```
myDB:
  Type: 'AWS::RDS::DBInstance'
  DeletionPolicy: Retain
  UpdateReplacePolicy: Retain
  Properties: {}
```

Opções de UpdateReplacePolicy

Excluir

O CloudFormation exclui o recurso e todo o conteúdo dele, se aplicável, durante a substituição do recurso. Você pode adicionar esta política a qualquer tipo de recurso. Por padrão, se você não especificar um `UpdateReplacePolicy`, CloudFormation excluirá seus recursos. No entanto, esteja ciente do seguinte:

Para buckets do Amazon S3, você deverá excluir todos os objetos no bucket para a exclusão para ser bem-sucedida.

Reter

O CloudFormation mantém o recurso sem excluir o recurso ou o conteúdo dele quando ele é substituído. Você pode adicionar esta política a qualquer tipo de recurso. Os recursos que são retidos continuam a existir e a incorrer em cobranças aplicáveis até que você exclua esses recursos.

Se um recurso é substituído, a `UpdateReplacePolicy` retém o recurso físico antigo, mas o remove do escopo do CloudFormation.

Snapshot

Para os recursos que oferecem suporte a snapshots, o CloudFormation cria um snapshot para o recurso antes de excluí-lo. Snapshots criados com essa política continuam a existir e continuam a incorrer em cobranças aplicáveis até que você os exclua.

> Note
>
> Se você especificar a opção `Snapshot` em `UpdateReplacePolicy` para um recurso que não oferece apoio a snapshots, o CloudFormation reverterá para a opção padrão, que é `Delete`.

Os recursos que oferecem suporte a snapshots incluem:

- `AWS::EC2::Volume`
- `AWS::ElastiCache::CacheCluster`
- `AWS::ElastiCache::ReplicationGroup`
- `AWS::Neptune::DBCluster`
- `AWS::RDS::DBCluster`
- `AWS::RDS::DBInstance`
- `AWS::Redshift::Cluster`

Referência à função intrínseca

O AWS CloudFormation fornece várias funções internas que ajudam você a gerenciar as pilhas. Use funções intrínsecas nos modelos para atribuir valores às propriedades que não estão disponíveis até o runtime.

> Note
>
> Você só pode usar funções intrínsecas em partes específicas de um modelo. Atualmente, você pode usar funções intrínsecas em propriedades de recurso, saídas, atributos de metadados

e atributos de política de atualização. Você também pode usar funções intrínsecas para criar condicionalmente recursos da pilha.

Tópicos

Fn::Base64

A função intrínseca `Fn::Base64` retorna a representação Base64 da string de entrada. Essa função normalmente é usada para passar dados codificados para instâncias do Amazon EC2 por meio da propriedade `UserData`.

Declaração

JSON

```
{ "Fn::Base64" : valueToEncode }
```

YAML

Sintaxe para o nome da função completo:

```
Fn::Base64: valueToEncode
```

Sintaxe para a forma resumida:

```
!Base64 valueToEncode
```

Note

Caso você use a forma curta e inclua imediatamente outra função no parâmetro `valueToEncode`, use o nome da função completo em pelo menos uma das funções. Por exemplo, a sintaxe a seguir não é válida:

```
!Base64 !Sub string
!Base64 !Ref logical_ID
```

Em vez disso, use o nome da função completo em pelo menos uma das funções, conforme mostrado nos seguintes exemplos:

```
!Base64
  "Fn::Sub": string

Fn::Base64:
  !Sub string
```

Parâmetros

valueToEncode

O valor da string que você deseja converter em Base64.

Valor de retorno:

A string original, em representação Base64.

Exemplo

JSON

```
{ "Fn::Base64" : "AWS CloudFormation" }
```

YAML

```
Fn::Base64: AWS CloudFormation
```

Funções compatíveis

Você pode usar qualquer função que retorne uma string dentro da função `Fn::Base64`.

Consulte também

- Referência à função intrínseca (p. 6709)

Fn::Cidr

A função intrínseca `Fn::Cidr` retorna uma matriz de blocos de endereços CIDR. O número de blocos CIDR retornados depende do parâmetro `count`.

Declaração

JSON

```
{ "Fn::Cidr" : [ipBlock, count, cidrBits]}
```

YAML

Sintaxe para o nome da função completo:

```
Fn::Cidr:
  - ipBlock
```

```
  - count
  - cidrBits
```

Sintaxe para a forma resumida:

```
!Cidr [ ipBlock, count, cidrBits ]
```

Parâmetros

ipBlock

O bloco de endereços CIDR especificado pelo usuário a ser dividido em blocos CIDR menores.

count

O número de CIDRs para gerar. O intervalo válido é entre 1 e 256.

cidrBits

O número de bits de sub-rede para o CIDR. Por exemplo, se você especificar um valor "8" para esse parâmetro, criará um CIDR com uma máscara de "/ 24".

> Note
>
> Bits de sub-rede é o inverso de uma máscara de sub-rede. Para calcular os bits de host necessários para um determinado número de bits de sub-rede, subtraia os bits de sub-rede de 32 para IPv4 ou de 128 para IPv6.

Valor de retorno

Uma matriz de blocos de endereços CIDR.

Exemplo

Uso básico

Este exemplo cria 6 CIDRs com uma máscara de sub-rede "/ 27" dentro de um CIDR com uma máscara de "/24".

JSON

```
{ "Fn::Cidr" : [ "192.168.0.0/24", "6", "5"] }
```

YAML

```
!Cidr [ "192.168.0.0/24", 6, 5 ]
```

Criar uma VPC habilitada para IPv6

Este modelo de exemplo cria uma sub-rede habilitada para IPv6.

JSON

```
{
  "Resources" : {
    "ExampleVpc" : {
      "Type" : "AWS::EC2::VPC",
      "Properties" : {
```

```
        "CidrBlock" : "10.0.0.0/16"
      }
    },
    "IPv6CidrBlock" : {
      "Type" : "AWS::EC2::VPCCidrBlock",
      "Properties" : {
        "AmazonProvidedIpv6CidrBlock" : true,
        "VpcId" : { "Ref" : "ExampleVpc" }
      }
    },
    "ExampleSubnet" : {
      "Type" : "AWS::EC2::Subnet",
      "DependsOn" : "IPv6CidrBlock",
      "Properties" : {
        "AssignIpv6AddressOnCreation" : true,
        "CidrBlock" : { "Fn::Select" : [ 0, { "Fn::Cidr" : [{ "Fn::GetAtt" :
 [ "ExampleVpc", "CidrBlock" ]}, 1, 8 ]}]},
        "Ipv6CidrBlock" : { "Fn::Select" : [ 0, { "Fn::Cidr" : [{ "Fn::Select" : [ 0,
 { "Fn::GetAtt" : [ "ExampleVpc", "Ipv6CidrBlocks" ]}]}, 1, 64 ]}]},
        "VpcId" : { "Ref" : "ExampleVpc" }
      }
    }
  }
}
```

YAML

```
Resources:
    ExampleVpc:
        Type: AWS::EC2::VPC
        Properties:
            CidrBlock: "10.0.0.0/16"
     IPv6CidrBlock:
        Type: AWS::EC2::VPCCidrBlock
        Properties:
            AmazonProvidedIpv6CidrBlock: true
            VpcId: !Ref ExampleVpc
     ExampleSubnet:
        Type: AWS::EC2::Subnet
        DependsOn: IPv6CidrBlock
        Properties:
            AssignIpv6AddressOnCreation: true
            CidrBlock: !Select [ 0, !Cidr [ !GetAtt ExampleVpc.CidrBlock, 1, 8 ]]
            Ipv6CidrBlock: !Select [ 0, !Cidr [ !Select [ 0, !GetAtt
 ExampleVpc.Ipv6CidrBlocks], 1, 64 ]]
            VpcId: !Ref ExampleVpc
```

Funções compatíveis

Você pode usar as seguintes funções em uma função `Fn::Cidr`:

- `Fn::Select` (p. 6743)
- `Ref` (p. 6752)

Funções de condição

Você pode usar funções intrínsecas, como `Fn::If`, `Fn::Equals` e `Fn::Not`, para criar condicionalmente recursos da pilha. Essas condições são avaliadas com base nos parâmetros de entrada que você declara ao criar ou atualizar uma pilha. Depois de definir todas as condições, você pode associá-las a recursos ou propriedades de recursos nas seções Recursos e Saídas de um modelo.

Você define todas as condições na seção Condições de um modelo, exceto para as condições `Fn::If`. Você pode usar a condição `Fn::If` no atributo de metadados, atualizar o atributo de política e valores de propriedade nas seções Recursos e Saídas de um modelo.

Você pode usar as condições quando você deseja reutilizar um modelo que pode criar recursos em contextos diferentes, como um ambiente de teste em comparação com um ambiente de produção. No modelo, você pode adicionar um parâmetro de entrada `EnvironmentType` que aceita **`prod`** ou **`test`** como entrada. Para o ambiente de produção, você pode incluir instâncias Amazon EC2 com determinados recursos. No entanto, para o ambiente de teste, você deve usar menos recursos para economizar custos. Com as condições, você pode definir quais recursos são criados e como eles são configurados para cada tipo de ambiente.

Para obter mais informações sobre a seção Condições, consulte Condições (p. 345).

> **Note**
>
> Você só pode fazer referência a outras condições e valores das seções Parâmetros e Mapeamentos de um modelo. Por exemplo, você pode fazer referência a um valor de um parâmetro de entrada, mas não ao ID lógico de um recurso em uma condição.

Tópicos

Associar uma condição

Para criar condicionalmente recursos, propriedades de recursos ou saídas, você deve associar uma condição a eles. Adicione a chave `Condition:` e o ID lógico da condição como um atributo para associar uma condição, como mostrado no seguinte trecho. O AWS CloudFormation cria o recurso `NewVolume` somente quando a condição `CreateProdResources` avalia como verdadeira.

