Microsoft

XaC Everything as Code – Tudo como Código

Quem sou eu?

George Luiz Bittencourt

- CSA na Microsoft focado em Apps & Infra.
- Mais de 20 anos de experiência com desenvolvimento e infraestrutura.
- LinkedIn: https://www.linkedin.com/in/glzbcrt/
- E-mail: george.bittencourt@microsoft.com

Agenda

- Quando não temos XaC
- O que é XaC?
- Vantagens
- Terraform
- Terraform Cloud
- Packer
- Ansible
- Helm
- Desafios

Quando não temos XaC...

- A pessoa sai da empresa e não tem documentação do ambiente e todo mundo tem medo de mexer, por que não sabe o que pode acontecer.
- Criar um ambiente novo toma muito tempo, pois não temos as dependências mapeadas e dependemos de pessoas para a criação.
- Com o passar do tempo perdemos o controle do ambiente, porque não temos um rastreio do que foi alterado tornando impossível auditorias.
- Não temos um processo formal de aprovação de alterações.
- Várias outras situações!

Pet vs Cattle

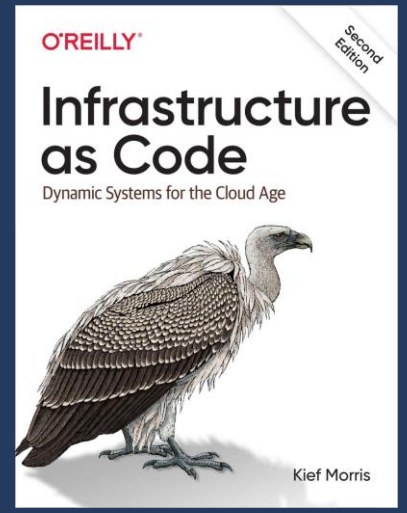
Criar componentes descartáveis!

O que é XaC?

Infrastructure as Code is an approach to infrastructure automation based on practices from software development.

It emphasizes **consistent**, **repeatable** routines for provisioning and changing systems and their configuration.

You make changes to code, then use automation to test and apply those changes to your systems.

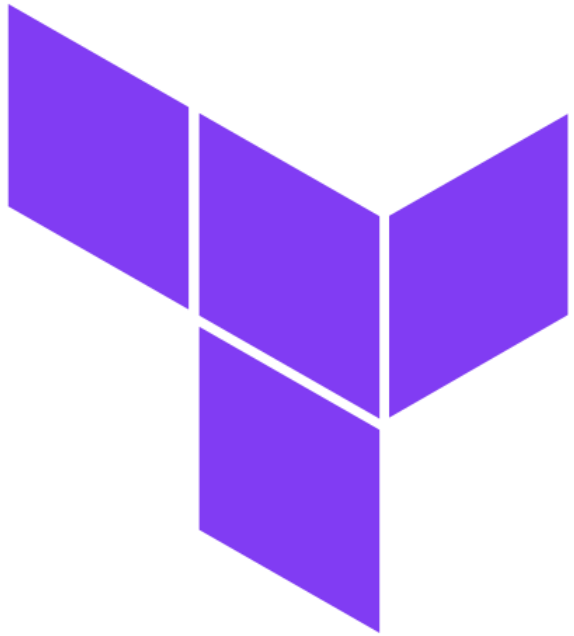


Vantagens

- O ambiente fica documentado.
- Alterações seguem um processo e somente com aprovações as alterações são aplicadas.
- É possível reverter alterações com maior facilidade, já que temos o histórico.
- Podemos recriar o ambiente ou partes dele com maior facilidade e agilidade.

Terraform

- Ferramenta criada em 2014 pela empresa Hashicorp.
- Desenvolvida de maneira open-source na linguagem Go e hospedada em hashicorp/terraform.
- Extensível através do uso de providers.
- Com o uso de módulos é possível criar códigos reaprovetáveis.
- A Microsoft fornece documentações especificas em Terraform on Azure.
- O ambiente é expresso usando a linguagem Hashicorp Configuration Language (HCL).



