```

```
    log_analytics_workspace_id = azurerm_log_analytics_workspace.aks_log_analytics.id
  }
 }
}
```

```
resource "azurerm_kubernetes_cluster" "cluster" {
  name                = "raksclusterr123"
  location            = "westus"
  resource_group_name = azurerm_resource_group.aks.name
  dns_prefix          = "aksrakshith-cluster"

  default_node_pool {
    name       = "default"
    node_count = "1"
    vm_size    = "standard_d2_v2"
  }

  identity {
    type = "SystemAssigned"
  }
  addon_profile {
    azure_policy {enabled =  true}
    oms_agent {
      enabled =  true
      log_analytics_workspace_id = azurerm_log_analytics_workspace.aks_log_analytics.id
    }
  }
}
```

Now we will add this also to staging and commit it and push it to github.and then we will have all the files both in our system and github:

Github link: https://github.com/kotianrakshith/AzureProj4

4. **Apply the terraform configuration**

First we will run initializing command to download required providers:

```
terrafrom init
```

```
rakshith [ ~/practice/AzureProj4 ]$ terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/random...
- Finding hashicorp/azurerm versions matching "~> 2.0"...
- Installing hashicorp/random v3.5.1...
- Installed hashicorp/random v3.5.1 (signed by HashiCorp)
- Installing hashicorp/azurerm v2.99.0...
- Installed hashicorp/azurerm v2.99.0 (signed by HashiCorp)
```

Then we will run terraform plan to check if there are any errors and if not, what are the resource which will be built:

```
terraform plan
```

```
commands will detect it and remind you to do so if necessary.
rakshith [ ~/practice/AzureProj4 ]$ terraform plan

Terraform used the selected providers to generate the following execution pla
  + create

Terraform will perform the following actions:
```

```
Plan: 16 to add, 0 to change, 0 to destroy.
```

Now we will run the apply command to build all this 16 resource

terraform apply

```
rakshith [ ~/practice/AzureProj4 ]$ terraform apply

Terraform used the selected providers to generate the follo
  + create

Terraform will perform the following actions:
```

This will prompt for approval, type yes

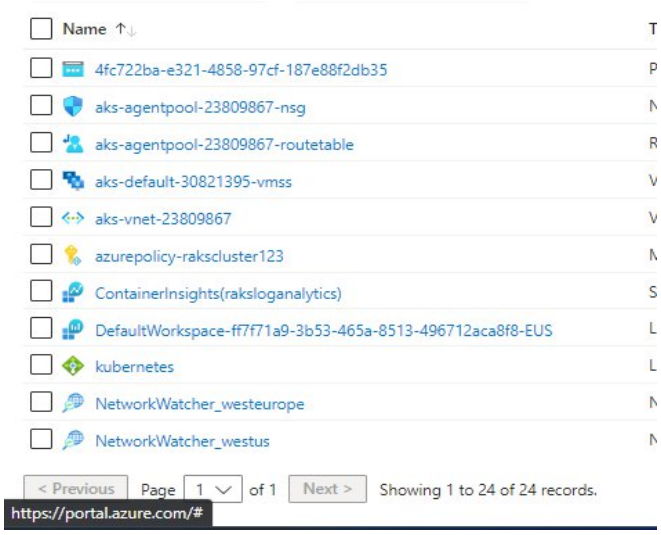
```
Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes
```

This will take some time and once all are created it will inform that all resource are created:

