**Terraform**

**Components of Terraform:**

1. **Provider**
2. **Resources**
3. **Variables**
4. **Statefile**
5. **Provisioners**
6. **Backends**
7. **Modules**
8. **Data Sources**
9. **Service Principals**

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Terraform init**Install Azurerm Plugin, it’s a plugin interact with azure cloud  
download the pluggin and it wil connect cloud using service principle

Terraform plan – it will show the plan  
It will create a statefile

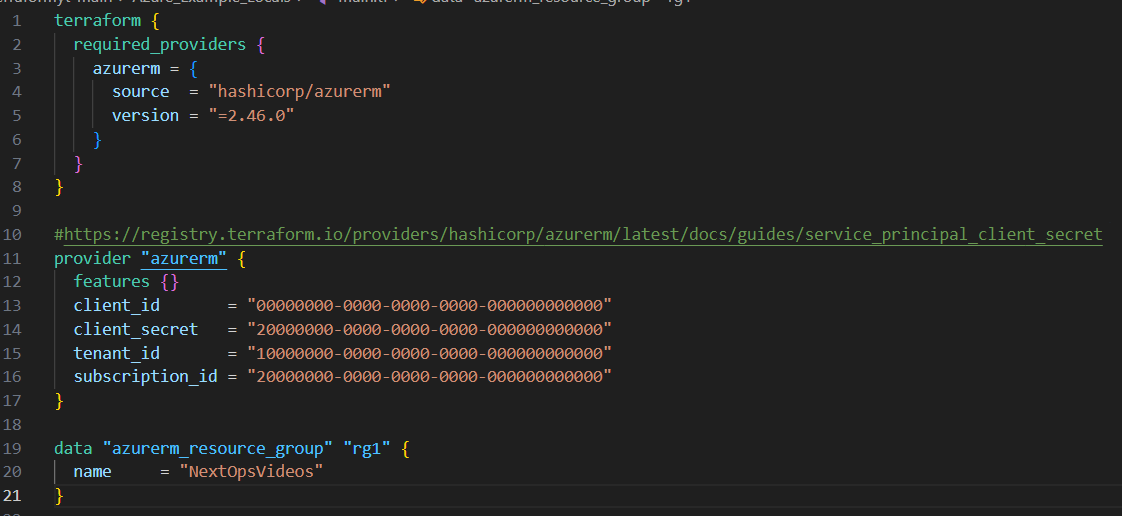
Terrafrom apply – deploy the companest

Go to azure AD, Create app registrations, certificate and secrets

Use Client id, tenant id, Sub id, secrte

Give privileges to service principle

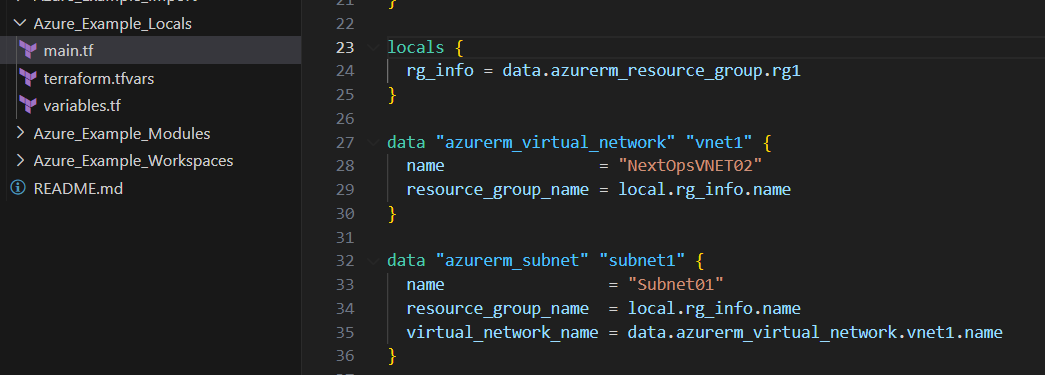
**Basic Terraform Structure:**



**Modules**

|  |  |
| --- | --- |
|  |  |

**Locals:**

****

**Terrarom Workspace:**

The terraform workspace new dev command is used to create a new workspace and switched to a new workspace.

The terraform workspace list command is used to list all existing workspaces.

