Terraform

Installation

- Use Homebrew package manager.

  ```
  brew tap hashicorp/tap
  brew install hashicorp/tap/terraform
  ```

- Try simple to use terraform on local machine.
  - Create file .tf

    ```
    #To create a file that contain some text

    resource "null_resource" "file-create" {
      provisioner "local-exec" {
        command = "echo 'Hello, World!' > hello.txt"
      }
    }
    ```

    `terraform init`

    `terraform plan`

    `terraform apply`

  - Now check inside project folder you will see `hello.txt` file, inside contain “Hello, world”

Terraform with azure

- After, installed already if we need to use terraform with any provider we need to do **Authenticating to Azure**
  - Authenticating to Azure using the Azure CLI
    - `az login` (login vai website method)
    - `az account set --subscription="SUBSCRIPTION_ID”`
- Let’s create simple script for create

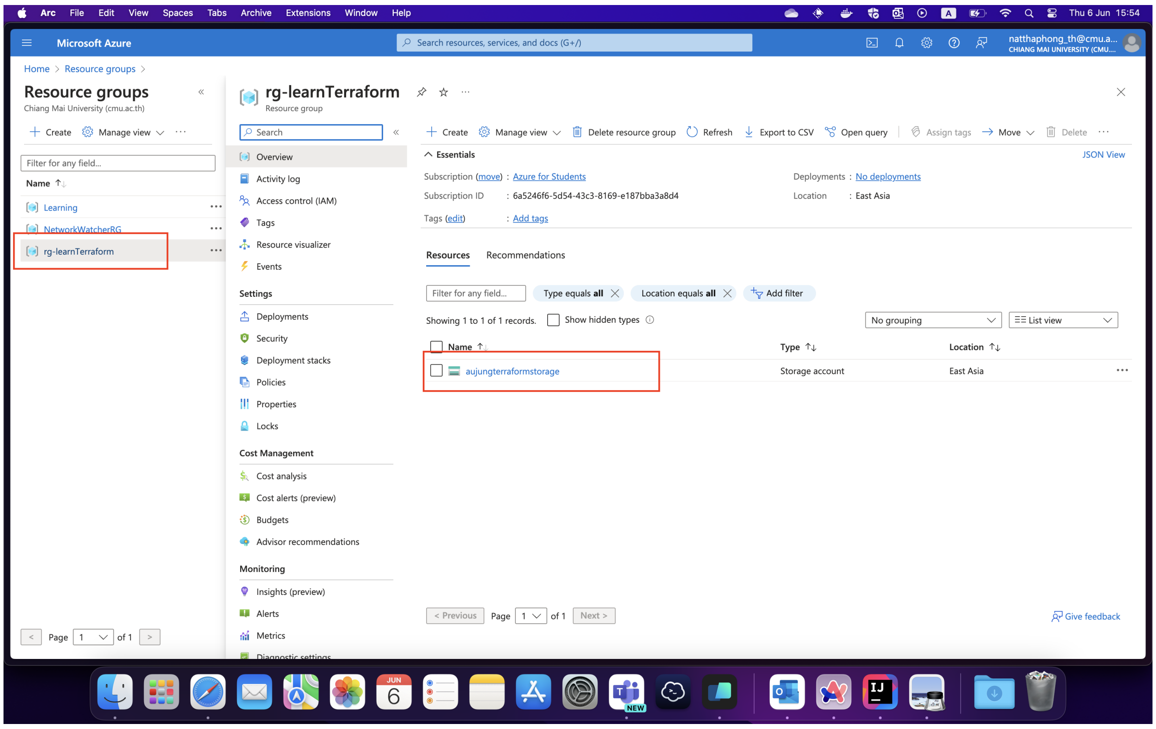
    - resource group
    - storage account including
        - web hosting file
- new file call main.tf

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

#create a resource group`
resource "azurerm_resource_group" "rg" {
  name     = "rg-learnTerraform"
  location = "East Asia"
}

#create a Storage Account
resource "azurerm_storage_account" "storage" {
  name                     = "aujungterraformstorage"
  resource_group_name      = azurerm_resource_group.rg.nam
  location                 = azurerm_resource_group.rg.loc
  account_tier             = "Standard"
  account_replication_type = "LRS"
  account_kind = "StorageV2" // StorageV2 is required for 

  static_website {
    index_document = "index.html"
  }
}

#add index.html file
resource "azurerm_storage_blob" "blob" {
  name                   = "index.html"
  storage_account_name   = azurerm_storage_account.storage
  storage_container_name = "$web"
  type                   = "Block"
  content_type = "text/html"
  source_content = "<html><body><h1>Hello, Terraform! from
}
```

- after that execute this respectively

