Terraform

As referências aos arquivos dessa sessão podem ser acessadas em
https://git.dds.ufvjm.edu.br/dicom/infra

Instalação (Linux)

```
# Baixar Terraform (versão 1.9+)
wget https://releases.hashicorp.com/terraform/1.9.0/terraform_1.9.0_linux_amd64.zip

# Descompactar
unzip terraform_1.9.0_linux_amd64.zip

# Mover para PATH
sudo mv terraform /usr/local/bin/

# Verificar instalação
terraform version
```

Saída esperada:

```
Terraform v1.9.0
```

Conceitos Básicos

- **Providers**: Conectam Terraform a plataformas (vSphere, AWS, etc)
- **Resources**: Recursos a serem criados (VMs, redes, discos)
- **Variables**: Variáveis/parâmetros configuráveis
- **Data Sources**: Consultar recursos existentes
- **State**: Arquivo que rastreia recursos criados (`terraform.tfstate`)

Estrutura de Um Projeto

```
projeto/
├── provider.tf      # Configuração do provider
```

```
├── variables.tf     # Declaração de variáveis
├── terraform.tfvars # Valores das variáveis (credenciais)
├── data.tf          # Data sources
└── main.tf          # Recursos principais
```

Provider vSphere

Arquivo `provider.tf`:

```
provider "vsphere" {
  user               = var.vsphere_user
  password           = var.vsphere_password
  vsphere_server     = var.vsphere_server
  allow_unverified_ssl = true
}
```

Referência: `base/provider.tf`

Variables

Arquivo `variables.tf`:

```
variable "vsphere_user" {
  default = "administrator@vsphere.local"
}

variable "vsphere_password" {}  # Sem default (variável obrigatória)

variable "vsphere_server" {
  default = "10.50.3.200"
}
```

Arquivo `terraform.tfvars` (não versionado):

```
vsphere_user     = "seu-usuario"
vsphere_password = "sua-senha"
```

Uso em código:

```
user = var.vsphere_user
```

Referência: `base/variables.tf`

Data Sources

Consultam recursos existentes no vSphere:

```
data "vsphere_datacenter" "dc" {
  name = "UFVJM"
}

data "vsphere_datastore" "datastore" {
  name          = "Dicom02"
  datacenter_id = data.vsphere_datacenter.dc.id
}

data "vsphere_virtual_machine" "template" {
  name          = "Template-Portal-Rancher"
  datacenter_id = data.vsphere_datacenter.dc.id
}
```

Uso:

```
datastore_id = data.vsphere_datastore.datastore.id
```

Referência: `base/data.tf`

Resource: Criar VM

```
resource "vsphere_virtual_machine" "vm" {
  name             = "minha-vm"
  resource_pool_id = data.vsphere_resource_pool.pool.id
  datastore_id     = data.vsphere_datastore.datastore.id

  num_cpus = 2
  memory   = 4096  # MB

  network_interface {
```

```
    network_id = var.vsphere_port_group_dicom_local
  }

  disk {
    label = "disk0"
    size  = 20  # GB
  }

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

Count: Múltiplas VMs

```
resource "vsphere_virtual_machine" "nodes" {
  count = 3  # Criar 3 VMs

  name = "node-${count.index + 1}"  # node-1, node-2, node-3
  # ... resto da configuração
}
```

Referência: `rancher-cluster/main.tf` (3 VMs)

Dynamic Blocks

Configuração condicional baseada em `count.index`:

```
resource "vsphere_virtual_machine" "nodes" {
  count = 6
  name  = "node-${count.index + 1}"

  # CPU/RAM diferentes para workers (nodes 4-6)
  num_cpus = count.index > 2 ? 4 : 2
  memory   = count.index > 2 ? 16384 : 8192

  # Rede adicional apenas para workers
  dynamic "network_interface" {
    for_each = count.index > 2 ? [1] : []
```

```
    content {
      network_id = var.vsphere_port_group_dicom_global
    }
  }

  # Disco adicional apenas para workers
  dynamic "disk" {
    for_each = count.index > 2 ? [1] : []
    content {
      label = "disk2"
      size  = 150
    }
  }
}
```

Explicação:

- `count.index > 2 ? valor_se_true : valor_se_false` (operador ternário)
- `for_each = [1]` → executa o bloco 1 vez
- `for_each = []` → não executa (lista vazia)

Referência: `prod-cluster/main.tf`

Funções

```
# Ler arquivo
file("metadata.yml")

# Codificar em base64
base64encode(file("metadata.yml"))

# Uso em VMs (cloud-init)
extra_config = {
  "guestinfo.metadata"          = base64encode(file("metadata-node1.yml"))
  "guestinfo.metadata.encoding" = "base64"
}
```

Cloud-Init Metadata

Arquivo `metadata-node1.yml`:

```
instance-id: rancher-node1
hostname: rancher-node1
network:
  version: 2
  ethernets:
    eth0:
      addresses:
        - 10.254.32.7/24
      gateway4: 10.254.32.1
      nameservers:
        addresses: [8.8.8.8, 8.8.4.4]
      dhcp4: false
```

Referência: `rancher-cluster/metadata-node1.yml`

Estrutura Compartilhada (Symlinks)

Problema: Evitar duplicação de `provider.tf`, `variables.tf` em cada módulo.

Solução: Diretório `/base/` com configurações compartilhadas.

```
# Script setup.sh cria symlinks automaticamente
cd rancher-cluster/
ln -s ../base/provider.tf provider.tf
ln -s ../base/variables.tf variables.tf
ln -s ../base/data.tf data.tf
```

Vantagem: Atualizar uma vez, todos os módulos usam.

Workflow Terraform

```
# 1. Inicializar (baixar providers)
terraform init

# 2. Planejar (preview das mudanças)
terraform plan

# 3. Aplicar (criar recursos)
terraform apply
```

```
# 4. Destruir (remover tudo)
terraform destroy

# 5. Ver estado atual
terraform show
```

Importante: `terraform.tfstate` rastreia recursos criados. **Não deletar!**

Revision #2
Created 6 February 2026 04:21:51 by Estevao Samuel Procopio Amaral
Updated 6 February 2026 04:29:20 by Estevao Samuel Procopio Amaral