The terraform workspace select dev command is used to choose a different workspace to use for further operations.

The terraform workspace delete dev command is used to delete an existing workspace.

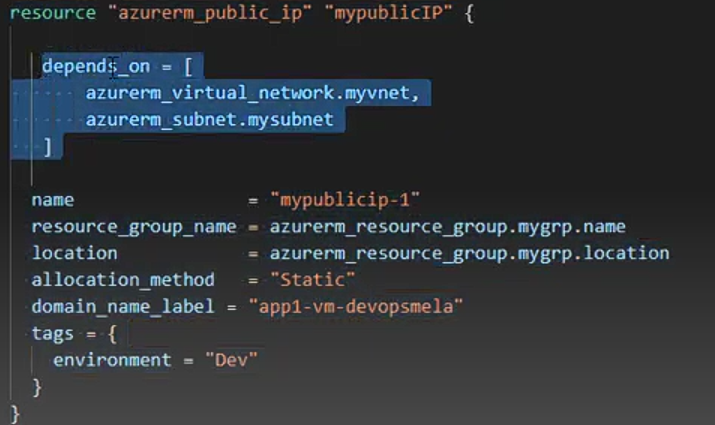
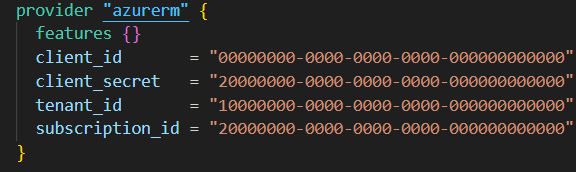
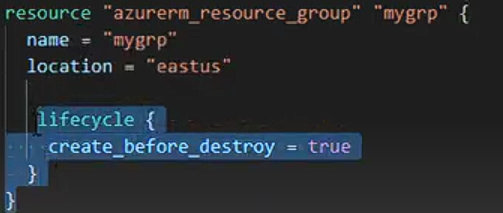
The terraform workspace show command is used to output the current workspace.

**Terraform Import**

Terraform import

* terraform import azurerm\_resource\_group.rg1 /subscriptions/MySubscriptionNumber/resourceGroups/MyResourceGroup

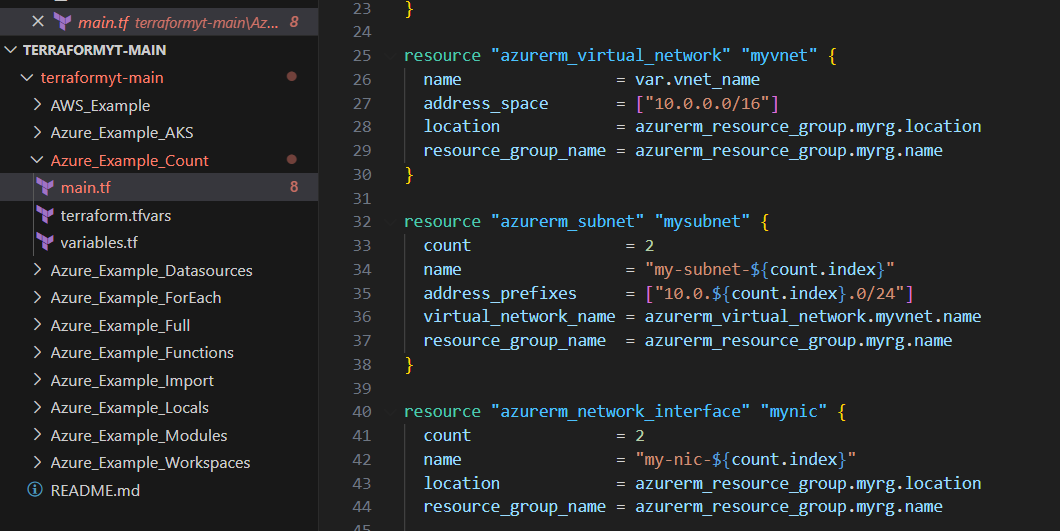
**Terraform Meta-Argument:**

1. **Depends\_on  
   **
2. **Provider  
   **
3. **Lifecycle  
   **
4. **Count**
5. **For\_each**