JSON

```
"NewVolume" : {
  "Type" : "AWS::EC2::Volume",
  "Condition" : "CreateProdResources",
  "Properties" : {
     "Size" : "100",
     "AvailabilityZone" : { "Fn::GetAtt" : [ "EC2Instance", "AvailabilityZone" ]}
}
```

YAML

```
NewVolume:
  Type: "AWS::EC2::Volume"
  Condition: CreateProdResources
  Properties:
    Size: 100
    AvailabilityZone: !GetAtt EC2Instance.AvailabilityZone
```

Fn::If

Para a função `Fn::If`, você só precisa especificar o nome da condição. O seguinte trecho mostra como usar `Fn::If` para especificar condicionalmente uma propriedade de recurso. Se a condição `CreateLargeSize` for verdadeira, o CloudFormation definirá o tamanho do volume como `100`. Se a condição for falsa, o CloudFormation definirá o tamanho do volume como `10`.

JSON

```
"NewVolume" : {
  "Type" : "AWS::EC2::Volume",
  "Properties" : {
    "Size" : {
      "Fn::If" : [
        "CreateLargeSize",
        "100",
        "10"
      ]},
    "AvailabilityZone" : { "Fn::GetAtt" : [ "Ec2Instance", "AvailabilityZone" ]}
  },
  "DeletionPolicy" : "Snapshot"
}
```

YAML

```
NewVolume:
  Type: "AWS::EC2::Volume"
  Properties:
    Size:
      !If [CreateLargeSize, 100, 10]
    AvailabilityZone: !GetAtt: Ec2Instance.AvailabilityZone
  DeletionPolicy: Snapshot
```

Condições aninhadas

Você também pode usar condições em outras condições. O seguinte trecho é da seção `Conditions` de um modelo. A condição `MyAndCondition` inclui a condição `SomeOtherCondition`:

JSON

```
"MyAndCondition": {
   "Fn::And": [
      {"Fn::Equals": ["sg-mysggroup", {"Ref": "ASecurityGroup"}]},
      {"Condition": "SomeOtherCondition"}
   ]
}
```

YAML

```
MyAndCondition: !And
  - !Equals ["sg-mysggroup", !Ref "ASecurityGroup"]
  - !Condition SomeOtherCondition
```

Fn::And

Retorna `true` se todas as condições especificadas forem verdadeiras, ou retornarem `false` se alguma das condições for falsa. O `Fn::And` atua como operador AND. O número mínimo de condições que você pode incluir é 2 e o máximo é 10.

Declaração

JSON

```
"Fn::And": [{condition}, {...}]
```

YAML

Sintaxe para o nome da função completo:

```
Fn::And: [condition]
```

Sintaxe para a forma resumida:

```
!And [condition]
```

Parâmetros

`condition`

Uma condição que avalia como `true` ou `false`.

Exemplo

O seguinte `MyAndCondition` avalia como verdadeiro se o nome do security group referenciado é igual a `sg-mysggroup` e se `SomeOtherCondition` avalia como verdadeiro:

JSON

```
"MyAndCondition": {
   "Fn::And": [
      {"Fn::Equals": ["sg-mysggroup", {"Ref": "ASecurityGroup"}]},
      {"Condition": "SomeOtherCondition"}
   ]
}
```

YAML

```
MyAndCondition: !And
  - !Equals ["sg-mysggroup", !Ref ASecurityGroup]
  - !Condition SomeOtherCondition
```

Fn::Equals

Compara se dois valores são iguais. Retorna `true` se os dois valores são iguais ou `false` se eles não são.

Declaração

JSON

```
"Fn::Equals" : ["value_1", "value_2"]
```

YAML

Sintaxe para o nome da função completo:

```
Fn::Equals: [value_1, value_2]
```

Sintaxe para a forma resumida:

```
!Equals [value_1, value_2]
```

Parâmetros

`value`

Um valor de qualquer tipo que você deseja comparar.

Exemplo

A seguinte condição `UseProdCondition` avaliará como verdadeira se o valor para o parâmetro `EnvironmentType` for igual a `prod`:

JSON

```
"UseProdCondition" : {
   "Fn::Equals": [
      {"Ref": "EnvironmentType"},
      "prod"
   ]
}
```

YAML

```
UseProdCondition:
  !Equals [!Ref EnvironmentType, prod]
```

Fn::If

Retorna um valor se a condição especificada avalia como `true` e outro valor se a condição especificada avalia como `false`. Atualmente, o CloudFormation é compatível com a função intrínseca `Fn::If` no atributo de metadados, a atualização do atributo de política e os valores de propriedade nas seções Recursos e Saídas de um modelo. Você pode usar o pseudoparâmetro `AWS::NoValue` como um valor de retorno para remover a propriedade correspondente.

Declaração

JSON

```
"Fn::If": [condition_name, value_if_true, value_if_false]
```

YAML

Sintaxe para o nome da função completo:

```
Fn::If: [condition_name, value_if_true, value_if_false]
```

Sintaxe para a forma resumida:

```
!If [condition_name, value_if_true, value_if_false]
```

Parâmetros

`condition_name`

Uma referência a uma condição na seção Condições. Use o nome da condição para fazer referência a ele.

`value_if_true`

Um valor a ser retornado se a condição especificada avalia como `true`.

`value_if_false`

Um valor a ser retornado se a condição especificada avalia como `false`.

Exemplos

Para visualizar mais exemplos, consulte Modelos de exemplo (p. 6722).

Exemplo 1

O seguinte trecho usa uma função `Fn::If` na propriedade `SecurityGroups` para um recurso do Amazon EC2. Se a condição `CreateNewSecurityGroup` for verdadeira, o CloudFormation usará o valor referenciado de `NewSecurityGroup` para especificar a propriedade `SecurityGroups`; caso contrário, o CloudFormation usará o valor referenciado de `ExistingSecurityGroup`.

JSON

```
"SecurityGroups" : [{
  "Fn::If" : [
    "CreateNewSecurityGroup",
    {"Ref" : "NewSecurityGroup"},
    {"Ref" : "ExistingSecurityGroup"}
  ]
}]
```

YAML

```
SecurityGroups:
  - !If [CreateNewSecurityGroup, !Ref NewSecurityGroup, !Ref ExistingSecurityGroup]
```

Exemplo 2

Na seção Saída de um modelo, você pode usar a função `Fn::If` para emitir condicionalmente informações. No seguinte trecho, se a condição `CreateNewSecurityGroup` for verdadeira, o CloudFormation emitirá o ID do grupo de segurança do recurso `NewSecurityGroup`. Se a condição for falsa, o CloudFormation emitirá o ID do grupo de segurança do recurso `ExistingSecurityGroup`.

JSON

```
"Outputs" : {
  "SecurityGroupId" : {
    "Description" : "Group ID of the security group used.",
    "Value" : {
```

```
      "Fn::If" : [
        "CreateNewSecurityGroup",
        {"Ref" : "NewSecurityGroup"},
        {"Ref" : "ExistingSecurityGroup"}
      ]
    }
  }
}
```

YAML

```
Outputs:
  SecurityGroupId:
    Description: Group ID of the security group used.
    Value: !If [CreateNewSecurityGroup, !Ref NewSecurityGroup, !Ref ExistingSecurityGroup]
```

Exemplo 3

O seguinte trecho usa o pseudoparâmetro `AWS::NoValue` em uma função `Fn::If`. A condição usa um snapshot para uma instância de banco de dados do Amazon RDS somente se o ID do snapshot é fornecido. Se a condição `UseDBSnapshot` for avaliada como verdadeira, o CloudFormation usará o valor do parâmetro `DBSnapshotName` para a propriedade `DBSnapshotIdentifier`. Se a condição for avaliada como falsa, o CloudFormation removerá a propriedade `DBSnapshotIdentifier`.

JSON

```
"MyDB" : {
  "Type" : "AWS::RDS::DBInstance",
  "Properties" : {
    "AllocatedStorage" : "5",
    "DBInstanceClass" : "db.t2.small",
    "Engine" : "MySQL",
    "EngineVersion" : "5.5",
    "MasterUsername" : { "Ref" : "DBUser" },
    "MasterUserPassword" : { "Ref" : "DBPassword" },
    "DBParameterGroupName" : { "Ref" : "MyRDSParamGroup" },
    "DBSnapshotIdentifier" : {
      "Fn::If" : [
        "UseDBSnapshot",
        {"Ref" : "DBSnapshotName"},
        {"Ref" : "AWS::NoValue"}
      ]
    }
  }
}
```

YAML

```
MyDB:
  Type: "AWS::RDS::DBInstance"
  Properties:
    AllocatedStorage: 5
    DBInstanceClass: db.t2.small
    Engine: MySQL
    EngineVersion: 5.5
    MasterUsername: !Ref DBUser
    MasterUserPassword: !Ref DBPassword
    DBParameterGroupName: !Ref MyRDSParamGroup
    DBSnapshotIdentifier:
      !If [UseDBSnapshot, !Ref DBSnapshotName, !Ref "AWS::NoValue"]
```

Exemplo 4

O seguinte trecho oferece uma política de atualização de escalabilidade automática somente se a condição `RollingUpdates` for verdadeira. Se a condição for avaliada como falsa, o CloudFormation removerá a política de atualização `AutoScalingRollingUpdate`.

