Aula 8 – Variáveis de Entrada e Saída (Input/Output)

Bem-vindo à nossa jornada pelo universo da Infraestrutura como Código (IaC), um campo que está revolucionando a forma como construímos e gerenciamos sistemas. Nesta aula, vamos mergulhar em um dos pilares da IaC: as **Variáveis de Entrada e Saída (Input/Output)**. Se você já se viu repetindo blocos de código ou configurando manualmente pequenos detalhes em diferentes ambientes, sabe o quão tedioso e propenso a erros isso pode ser. As variáveis são a chave para superar esses desafios, transformando seu código em algo flexível, reutilizável e muito mais poderoso.

Imagine que você está construindo uma casa. Cada casa tem características em comum – paredes, telhado, portas – mas cada cliente pode querer um número diferente de quartos, uma cor específica para a fachada ou um tipo particular de piso. Se você tivesse que redesenhar a planta inteira para cada pequena mudança, o processo seria exaustivo. No mundo da IaC, as variáveis funcionam como esses "parâmetros" ajustáveis da planta, permitindo que você defina a estrutura básica uma única vez e, em seguida, personalize-a com facilidade para diferentes necessidades ou ambientes.

Ao final desta aula, você será capaz de compreender a importância das variáveis para a parametrização e reutilização do código de infraestrutura. Exploraremos os diversos tipos de variáveis disponíveis, aprenderá a gerenciar suas configurações através de arquivos dedicados e pela linha de comando, e descobrirá como o bloco output é fundamental para expor informações cruciais da sua infraestrutura. Prepare-se para elevar o nível da sua automação e tornar seu trabalho com IaC mais eficiente e seguro.

A Necessidade de Parametrização: Adeus ao Código Rígido

No início da automação de infraestrutura, era comum ver scripts que continham valores fixos, ou "hardcoded", para configurações como nomes de servidores, tamanhos de disco ou regiões de nuvem. Embora funcionasse para um ambiente específico, essa abordagem rapidamente se tornava um pesadelo. Qualquer pequena alteração ou a necessidade de replicar a infraestrutura em um novo ambiente exigia a modificação manual de cada linha de código afetada, introduzindo riscos de erros e consumindo um tempo valioso.

> **Analogia do Chef:** Pense em um chef que precisa cozinhar o mesmo prato para diferentes clientes. Se ele tiver uma receita que especifica "200g de sal", mas um cliente prefere menos sal e outro mais, ele teria que reescrever a receita inteira para cada um. Seria muito mais prático se a receita dissesse "quantidade de sal: [variável]", e ele pudesse ajustar essa variável para cada pedido.

No contexto da IaC, a parametrização com variáveis é exatamente isso: uma forma de tornar suas "receitas" de infraestrutura adaptáveis, sem a necessidade de reescrever o código base.

Código Rígido	Código Parametrizado	Resultado
Valores fixos que exigem edição manual para cada mudança	Variáveis ajustáveis que se adaptam a diferentes contextos	Agilidade, consistência e redução de erros

Essa flexibilidade é crucial em cenários modernos, onde a infraestrutura precisa ser ágil e responder rapidamente às demandas de negócio. Seja para provisionar ambientes de desenvolvimento, teste ou produção, ou para escalar recursos sob demanda, a capacidade de ajustar configurações com variáveis é o que diferencia um código IaC robusto de um script engessado. Isso nos leva diretamente ao conceito de reutilização, onde um único conjunto de código pode ser aplicado em múltiplos contextos, economizando tempo e garantindo consistência.

Variáveis em Ação: Tornando seu Código Reutilizável

A verdadeira magia das variáveis reside em sua capacidade de transformar um código específico em um modelo genérico. Ao invés de definir um nome de bucket S3 como `meu-bucket-producao-regiao-a`, você pode usar uma variável como `nome_do_bucket` e atribuir a ela o valor desejado no momento da execução. Isso significa que o mesmo código pode ser usado para criar `meu-bucket-desenvolvimento-regiao-b` ou `meu-bucket-homologacao-regiao-c` sem uma única alteração no arquivo principal da infraestrutura.

