



PowerCLI

Automating VM Deployment on vSphere

With PowerCLI and Terraform



HashiCorp

Terraform

Who Am I?

Stephan McTighe

He | Him



I'm an infrastructure engineer focused on virtualisation and VMware solutions.

VMware vExpert.



<https://stephanmctighe.com>



me@stephanmctighe.com



www.linkedin.com/in/stephanmctighe



[@vStephanMcTighe](https://twitter.com/vStephanMcTighe)



<https://github.com/smctighevcv>





Automation; Why?

In this day and age, time is money. You don't want to be giving time to repeatable tasks, remediating inconsistencies or human error when its avoidable.

Consistency

Deploying from templates.

Speed

Quick deployments of VM's when you need them.

Repeatability

Deploying a VM the same way each time quickly and consistently.

Scalability

Standing up multiple VM's or an entire environment in one go.

Usability

One Line deployment.

Where to begin?

One of the best ways to ensure your virtual machines are deployed consistently is by starting with a template built to your organisations operational and security standards.

Your template could include:

- Security Baselines
- Standard Software Installations
- Locale settings

| Images | | |
|----------------------------------|--------------|--|
| ACTIONS | | |
| Summary Templates Other Types | | |
| VM Templates OVF & OVA Templates | | |
| Name | Content V... | Guest OS |
| cen-7 | 6 | CentOS 7 (64-bit) |
| win-2019-std-core | 5 | Microsoft Windows Server 2019 (64-bit) |
| win-2019-std-gui | 5 | Microsoft Windows Server 2019 (64-bit) |

Template Management



Packer by HashiCorp, is a great option for deploying templates in a repeatable fashion.

- Easy Installation
- Simple Command Lines
- Minimal Components
- Extensively Customizable

```
"builders": [{
  "type": "vSphere-iso",
  "scsi_controller": "[[user 'vSphere-server']]",
  "username": "[[user 'vSphere-user']]",
  "password": "[[user 'vSphere-password']]",
  "insecure_connection": "[[user 'insecure_connection']]",
  "datacenter": "[[user 'vSphere-datacenter']]",
  "cluster": "[[user 'vSphere-compute-cluster']]",
  "folder": "[[user 'vSphere-folder']]",
  "datastore": "[[user 'vSphere-datastore']]",
  "content_library_destination": {
    "library": "[[user 'content_library_destination']]",
    "name": "[[user 'template_library_name']]",
    "destroy": "[[user 'library_on_destroy']]",
    "ovf": "[[user 'ovf']]"
  },
  "vm_name": "[[user 'vm_name']]",
  "cpus": "[[user 'CPUs']]"
}]
```

JSON

```
source "vSphere-iso" "centos-8" {
  CPUs           = var.CPUs
  RAM             = var.RAM
  RAM_reserve_all = var.ram_reserve_all
  boot_command    = var.boot_command
  boot_order      = var.boot_order
  boot_wait       = var.boot_wait
  cluster         = var.vSphere_compute_cluster
  content_library_destination {
    destroy = var.library_on_destroy
    library = var.content_library_destination
    name    = var.template_library_name
    ovf     = var.ovf
  }
  datacenter = var.vSphere_datacenter
  datastore  = var.vSphere_datastore
  disk_controller_type = var.disk_controller_type
  firmware   = var.firmware
}

build {
  name = "Centos-8"
  sources = ["source.vSphere-iso.centos-8"]
}
```

HCL

PowerCLI



PowerCLI

- Readily available
- Common Language
- New Features

```
Class Clusters : System.Management.Automation.IValidateSetValuesGenerator {  
    [String[]] GetValidValues() {  
        $Cluster = (Get-Cluster).Name  
        return [String[]] $cluster  
    }  
}
```

```
process {  
    Write-Host "INFO: Building VM..." -ForegroundColor Green  
    New-VM -Name $VMName -OSCustomizationSpec $CustomSpec -Template $Template -resourcePool $Cluster -datastore $Datastore > $null  
    $newVM = Get-VM $VMName | Set-VM -MemoryGB $Memory -NumCPU $vCPU -CoresPerSocket $vCPU -Notes "$($_.Notes) Built $(Get-Date)" -Confirm:$false  
    $HostPG = $newVM | Get-VMHost | Get-VirtualPortGroup | Where {$_.Name -eq $Network} -WarningAction SilentlyContinue  
    $newVM | Get-NetworkAdapter | Set-NetworkAdapter -Portgroup $HostPG -Confirm:$false -WarningAction SilentlyContinue > $null  
}
```

Demo

Deploying Virtual Machines with PowerCLI



PowerCLI

Demo Summary

- Common Language – PowerShell!
- Wide Community Support
- Powerful
- Can become complex
- New Features



Terraform



- Infrastructure as Code
- Good Support Documentation
- HCL – Human and Machine Friendly

```
resource "vsphere_virtual_machine" "tfdemo3" {  
  name           = var.computer_name  
  resource_pool_id = data.vsphere_compute_cluster.cluster.resource_pool_id  
  datastore_id    = data.vsphere_datastore.datastore.id  
  
  num_cpus           = var.num_cpus  
  num_cores_per_socket = var.num_cores_per_socket  
  memory             = var.memory  
  folder             = var.folder  
}
```

Common Commands

terraform init
terraform plan
terraform apply
terraform destroy



Demo

Deploying Virtual Machines with Terraform



Demo Summary

- Reusable code
- Version Control
- Multi platform/cloud
- Potentially a new concept
- Can destroy as easily as create!



Thank You

Any Questions... ?



<https://stephanmctighe.com>



me@stephanmctighe.com



www.linkedin.com/in/stephanmctighe



[@vStephanMcTighe](https://twitter.com/vStephanMcTighe)



<https://github.com/smctighevc>