Terraform: HCL

- É uma linguagem declarativa.
- É possível criar várias instâncias de um mesmo recurso utilizando operadores como *count* e *for_each*.
- Através do uso de arquivos de variáveis é possível externalizar os valores.
- Existe um conjunto grande de funções para ler arquivos, efetuar chamadas HTTP, etc.
- É possível executar um provisionador quando o recurso é criado permitindo combinar ferramentas.
- Algumas situações não são possíveis de expressar em HCL.
- Existem várias extensões para o Visual Studio Code para facilitar o desenvolvimento.
- A ferramenta Terragrunt adiciona algumas ferramentas extras aos fluxos de trabalho do Terraform.

Terraform: Módulos

- Da mesma forma que uma função em programação módulos permitem reaproveitar códigos.
- Com módulos garantimos consistências, boas práticas e encapsulamento.
- É possível através da passagem de parâmetros alterar o comportamento do módulo.
- Módulos podem ser aninhados.
- No Terrafom Registry existe uma grande quantidade de módulos já disponíveis.

Terraform: DAG

- O Terraform cria um *directed acyclic graph* (DAG) quando executado.
- Os dados desse *graph* são utilizados para sequenciar a criação dos recursos.
- A ordem é definida com base nas dependências ímplicitas e explícitas entre os recursos.

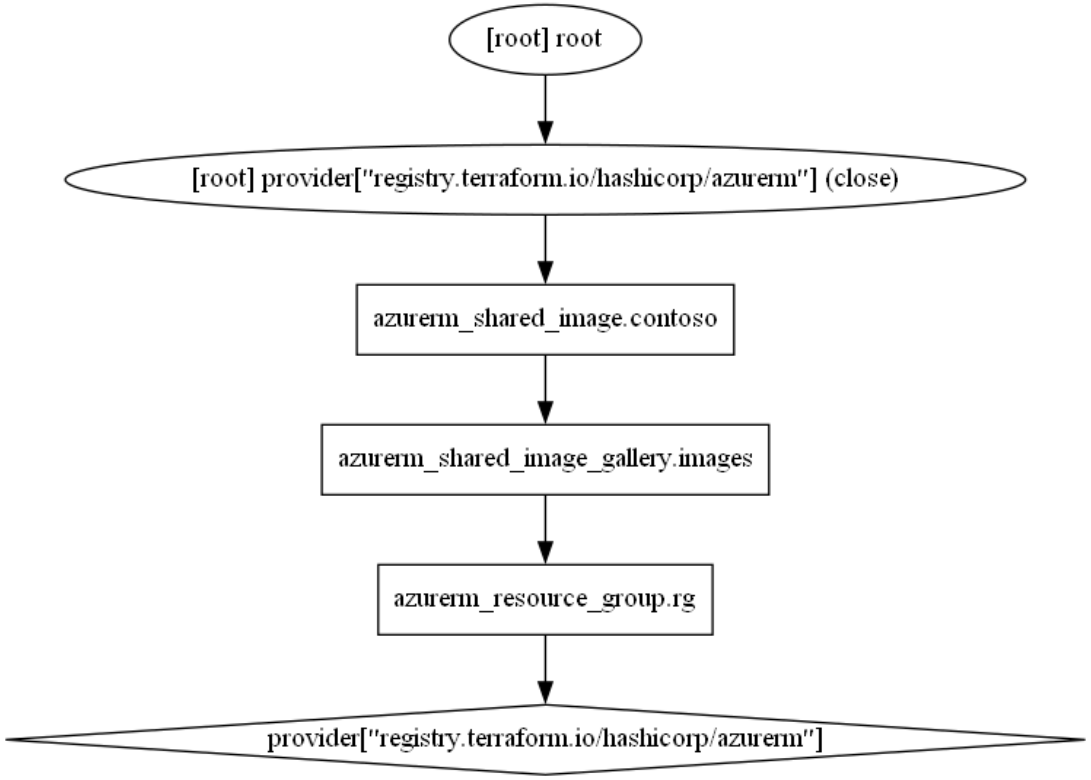
```
resource "azurerm_resource_group" "rg" {
  name     = "iac"
  location = "East US"
}

resource "azurerm_shared_image_gallery" "images" {
  name                = "images"
  resource_group_name = azurerm_resource_group.rg.name
  location            = azurerm_resource_group.rg.location

  depends_on = [
    azurerm_resource_group.rg
  ]
}
```