JSON

```
"UpdatePolicy": {
  "AutoScalingRollingUpdate": {
    "Fn::If": [
      "RollingUpdates",
      {
        "MaxBatchSize": "2",
        "MinInstancesInService": "2",
        "PauseTime": "PT0M30S"
      },
      {
        "Ref" : "AWS::NoValue"
      }
    ]
  }
}
```

YAML

```
UpdatePolicy:
  AutoScalingRollingUpdate:
    !If
      - RollingUpdates
      -
        MaxBatchSize: 2
        MinInstancesInService: 2
        PauseTime: PT0M30S
      - !Ref "AWS::NoValue"
```

Fn::Not

Retorna `true` para uma condição que avalia como `false` ou retorna `false` para uma condição que avalia como `true`. `Fn::Not` funciona como o operador NOT.

Declaração

JSON

```
"Fn::Not": [{condition}]
```

YAML

Sintaxe para o nome da função completo:

```
Fn::Not: [condition]
```

Sintaxe para a forma resumida:

```
!Not [condition]
```

Parâmetros

`condition`

Uma condição como `Fn::Equals` que avalia como `true` ou `false`.

Exemplo

A seguinte condição `EnvCondition` será avaliada como verdadeira se o valor para o parâmetro `EnvironmentType` não for igual a `prod`:

JSON

```
"MyNotCondition" : {
   "Fn::Not" : [{
      "Fn::Equals" : [
         {"Ref" : "EnvironmentType"},
         "prod"
      ]
   }]
}
```

YAML

```
MyNotCondition:
  !Not [!Equals [!Ref EnvironmentType, prod]]
```

Fn::Or

Retorna `true` se alguma das condições especificadas forem verdadeiras, ou retornarem `false` se todas as condições forem falsas. O `Fn::Or` atua como operador OR. O número mínimo de condições que você pode incluir é 2 e o máximo é 10.

Declaração

JSON

```
"Fn::Or": [{condition}, {...}]
```

YAML

Sintaxe para o nome da função completo:

```
Fn::Or: [condition, ...]
```

Sintaxe para a forma resumida:

```
!Or [condition, ...]
```

Parâmetros

`condition`

Uma condição que avalia como `true` ou `false`.

Exemplo

O seguinte `MyOrCondition` avalia como verdadeiro se o nome do security group referenciado é igual a `sg-mysggroup` ou se `SomeOtherCondition` avalia como verdadeiro:

JSON

```
"MyOrCondition" : {
   "Fn::Or" : [
      {"Fn::Equals" : ["sg-mysggroup", {"Ref" : "ASecurityGroup"}]},
      {"Condition" : "SomeOtherCondition"}
   ]
}
```

YAML

```
MyOrCondition:
  !Or [!Equals [sg-mysggroup, !Ref ASecurityGroup], Condition: SomeOtherCondition]
```

Funções compatíveis

Você pode usar as seguintes funções na condição `Fn::If`:

- `Fn::Base64`
- `Fn::FindInMap`
- `Fn::GetAtt`
- `Fn::GetAZs`
- `Fn::If`
- `Fn::Join`
- `Fn::Select`
- `Fn::Sub`
- `Ref`

Você pode usar as seguintes funções em todas as outras funções de condição, como `Fn::Equals` e `Fn::Or`:

- `Fn::FindInMap`
- `Ref`
- Outras funções de condição

Modelos de exemplo

Crie condicionalmente recursos para uma produção, desenvolvimento ou teste de pilha

Em alguns casos, você pode criar pilhas que são semelhantes, mas com pequenos ajustes. Por exemplo, você pode ter um modelo que você usa para aplicativos de produção. Você deseja criar a mesma pilha de produção, para que você possa usá-la para desenvolvimento ou teste. No entanto, para desenvolvimento e teste, você pode não exigir toda a capacidade extra incluída em uma pilha de nível de produção. Em vez disso, você pode usar um parâmetro de entrada no tipo de ambiente para criar condicionalmente

os recursos de pilha que são específicos para produção, desenvolvimento ou testes, como mostrado no seguinte exemplo:

Example JSON

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",

  "Mappings" : {
    "RegionMap" : {
      "us-east-1"      : { "AMI" : "ami-0ff8a91507f77f867"},
      "us-west-1"      : { "AMI" : "ami-0bdb828fd58c52235"},
      "us-west-2"      : { "AMI" : "ami-a0cfeed8"},
      "eu-west-1"      : { "AMI" : "ami-047bb4163c506cd98"},
      "sa-east-1"      : { "AMI" : "ami-07b14488da8ea02a0"},
      "ap-southeast-1" : { "AMI" : "ami-08569b978cc4dfa10"},
      "ap-southeast-2" : { "AMI" : "ami-09b42976632b27e9b"},
      "ap-northeast-1" : { "AMI" : "ami-06cd52961ce9f0d85"}
    }
  },

  "Parameters" : {
    "EnvType" : {
      "Description" : "Environment type.",
      "Default" : "test",
      "Type" : "String",
      "AllowedValues" : ["prod", "dev", "test"],
      "ConstraintDescription" : "must specify prod, dev, or test."
    }
  },

  "Conditions" : {
    "CreateProdResources" : {"Fn::Equals" : [{"Ref" : "EnvType"}, "prod"]},
    "CreateDevResources" : {"Fn::Equals" : [{"Ref" : "EnvType"}, "dev"]}
  },

  "Resources" : {
    "EC2Instance" : {
      "Type" : "AWS::EC2::Instance",
      "Properties" : {
        "ImageId" : { "Fn::FindInMap" : [ "RegionMap", { "Ref" : "AWS::Region" }, "AMI" ]},
        "InstanceType" : { "Fn::If" : [
          "CreateProdResources",
          "c1.xlarge",
          {"Fn::If" : [
            "CreateDevResources",
            "m1.large",
            "m1.small"
          ]}
        ]}
      }
    },

    "MountPoint" : {
      "Type" : "AWS::EC2::VolumeAttachment",
      "Condition" : "CreateProdResources",
      "Properties" : {
        "InstanceId" : { "Ref" : "EC2Instance" },
        "VolumeId"  : { "Ref" : "NewVolume" },
        "Device" : "/dev/sdh"
      }
    },

    "NewVolume" : {
      "Type" : "AWS::EC2::Volume",
```

```
      "Condition" : "CreateProdResources",
      "Properties" : {
        "Size" : "100",
        "AvailabilityZone" : { "Fn::GetAtt" : [ "EC2Instance", "AvailabilityZone" ]}
      }
    }
  }
}
```

Example YAML

```
AWSTemplateFormatVersion: "2010-09-09"

Mappings:
  RegionMap:
    us-east-1:
      AMI: "ami-0ff8a91507f77f867"
    us-west-1:
      AMI: "ami-0bdb828fd58c52235"
    us-west-2:
      AMI: "ami-a0cfeed8"
    eu-west-1:
      AMI: "ami-047bb4163c506cd98"
    sa-east-1:
      AMI: "ami-07b14488da8ea02a0"
    ap-southeast-1:
      AMI: "ami-08569b978cc4dfa10"
    ap-southeast-2:
      AMI: "ami-09b42976632b27e9b"
    ap-northeast-1:
      AMI: "ami-06cd52961ce9f0d85"

Parameters:
  EnvType:
    Description: Environment type.
    Default: test
    Type: String
    AllowedValues: [prod, dev, test]
    ConstraintDescription: must specify prod, dev, or test.

Conditions:
  CreateProdResources: !Equals [!Ref EnvType, prod]
  CreateDevResources: !Equals [!Ref EnvType, "dev"]