```
$ terraform init
$ terraform plan
$ terraform apply
```

- Everything work fine we will see an resource that was created in azure portal like this.



rg-learnTerraform
Resource group

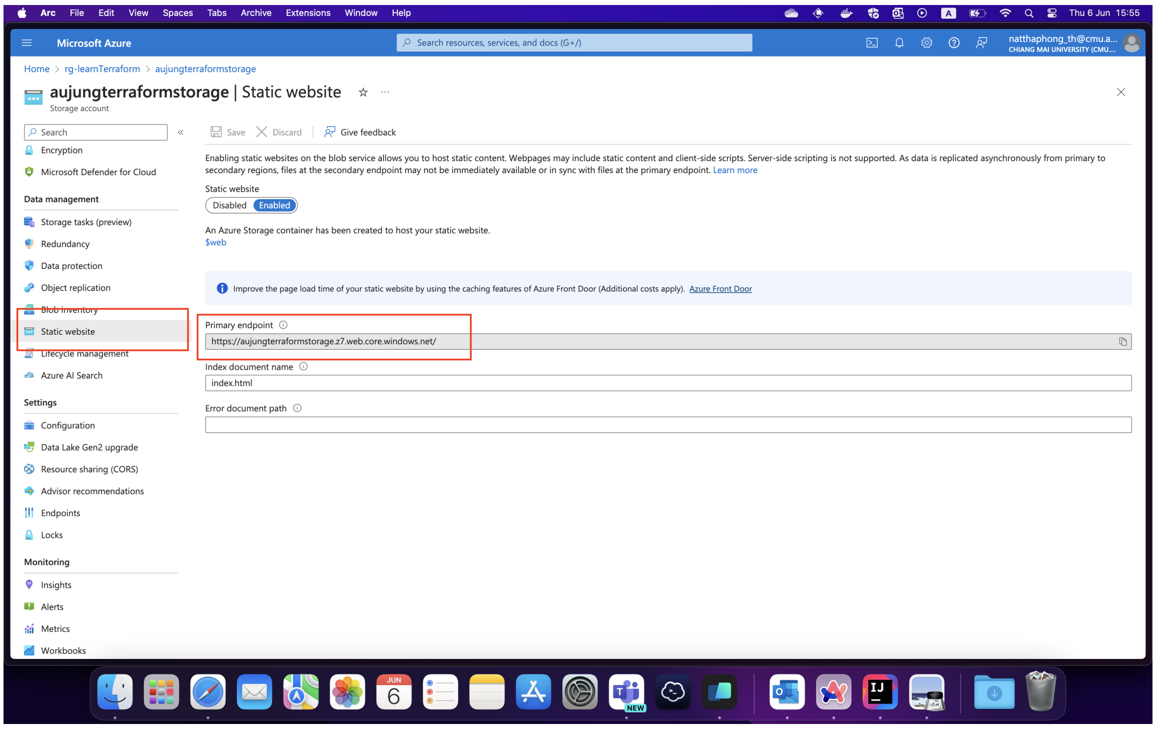
Essentials

Subscription (move) : Azure for Students
Subscription ID : 6a5246f6-5d54-43c3-8169-e187bba3a8d4
Tags (edit) : Add tags
Deployments : No deployments
Location : East Asia

Showing 1 to 1 of 1 records.

Name	Type	Location
aujungterraformstorage	Storage account	East Asia

- Let's see an static web that we created via



aujungterraformstorage | Static website

Storage account

Enabling static websites on the blob service allows you to host static content. Webpages may include static content and client-side scripts. Server-side scripting is not supported. As data is