IMPLÍCITO

EXPLÍCITO



Terraform: Providers

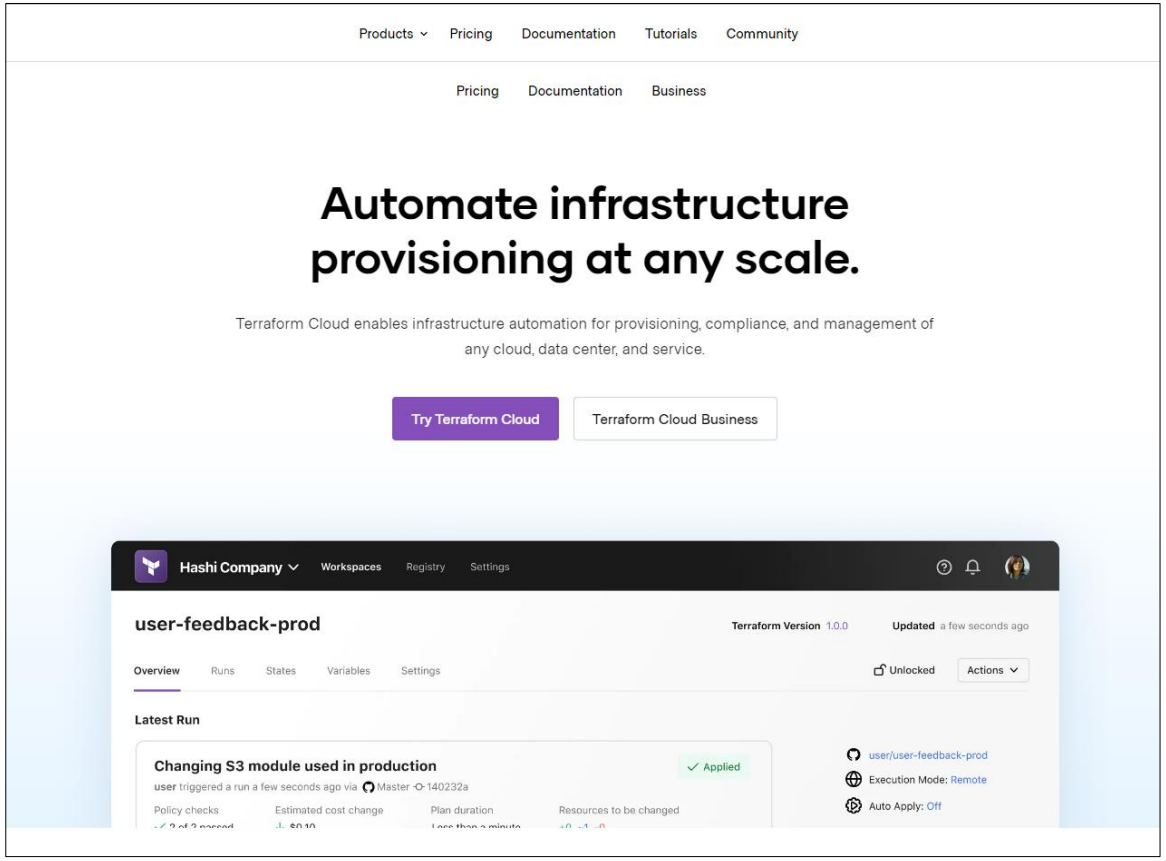
- Desacopla o Terraform dos recursos gerenciados em si.
- É possível criar providers na linguagem Go.
- Exitem mais de 2.600 providers no registro público do Terraform em providers.
- Eles são instalados sob demanda e são versionados utilizando *semantic versioning*.
- Podemos utilizar vários *providers* em um