Resources:
  EC2Instance:
    Type: "AWS::EC2::Instance"
    Properties:
      ImageId: !FindInMap [RegionMap, !Ref "AWS::Region", AMI]
      InstanceType: !If [CreateProdResources, c1.xlarge, !If [CreateDevResources, m1.large,
 m1.small]]
  MountPoint:
    Type: "AWS::EC2::VolumeAttachment"
    Condition: CreateProdResources
    Properties:
      InstanceId: !Ref EC2Instance
      VolumeId: !Ref NewVolume
      Device: /dev/sdh
  NewVolume:
    Type: "AWS::EC2::Volume"
    Condition: CreateProdResources
    Properties:
      Size: 100
      AvailabilityZone: !GetAtt EC2Instance.AvailabilityZone
```

Você pode especificar `prod`, `dev` ou `test` para o parâmetro `EnvType`. Para cada tipo de ambiente, o modelo especifica um tipo de instância diferente. Os tipos de instância podem variar desde um tipo de instância grande, otimizada por computação, até um tipo de instância pequeno, de uso geral. Para especificar condicionalmente o tipo de instância, o modelo define duas condições na seção Condições do modelo: `CreateProdResources`, que avalia como verdadeiro se o valor do parâmetro `EnvType` é igual a `prod` e `CreateDevResources`, que avalia como verdadeiro se o valor do parâmetro é igual a `dev`.

Na propriedade `InstanceType`, o modelo aninha duas funções intrínsecas `Fn::If` para determinar qual tipo de instância deverá ser usada. Se a condição `CreateProdResources` for verdadeira, o tipo de instância será `c1.xlarge`. Se a condição for falsa, a condição `CreateDevResources` será avaliada. Se a condição `CreateDevResources` for verdadeira, o tipo de instância será `m1.large`; caso contrário, o tipo de instância será `m1.small`.

Além do tipo de instância, o ambiente de produção cria e anexa um volume do Amazon EC2 à instância. Os recurso `MountPoint` e `NewVolume` estão associados à condição `CreateProdResources` para que os recursos sejam criados somente se a condição for verdadeira.

Atribuição condicional de uma propriedade de recurso

Neste exemplo, você pode criar uma instância de banco de dados do Amazon RDS a partir de um snapshot. Se você especificar o parâmetro `DBSnapshotName`, o CloudFormation usará o valor do parâmetro como o nome do snapshot ao criar a instância de banco de dados. Se você mantiver o valor padrão (string vazia), o CloudFormation removerá a propriedade `DBSnapshotIdentifier` e criará uma instância de banco de dados do zero.

O exemplo define os parâmetros `DBUser` e `DBPassword` com a propriedade `NoEcho` definida como `true`. Se você definir o atributo `NoEcho` como `true`, o CloudFormation retornará o valor do parâmetro mascarado como asteriscos (*****) para qualquer chamada que descreva a pilha ou os eventos de pilha, exceto informações armazenadas nos locais especificados abaixo.

> **Important**
>
> O uso do atributo `NoEcho` não mascara nenhuma informação armazenada no seguinte:
>
> - A seção de modelo de `Metadata`. O CloudFormation não transforma, modifica nem edita nenhuma informação incluída na seção `Metadata`. Para obter mais informações, consulte Metadados.
> - A seção `Outputs` do modelo. Para obter mais informações, consulte Saídas.
> - O atributo `Metadata` da definição de um recurso. Para obter mais informações, Atributo de metadados.
>
> É altamente recomendável que não use esses mecanismos para incluir informações confidenciais, como senhas ou segredos.

> **Important**
>
> Em vez de incorporar informações confidenciais diretamente aos modelos do CloudFormation, recomendamos usar os parâmetros dinâmicos no modelo da pilha para fazer referência a informações confidenciais armazenadas e gerenciadas fora do CloudFormation, como no AWS Systems Manager Parameter Store e no AWS Secrets Manager.
> Para obter mais informações, consulte a melhor prática Não incorporar credenciais em seus modelos.

Example JSON

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",
```

```
 "Parameters": {
   "DBUser": {
     "NoEcho": "true",
     "Description" : "The database admin account username",
     "Type": "String",
     "MinLength": "1",
     "MaxLength": "16",
     "AllowedPattern" : "[a-zA-Z][a-zA-Z0-9]*",
     "ConstraintDescription" : "must begin with a letter and contain only alphanumeric
characters."
   },
   "DBPassword": {
     "NoEcho": "true",
     "Description" : "The database admin account password",
     "Type": "String",
     "MinLength": "1",
     "MaxLength": "41",
     "AllowedPattern" : "[a-zA-Z0-9]*",
     "ConstraintDescription" : "must contain only alphanumeric characters."
   },
   "DBSnapshotName": {
     "Description": "The name of a DB snapshot (optional)",
     "Default": "",
     "Type": "String"
   }
 },

 "Conditions": {
   "UseDBSnapshot": {"Fn::Not": [{"Fn::Equals" : [{"Ref" : "DBSnapshotName"}, ""]}]}
 },

 "Resources" : {
   "MyDB" : {
     "Type" : "AWS::RDS::DBInstance",
     "Properties" : {
       "AllocatedStorage" : "5",
       "DBInstanceClass" : "db.t2.small",
       "Engine" : "MySQL",
       "EngineVersion" : "5.5",
       "MasterUsername" : { "Ref" : "DBUser" },
       "MasterUserPassword" : { "Ref" : "DBPassword" },
       "DBParameterGroupName" : { "Ref" : "MyRDSParamGroup" },
       "DBSnapshotIdentifier" : {
         "Fn::If" : [
           "UseDBSnapshot",
           {"Ref" : "DBSnapshotName"},
           {"Ref" : "AWS::NoValue"}
         ]
       }
     }
   },

   "MyRDSParamGroup" : {
     "Type": "AWS::RDS::DBParameterGroup",
     "Properties" : {
       "Family" : "MySQL5.5",
       "Description" : "CloudFormation Sample Database Parameter Group",
         "Parameters" : {
           "autocommit" : "1" ,
           "general_log" : "1",
           "old_passwords" : "0"
       }
     }
   }
 }
```

```
}
```

Example YAML

```
AWSTemplateFormatVersion: "2010-09-09"
Parameters:
  DBUser:
    NoEcho: true
    Description: The database admin account username
    Type: String
    MinLength: 1
    MaxLength: 16
    AllowedPattern: "[a-zA-Z][a-zA-Z0-9]*"
    ConstraintDescription: must begin with a letter and contain only alphanumeric
 characters.
  DBPassword:
    NoEcho: true
    Description: The database admin account password
    Type: String
    MinLength: 1
    MaxLength: 41
    AllowedPattern: "[a-zA-Z0-9]*"
    ConstraintDescription: must contain only alphanumeric characters.
  DBSnapshotName:
    Description: The name of a DB snapshot (optional)
    Default: ""
    Type: String
Conditions:
  UseDBSnapshot: !Not [!Equals [!Ref DBSnapshotName, ""]]
Resources:
  MyDB:
    Type: "AWS::RDS::DBInstance"
    Properties:
      AllocatedStorage: 5
      DBInstanceClass: db.t2.small
      Engine: MySQL
      EngineVersion: 5.5
      MasterUsername: !Ref DBUser
      MasterUserPassword: !Ref DBPassword
      DBParameterGroupName: !Ref MyRDSParamGroup
      DBSnapshotIdentifier: !If [UseDBSnapshot, !Ref DBSnapshotName, !Ref "AWS::NoValue"]
  MyRDSParamGroup:
    Type: "AWS::RDS::DBParameterGroup"
    Properties:
      Family: MySQL5.5
      Description: CloudFormation Sample Database Parameter Group
      Parameters:
        autocommit: 1
        general_log: 1
        old_passwords: 0
```

A condição `UseDBSnapshot` será avaliada como verdadeira somente se `DBSnapshotName` não for uma string vazia. Se a condição `UseDBSnapshot` for avaliada como verdadeira, o CloudFormation usará o valor do parâmetro `DBSnapshotName` para a propriedade `DBSnapshotIdentifier`. Se a condição for avaliada como falsa, o CloudFormation removerá a propriedade `DBSnapshotIdentifier`. O pseudoparâmetro `AWS::NoValue` remove a propriedade de recurso correspondente quando é usada como um valor de retorno.

Uso condicional de um recurso existente

Neste exemplo, você pode usar um security group do Amazon EC2 que já tenha sido criado ou você pode criar um novo security group, que está especificado no modelo. Para o parâmetro `ExistingSecurityGroup`, você pode especificar o nome do security group `default` ou `NONE`. Se você

especificar `default`, o CloudFormation usará um grupo de segurança que já foi criado e é chamado de `default`. Se você especificar `NONE`, o CloudFormation criará o grupo de segurança que está definido no modelo.

Example JSON

```
{
  "Parameters" : {
    "ExistingSecurityGroup" : {
      "Description" : "An existing security group ID (optional).",
      "Default" : "NONE",
      "Type" : "String",
      "AllowedValues" : ["default", "NONE"]
    }
  },

  "Conditions" : {
    "CreateNewSecurityGroup" : {"Fn::Equals" : [{"Ref" : "ExistingSecurityGroup"},
 "NONE"] }
  },

  "Resources" : {
    "MyInstance" : {
      "Type" : "AWS::EC2::Instance",
        "Properties" : {
          "ImageId" : "ami-0ff8a91507f77f867",
          "SecurityGroups" : [{
            "Fn::If" : [
              "CreateNewSecurityGroup",
              {"Ref" : "NewSecurityGroup"},
              {"Ref" : "ExistingSecurityGroup"}
            ]
          }]
        }
    },

    "NewSecurityGroup" : {
      "Type" : "AWS::EC2::SecurityGroup",
      "Condition" : "CreateNewSecurityGroup",
      "Properties" : {
        "GroupDescription" : "Enable HTTP access via port 80",
        "SecurityGroupIngress" : [ {
          "IpProtocol" : "tcp",
          "FromPort" : 80,
          "ToPort" : 80,
          "CidrIp" : "0.0.0.0/0"
        } ]
      }
    }
  },

  "Outputs" : {
    "SecurityGroupId" : {
      "Description" : "Group ID of the security group used.",
      "Value" : {
        "Fn::If" : [
          "CreateNewSecurityGroup",
          {"Ref" : "NewSecurityGroup"},
          {"Ref" : "ExistingSecurityGroup"}
        ]
      }
    }
  }
}
```

Example YAML

```
Parameters:
  ExistingSecurityGroup:
    Description: An existing security group ID (optional).
    Default: NONE
    Type: String
    AllowedValues:
      - default
      - NONE
Conditions:
  CreateNewSecurityGroup: !Equals [!Ref ExistingSecurityGroup, NONE]
Resources:
  MyInstance:
    Type: "AWS::EC2::Instance"
    Properties:
      ImageId: "ami-0ff8a91507f77f867"
      SecurityGroups: !If [CreateNewSecurityGroup, !Ref NewSecurityGroup, !Ref
 ExistingSecurityGroup]
  NewSecurityGroup:
    Type: "AWS::EC2::SecurityGroup"
    Condition: CreateNewSecurityGroup
    Properties:
      GroupDescription: Enable HTTP access via port 80
      SecurityGroupIngress:
        -
          IpProtocol: tcp
          FromPort: 80
          ToPort: 80
          CidrIp: 0.0.0.0/0
Outputs:
  SecurityGroupId:
    Description: Group ID of the security group used.
    Value: !If [CreateNewSecurityGroup, !Ref NewSecurityGroup, !Ref ExistingSecurityGroup]
```

Para determinar se deseja criar o recurso `NewSecurityGroup`, o recurso está associada à condição `CreateNewSecurityGroup`. O recurso é criado somente quando a condição é verdadeira (quando o parâmetro `ExistingSecurityGroup` é igual a `NONE`).

Na propriedade `SecurityGroups`, o modelo usa a função intrínseca `Fn::If` para determinar qual tipo de security group deverá ser usado. Se a condição `CreateNewSecurityGroup` avaliar como verdadeira, a propriedade de security group fará referência ao recurso `NewSecurityGroup`. Se a condição `CreateNewSecurityGroup` avaliar como falsa, a propriedade de security group fará referência ao parâmetro `ExistingSecurityGroup` (o security group `default`).