replicated asynchronously from primary to secondary regions, files at the secondary endpoint may not be immediately available or in sync with files at the primary endpoint. Learn more

Static website: Disabled [✓ Enabled]

An Azure Storage container has been created to host your static website.
$web

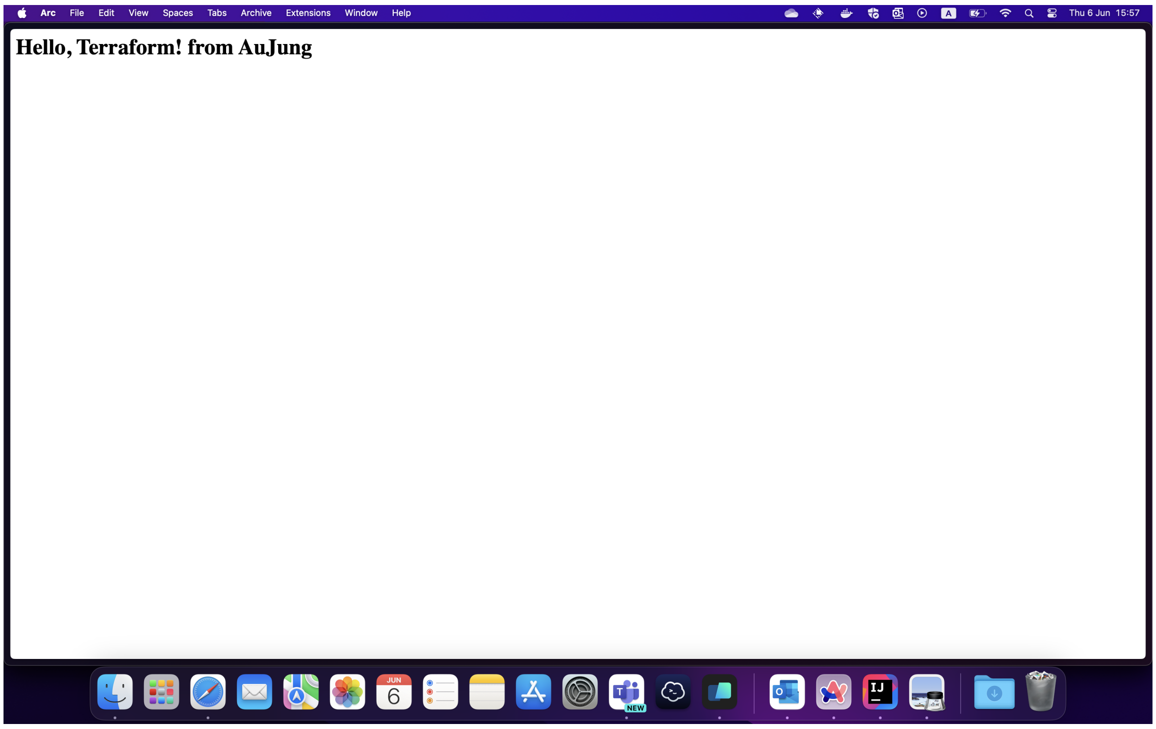
Improve the page load time of your static website by using the caching features of Azure Front Door (Additional costs apply). Azure Front Door

Primary endpoint: https://aujungterraformstorage.z7.web.core.windows.net/

Index document name: index.html

Error document path:

- the result on that web should be this.


Arc File Edit View Spaces Tabs Archive Extensions Window Help
Thu 6 Jun 15:57

Hello, Terraform! from AuJung

Create Vm and setup docker with terraform