Benefícios da Reutilização

- **Economia de tempo:** Escreva uma vez, use em múltiplos contextos
- **Consistência:** Reduz desvios e erros manuais
- **Padronização:** Pilar fundamental para GitOps
- **Manutenibilidade:** Facilita atualizações e correções

Aplicação Prática

Essa abordagem não apenas economiza tempo, mas também fortalece a consistência. Quando você tem um único "modelo" de infraestrutura, as chances de introduzir desvios ou erros manuais são drasticamente reduzidas.

É como ter um carimbo que você pode personalizar com diferentes textos antes de usar; o carimbo base é sempre o mesmo, mas o resultado final se adapta à sua necessidade.

Reutilização em Módulos

A reutilização se estende a módulos e componentes. Você pode criar um módulo de rede que aceita variáveis para o CIDR da VPC, sub-redes e grupos de segurança. Esse módulo pode então ser "invocado" em diferentes projetos, cada um com suas próprias configurações de rede, mas todos se beneficiando da mesma lógica de provisionamento testada e aprovada. Isso acelera o desenvolvimento, melhora a qualidade e facilita a manutenção da infraestrutura ao longo do tempo.

Desvendando os Tipos de Variáveis: A Linguagem da Flexibilidade

Assim como em qualquer linguagem de programação, as variáveis em IaC possuem tipos que definem o tipo de dado que elas podem armazenar. Compreender esses tipos é fundamental para escrever um código robusto e evitar erros de configuração. Cada tipo tem seu propósito e sua forma de ser utilizado, garantindo que a informação seja interpretada corretamente pela ferramenta de IaC.

Vamos começar com os tipos mais básicos, que são a base para a construção de configurações mais complexas. Eles são como os ingredientes primários em uma receita: farinha, água, sal. Você pode combiná-los de diversas formas, mas precisa saber o que cada um é e como se comporta.

Variáveis Simples: String, Number e Bool

String (Cadeia de Caracteres)

Este é o tipo mais comum e versátil. Uma string é uma sequência de caracteres, usada para armazenar texto. Pense em nomes de recursos, descrições, URLs ou qualquer valor que seja essencialmente textual.

- **Exemplo:** `nome_servidor = "web-app-producao"`, `regiao = "us-east-1"`
- **Aplicação:** Definir nomes de instâncias, tags, caminhos de arquivos, mensagens de log

Number (Número)

Usado para armazenar valores numéricos, sejam eles inteiros ou decimais. É ideal para configurações que envolvem contagens, tamanhos, portas ou capacidades.

- **Exemplo:** `numero_instancias = 3`, `tamanho_disco_gb = 50`
- **Aplicação:** Especificar a quantidade de recursos, portas de rede, limites de memória

Bool (Booleano)

Representa um valor lógico que pode ser `true` (verdadeiro) ou `false` (falso). É perfeito para ativar ou desativar funcionalidades, definir permissões ou controlar fluxos condicionais.

- **Exemplo:** `habilitar_monitoramento = true`, `publico = false`
- **Aplicação:** Controlar se um recurso é público ou privado, ativar ou desativar um recurso específico, definir flags de segurança

Esses tipos básicos formam a espinha dorsal da maioria das configurações, permitindo que você defina parâmetros claros e diretos para sua infraestrutura.

Tipos de Variáveis Avançados: Listas, Mapas e Objetos

À medida que a complexidade da sua infraestrutura cresce, a necessidade de agrupar e estruturar informações de forma mais sofisticada também aumenta. É aqui que entram os tipos de variáveis mais avançados: `list`, `map` e `object`. Eles permitem que você organize dados relacionados, tornando seu código mais legível e gerenciável, especialmente quando você precisa lidar com coleções de itens ou configurações hierárquicas.

