Terraform for VMware vSphere | Quick Guide

Version 25.06

Ahmed Abdelwahed
ahmed@abdelwahed.me
www.abdelwahed.me
LinkedIn
GitHub

Step-by-Step: Setting Up Terraform on Windows

Step 1: Install Terraform

1. **Download Terraform:**
   - Go to the Terraform Downloads Page.
   - Download the Windows version of Terraform.
2. **Extract Terraform:**
   - Extract the .zip file to a folder (e.g., C:\Terraform).
3. **Add Terraform to PATH:**
   - Open **Control Panel > System > Advanced System Settings > Environment Variables**.
   - Under **System Variables**, locate Path and click **Edit**.
   - Add the directory where you extracted Terraform (e.g., C:\Terraform).
   - Click **OK** to save.
4. **Verify Installation:**
   - Open a **Command Prompt** or **PowerShell** and type:
     ```
     terraform --version
     ```
   - You should see the installed version of Terraform.

Step 2: Prepare Your Terraform Configuration Files

1. **Create a New Directory:**
   - Create a folder for your Terraform project (e.g., C:\MyTerraformProject).
   - Navigate to this folder in Command Prompt or PowerShell:
     ```
     cd C:\MyTerraformProject
     ```
2. **Write Terraform Configuration Files:**
   - Create a .tf file (e.g., main.tf) with your desired configuration using a text editor like Notepad++ or Visual Studio Code.

Step 3: Run terraform init

1. **Navigate to the Project Directory:**
   ```
   cd C:\MyTerraformProject
   ```
2. **Initialize Terraform:**
   - Run the following command:
     ```
     terraform init
     ```
3. **What Happens:**
   - Terraform downloads the necessary provider plugins.
   - It sets up the backend configuration for state management.
   - It ensures your working directory is ready for Terraform operations.
4. **Check for Success:**
   - If successful, you'll see output similar to:
     ```
     Terraform has been successfully initialized!
     ```

Lab1: Create 1 Linux VM from a template

main.tf

```
provider "vsphere" {
  vsphere_server = "vcenter.abdelwahed.me"
  user           = "administrator@vsphere.local"
  password       = "MyPassword12345"
  allow_unverified_ssl = true # Disable SSL verification
}
data "vsphere_datacenter" "dc" {
  name = "vDatacenter"
}
data "vsphere_compute_cluster" "cluster" {
  name          = "vCluster"
  datacenter_id = data.vsphere_datacenter.dc.id
}
data "vsphere_datastore" "ds" {
  name          = "NDatastore1"
  datacenter_id = data.vsphere_datacenter.dc.id
}
data "vsphere_network" "net" {
  name          = "DPortGroup"
  datacenter_id = data.vsphere_datacenter.dc.id
}
data "vsphere_virtual_machine" "template" {
  name          = "Linux"  # Template name from your environment
  datacenter_id = data.vsphere_datacenter.dc.id
}
resource "vsphere_virtual_machine" "vm" {
  name             = "Linux1"
  resource_pool_id = data.vsphere_compute_cluster.cluster.resource_pool_id
  datastore_id     = data.vsphere_datastore.ds.id
  num_cpus = 2
  memory   = 1024  # Set memory to 1 GB
  guest_id = data.vsphere_virtual_machine.template.guest_id
  firmware = "efi"  # Set firmware to EFI
  network_interface {
    network_id   = data.vsphere_network.net.id
    adapter_type = "vmxnet3"
  }
  disk {
    label            = "disk0"
    size             = data.vsphere_virtual_machine.template.disks.0.size
    eagerly_scrub    = false
    thin_provisioned = true
  }
  clone {
    template_uuid = data.vsphere_virtual_machine.template.id
```

```
      customize {
        linux_options {
          host_name = "linux1"
          domain    = "local"
        }
        network_interface {
          ipv4_address = "192.168.221.100"  # Adjust as needed
          ipv4_netmask = 24
        }
        ipv4_gateway = "192.168.221.2"  # Updated gateway
        dns_server_list = ["192.168.221.200"]  # DNS server
      }
    }
  }
```