Terraform: State

- É uma forma de banco de dados estruturado em formato JSON.
- Armazena um grande conjunto de informações sobre cada recurso criado.
- O Terraform usa ele para controlar, por exemplo, quais recursos foram criados por ele impedindo com isso de excluir recursos de maneira incorreta.
- Quanto mais recursos são criados, maior ele fica o que pode se tornar um problema.
- Somente um usuário pode modificar o *state* por execução.
- Através do comando *terraform state* é possível controlar alguns aspectos dele.

Terraform Cloud

- É uma oferta da Hashicorp de Terraform na nuvem.
- Até 5 usuários são gratuitos.
- Armazena e gerencia o acesso ao *state*.
- Integra com os principais VCS do mercado.
- Com integrações é possível criar políticas e também validar o impacto das alterações, como o aumento de custo.
- As credenciais são persistidas de forma centralizada e controladas.



LABORATÓRIO

Packer

- Ferramenta criada pela empresa Hashicorp.
- Desenvolvida de maneira open-source na linguagem Go e hospedada em hashicorp/packer.
- Permite a criação de imagens bases.
- Essas imagens podem ser utilizadas na criação de máquinas virtuais.
- É extensível através de *builders* que abstraem os detalhes da nuvem onde a máquina virtual será criada.
- Através do uso de *provisioners* conseguimos executar scripts através de SSH ou WinRM.
- Com o uso de Azure Policy podemos restringir que imagens podem ser utilizadas.

HashiCorp
Packer

LABORATÓRIO

Ansible

- Ferramenta criada em 2012 por Michael DeHaan. Atualmente pertence a empresa Red Hat.
- Desenvolvida de maneira open-source na linguagem Python e hospedada em ansible/ansible.
- É uma ferramente *agentless*.
- Utiliza conexões SSH ou WinRM para a execução dos scripts.
- Usa uma linguagem declarativa.