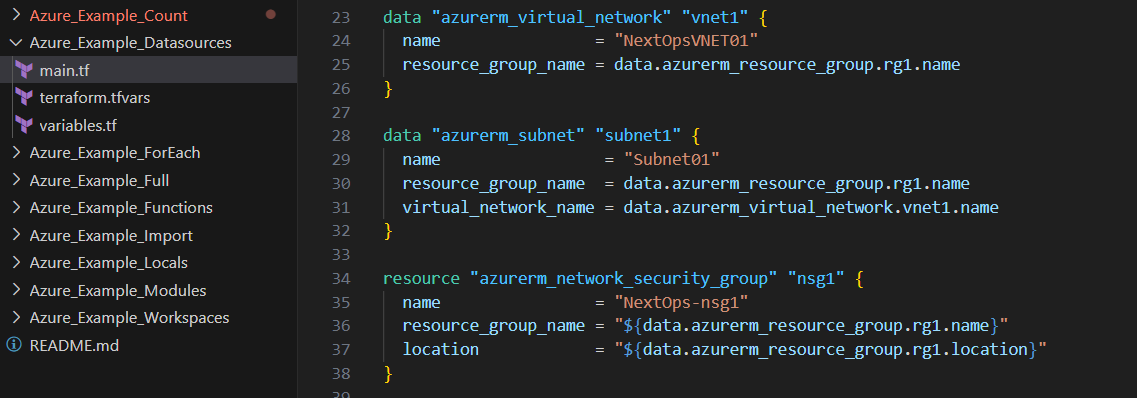
**4.For Each:**

|  |  |
| --- | --- |
|  |  |

**5.Count:**

****

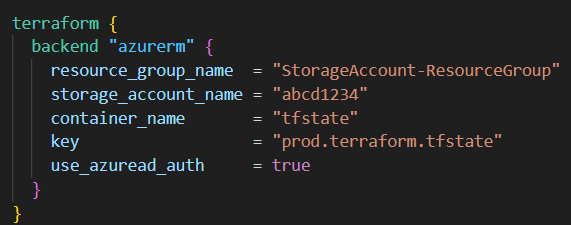
**Data Sources:  
while giving destroy command data resource not delete**

****

**Provisioners**

1. **File**
2. **local exec**
3. **Remote-exec**

**Terraform Backend:**

****

**Terraform Taint and untaint**

Resource will be **destroyed and recreated** during the next apply operation. When state undesirable or **unexpected state**, but its configuration **hasn’t changed**. Terraform basically **forces the recreation or Redeploy** the resources even if the configuration matches the current state.

* terraform **taint** module.example.aws\_instance.example
* terraform **untaint** module.example.aws\_instance.example
* terraform **apply** -**replace**=" module.example.aws\_instance.example "

**Key Vault Task in Terraform**

* Before adding you are authorized to create a secrets, IAM add role assignments, Contributor group
* Add these 2 secrets in azure key vault
* Add service principle and create a access the policy 🡪 **get, list update, create import, delete, recover, Backup**

Note:

Terrafrom.tfstate file and Terrafrom.tfstate.backup file, this back file will create a 2 place of time, 1. Terrform destroy or taint

In that if you open tfstate file all will be deleted, if you want restore, just replace Terrafrom.tfstate and remove .back in above file and apply commads use it  
Terrafrom.tfstate.backup to Terrafrom.tfstate