Summary of Terraform Code Actions

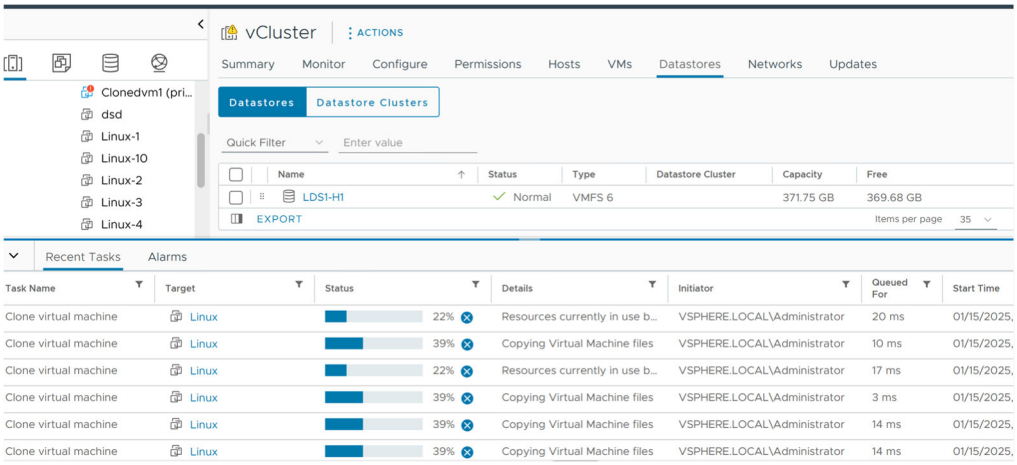
- **Configures the vSphere Provider**
  - Connects to the vCenter server at vcenter.abdelwahed.me using the specified credentials.
  - Disables SSL certificate verification (allow_unverified_ssl = true).
- **Reads Existing vSphere Infrastructure Objects**
  - Loads a datacenter named **vDatacenter**.
  - Loads a compute cluster named **vCluster** within the specified datacenter.
  - Loads a datastore named **NDatastore1** from the same datacenter.
  - Loads a network named **DPortGroup** from the same datacenter.
  - Loads a VM template named **Linux** to use as the cloning source.
- **Creates a New Virtual Machine (VM) Named Linux1**
  - Places the VM in the resource pool of the vCluster compute cluster.
  - Uses the NDatastore1 for VM storage.
  - Assigns **2 CPUs** and **1 GB memory** to the VM.
  - Sets the VM's firmware to **EFI**.
  - Configures a single network interface:
    - Connects to the DPortGroup network.
    - Uses the vmxnet3 adapter type.
  - Configures a single virtual disk:
    - Size matches the first disk of the Linux template.
    - Disk is thin-provisioned (saves space).
    - Disk is not eagerly scrubbed.
- **Clones the New VM from the Linux Template**
  - Specifies the template's UUID for cloning.
  - Customizes the guest OS:
    - Sets hostname to **linux1** and domain to **local**.
    - Sets static IPv4 address to 192.168.221.100, netmask to /24.
    - Sets gateway to 192.168.221.2.
    - Uses DNS server 192.168.221.200.