> **Analogia:** Imagine que você não precisa apenas de um ingrediente (string, number, bool), mas de um conjunto de ingredientes para uma receita, ou de uma lista de substituições para um item específico. Esses tipos avançados oferecem essa capacidade de agrupamento e estruturação.

Variáveis Estruturadas: List, Map e Object

List (Lista/Array)

Uma list é uma coleção ordenada de valores do mesmo tipo. É útil quando você precisa definir múltiplos itens semelhantes, como uma lista de nomes de sub-redes, IPs permitidos ou usuários.

Exemplo: `subredes = ["10.0.1.0/24", "10.0.2.0/24"]`, `portas_abertas = [80, 443, 22]`

Aplicação: Definir múltiplos valores para um único atributo (ex: várias zonas de disponibilidade, IPs de origem permitidos em um firewall)

Map (Mapa/Dicionário)

Um map é uma coleção de pares chave-valor, onde cada chave é uma string única e mapeia para um valor. É ideal para configurações que exigem rótulos ou identificadores para seus valores, como tags de recursos ou configurações específicas de ambiente.

Exemplo: `tags_comuns = { "Ambiente" = "Producao", "Projeto" = "WebApp" }`, `config_db = { "usuario" = "admin", "porta" = 5432 }`

Aplicação: Definir metadados para recursos (tags), configurações de banco de dados, parâmetros de aplicação

Object (Objeto)

Um object é semelhante a um map, mas com uma estrutura mais rígida e tipada. Ele define um conjunto fixo de atributos, cada um com um nome e um tipo específico. É excelente para criar estruturas de dados complexas e bem definidas.

Aplicação: Modelar configurações complexas com múltiplos atributos de diferentes tipos, como configurações de rede para um grupo de instâncias ou detalhes de um serviço específico

Exemplo de Object

```
variable "usuario_app" {
  type = object({
    nome  = string
    email = string
    admin = bool
  })
  default = {
    nome  = "joao.silva"
    email = "joao.silva@empresa.com"
    admin = false
  }
}
```

A escolha do tipo de variável correto não é apenas uma questão de sintaxe, mas de design. Usar o tipo adequado melhora a clareza do seu código, facilita a validação e previne erros, tornando sua infraestrutura mais robusta e fácil de manter.

Gerenciando Variáveis: Arquivos .tfvars para Organização

Com a crescente complexidade da infraestrutura e a necessidade de gerenciar diferentes ambientes (desenvolvimento, homologação, produção), a simples passagem de variáveis pela linha de comando pode se tornar impraticável. É aqui que os arquivos de variáveis, com a extensão `.tfvars`, entram em cena como uma solução elegante e organizada. Eles permitem que você defina todos os valores das suas variáveis em um ou mais arquivos, separando as configurações específicas do ambiente do código principal da sua infraestrutura.

Por que usar .tfvars?

- Separação clara entre código e configuração
- Gerenciamento simplificado de múltiplos ambientes
- Versionamento de configurações no Git
- Rastreabilidade e auditoria completa
- Alinhamento com práticas de GitOps

> **Analogia do Livro de Receitas:** Pense em um livro de receitas onde a receita base está em um capítulo, mas as variações regionais ou sazonais dos ingredientes estão em apêndices separados. Você não precisa reescrever a receita principal; apenas consulta o apêndice relevante.

Como Usar Arquivos .tfvars

01

Crie o arquivo

Basta criar um arquivo com a extensão `.tfvars` (ex: `producao.tfvars`) no mesmo diretório do seu código IaC ou em um subdiretório.

02

Defina as variáveis

Dentro do arquivo, atribua valores às suas variáveis declaradas, usando a sintaxe `nome_da_variavel = valor`.