Por fim, o modelo condicionalmente emite o ID do security group. Se a condição `CreateNewSecurityGroup` for avaliada como verdadeira, o CloudFormation emitirá o ID do grupo de segurança do recurso `NewSecurityGroup`. Se a condição for falsa, o CloudFormation emitirá o ID do grupo de segurança do recurso `ExistingSecurityGroup`.

Condição

A função intrínseca `Condition` retorna o resultado avaliado da condição especificada.

Quando estiver declarando uma condição em um modelo e precisar usar outra condição na avaliação, você poderá usar `Condition` para se referir a essa outra condição. Isso é usado quando uma condição é declarada na seção Condições do modelo.

Declaração

JSON

```
{ "Condition" : "conditionName" }
```

YAML

Sintaxe para o nome da função completo:

```
Condition: conditionName
```

Sintaxe do nome abreviado da função:

```
!Condition conditionName
```

Parâmetros

`conditionName`

O nome da condição à qual você deseja fazer referência.

Valor de retorno

O resultado booliano da condição referenciada.

Exemplo

O seguinte trecho é da seção `Conditions` de um modelo. A condição `MyAndCondition` inclui a condição `SomeOtherCondition`:

JSON

```
"MyAndCondition": {
   "Fn::And": [
      {"Fn::Equals": ["sg-mysggroup", {"Ref": "ASecurityGroup"}]},
      {"Condition": "SomeOtherCondition"}
   ]
}
```

YAML

```
MyAndCondition: !And
  - !Equals ["sg-mysggroup", !Ref "ASecurityGroup"]
  - !Condition SomeOtherCondition
```

Funções compatíveis

Não use quaisquer funções na função `Condition`. É necessário especificar uma string que seja um nome de condição.

Fn::FindInMap

A função intrínseca `Fn::FindInMap` retorna o valor correspondente às chaves em um mapa de dois níveis que é declarado na seção `Mappings`.

Declaração

JSON

```
{ "Fn::FindInMap" : [ "MapName", "TopLevelKey", "SecondLevelKey"] }
```

YAML

Sintaxe para o nome da função completo:

```
Fn::FindInMap: [ MapName, TopLevelKey, SecondLevelKey ]
```

Sintaxe para a forma resumida:

```
!FindInMap [ MapName, TopLevelKey, SecondLevelKey ]
```

> **Note**
>
> Você não pode aninhar duas instâncias de duas funções na forma abreviada.

Parâmetros

MapName

O nome lógico de um mapeamento declarado na seção de mapeamento que contém as chaves e os valores.

TopLevelKey

O nome da chave de nível superior. Seu valor é uma lista de pares de chave/valor.

SecondLevelKey

O nome da chave de segundo nível, que é definido para uma das chaves na lista atribuída ao `TopLevelKey`.

Valor de retorno:

O valor que é atribuído ao `SecondLevelKey`.

Exemplo

O exemplo a seguir mostra como usar `Fn::FindInMap` em um modelo com uma seção `Mappings` que contém um único mapa, `RegionMap`, que associa AMIs com regiões da AWS.

- O mapa tem 5 chaves de nível superior que correspondem a várias regiões da AWS.
- Cada chave de nível superior é atribuída a uma lista com duas chaves de segundo nível, `"HVM64"` e `"HVMG2"`, que correspondem à arquitetura da AMI.
- Cada uma das chaves de segundo nível é atribuída com um nome de AMI apropriada.

O modelo de exemplo a seguir contém um recurso `AWS::EC2::Instance` cuja propriedade `ImageId` é definida pela função `FindInMap`.

`MapName` é definido como o mapa de interesse, `"RegionMap"` neste exemplo. `TopLevelKey` é definido como a região em que a pilha é criada, que é determinada usando o pseudoparâmetro `"AWS::Region"`. `SecondLevelKey` é definido como a arquitetura desejada, `"HVM64"` para este exemplo.

`FindInMap` retorna a AMI atribuída à `FindInMap`. Para uma instância HVM64 em us-east-1, `FindInMap` retornaria `"ami-0ff8a91507f77f867"`.

JSON

```
{
  ...
  "Mappings" : {
    "RegionMap" : {
      "us-east-1" : {
        "HVM64" : "ami-0ff8a91507f77f867", "HVMG2" : "ami-0a584ac55a7631c0c"
      },
      "us-west-1" : {
        "HVM64" : "ami-0bdb828fd58c52235", "HVMG2" : "ami-066ee5fd4a9ef77f1"
      },
      "eu-west-1" : {
        "HVM64" : "ami-047bb4163c506cd98", "HVMG2" : "ami-0a7c483d527806435"
      },
      "ap-southeast-1" : {
        "HVM64" : "ami-08569b978cc4dfa10", "HVMG2" : "ami-0be9df32ae9f92309"
      },
      "ap-northeast-1" : {
        "HVM64" : "ami-06cd52961ce9f0d85", "HVMG2" : "ami-053cdd503598e4a9d"
      }
    }
  },

  "Resources" : {
    "myEC2Instance" : {
      "Type" : "AWS::EC2::Instance",
      "Properties" : {
        "ImageId" : {
          "Fn::FindInMap" : [
            "RegionMap",
            {
              "Ref" : "AWS::Region"
            },
            "HVM64"
          ]
        },
        "InstanceType" : "m1.small"
      }
    }
  }
}
```

YAML

```
Mappings:
  RegionMap:
    us-east-1:
      HVM64: "ami-0ff8a91507f77f867"
      HVMG2: "ami-0a584ac55a7631c0c"
    us-west-1:
      HVM64: "ami-0bdb828fd58c52235"
      HVMG2: "ami-066ee5fd4a9ef77f1"
    eu-west-1:
      HVM64: "ami-047bb4163c506cd98"
      HVMG2: "ami-31c2f645"
```

```
    ap-southeast-1:
      HVM64: "ami-08569b978cc4dfa10"
      HVMG2: "ami-0be9df32ae9f92309"
    ap-northeast-1:
      HVM64: "ami-06cd52961ce9f0d85"
      HVMG2: "ami-053cdd503598e4a9d"
Resources:
  myEC2Instance:
    Type: "AWS::EC2::Instance"
    Properties:
      ImageId: !FindInMap
        - RegionMap
        - !Ref 'AWS::Region'
        - HVM64
      InstanceType: m1.small
```

Funções compatíveis

Você pode usar as seguintes funções em uma função `Fn::FindInMap`:

- `Fn::FindInMap`
- `Ref`

Fn::GetAtt

A função intrínseca `Fn::GetAtt` retorna o valor de um atributo obtido de um recurso no modelo. Para obter mais informações sobre valores de retorno `GetAtt` para um recurso específico, consulte a documentação desse recurso na Referência de propriedades e recursos (p. 762).

Declaração

JSON

```
{ "Fn::GetAtt" : [ "logicalNameOfResource", "attributeName" ] }
```

YAML

Sintaxe para o nome da função completo:

```
Fn::GetAtt: [ logicalNameOfResource, attributeName ]
```

Sintaxe para a forma resumida:

```
!GetAtt logicalNameOfResource.attributeName
```

Parâmetros

`logicalNameOfResource`

O nome lógico (também chamado de ID lógico) do recurso que contém o atributo que você deseja.

`attributeName`

O nome do atributo específico do recurso cujo valor é desejado. Consulte a página de referência do recurso para obter mais detalhes sobre os atributos disponíveis para esse tipo de recurso.

Valor de retorno

O valor de atributo.

Exemplos

Retornar uma string

Este snippet retorna uma string que contém o nome DNS do load balancer com o nome lógico `myELB`.

JSON

```
"Fn::GetAtt" : [ "myELB" , "DNSName" ]
```

YAML

```
!GetAtt myELB.DNSName
```

Retornar várias strings

O modelo de exemplo a seguir retorna `SourceSecurityGroup.OwnerAlias` e `SourceSecurityGroup.GroupName` do load balancer com o nome lógico `myELB`.

JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Resources": {
        "myELB": {
            "Type": "AWS::ElasticLoadBalancing::LoadBalancer",
            "Properties": {
                "AvailabilityZones": [
                    "eu-west-1a"
                ],
                "Listeners": [
                    {
                        "LoadBalancerPort": "80",
                        "InstancePort": "80",
                        "Protocol": "HTTP"
                    }
                ]
            }
        },
        "myELBIngressGroup": {
            "Type": "AWS::EC2::SecurityGroup",
            "Properties": {
                "GroupDescription": "ELB ingress group",
                "SecurityGroupIngress": [
                    {
                        "IpProtocol": "tcp",
                        "FromPort": 80,
                        "ToPort": 80,
                        "SourceSecurityGroupOwnerId": {
                            "Fn::GetAtt": [
                                "myELB",
                                "SourceSecurityGroup.OwnerAlias"
                            ]
                        },
                        "SourceSecurityGroupName": {
                            "Fn::GetAtt": [
```

```
                                "myELB",
                                "SourceSecurityGroup.GroupName"
                            ]
                        }
                    }
                ]
            }
        }
    }
}
```

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  myELB:
    Type: AWS::ElasticLoadBalancing::LoadBalancer
    Properties:
      AvailabilityZones:
        - eu-west-1a
      Listeners:
        - LoadBalancerPort: '80'
          InstancePort: '80'
          Protocol: HTTP
  myELBIngressGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: ELB ingress group
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: 80
          ToPort: 80
          SourceSecurityGroupOwnerId: !GetAtt myELB.SourceSecurityGroup.OwnerAlias
          SourceSecurityGroupName: !GetAtt myELB.SourceSecurityGroup.GroupName
```

Funções compatíveis

Para o nome do recurso lógico `Fn::GetAtt`, você não pode usar funções. É necessário especificar uma string que seja um ID lógico do recurso.