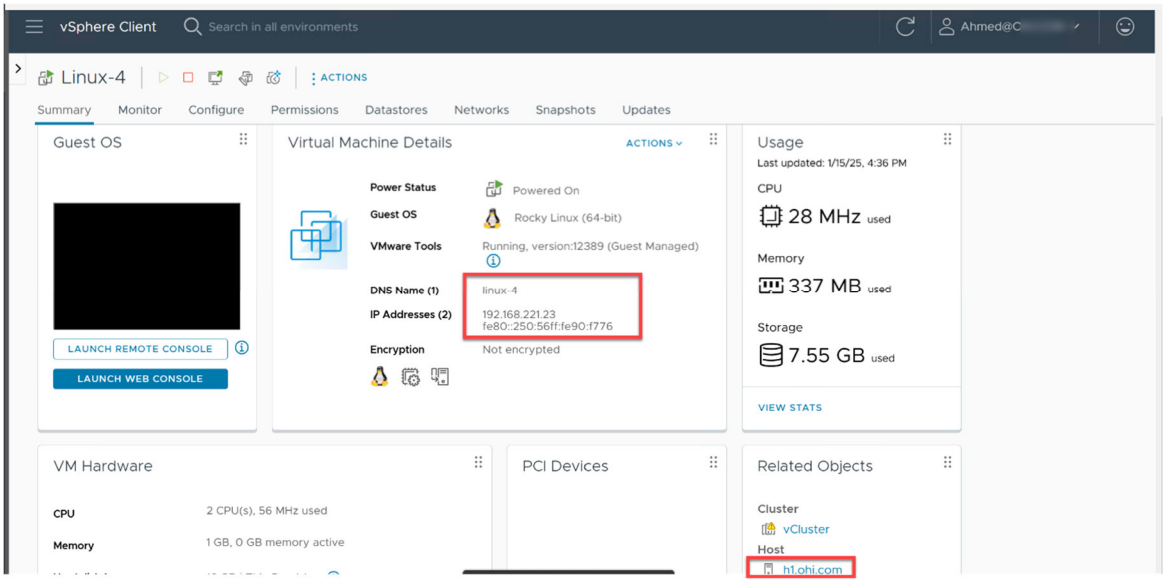
Lab2: Deploy 10 Virtual Machines with Custom IP Range in vSphere Cluster

main.tf

```
provider "vsphere" {
  vsphere_server = "vcenter.abdelwahed.me"

  user           = "administrator@vsphere.local"
  password       = "Mypassword12345"
  allow_unverified_ssl = true # Disable SSL verification
}
data "vsphere_datacenter" "dc" {
  name = "vDatacenter"
}
data "vsphere_compute_cluster" "cluster" {
  name          = "vCluster"
  datacenter_id = data.vsphere_datacenter.dc.id
}
data "vsphere_datastore" "ds" {
  name          = "NDatastore1"
  datacenter_id = data.vsphere_datacenter.dc.id
}
data "vsphere_network" "net" {
  name          = "DPortGroup"
  datacenter_id = data.vsphere_datacenter.dc.id
}
data "vsphere_virtual_machine" "template" {
  name          = "Linux"  # Template name from your environment
  datacenter_id = data.vsphere_datacenter.dc.id
}
resource "vsphere_virtual_machine" "vm" {
  count           = 2  # Create 10 VMs
  name            = "Linux-${count.index + 1}"  # Name VMs as Linux-1, Linux-2, etc.
  resource_pool_id = data.vsphere_compute_cluster.cluster.resource_pool_id
  datastore_id    = data.vsphere_datastore.ds.id
  num_cpus = 2
  memory   = 1024  # Set memory to 1 GB
  guest_id = data.vsphere_virtual_machine.template.guest_id
  firmware = "efi"  # Set firmware to EFI
  network_interface {
    network_id   = data.vsphere_network.net.id
    adapter_type = "vmxnet3"
  }

  disk {
    label            = "disk0"
    size             = data.vsphere_virtual_machine.template.disks.0.size
    eagerly_scrub    = false
```

```
    thin_provisioned = true
  }
  clone {
    template_uuid = data.vsphere_virtual_machine.template.id

    customize {
      linux_options {
        host_name = "linux-${count.index + 1}"  # Set hostname
        domain    = "local"
      }
      network_interface {
        ipv4_address = "192.168.221.${20 + count.index}"  # Assign IPs from 192.168.221.20 to
192.168.221.30
        ipv4_netmask = 24
      }
      ipv4_gateway = "192.168.221.2"  # Default gateway
      dns_server_list = ["192.168.221.200"]  # DNS server
    }
  }

}
```