|  |  |
| --- | --- |
|  |  |

**Terraform CICD Code**

|  |  |
| --- | --- |
| **trigger:**  **- main**  **pool:**  **vmImage: ubuntu-latest**  **jobs:**  **- job: Validate**  **displayName: Terraform Validate**  **pool:**  **vmImage: ubuntu-latest**  **steps:**  **- checkout: self**  **- task: TerraformInstaller@0**  **displayName: Install Terraform latest**  **- task: TerraformTaskV2@2**  **displayName: 'Terraform : Init'**  **inputs:**  **workingDirectory: $(System.DefaultWorkingDirectory)/dev/**  **backendServiceArm: 'sc-dowdtf'**  **backendAzureRmResourceGroupName: dowd-devops-rg**  **backendAzureRmStorageAccountName: dowdtf**  **backendAzureRmContainerName: tfstatedowd**  **backendAzureRmKey: terraform.tfstate**  **- task: TerraformTaskV2@2**  **displayName: ‘Validate'**  **inputs:**  **command: validate** | terraform {  required\_providers {  azurerm = {  source = "hashicorp/azurerm"  version = "=2.46.0"  }  }  backend "azurerm" {  resource\_group\_name = "dowd-devops-rg"  storage\_account\_name = "dowdtf"  container\_name = "tfstatedowd"  key = "terraform.tfstate"  }  }  provider "azurerm" {  features {}  }  data "azurerm\_client\_config" "current" {}  resource "azurerm\_resource\_group" "dowd-rg" {  name = "dowd-tf"  location = "uksouth"  }  resource "azurerm\_storage\_account" "dowdsa" {  name = "dowdsatf"  resource\_group\_name = azurerm\_resource\_group.dowd-rg.name  location = azurerm\_resource\_group.dowd-rg.location  account\_tier = "Standard"  account\_replication\_type = "LRS"  } |
| **- job: Deploy**  **displayName: Terraform Deploy**  **pool:**  **vmImage: ubuntu-latest**  **steps:**  **- checkout: self**  **- task: TerraformInstaller@0**  **displayName: Install Terraform latest**  **- task: TerraformTaskV2@2**  **displayName: 'Terraform : Init'**  **inputs:**  **workingDirectory: $(System.DefaultWorkingDirectory)/dev/**  **backendServiceArm: 'sc-dowdtf'**  **backendAzureRmResourceGroupName: dowd-devops-rg**  **backendAzureRmStorageAccountName: dowdtf**  **backendAzureRmContainerName: tfstatedowd**  **backendAzureRmKey: terraform.tfstate**  **- task: TerraformTaskV2@2**  **displayName: 'Terraform : Plan'**  **inputs:**  **command: plan**  **environmentServiceNameAzureRM: 'sc-dowdtf'**  **- task: TerraformTaskV2@2**  **displayName: 'Apply'**  **inputs:**  **command: apply**  **commandOption: --auto-approve**  **environmentServiceNameAzureRM: 'sc-dowdtf'** |