```
#main.tf
```

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


# Create a resource group
resource "azurerm_resource_group" "rg" {
  location = "East Asia"
  name     = "aujung-rg"
}

# Create a virtual network
resource "azurerm_virtual_network" "vnet" {
  name = "aujung-vnet"
  address_space = ["10.0.0.0/16"]
  location = azurerm_resource_group.rg.location
  resource_group_name = azurerm_resource_group.rg.name
}

# Create a subnet
resource "azurerm_subnet" "subnet" {
  name                 = "aujung-subnet"
  resource_group_name  = azurerm_resource_group.rg.name
  virtual_network_name = azurerm_virtual_network.vnet.name
  address_prefixes     = ["10.0.0.0/24"]

}

# Create a public IP
resource "azurerm_public_ip" "ip" {
  location                = azurerm_resource_group.rg.locatio
  name                    = "aujung-ip"
  resource_group_name     = azurerm_resource_group.rg.name
  allocation_method       = "Dynamic"
}

# Create a network interface
```

```
resource "azurerm_network_interface" "nic" {
  location            = azurerm_resource_group.rg.location
  name                = "aujung-nic"
  resource_group_name = azurerm_resource_group.rg.name

  ip_configuration {
    name                          = "aujung-ipconfig"
    subnet_id                     = azurerm_subnet.subnet.id
    private_ip_address_allocation = "Dynamic"
    public_ip_address_id = azurerm_public_ip.ip.id
  }
}

# Setup inbound security rules
resource "azurerm_network_security_group" "nsg" {
  location            = azurerm_resource_group.rg.location
  name                = "aujung-nsg"
  resource_group_name = azurerm_resource_group.rg.name

  security_rule {
    name                       = "SSH"
    priority                   = 1001
    direction                  = "Inbound"
    access                     = "Allow"
    protocol                   = "Tcp"
    source_port_range          = "*"
    destination_port_range     = "22"
    source_address_prefix      = "*"
    destination_address_prefix = "*"
  }

  security_rule {
    name                       = "HTTP"
    priority                   = 1002
    direction                  = "Inbound"
    access                     = "Allow"
    protocol                   = "Tcp"
    source_port_range          = "*"
    destination_port_range     = "80"
    source_address_prefix      = "*"
    destination_address_prefix = "*"
  }
}

# Apply the network security group to vm's network interface
resource "azurerm_network_interface_security_group_associatio
  network_interface_id      = azurerm_network_interface.nic.i
  network_security_group_id = azurerm_network_security_group.
}

# Create a vm
resource "azurerm_linux_virtual_machine" "vm" {
  admin_username        = "aujung"
  location              = azurerm_resource_group.rg.location
  name                  = "aujung-vm"
  network_interface_ids = [azurerm_network_interface.nic.id]
  resource_group_name = azurerm_resource_group.rg.name
  size                  = "Standard_B1ls"

  admin_ssh_key {
    public_key = file("~/.ssh/id_rsa.pub")
    username   = "aujung"
  }

  os_disk {
    caching              = "ReadWrite"
    storage_account_type = "Standard_LRS"
    disk_size_gb = 30
  }

  source_image_reference {
    publisher = "Canonical"
    offer     = "0001-com-ubuntu-server-jammy"
    sku       = "22_04-lts"
    version   = "latest"
  }
}
```

```
# setup-docker.tf

# Install Docker on the virtual machine
resource "null_resource" "install_docker" {
  triggers = {
    vm_id = azurerm_linux_virtual_machine.vm.id
  }

  connection {
    type        = "ssh"
    host        = azurerm_linux_virtual_machine.vm.public_ip_
    user        = azurerm_linux_virtual_machine.vm.admin_user
    private_key = file("~/.ssh/id_rsa")
  }

  # Remove old Docker packages
  provisioner "remote-exec" {
    inline = [
      "sudo apt-get remove -y docker docker-engine docker.io 
    ]
  }

  # Install Docker
  provisioner "remote-exec" {
    inline = [
      "sudo apt-get update -y",
      "sudo apt-get install -y -o=APT::Get::Assume-Yes=true c
      "sudo rm -rf /etc/apt/keyrings",
      "sudo mkdir -p /etc/apt/keyrings",
      "curl -fsSL https://download.docker.com/linux/ubuntu/gp
      sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg",
      "echo \"deb [arch=$(dpkg --print-architecture)
      signed-by=/etc/apt/keyrings/docker.gpg]
      https://download.docker.com/linux/ubuntu $(lsb_release 
      sudo tee /etc/apt/sources.list.d/docker.list > /dev/nul
      "sudo apt-get update -y",
      "sudo apt-get install -y -o=APT::Get::Assume-
      Yes=true
      docker-ce
      docker-ce-cli
      containerd.io",
      "sudo usermod -aG docker ${azurerm_linux_virtual_machin
    ]
  }

  # Configure Docker daemon
  provisioner "remote-exec" {
    inline = [
      "sudo mkdir -p /etc/docker",
      "sudo tee /etc/docker/daemon.json > /dev/null <<EOF",
      "{",
      "  \"log-driver\": \"json-file\",",
      "  \"log-opts\": {",
      "    \"max-size\": \"10m\",",
      "    \"max-file\": \"3\"",
      "  }",
      "}",
      "EOF",
      "sudo systemctl daemon-reload",
      "sudo systemctl restart docker",
    ]
  }
}

# Depoly nginx container

resource "null_resource" "nginx" {
  depends_on = [null_resource.install_docker]

  triggers = {
    vm_id = azurerm_linux_virtual_machine.vm.id
  }
  connection {
    type = "ssh"
    host = azurerm_linux_virtual_machine.vm.public_ip_address
    user = azurerm_linux_virtual_machine.vm.admin_username
    private_key = file("~/.ssh/id_rsa")
  }

  provisioner "remote-exec" {
    inline = [
      "sudo docker run -d -p 80:80 --name nginx nginx",
    ]
  }
}
```