03

Aplique o arquivo

Ao executar seu comando de IaC, você pode especificar o arquivo .tfvars a ser usado: `terraform apply -var-file="producao.tfvars"`

Exemplo de Arquivo .tfvars

```
# producao.tfvars
regiao = "sa-east-1"
numero_instancias = 5
tags_comuns = {
  Ambiente = "Producao"
  Projeto  = "E-commerce"
}
```

Você pode usar múltiplos arquivos `.tfvars`, e eles serão processados em ordem, com os valores dos arquivos posteriores sobrescrevendo os anteriores em caso de conflito. Isso oferece uma flexibilidade enorme para gerenciar configurações complexas e hierárquicas.

Passagem de Variáveis pela Linha de Comando: Ajustes Rápidos e Prioridade

Embora os arquivos `.tfvars` sejam excelentes para gerenciar conjuntos de variáveis para ambientes inteiros, há momentos em que você precisa fazer um ajuste rápido ou sobrescrever um valor específico sem modificar um arquivo. É aí que a passagem de variáveis pela linha de comando se torna extremamente útil. Essa abordagem oferece uma maneira direta e imediata de fornecer valores para suas variáveis, sendo ideal para testes rápidos, automações pontuais ou para fornecer informações sensíveis que não devem ser armazenadas em arquivos.

Quando Usar

- Testes rápidos de configuração
- Automações pontuais
- Sobrescrever valores específicos
- Fornecer informações sensíveis temporariamente

Hierarquia de Precedência

1. **Linha de comando** (maior precedência)
2. Arquivos .tfvars
3. Valores padrão no código

O valor da linha de comando sempre prevalece sobre outras formas de atribuição.

Como Passar Variáveis pela Linha de Comando

Você pode usar a flag `-var` seguida do nome da variável e seu valor:

```
terraform apply -var="regiao=us-east-1" -var="numero_instancias=2"
```

Variáveis Complexas (Listas e Mapas)

Para variáveis que são listas ou mapas, a sintaxe pode ser um pouco mais complexa, exigindo que o valor seja passado como uma string JSON:

```
terraform apply -var='subredes=["10.0.1.0/24", "10.0.2.0/24"]'
terraform apply -var='tags_comuns={"Ambiente":"Dev","Projeto":"API"}'
```

Considerações de Segurança (DevSecOps): Embora conveniente, evite passar informações sensíveis (como senhas ou chaves de API) diretamente na linha de comando em scripts automatizados ou ambientes compartilhados, pois elas podem ser expostas no histórico de comandos. Para segredos, é preferível usar gerenciadores de segredos integrados ou variáveis de ambiente, que são abordagens mais seguras e alinhadas com as práticas de **DevSecOps**.

O Bloco output: Expondo Informações Cruciais da Infraestrutura

Após provisionar sua infraestrutura, você frequentemente precisará de informações sobre os recursos que foram criados. Qual é o endereço IP público daquele servidor? Qual é o endpoint do banco de dados? Qual é o nome do bucket S3 que foi gerado? O bloco `output` é a ferramenta que permite que sua ferramenta de IaC exponha esses dados de forma estruturada e acessível, tornando-os fáceis de serem consumidos por você, por outros scripts ou por outras ferramentas.

O que é o Output?

Pense no output como a "nota fiscal" ou o "relatório final" da sua construção. Depois que a casa está pronta, você recebe um documento com o endereço, o número de quartos, a metragem quadrada e outras informações importantes.

O bloco output faz exatamente isso para sua infraestrutura: ele sumariza e apresenta os dados mais relevantes dos recursos provisionados.

Por que é Importante?

- Facilita integração com CI/CD
- Automatiza configurações downstream
- Elimina consultas manuais
- Fornece dados para testes automatizados
- Documenta recursos criados

Como Usar o Bloco output

Você define um bloco output com um nome e um valor, que geralmente é uma referência a um atributo de um recurso criado.

```
output "ip_publico_servidor" {
  description = "O endereço IP público da instância web."
  value       = aws_instance.web_server.public_ip
}

output "endpoint_banco_dados" {
  description = "O endpoint de conexão do banco de dados."
  value       = aws_db_instance.main_db.address
  sensitive   = true  # Marca como sensível para não exibir em logs
}
```

description

(Opcional) Uma descrição clara do que o output representa.

value

O valor que será exposto. Pode ser uma string, um número, uma lista, um mapa, ou uma combinação deles, geralmente referenciando atributos de recursos.

sensitive = true

(Opcional) Se o valor contiver informações sensíveis (senhas, chaves), marcá-lo como sensitive fará com que ele não seja exibido em logs ou na saída padrão do comando apply, aumentando a segurança.