```
Apply complete! Resources: 16 added, 0 changed, 0 destroyed.
rakshith [ ~/practice/AzureProj4 ]$
```

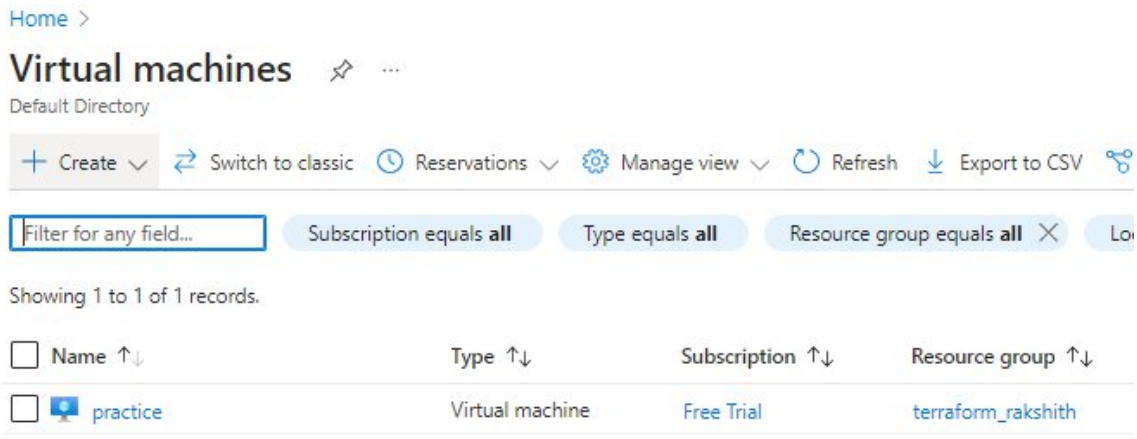
5. **Check the deployed infrastructure**

First you can go to All resource in azure portal and you can see all the resource that is created

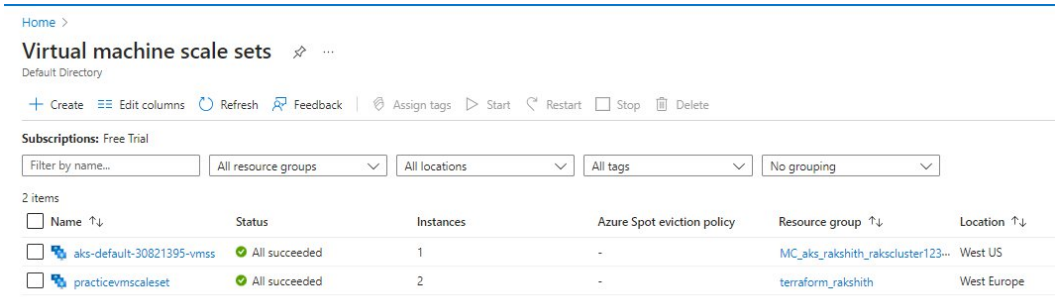


Now you can check the required deployment:

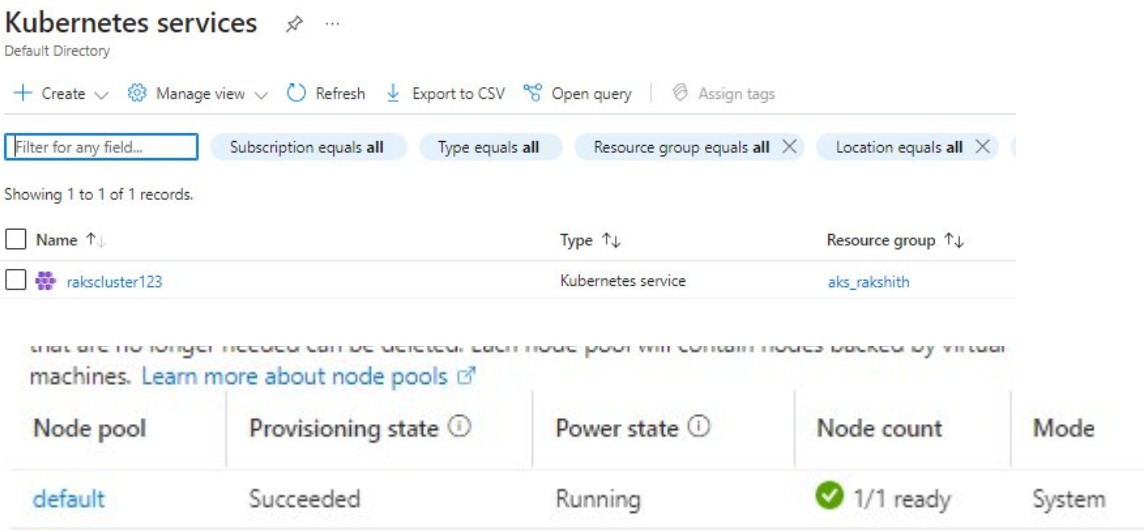
VM: You can see our virtual machine is created in the resource group we specified



VMSS: You can see that our scale set is created(One for AKS cluster, One which we created)



AKS: You can go to kubernetes service and check that your cluster is created in mentioned resource group:



That completes the building infrastrcutre using IaC tool Terraform. We can see all the required dependency are created with minimal time and the code can be reused.

6. **Destroy the infrastructure:**

You can delete all the infrastructure using simple command: terraform destroy

```
rakshith [ ~/practice/AzureProj4 ]$  terraform destroy
random_string.fqdn: Refreshing state... [id=qmyjzw]
azurerm_resource_group.aks: Refreshing state... [id=/subscriptions/f
```

It will ask for approval:

```
Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value: yes
```

Once completed all the resource will be deleted. You can check in all resources if ther resources has been removed:

Name	Type
DefaultWorkspace-ff7f71a9-3b53-465a-8513-496712aca8f8-EUS	Log Analytics workspa
NetworkWatcher_westeurope	Network Watcher
NetworkWatcher_westus	Network Watcher
shellstorageaccount2105	Storage account

< Previous Page 1 of 1 Next > Showing 1 to 4 of 4 records.