Para o nome de atributo `Fn::GetAtt`, você pode usar a função `Ref`.

Fn::GetAZs

A função intrínseca `Fn::GetAZs` retorna uma matriz que lista as zonas de disponibilidade para uma região específica em ordem alfabética. Como os clientes têm acesso a diferentes zonas de disponibilidade, a função intrínseca `Fn::GetAZs` permite que os autores do modelo gravem modelos que se adaptam ao acesso do usuário da chamada. Dessa forma, não é necessário codificar uma lista completa das zonas de disponibilidade de uma região especificada.

> **Important**
>
> Para a plataforma EC2-Classic, a função `Fn::GetAZs` gera todas as zonas de disponibilidade de uma região. Para a plataforma EC2-VPC, a função `Fn::GetAZs` gera apenas as zonas de disponibilidade que têm uma sub-rede padrão, a menos que nenhuma das zonas de disponibilidade tenha uma sub-rede padrão; neste caso, todas as zonas de disponibilidade são geradas.
> Da mesma forma que a resposta do comando `describe-availability-zones`da AWS CLI, a ordem dos resultados da função `Fn::GetAZs` não é garantida e pode ser alterada quando novas zonas de disponibilidade forem adicionadas.

Permissões IAM

As permissões necessárias para usar a função `Fn::GetAZs` dependem da plataforma em que você estiver iniciando as instâncias do Amazon EC2. Para ambas as plataformas, são necessárias permissões para as ações do Amazon EC2 `DescribeAvailabilityZones` e `DescribeAccountAttributes`. Para o EC2-VPC, também são necessárias permissões para a ação do Amazon EC2`DescribeSubnets`.

Declaração

JSON

```
{ "Fn::GetAZs" : "região" }
```

YAML

Sintaxe para o nome da função completo:

```
Fn::GetAZs: região
```

Sintaxe para a forma resumida:

```
!GetAZs região
```

Parâmetros

região

O nome da região para a qual você deseja obter as zonas de disponibilidade.

É possível usar o pseudoparâmetro `AWS::Region` para especificar a região em que a pilha é criada. Especificar uma string vazia é equivalente a especificar `AWS::Region`.

Valor de retorno

A lista de zonas de disponibilidade para a região.

Exemplos

Avaliar uma região

Para esses exemplos, o CloudFormation avalia `Fn::GetAZs` para a matriz a seguir, assumindo que o usuário criou a pilha na região `us-east-1`:

```
[ "us-east-1a", "us-east-1b", "us-east-1c", "us-east-1d" ]
```

JSON

```
{ "Fn::GetAZs" : "" }
{ "Fn::GetAZs" : { "Ref" : "AWS::Region" } }
{ "Fn::GetAZs" : "us-east-1" }
```

YAML

```
Fn::GetAZs: ""
Fn::GetAZs:
```

```
  Ref: "AWS::Region"
Fn::GetAZs: us-east-1
```

Especificar a zona de disponibilidade de uma sub-rede

O exemplo a seguir usa `Fn::GetAZs` para especificar uma zona de disponibilidade da sub-rede:

JSON

```
"mySubnet" : {
  "Type" : "AWS::EC2::Subnet",
  "Properties" : {
    "VpcId" : {
      "Ref" : "VPC"
    },
    "CidrBlock" : "10.0.0.0/24",
    "AvailabilityZone" : {
      "Fn::Select" : [
        "0",
        {
          "Fn::GetAZs" : ""
        }
      ]
    }
  }
}
```

YAML

```
mySubnet:
  Type: "AWS::EC2::Subnet"
  Properties:
    VpcId:
      !Ref VPC
    CidrBlock: 10.0.0.0/24
    AvailabilityZone:
      Fn::Select:
        - 0
        - Fn::GetAZs: ""
```

Funções aninhadas com YAML em forma abreviada

Os exemplos a seguir mostram os padrões válidos para usar as funções intrínsecas aninhadas usando a forma abreviada YAML. Você não pode aninhar funções em forma abreviada consecutivamente, portanto, um padrão como `!GetAZs !Ref` é inválido.

YAML

```
AvailabilityZone: !Select
  - 0
  - !GetAZs
    Ref: 'AWS::Region'
```

YAML

```
AvailabilityZone: !Select
```

```
  - 0
  - Fn::GetAZs: !Ref 'AWS::Region'
```

Funções compatíveis

Use a função `Ref` na função `Fn::GetAZs`.

Fn::ImportValue

A função intrínseca `Fn::ImportValue` retorna o valor de uma saída exportada (p. 354) por outra pilha. Normalmente, você usa essa função para criar referências de pilha cruzada (p. 404). Nos trechos do modelo de exemplo a seguir, Stack A exporta valores do security group da VPC, e Stack B os importa.

> **Note**
>
> As seguintes restrições se aplicam a referências de pilha cruzada:
>
> - Para cada conta da AWS, os nomes de `Export` devem ser exclusivos dentro de uma região.
> - Você não pode criar referências de pilha cruzada entre regiões. Você pode usar a função intrínseca `Fn::ImportValue` para importar apenas valores que foram exportados dentro da mesma região.
> - Para saídas, o valor da propriedade `Name` de um `Export` não pode usar funções `Ref` ou `GetAtt` que dependam de um recurso.
>
> Da mesma maneira, a função `ImportValue` não pode incluir funções `Ref` ou `GetAtt` que dependam de um recurso.
> - Você não poderá excluir uma pilha, se outra pilha referenciar uma das saídas.
> - Você não pode modificar nem remover um valor de saída referenciado por outra pilha.

JSON

Exportação da Stack A

```
"Outputs" : {
  "PublicSubnet" : {
    "Description" : "The subnet ID to use for public web servers",
    "Value" :  { "Ref" : "PublicSubnet" },
    "Export" : { "Name" : {"Fn::Sub": "${AWS::StackName}-SubnetID" }}
  },
  "WebServerSecurityGroup" : {
    "Description" : "The security group ID to use for public web servers",
    "Value" :  { "Fn::GetAtt" : ["WebServerSecurityGroup", "GroupId"] },
    "Export" : { "Name" : {"Fn::Sub": "${AWS::StackName}-SecurityGroupID" }}
  }
}
```

YAML

Exportação da Stack A

```
Outputs:
  PublicSubnet:
    Description: The subnet ID to use for public web servers
    Value:
      Ref: PublicSubnet
    Export:
```

```
      Name:
        'Fn::Sub': '${AWS::StackName}-SubnetID'
  WebServerSecurityGroup:
    Description: The security group ID to use for public web servers
    Value:
      'Fn::GetAtt':
        - WebServerSecurityGroup
        - GroupId
    Export:
      Name:
        'Fn::Sub': '${AWS::StackName}-SecurityGroupID'
```

JSON

Importação da Stack B

```
"Resources" : {
  "WebServerInstance" : {
    "Type" : "AWS::EC2::Instance",
    "Properties" : {
      "InstanceType" : "t2.micro",
      "ImageId" : "ami-a1b23456",
      "NetworkInterfaces" : [{
        "GroupSet" : [{"Fn::ImportValue" : {"Fn::Sub" : "${NetworkStackNameParameter}-
SecurityGroupID"}}],
        "AssociatePublicIpAddress" : "true",
        "DeviceIndex" : "0",
        "DeleteOnTermination" : "true",
        "SubnetId" : {"Fn::ImportValue" : {"Fn::Sub" : "${NetworkStackNameParameter}-
SubnetID"}}
      }]
    }
  }
}
```

YAML

Importação da Stack B

```
Resources:
  WebServerInstance:
    Type: 'AWS::EC2::Instance'
    Properties:
      InstanceType: t2.micro
      ImageId: ami-a1b23456
      NetworkInterfaces:
        - GroupSet:
            - !ImportValue
              'Fn::Sub': '${NetworkStackNameParameter}-SecurityGroupID'
          AssociatePublicIpAddress: 'true'
          DeviceIndex: '0'
          DeleteOnTermination: 'true'
          SubnetId: !ImportValue
            'Fn::Sub': '${NetworkStackNameParameter}-SubnetID'
```

Declaração

JSON

```
{ "Fn::ImportValue" : sharedValueToImport }
```

YAML

Você pode usar o nome completo da função:

```
Fn::ImportValue: sharedValueToImport
```

Como alternativa, você pode usar a forma abreviada:

```
!ImportValue sharedValueToImport
```

> **Important**
>
> Você não pode usar a forma abreviada de `!ImportValue` quando ele contém um `!Sub`. O exemplo a seguir é válido para o AWS CloudFormation, mas não para o YAML:
>
> ```
> !ImportValue
>   !Sub "${NetworkStack}-SubnetID"
> ```
>
> Em vez disso, você deve usar o nome completo da função, por exemplo:
>
> ```
> Fn::ImportValue:
>   !Sub "${NetworkStack}-SubnetID"
> ```

Parâmetros

sharedValueToImport

O valor de saída da pilha que você deseja importar.

Valor de retorno

O valor de saída da pilha.

Exemplo

JSON

```
{ "Fn::ImportValue" : {"Fn::Sub": "${NetworkStackNameParameter}-SubnetID" } }
```

YAML

```
Fn::ImportValue:
  !Sub "${NetworkStackName}-SecurityGroupID"
```

Funções compatíveis

Você pode usar as seguintes funções na função `Fn::ImportValue`. O valor dessas funções não depende de um recurso.