Após a aplicação da infraestrutura, os valores dos outputs são exibidos no console e podem ser consultados posteriormente.

Conectando Variáveis e Outputs com o Mundo Real: GitOps, DevSecOps e AIOps

A compreensão e o uso eficaz de variáveis e blocos output são mais do que apenas boas práticas; eles são habilitadores essenciais para as metodologias e tendências mais avançadas em operações de TI. A infraestrutura moderna exige agilidade, segurança e inteligência, e esses conceitos são a base para alcançá-las.

GitOps como Padrão

No **GitOps**, o repositório Git é a "única fonte da verdade" para o estado desejado da sua infraestrutura. Variáveis desempenham um papel crucial aqui, permitindo que o mesmo código base de infraestrutura seja aplicado a diferentes ambientes (desenvolvimento, teste, produção) simplesmente alterando os valores em arquivos .tfvars versionados.

Isso significa que a diferença entre um ambiente de desenvolvimento e um de produção pode ser tão simples quanto um conjunto de valores de variáveis, todos rastreados e auditáveis no Git. Os outputs, por sua vez, podem ser usados para alimentar outros repositórios Git ou sistemas que dependem das informações da infraestrutura provisionada, fechando o ciclo do GitOps.

Segurança Integrada (DevSecOps)

A abordagem **DevSecOps** enfatiza a segurança desde o início do ciclo de vida do desenvolvimento. Variáveis são fundamentais para isso:

- **Configuração Segura:** Variáveis podem ser usadas para injetar configurações de segurança (ex: regras de firewall, políticas de IAM, chaves de criptografia) de forma parametrizada, garantindo que as melhores práticas de segurança sejam aplicadas consistentemente em todos os ambientes.
- **Gerenciamento de Segredos:** Em vez de hardcoding, variáveis são o mecanismo para integrar gerenciadores de segredos (como HashiCorp Vault, AWS Secrets Manager) ao seu código IaC, passando credenciais e chaves de forma segura.
- **Outputs Sensíveis:** O atributo `sensitive = true` nos blocos output é uma prática de DevSecOps, prevenindo que informações confidenciais sejam expostas em logs ou na saída padrão, reduzindo o risco de vazamento de dados.

AIOps e Automação Inteligente

A **AIOps** busca otimizar operações de TI usando Inteligência Artificial para prever falhas, automatizar remediações e otimizar recursos. Variáveis e outputs são a ponte entre a inteligência artificial e a infraestrutura:

- **Inputs Dinâmicos:** Sistemas de AIOps podem gerar valores para variáveis de entrada com base em análises de dados (ex: ajustar o número de instâncias com base na previsão de tráfego, ou o tamanho do disco com base no uso histórico).
- **Outputs para Monitoramento:** Os outputs da infraestrutura provisionada podem ser consumidos por ferramentas de monitoramento e AIOps, fornecendo dados essenciais (endpoints, IPs, IDs de recursos) para que a IA possa monitorar, analisar e agir sobre a infraestrutura de forma inteligente.

Conclusão: Em suma, variáveis e outputs não são apenas recursos técnicos; são ferramentas estratégicas que permitem que sua organização adote práticas modernas, seguras e inteligentes na gestão da infraestrutura.