**Git**

**How to undo last commit**

**Git reset HEAD~1**

**Git revert commit\_id**

**How to deploy multiple resources at single deployment**

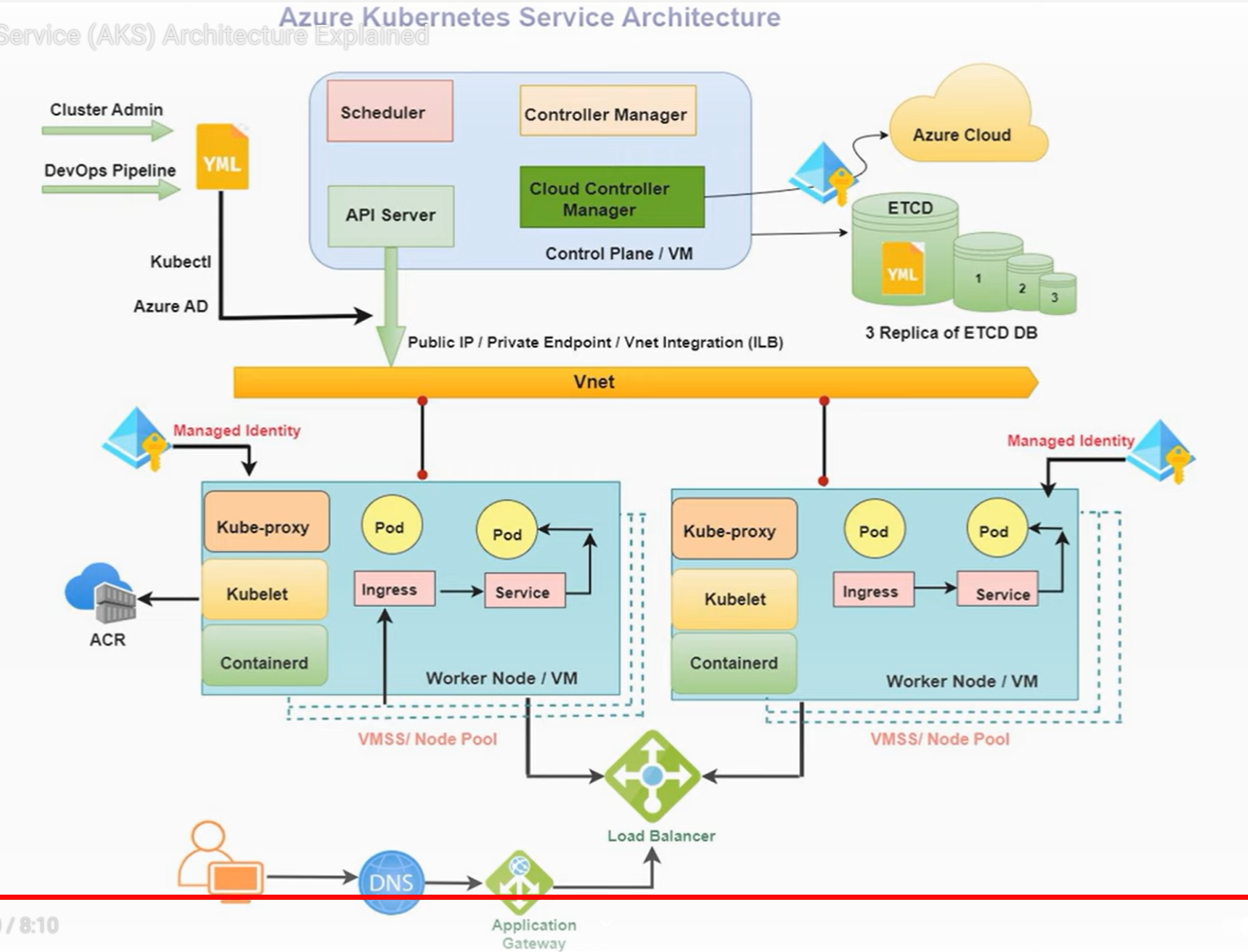
**Through function**

1. **Count**
2. **For each**

|  |  |
| --- | --- |
| **Count** |  |
| **resource "azurerm\_subnet" "mysubnet" {**  **count = 2**  **name = "my-subnet-${count.index}"**  **address\_prefixes = ["10.0.${count.index}.0/24"]**  **virtual\_network\_name = azurerm\_virtual\_network.myvnet.name**  **resource\_group\_name = azurerm\_resource\_group.myrg.name**  **}**  **ip\_configuration {**  **name = "my-ip-config"**  **subnet\_id = azurerm\_subnet.mysubnet[count.index].id**  **private\_ip\_address\_allocation = "Dynamic"**  **}**  **}**  **-${count.index}" (0,1,2,3) : Interpolation** | **variable "resourcedetails" {**  **type = map(object({**  **name = string**  **location = string**  **size = string**  **rg\_name = string**  **vnet\_name = string**  **subnet\_name = string**  **}))**  **default = {**  **westus = {**  **rg\_name = "westus-rg"**  **name = "west-vm"**  **location = "westus2"**  **size = "Standard\_B2s"**  **vnet\_name = "west-vnet"**  **subnet\_name = "west-subnet"**  **}**  **eastus = {**  **rg\_name = "eastus-rg"**  **name = "east-vm"**  **location = "eastus"**  **size = "Standard\_B1s"**  **vnet\_name = "east-vnet"**  **subnet\_name = "east-subnet"**  **}**  **}**  **}** |
| **For each** |
| **resource "azurerm\_resource\_group" "myrg" {**  **for\_each = var.resourcedetails**  **name = each.value.rg\_name**  **location = each.value.location**  **}**  **resource "azurerm\_virtual\_network" "myvnet" {**  **for\_each = var.resourcedetails**  **name = each.value.vnet\_name**  **address\_space = ["10.0.0.0/16"]**  **location = azurerm\_resource\_group.myrg[each.key].location**  **resource\_group\_name = azurerm\_resource\_group.myrg[each.key].name**  **}** |

**What is SSL and TLS ?**

[Azure Kubernetes Service (AKS) Architecture Explained (youtube.com)](https://www.youtube.com/watch?v=lF8_NhLp4uI&list=PLWgk5rA0Qxeb6_2FRqQlbxmgj3BPSwZco&index=39)

****