Summary of Terraform Code Actions

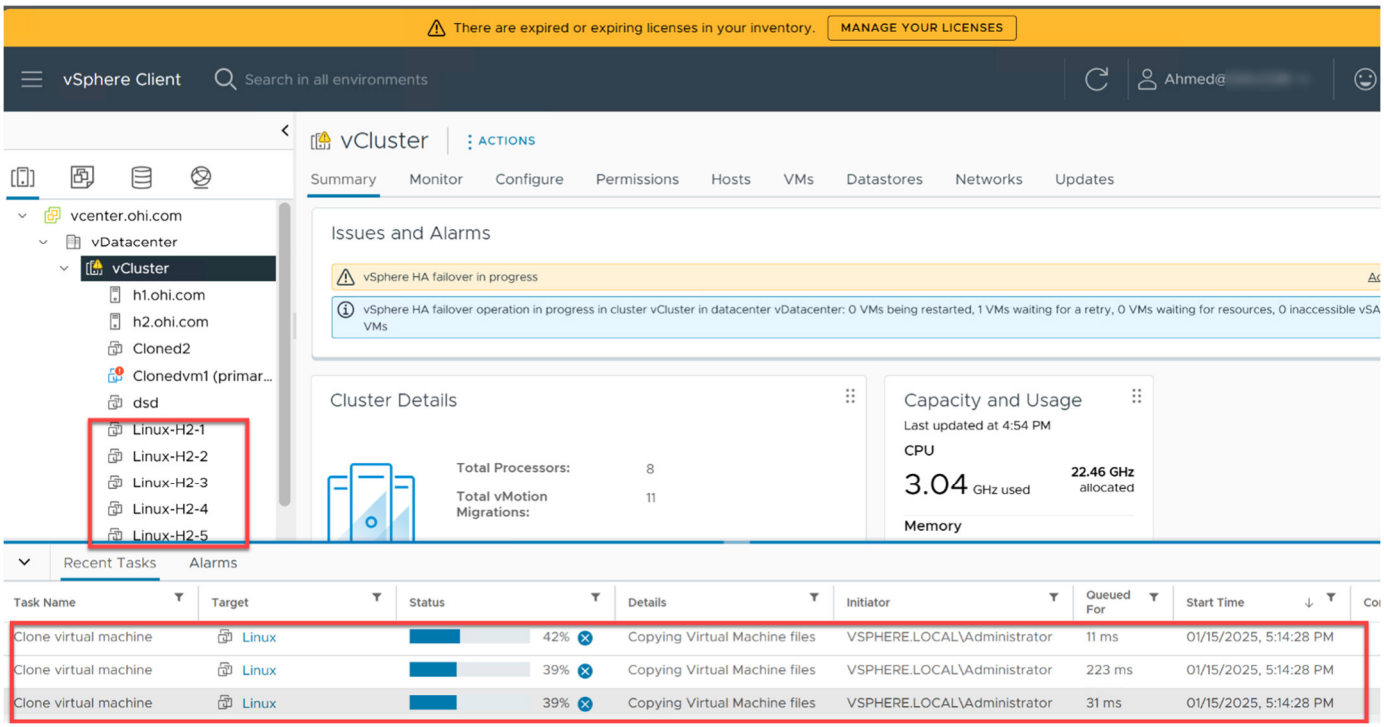
- **Configures the vSphere Provider**
  - Connects to the vCenter server at vcenter.abdelwahed.me using specified admin credentials.
  - SSL verification is disabled (allow_unverified_ssl = true).
- **Reads Existing vSphere Infrastructure Objects**
  - Loads datacenter named **vDatacenter**.
  - Loads compute cluster named **vCluster** in that datacenter.
  - Loads datastore named **NDatastore1** from the datacenter.
  - Loads network named **DPortGroup** from the datacenter.
  - Loads VM template named **Linux** from the datacenter.
- **Creates Multiple VMs in a Loop (Using count)**
  - **Creates 2 VMs** (based on count = 2; comment suggests 10, but code creates 2).
  - Each VM is named **Linux-1**, **Linux-2**, etc. (Linux-${count.index + 1}).
  - Each VM is placed in the resource pool of the vCluster and uses the NDatastore1 datastore.
  - Each VM gets:
    - **2 CPUs** and **1 GB memory**.
    - **EFI firmware**.
    - Network interface attached to DPortGroup using vmxnet3 adapter.
    - One thin-provisioned disk matching the template size.
- **Clones Each VM from the Linux Template**
  - Uses the specified template for each VM.
  - Customizes the guest OS for each VM:
    - Sets hostname to linux-1, linux-2, etc.
    - Sets domain to local.
    - Assigns a unique static IPv4 address for each VM:
      - First VM: 192.168.221.20
      - Second VM: 192.168.221.21
      - (Would continue for additional VMs if count were increased)
    - Sets network mask to /24.
    - Sets default gateway to 192.168.221.2.
    - Configures DNS server as 192.168.221.200.

