Hi, and welcome to the latest instalment of my ADVENTURES… IN… TERRAFORM!!!

This time, I have been experimenting with terraforming our on premises ESX infrastructure.

I wanted to see if I could create a new virtual machine from an existing template using the VMware vSphere Provider. Well, did you achieve it?... I hear you ask… you’ll have to keep reading to find out!

Warning!

This article contains gratuitous Harry Potter references and assumes both prior knowledge of ESX infrastructure and a basic knowledge of terraform.

The challenge.

To create a new Virtual Machine called ‘My-First-ESX-VM’ based on the VM template ‘Marauders Map’ using existing infrastructure.

The VM will reside in the ‘Gryffindor’ resource pool, which is located in the ‘Hogwarts School’ cluster of ESX servers. The VM will be stored in the ‘School Library’ data store and will be connected to the ‘dumbledore’s army’ network.

I have code snippets I will list below can all be found in my git repo <https://github.com/richfrankrepo/Linkedin/tree/master/Terraform/Part4>.

The code below shows the declaration of this various infrastructure as ‘data sources ‘.

{insert code snippet here}

The first item is the datacenter data source which is used to discover the ID property of your vSphere Datacenter and is referenced during the declaration of the datastore, cluster, network, resource pool and template objects.

I’m getting ahead of myself… meet the vSphere provider, which is the star of the show and makes all this magic possible.

provider "vsphere" {

    user           = var.vsphere\_user

    password       = var.vsphere\_password

    vsphere\_server = var.vsphere\_server

    # If you have a self-signed cert

    allow\_unverified\_ssl = true

}

variable "vsphere\_user" {

    type = string

    default = "harry.potter@hogwarts-witchcraf-wizardry.sch"

}

variable "vsphere\_password" {

    type = string

    default = "isolemnlyswearthatiamuptonogood"

}

variable "vsphere\_server" {

    type = string

    default "4 privet drive"

}

You’ll notice I’m using input variables to make this reusable and prevent me from having to hard code my login credentials to the server.

So, on to the main course…. how do we actually declare the new VM?

resource "vsphere\_virtual\_machine" "vm" {

    name              = "My-First-ESX-VM"

    resource\_pool\_id  = data.vsphere\_compute\_cluster.cluster.resource\_pool\_id

    datastore\_id      = data.vsphere\_datastore.datastore.id

    folder            = "Restricted Section"

    num\_cpus = 2

    memory   = 4096

    guest\_id = data.vsphere\_virtual\_machine.template.guest\_id

    scsi\_type = data.vsphere\_virtual\_machine.template.scsi\_type

    firmware = data.vsphere\_virtual\_machine.template.firmware

    network\_interface {

      network\_id   = data.vsphere\_network.network.id

      adapter\_type = data.vsphere\_virtual\_machine.template.network\_interface\_types[0]

    }

    disk {

      label            = "disk0"

      size             = data.vsphere\_virtual\_machine.template.disks.0.size

      eagerly\_scrub    = data.vsphere\_virtual\_machine.template.disks.0.eagerly\_scrub

      thin\_provisioned = data.vsphere\_virtual\_machine.template.disks.0.thin\_provisioned

    }

    clone {

      template\_uuid = data.vsphere\_virtual\_machine.template.id

      customize {

            network\_interface{}

            windows\_options {

              computer\_name  = "My-First-ESX-VM"

              workgroup      = "MyWorkGroup"

              admin\_password = "P455w0rd!"

            }

      }

    }

}

You will notice that I am using the ‘data.vsphere\_virtual\_machine.template’ object to set many of the VM’s properties, this is because I want it to match exactly, the configuration of the VM template.

Pay particular attention to the line…

firmware = data.vsphere\_virtual\_machine.template.firmware

I forgot to add this initially and although the build would complete, the server would not boot into the OS as default firmware setting was different to that specified in my template.

In the following line, I set the template\_uuid property to the id of the ‘Marauders Map’ template

template\_uuid = data.vsphere\_virtual\_machine.template.id

The ‘customize’ configuration block is where you add whatever custom settings you want. I’ve used it to set the NIC card to use DHCP and configure the computer name, workgroup and admin password properties.

      customize {

            network\_interface{}

            windows\_options {

              computer\_name  = "My-First-ESX-VM"

              workgroup      = "MyWorkGroup"

              admin\_password = "P455w0rd!"

            }

      }

It goes without saying that if you want to use this example to create your own VM, you’ll have to replace all my Harry Potter themed values with more appropriate ones… unless your environment has a very relaxed attitude to naming conventions and security 😊.

Thanks for your time and attention, I hope you found this illuminating.

If you want to dive deeper, check out this rather useful article on the Terraform website which, though sadly lacking in magic, goes into a great more detail.

<https://www.terraform.io/docs/providers/vsphere/r/virtual_machine.html#virtual-machine-customization>

All that’s left to do is click the secret button on my mouse and recite,

‘Mischief managed’