- `Fn::Base64`

- Fn::FindInMap
- Fn::If
- Fn::Join
- Fn::Select
- Fn::Split
- Fn::Sub
- Ref

Fn::Join

A função intrínseca Fn::Join anexa um conjunto de valores em um único valor, separados pelo delimitador especificado. Se um delimitador é a string vazia, o conjunto de valores é concatenado sem delimitador.

Declaração

JSON

```
{ "Fn::Join" : [ "delimiter", [ comma-delimited list of values ] ] }
```

YAML

Sintaxe para o nome da função completo:

```
Fn::Join: [ delimiter, [ comma-delimited list of values ] ]
```

Sintaxe para a forma resumida:

```
!Join [ delimiter, [ comma-delimited list of values ] ]
```

Parâmetros

delimitador

O valor que você deseja que ocorra entre estilhaços. O delimitador ocorrerá apenas entre estilhaços. Não encerrará o valor final.

ListOfValues

A lista de valores a ser combinados.

Valor de retorno

A string combinada.

Exemplos

Unir uma matriz de strings simples

O seguinte exemplo retorna: "a:b:c".

JSON

```
"Fn::Join" : [ ":", [ "a", "b", "c" ] ]
```

YAML

```
!Join [ ":", [ a, b, c ] ]
```

Unir usando a função Ref com parâmetros

O exemplo a seguir usa `Fn::Join` para construir um valor de string. Ele usa a função `Ref` com o parâmetro `AWS::Partition` e o pseudoparâmetro `AWS::AccountId`.

JSON

```
{
  "Fn::Join": [
    "", [
      "arn:",
      {
        "Ref": "AWS::Partition"
      },
      ":s3:::elasticbeanstalk-*-",
      {
        "Ref": "AWS::AccountId"
      }
    ]
  ]
}
```

YAML

```
!Join
  - ''
  - - 'arn:'
    - !Ref AWS::Partition
    - ':s3:::elasticbeanstalk-*-'
    - !Ref AWS::AccountId
```

> **Note**
>
> Consulte também a função `Fn::Sub` (p. 6747) para funcionalidade semelhante.

Funções compatíveis

No delimitador `Fn::Join`, não é possível usar nenhuma função. É necessário especificar um valor de string.

Para a lista de valores `Fn::Join`, use as seguintes funções:

- `Fn::Base64`
- `Fn::FindInMap`
- `Fn::GetAtt`
- `Fn::GetAZs`
- `Fn::If`
- `Fn::ImportValue`

- Fn::Join
- Fn::Split
- Fn::Select
- Fn::Sub
- Ref

Fn::Select

A função intrínseca Fn::Select retorna um único objeto a partir de uma lista de objetos por índice.

> Important
>
> O Fn::Select não verifica valores nulos ou se o índice está fora dos limites da série. Ambas as condições resultarão em um erro de pilha, portanto, você deverá ter certeza de que o índice escolhido é válida e se a lista contém valores não nulos.

Declaração

JSON

```
{ "Fn::Select" : [ index, listOfObjects ] }
```

YAML

Sintaxe para o nome da função completo:

```
Fn::Select: [ index, listOfObjects ]
```

Sintaxe para a forma resumida:

```
!Select [ index, listOfObjects ]
```

Parâmetros

índice

> O índice do objeto para recuperar. Isto deve ser um valor de zero a N-1, onde N representa o número de elementos na série.

listOfObjects

> A lista de objetos para selecionar. Essa lista não deve ser nula nem pode ter entradas nulas.

Valor de retorno

O objeto selecionado.

Exemplos

Exemplo básico

O seguinte exemplo retorna: "grapes".

JSON

```
{ "Fn::Select" : [ "1", [ "apples", "grapes", "oranges", "mangoes" ] ] }
```

YAML

```
!Select [ "1", [ "apples", "grapes", "oranges", "mangoes" ] ]
```

Tipo de parâmetros da lista delimitada por vírgula

Você pode usar `Fn::Select` para selecionar um objeto a partir de um parâmetro `CommaDelimitedList`. Você pode usar um parâmetro `CommaDelimitedList` para combinar os valores dos parâmetros relacionados, que reduz o número total de parâmetros no seu modelo. Por exemplo, o parâmetro a seguir especifica uma lista delimitada por vírgulas de três blocos CIDR:

JSON

```
"Parameters" : {
  "DbSubnetIpBlocks": {
    "Description": "Comma-delimited list of three CIDR blocks",
    "Type": "CommaDelimitedList",
      "Default": "10.0.48.0/24, 10.0.112.0/24, 10.0.176.0/24"
  }
}
```

YAML

```
Parameters:
  DbSubnetIpBlocks:
    Description: "Comma-delimited list of three CIDR blocks"
    Type: CommaDelimitedList
    Default: "10.0.48.0/24, 10.0.112.0/24, 10.0.176.0/24"
```

Para especificar um dos três blocos de CIDR, use `Fn::Select` na seção Recursos do mesmo modelo, conforme mostrado no snippet de amostra a seguir:

JSON

```
"Subnet0": {
  "Type": "AWS::EC2::Subnet",
    "Properties": {
      "VpcId": { "Ref": "VPC" },
      "CidrBlock": { "Fn::Select" : [ "0", {"Ref": "DbSubnetIpBlocks"} ] }
    }
}
```

YAML

```
Subnet0:
  Type: "AWS::EC2::Subnet"
  Properties:
    VpcId: !Ref VPC
    CidrBlock: !Select [ 0, !Ref DbSubnetIpBlocks ]
```

Funções aninhadas com YAML em forma abreviada

Os exemplos a seguir mostram os padrões válidos para usar as funções intrínsecas aninhadas com a forma abreviada `!Select`. Você não pode aninhar funções em forma abreviada consecutivamente, portanto, um padrão como `!GetAZs !Ref` é inválido.

YAML

```
AvailabilityZone: !Select
  - 0
  - !GetAZs
    Ref: 'AWS::Region'
```

YAML

```
AvailabilityZone: !Select
  - 0
  - Fn::GetAZs: !Ref 'AWS::Region'
```

Funções compatíveis

Para obter o valor de índice de `Fn::Select`, você pode usar as funções `Ref` e `Fn::FindInMap`.

Para obter a lista de objetos `Fn::Select`, use as funções a seguir:

- `Fn::FindInMap`
- `Fn::GetAtt`
- `Fn::GetAZs`
- `Fn::If`
- `Fn::Split`
- `Ref`

Fn::Split

Para dividir uma string em uma lista de valores de string, para que você possa selecionar um elemento na lista de strings resultante, use a função intrínseca `Fn::Split`. Especifique o local das divisões com um delimitador, como `,` (uma vírgula). Após dividir uma string, use a função `Fn::Select` (p. 6743) para escolher um elemento específico.

Por exemplo, se uma string de IDs de sub-rede delimitada por vírgulas for importada para seu modelo de pilha, você poderá dividir a string em cada vírgula. Na lista de IDs de sub-rede, use a função intrínseca `Fn::Select` para especificar um ID de sub-rede para um recurso.

Declaração

JSON

```
{ "Fn::Split" : [ "delimiter", "source string" ] }
```

YAML

Sintaxe para o nome da função completo:

```
Fn::Split: [ delimiter, source string ]
```

Sintaxe para a forma resumida:

```
!Split [ delimiter, source string ]
```

Parâmetros

Você deve especificar os dois parâmetros.

delimitador

Um valor de string que determina onde a string de origem será dividida.

string de origem

O valor de string que você deseja dividir.

Valor de retorno

Uma lista de valores de string.

Exemplos

Os exemplos a seguir demonstram o comportamento da função `Fn::Split`.

Lista simples

O exemplo a seguir divide uma string em cada barra vertical (`|`). A função retorna `["a", "b", "c"]`.

JSON

```
{ "Fn::Split" : [ "|" , "a|b|c" ] }
```

YAML

```
!Split [ "|" , "a|b|c" ]
```

Lista com valores de string vazios

Se você dividir uma string com delimitadores consecutivos, a lista resultante incluirá uma string vazia. O exemplo a seguir mostra como uma string com dois delimitadores consecutivos e um delimitador anexado é dividida. A função retorna `["a", "", "c", ""]`.

JSON

```
{ "Fn::Split" : [ "|" , "a||c|" ] }
```

YAML

```
!Split [ "|" , "a||c|" ]
```

Divisão de um valor de saída importado

O exemplo a seguir divide um valor de saída importado e seleciona o terceiro elemento na lista resultante de IDs de sub-rede, conforme especificado pela função `Fn::Select`.

JSON

```
{ "Fn::Select" : [ "2", { "Fn::Split": [",", {"Fn::ImportValue": "AccountSubnetIDs"}]}] }
```

YAML

```
!Select [2, !Split [",", !ImportValue AccountSubnetIDs]]
```

Funções compatíveis

No delimitador `Fn::Split`, não é possível usar nenhuma função. É necessário especificar um valor de string.

Para a lista de valores `Fn::Split`, use as seguintes funções:

- `Fn::Base64`
- `Fn::FindInMap`
- `Fn::GetAtt`
- `Fn::GetAZs`
- `Fn::If`
- `Fn::ImportValue`
- `Fn::Join`
- `Fn::Select`
- `Fn::Sub`
- `Ref`

Fn::Sub

A função intrínseca `Fn::Sub` substitui variáveis em uma sequência de entrada por valores especificados por você. Em seus modelos, você pode usar essa função para construir comandos ou saídas que incluem valores que não estão disponíveis até que você crie ou atualize uma pilha.

Declaração

As seções a seguir mostram a sintaxe da função.