Lab3: Deploy 5 Virtual Machines on Host h2.abdelwahed.me

main.tf

```
provider "vsphere" {
  vsphere_server = "vcenter.abdelwahed.me"
  user           = "administrator@vsphere.local"
  password       = "Mypassword12345"
  allow_unverified_ssl = true   # Disable SSL verification
}
data "vsphere_datacenter" "dc" {
  name = "vDatacenter"
}
data "vsphere_host" "host_h2" {
  name          = "h2.abdelwahed.me"  # Corrected host name
  datacenter_id = data.vsphere_datacenter.dc.id
}
data "vsphere_datastore" "ds" {
  name          = "NDatastore1"
  datacenter_id = data.vsphere_datacenter.dc.id
}
data "vsphere_network" "net" {
  name          = "DPortGroup"
  datacenter_id = data.vsphere_datacenter.dc.id
}
data "vsphere_virtual_machine" "template" {
  name          = "Linux"  # Template name from your environment
  datacenter_id = data.vsphere_datacenter.dc.id
}
resource "vsphere_virtual_machine" "vm" {
  count           = 5  # Create 5 VMs
  name            = "Linux-H2-${count.index + 1}"  # Name VMs as Linux-H2-1, Linux-H2-2,
etc.
  resource_pool_id = data.vsphere_host.host_h2.resource_pool_id  # Assign VMs to H2's
resource pool
  datastore_id    = data.vsphere_datastore.ds.id
  num_cpus = 2
  memory   = 1024  # Set memory to 1 GB
  guest_id = data.vsphere_virtual_machine.template.guest_id
  firmware = "efi"  # Set firmware to EFI
  network_interface {
    network_id   = data.vsphere_network.net.id
    adapter_type = "vmxnet3"
  }
```

```
  disk {
    label            = "disk0"
    size             = data.vsphere_virtual_machine.template.disks.0.size
    eagerly_scrub    = false
    thin_provisioned = true
  }
  clone {
    template_uuid = data.vsphere_virtual_machine.template.id
    customize {
      linux_options {
        host_name = "linux-h2-${count.index + 1}"  # Set hostname
        domain    = "local"
      }
      network_interface {
        ipv4_address = "192.168.221.${40 + count.index}"  # Assign IPs from
192.168.221.40 to 192.168.221.44
        ipv4_netmask = 24
      }
      ipv4_gateway = "192.168.221.2"  # Default gateway
      dns_server_list = ["192.168.221.200"]  # DNS server
    }
  }
}
```



Summary of Terraform Code Actions

- **Configures the vSphere Provider**
  - Connects to the vCenter server at vcenter.abdelwahed.me using admin credentials.
  - Disables SSL certificate verification.
- **Reads Existing vSphere Infrastructure Objects**
  - Loads the datacenter named **vDatacenter**.
  - Loads the ESXi host named **h2.abdelwahed.me** in that datacenter.
  - Loads the datastore named **NDatastore1**.
  - Loads the network named **DPortGroup**.
  - Loads the VM template named **Linux**.
- **Creates Multiple Virtual Machines (VMs) on a Specific Host**
  - **Creates 5 VMs** (using count = 5).
  - Names each VM as **Linux-H2-1**, **Linux-H2-2**, ..., **Linux-H2-5**.
  - Assigns each VM to the resource pool of the ESXi host h2.abdelwahed.me.
  - Stores each VM in the datastore NDatastore1.
  - **VM Configuration:**
    - 2 CPUs and 1 GB memory per VM.
    - EFI firmware.
    - One network interface using vmxnet3 adapter, connected to DPortGroup.
    - One thin-provisioned disk matching the size of the template.
- **Clones Each VM from the Linux Template**
  - Each VM is a clone of the **Linux** template.
  - Customizes guest OS for each VM:
    - Hostname as linux-h2-1, linux-h2-2, ..., linux-h2-5.
    - Domain set to local.
    - Assigns a unique static IPv4 address to each VM:
      - 192.168.221.40 through 192.168.221.44
    - Sets network mask to /24.
    - Sets gateway to 192.168.221.2.
    - Sets DNS server to 192.168.221.200.