Síntese e Aplicação Prática

Nesta aula, exploramos o papel fundamental das variáveis de entrada e saída na construção de uma infraestrutura como código robusta e flexível. Vimos como as variáveis permitem parametrizar seu código, tornando-o reutilizável para diferentes ambientes e necessidades, eliminando a repetição e reduzindo erros. Discutimos os diversos tipos de variáveis – desde os simples string, number e bool até os estruturados list, map e object – e como cada um serve a um propósito específico na organização dos seus dados.

Aprendemos a gerenciar esses parâmetros de forma eficiente, utilizando arquivos `.tfvars` para configurações de ambiente e a linha de comando para ajustes rápidos ou sobrescritas pontuais, sempre com atenção às melhores práticas de segurança. Finalmente, desvendamos o poder do bloco `output` para expor informações cruciais da infraestrutura provisionada, facilitando a integração com outros sistemas e o consumo por automações.

Em prática:

Use variáveis para tudo que é configurável

Sempre use variáveis para qualquer valor que possa mudar entre ambientes ou que precise ser configurável.

Organize com arquivos .tfvars

Organize suas variáveis em arquivos .tfvars específicos para cada ambiente (ex: dev.tfvars, prod.tfvars).

Exponha informações com outputs

Utilize o bloco output para expor endereços IP, endpoints de serviços, IDs de recursos e outras informações que serão consumidas por aplicações ou outros módulos.

Proteja dados sensíveis

Marque outputs sensíveis com `sensitive = true` para proteger informações confidenciais.

Integre com CI/CD

Incorpore variáveis e outputs em seus pipelines de CI/CD para automação e integração contínua, alinhando-se com GitOps e DevSecOps.

Autoavaliação

Questão 1

Qual é o principal benefício de utilizar variáveis em Infraestrutura como Código?

1. Reduzir o tempo de compilação do código.
2. Aumentar a complexidade do código para maior segurança.
3. Parametrizar o código, tornando-o reutilizável e adaptável a diferentes ambientes.
4. Eliminar a necessidade de versionamento do código.

Questão 2

Você precisa definir uma lista de portas de rede que devem ser abertas em um grupo de segurança. Qual tipo de variável seria o mais adequado para essa finalidade?

1. string
2. number
3. list
4. bool

Questão 3

Ao gerenciar variáveis, qual método tem a maior precedência em caso de conflito de valores para a mesma variável?

1. Valores padrão definidos no código.
2. Variáveis definidas em arquivos .tfvars.
3. Variáveis passadas pela linha de comando com a flag -var.
4. Variáveis de ambiente.

Questão 4

Qual é a finalidade do atributo `sensitive = true` em um bloco output?

1. Indicar que o output é opcional e pode ser omitido.
2. Marcar o valor do output para que não seja exibido em logs ou na saída padrão, protegendo informações confidenciais.
3. Aumentar a performance da aplicação da infraestrutura.
4. Definir o tipo de dado do valor do output como sensível.

Gabarito

- **1.** c)
- **2.** c)
- **3.** c)
- **4.** b)

Questão Discursiva

> Explique como a utilização de variáveis e blocos output contribui para a implementação das metodologias GitOps e DevSecOps em um ambiente de Infraestrutura como Código.

Próximos Passos e Recursos

Próxima Aula

Na Aula 9, aprofundaremos em outro conceito crucial para a gestão da infraestrutura: **O Arquivo de Estado (State File) do Terraform**. Entenderemos como ele funciona, sua importância para o controle do ciclo de vida dos recursos e as melhores práticas para seu gerenciamento e segurança.

Recursos Adicionais

Documentação Oficial da Ferramenta de IaC

Para detalhes específicos sobre a sintaxe e uso de variáveis e outputs.

Artigos sobre GitOps e DevSecOps

Para aprofundar a conexão entre as práticas de IaC e essas metodologias.

Tutoriais Práticos de IaC

Para aplicar os conceitos aprendidos em cenários reais de provisionamento de infraestrutura.

NOTA IMPORTANTE: As informações regulatórias/legais/técnicas desta aula estão atualizadas até 2025. Consulte sempre fontes oficiais para verificar alterações.