Create new infrastructure with Ansible and Terraform

Howto automate the process for creating and maintaining infrastructure

Jan Baer

May 4, 2022

What you can expect from this talk

- Howto create new infrastructure with Ansible and Terraform

What you can expect from this talk

- Howto create new infrastructure with Ansible and Terraform
- How to keep definitions of VMs independent from the cloud provider

What you can expect from this talk

- Howto create new infrastructure with Ansible and Terraform
- How to keep definitions of VMs independent from the cloud provider
- My learnings from working with Terraform

What you can expect from this talk

- Howto create new infrastructure with Ansible and Terraform
- How to keep definitions of VMs independent from the cloud provider
- My learnings from working with Terraform
- Tips&Tricks for using Terraform

About the project

- Migration from +40 VMs from OfficeIT to external cloud provider

About the project

- Migration from +40 VMs from OfficeIT to external cloud provider
- Create Firewall-rules with Terraform to have one source of truth

About the project

- Migration from +40 VMs from OfficeIT to external cloud provider
- Create Firewall-rules with Terraform to have one source of truth
- Creating an HA-Kubernetes-cluster with 3 control-planes and finally 8 worker-nodes

About Terraform

- Terraform codifies cloud APIs into declarative configuration files.
- Made by HashiCorp (Vagrant, Vault, Consul, Packer)
- Describe infrastructure on various providers with Terraform's configuration language (HCL)
- Use the Terraform CLI to manage configuration, plugins, infrastructure, and state

Why using two tools for the infrastructure

- Ansible can do things that also Terraform can do
- Terraform is more specialized for creating infrastructure
- Ansible is more specialized for provisioning and maintaining the software

Decisions

- Only automate things, I have to configure more than once (VMs, IP-Groups, Firewall-rules)
- Save state in Git (Do not use any cloud-storage)
- Use Ansible with local-exec
- Separate definitions from provider specific code.
- Use fixed IP addresses

How to configure Terraform

- Define variables with defaults and descriptions in **variables.tf**
- Mark secret variables as sensitive

```
variable "vcd_user" {
  type = string
  description = "vCloud user"
}
variable "vcd_pass" {
  type = string
  description = "vCloud pass"
  sensitive = true
}

variable "vcd_url" {
  type = string
}
variable "org_name" {
  type = string
}
```

How to configure Terraform

- Define values for variables in **terraform.tfvars**

```
# vcd_user = "" should be defined in the env variable TF_VAR_vcd_user
# vcd_pass = "" should be defined in the env variable TF_VAR_vcd_pass

org_name = "CVBU-0001"

vcd_url = "https://vcloud.uptime.de/api"

vdc_name = "CVBU-0001_DEHH02_vDC-02"
net_name = "CVBU-0001_DEHH02_OrgNetz-02"
edge_name = "CVBU-0001_DEHH02_EdgeGW-02"
vapp_name = "BU-DEV-01"

catalog_name = "Check24"
catalog_item = "template_ubuntu_20.04.3-server"

base_cidr_block = "10.2.201.0/24"
domain_name = "intern.bu.check24.de"
```

- Howto pass password to Terraform

```
export TF_VAR_vcd_pass=$(gopass show /check24/uptime/jan.ba
```

Configure the provider to talk with the cloud

```
terraform {
  required_providers {
    vcd = {
      source = "vmware/vcd"
      version = "≥3.5.0"
    }
  }
}

# Connect VMware vCloud Director Provider
provider "vcd" {
  user                 = var.vcd_user
  password             = var.vcd_pass
  org                  = var.org_name
  url                  = var.vcd_url
  max_retry_timeout    = 60
  allow_unverified_ssl = true
}
```

Docs

How to define a VPC

```
bu_dev_k8s_server = {
  environment = "development"
  vm_count = 3
  ip_start = 120
  cpus     = 4
  memory   = 6
  hd_size  = 75
  initscript = format("%s\n%s",
    file("${path.module}/scripts/extend_root_full.sh"),
    file("${path.module}/scripts/turn_swap_off.sh"),
  )
}
```

How to define a IP-Set

```
locals {
  ip_sets = {
    c24_muc_all = {
      ip_addresses = ["[redacted]/12"]
      description  = "ip-address ranges for all ser
    }
```

How to define a Firewall rule

```
# Allow all internal servers to query any DNS-server outside
resource "vcd_nsxv_firewall_rule" "out_int_any_dns" {
  org          = var.context.org
  vdc          = var.context.vdc
  edge_gateway = var.context.edge
  name         = "OUT_INT_ANY_DNS"
  source {
    gateway_interfaces = ["internal"]
  }
  destination {
    ip_addresses = ["any"]
  }
  service {
    protocol = "tcp"
    port     = "53"
  }
  service {
    protocol = "udp"
    port     = "53"
  }
}
```

How to run Terraform

- `terraform plan` shows you, what will be changed
- `terraform apply` will apply you changes and create all the things
- `terraform destroy` will destroy everything or a specific resource

How Terraform is using Ansible

- Calling *null_resource* whenever a new VPC will be created
- Passing environment and computer name from the VPC definition
- Run `provision.sh` in the working-dir with setting the VPC name as the limit

```
resource "null_resource" "ansible_provisioning" {
  depends_on = [vcd_vapp_vm.vapp_vm]

  count = var.vm_count

  triggers = {
    vpc_name = vcd_vapp_vm.vapp_vm[count.index].computer_name
  }

  provisioner "local-exec" {
    working_dir = local.ansible_working_dir
    command = "./provision.sh --limit=${self.triggers.vpc_name}"
  }
}
```

Showtime

Video

Keep in mind

- The following things can be changed while VM is running: Add CPU or Memory
- The following changed requires to turn off the VM: Reduce CPU or Memory
- The following changes will destroy and recreate the VM: changing the size of the HD
- Terraform will show you, when changes can be applied in-place or when the resource will be destroyed
- *null_resource* will be triggered only once when the resource was not existing before

Tips&Tricks

- Use additional internal_disk in case you want to add more diskspace afterwards without recreating the VM
- Don't be afraid of recreating a VM
- Use an *ext_disk* in case you don't want to loose the data when VM will be destroyed
- Use *dynamic* for optional options in a resource
- Address specific resource with **–target** to run Terraforming faster
- You can import the state of existing resources
- Don't make any manual changes for the Terraformed resources

```
terraform apply --target="module.vpcs.module.vpc[\"bu_int_r
```

Questions???

Thank you...