Ansible: Componentes

INVENTORY

- Estático
- Dinâmico
- Roles

PLAYBOOKS

- Conjunto de tasks
- Loops
- Condicionais
- Variáveis

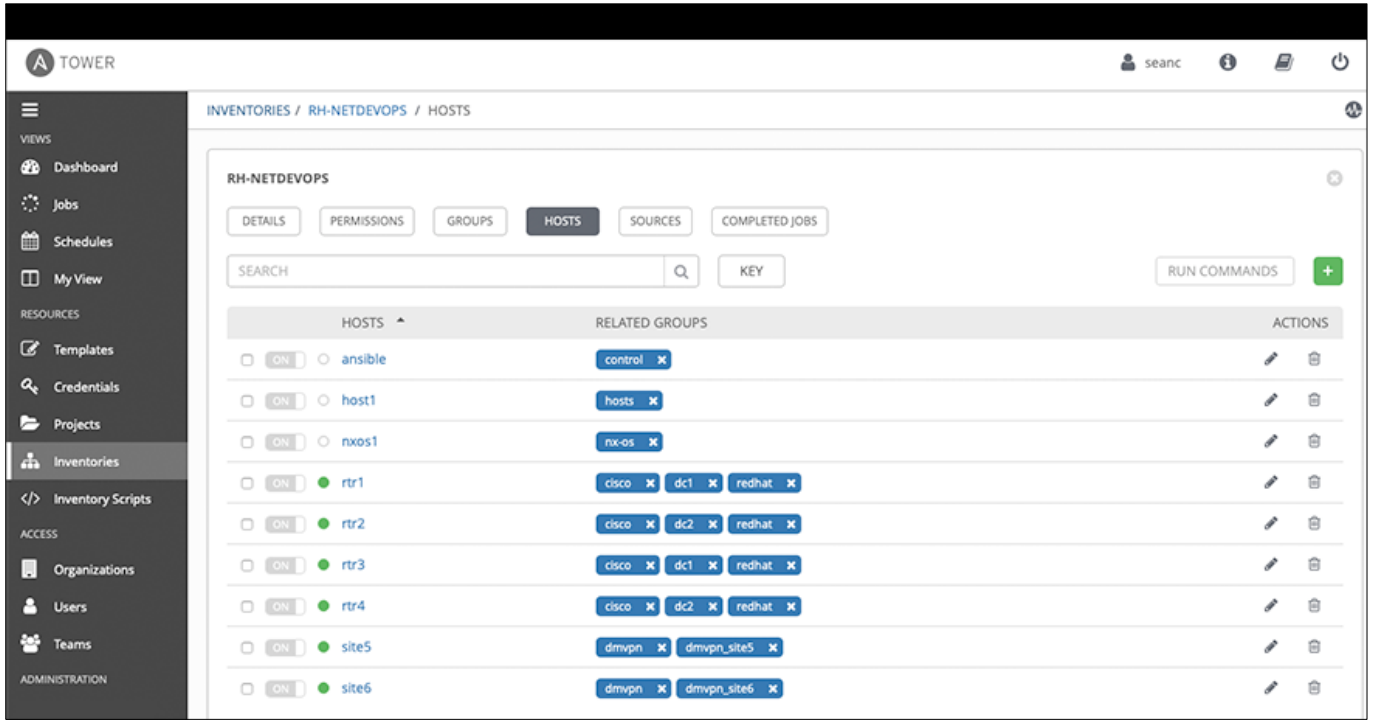
MODULES

- Padrões
- Comunidade
- Extensível

LABORATÓRIO

Ansible: Tower

- Fornece uma interface gráfica para o gerenciamento e execução dos playbooks.
- Administra o inventário de forma centralizada.
- Controle de acesso para usuários e grupos.
- Reaproveitamente de código.



Helm

- Gerenciador de pacotes para o Kubernetes. Projeto graduado no CNCF.
- É open-source e desenvolvido em Go em helm/helm.
- Através de um provider para o Terraform é possível integrar ele ao Kubernetes.
- As aplicações são instaladas através de *charts* e esses por sua vez são armazenados em *repositories*.
- *Charts* nada mais são que templates textuais processados em tempo de instalação efetuando substituições antes de serem enviadas para o Kubernetes.
- Existe a versão da aplicação e do *chart*.
- Quando um *chart* é instalado em um cluster é criada uma *release*. Essa pode ser atualizada e removida.
- O Artifact Hub, que também é um projeto CNCF, indexa milhares de *charts* disponíveis.

LABORATÓRIO

Outras Ferramentas

- Powershell Desired State Configuration (DSC)
- Pulumi
- Puppet
- Chef
- Bicep
- Vagrant
- Atlantis

Desafios

- Garantir a sincronizacão entre o código e o ambiente, conhecido como *configuration drift*.
  - Controlar o acesso dos usuários impedindo alterações não autorizadas.
  - Executar de maneira contínua as ferramentas para detectar as diferenças.
  - Adotar a política de sempre iniciar pelo código, mesmo quando efetuamos *troubleshoots*.
- Escolher a ferramenta certa.
  - Existem várias ferramentas no mercado e escolher a certa nem sempre é fácil.
  - Algumas funcionalidades precisam ser utilizadas com cuidado para não complicarem o ambiente. Exemplo: Terraform provisioners.
- Orquestrar as várias ferramentas.
  - Raramente uma única ferramenta é suficiente para criar o ambiente.
  - O gatilho das ferramentas é manual ou automático?
- Refatorar ambientes.
  - Criar padrões de nomes.
  - Lembre-se: é um código e precisa ser tratado como tal. Aplique princípios de engenharia de software.

Microsoft

Obrigado!