Lab4: Deploy Multiple VMs With Different Sizes

main.tf

```
provider "vsphere" {
  vsphere_server       = "vcenter.example.com"
  user                 = "administrator@vsphere.local"
  password             = "MyPassword12345"
  allow_unverified_ssl = true
}
data "vsphere_datacenter" "dc" {
  name = "vDatacenter"
}
data "vsphere_compute_cluster" "cluster" {
  name          = "vCluster"
  datacenter_id = data.vsphere_datacenter.dc.id
}
data "vsphere_datastore" "ds" {
  name          = "NDatastore1"
  datacenter_id = data.vsphere_datacenter.dc.id
}
data "vsphere_network" "net" {
  name          = "DPortGroup"
  datacenter_id = data.vsphere_datacenter.dc.id
}
data "vsphere_virtual_machine" "template" {
  name          = "Linux"
  datacenter_id = data.vsphere_datacenter.dc.id
}
variable "vm_configs" {
  default = [
    { name = "web",  cpu = 2, mem = 2048, ip = "192.168.221.100" },
    { name = "db",   cpu = 4, mem = 4096, ip = "192.168.221.101" },
    { name = "test", cpu = 1, mem = 1024, ip = "192.168.221.102" },
  ]
}
resource "vsphere_virtual_machine" "custom_vms" {
  count            = length(var.vm_configs)
  name             = var.vm_configs[count.index]["name"]
  resource_pool_id = data.vsphere_compute_cluster.cluster.resource_pool_id
  datastore_id     = data.vsphere_datastore.ds.id
```

```
  num_cpus = var.vm_configs[count.index]["cpu"]
  memory   = var.vm_configs[count.index]["mem"]
  guest_id = data.vsphere_virtual_machine.template.guest_id
  firmware = "efi"
  network_interface {
    network_id   = data.vsphere_network.net.id
    adapter_type = "vmxnet3"
  }
  disk {
    label            = "disk0"
    size             = data.vsphere_virtual_machine.template.disks.0.size
    eagerly_scrub    = false
    thin_provisioned = true
  }
  clone {
    template_uuid = data.vsphere_virtual_machine.template.id
    customize {
      linux_options {
        host_name = var.vm_configs[count.index]["name"]
        domain    = "local"
      }
      network_interface {
        ipv4_address = var.vm_configs[count.index]["ip"]
        ipv4_netmask = 24
      }
      ipv4_gateway    = "192.168.221.2"
      dns_server_list = ["192.168.221.200"]
    }
  }
}
```

Summary of Terraform Code Actions

- **Configures the vSphere Provider**
  - Connects to the vCenter server at vcenter.example.com using administrative credentials.
  - Disables SSL verification for vCenter connection.
- **Reads Existing vSphere Infrastructure**
  - Loads a datacenter named vDatacenter.
  - Loads a compute cluster named vCluster within the specified datacenter.
  - Loads a datastore named NDatastore1 within the datacenter.
  - Loads a network named DPortGroup within the datacenter.
  - Loads a virtual machine template named Linux within the datacenter.
- **Defines a List of VM Configurations**
  - The vm_configs variable lists three VMs, each with:
    - A unique name (web, db, test)
    - Different CPU and memory values
    - A unique static IPv4 address
- **Creates Multiple Virtual Machines Using a Loop**
  - For each configuration in vm_configs, a new VM is created with settings from the variable.
  - VMs are placed in the specified compute cluster and datastore.
  - Each VM is assigned its specified CPU, memory, and static IP.
  - Each VM is cloned from the Linux template, with the same disk size as the template, thin-provisioned.
  - The network adapter uses vmxnet3 and connects to DPortGroup.
  - Each VM is set up with EFI firmware.
  - Linux guest OS customization is performed for hostname, domain, network, gateway, and DNS.