JSON

```
{ "Fn::Sub" : [ String, { Var1Name: Var1Value, Var2Name: Var2Value } ] }
```

Se você estiver substituindo apenas parâmetros do modelo, IDs lógicos de recursos ou atributos de recursos no parâmetro *String*, não especifique um mapa de variáveis.

```
{ "Fn::Sub" : String }
```

YAML

Sintaxe para o nome da função completo:

```
Fn::Sub:
  - String
  - Var1Name: Var1Value
    Var2Name: Var2Value
```

Sintaxe para a forma resumida:

```
!Sub
  - String
  - Var1Name: Var1Value
    Var2Name: Var2Value
```

Se você estiver substituindo apenas parâmetros do modelo, IDs lógicos de recursos ou atributos de recursos no parâmetro *String*, não especifique um mapa de variáveis.

Sintaxe para o nome da função completo:

```
Fn::Sub: String
```

Sintaxe para a forma resumida:

```
!Sub String
```

Parâmetros

`String`

Uma sequência com variáveis que o AWS CloudFormation substitui por seus valores associados em runtime. Escreva variáveis como `${MyVarName}`. As variáveis podem ser nomes de parâmetros do modelo, IDs lógicos de recursos, atributos de recursos ou uma variável em um mapa de chave/valor. Se você especificar apenas nomes de parâmetros do modelo, IDs lógicos de recursos e atributos de recursos, não especifique um mapa de chave/valor.

Se você especificar nomes de parâmetros do modelo ou IDs lógicas de recursos, como `${InstanceTypeParameter}`, o CloudFormation retornará os mesmos valores como se você tivesse usado a função intrínseca `Ref`. Se você especificar atributos de recursos, como `${MyInstance.PublicIp}`, o CloudFormation retornará os mesmos valores como se você tivesse usado a função intrínseca `Fn::GetAtt`.

Para escrever um cifrão e chaves (`${}`) literalmente, adicione um ponto de exclamação (`!`) após a chave de abertura, como `${!Literal}`. O CloudFormation resolve esse texto como `${Literal}`.

`VarName`

O nome de uma variável que você incluiu no parâmetro `String`.

`VarValue`

O valor que o CloudFormation substitui para o nome da variável associada em runtime.

Valor de retorno

O CloudFormation retorna a sequência original, substituindo os valores de todas as variáveis.

Exemplos

Os exemplos a seguir demonstram como usar a função `Fn::Sub`.

Fn::Sub com um mapeamento

O exemplo a seguir usa um mapeamento para substituir a variável `${Domain}` pelo valor resultante da função `Ref`.

JSON

```
{ "Fn::Sub": [ "www.${Domain}", { "Domain": {"Ref" : "RootDomainName" }} ]}
```

YAML

```
Name: !Sub
  - www.${Domain}
  - { Domain: !Ref RootDomainName }
```

Fn::Sub sem um mapeamento

O exemplo a seguir usa Fn::Sub com os pseudoparâmetros `AWS::Region` e `AWS::AccountId` e o ID lógico de recurso `vpc` para criar um Nome de recurso da Amazon (ARN) para uma VPC.

JSON

```
{ "Fn::Sub": "arn:aws:ec2:${AWS::Region}:${AWS::AccountId}:vpc/${vpc}" }
```

YAML

```
!Sub 'arn:aws:ec2:${AWS::Region}:${AWS::AccountId}:vpc/${vpc}'
```

Comandos UserData

O exemplo a seguir usa `Fn::Sub` para substituir os pseudoparâmetros `AWS::StackName` e `AWS::Region` do nome real da pilha e da região em tempo de execução.

JSON

Para facilitar a leitura, o exemplo do JSON usa a função `Fn::Join` para separar cada comando, em vez de especificar todo o script de dados do usuário em um único valor de sequência.

```
"UserData": { "Fn::Base64": { "Fn::Join": ["\n", [
  "#!/bin/bash -xe",
  "yum update -y aws-cfn-bootstrap",
  { "Fn::Sub": "/opt/aws/bin/cfn-init -v --stack ${AWS::StackName} --resource LaunchConfig
 --configsets wordpress_install --region ${AWS::Region}" },
  { "Fn::Sub": "/opt/aws/bin/cfn-signal -e $? --stack ${AWS::StackName} --resource
 WebServerGroup --region ${AWS::Region}" }]]
}}
```

YAML

O exemplo do YAML usa um bloco literal para especificar o script de dados do usuário.

```
UserData:
  Fn::Base64:
    !Sub |
      #!/bin/bash -xe
      yum update -y aws-cfn-bootstrap
      /opt/aws/bin/cfn-init -v --stack ${AWS::StackName} --resource LaunchConfig --
configsets wordpress_install --region ${AWS::Region}
      /opt/aws/bin/cfn-signal -e $? --stack ${AWS::StackName} --resource WebServerGroup --
region ${AWS::Region}
```

Funções compatíveis

No parâmetro `String`, não é possível usar funções. É necessário especificar um valor de string.

Para os parâmetros `VarName` e `VarValue`, você pode usar as seguintes funções:

- `Fn::Base64`
- `Fn::FindInMap`
- `Fn::GetAtt`
- `Fn::GetAZs`
- `Fn::If`
- `Fn::ImportValue`
- `Fn::Join`
- `Fn::Select`
- `Ref`

Fn::Transform

A função intrínseca `Fn::Transform` especifica uma macro para realizar o processamento personalizado em parte de um modelo de pilha. Macros permitem realizar o processamento personalizado em modelos, desde ações simples, como operações de localizar e substituir, até transformações extensas de modelos inteiros. Para obter mais informações, consulte Usar macros do AWS CloudFormation para realizar processamento personalizado em modelos (p. 647).

Você também pode usar `Fn::Transform` para chamar a transformação `Transformação AWS::Include (p. 6758)`, que é uma macro hospedado pelo AWS CloudFormation.

Declaração

JSON

Sintaxe para o nome da função completo:

```
{
    "Fn::Transform": {
        "Name": "macro name",
        "Parameters": {
            "Key": "value"
        }
```

```
    }
}
```

Sintaxe para a forma resumida:

```
{
    "Transform": {
        "Name": "macro name",
        "Parameters": {
            "Key": "value"
        }
    }
}
```

YAML

Sintaxe para o nome da função completo:

```
Fn::Transform:
  Name : macro name
  Parameters :
          Key : value
```

Sintaxe para a forma resumida:

```
Transform:
  Name: macro name
  Parameters:
    Key: value
```

Parâmetros

Nome

O nome da macro da qual você deseja realizar o processamento.

Parâmetros

Os parâmetros de lista, especificados como pares de chave/valor a serem transferidos para a macro.

Valor de retorno

O trecho de modelo processado a ser incluído no modelo de pilha.

Exemplos

O exemplo a seguir chama a transformação `AWS::Include`, especificando que o local de onde recuperar um trecho de modelo é transmitido no parâmetro `InputValue`.

JSON

```
{
   "Fn::Transform" : {
       "Name" : "AWS::Include",
       "Parameters" : {
```

```
            "Location" : { "Ref" : "InputValue" }
        }
    }
}
```

YAML

```
'Fn::Transform':
    Name: 'AWS::Include'
    Parameters: {Location: {Ref: InputValue}}
```

O exemplo a seguir chama a transformação `AWS::Include`, especificando que o local de onde recuperar um trecho de modelo está localizado no mapeamento `RegionMap`, sob a chave `us-east-1` e a chave aninhada `s3Location`.

JSON

```
{
   "Fn::Transform" : {
       "Name" : "AWS::Include",
       "Parameters" : {
           "Location" : {"Fn::FindInMap" : ["RegionMap", "us-east-1", "s3Location"] }
        }
    }
}
```

YAML

```
'Fn::Transform':
    Name: 'AWS::Include'
    Parameters: {Location: {'Fn::FindInMap': [RegionMap, us-east-1, s3Location]}}
```

Funções compatíveis

Nenhuma. O CloudFormation transmite qualquer chamada de função intrínseca incluída em `Fn::Transform` à macro especificada como strings de literal. Para obter mais informações, consulte Interface de funções de macro do AWS CloudFormation (p. 649).

Ref

A função intrínseca `Ref` retorna o valor do recurso ou parâmetro especificado.

- Quando você especifica um nome lógico do parâmetro, ele retorna o valor do parâmetro.
- Quando você especifica o nome lógico de um recurso, ele retorna um valor que você pode geralmente usa para fazer referência a esse recurso, como uma ID física (p. 351).

Quando você estiver declarando um recurso em um modelo e precisar especificar outro recurso de modelo pelo nome, use o `Ref` para fazer referência àquele outro recurso. Em geral, o `Ref` retorna o nome do recurso. Por exemplo, uma referência a um AWS::AutoScaling::AutoScalingGroup retorna o nome do recurso do grupo de Auto Scaling.

Para alguns recursos, um identificador é retornado com outro significado importante no contexto do recurso. Um recurso AWS::EC2::EIP, por exemplo, retorna o endereço IP, e um AWS::EC2::Instance retorna o ID da instância.

> **Tip**
>
> Você também pode usar `Ref` para adicionar valores a mensagens de saída.

Para obter mais informações sobre valores de retorno `Ref` para um recurso ou uma propriedade específica, consulte a documentação desse recurso ou dessa propriedade na Referência de propriedades e recursos (p. 762).

Declaração

JSON

```
{ "Ref" : "logicalName" }
```

YAML

Sintaxe para o nome da função completo:

```
Ref: logicalName
```

Sintaxe para a forma resumida:

```
!Ref logicalName
```

Parâmetros

logicalName

O nome lógico do recurso ou parâmetro que você deseja cancelar.

Valor de retorno

O ID físico do recurso ou o valor do parâmetro.

Exemplo

A seguinte declaração de recurso para um endereço IP elástico precisa do ID de instância de uma instância EC2 e usa a função `Ref` para especificar o ID da instância do recurso MyEC2Instance:

JSON

```
"MyEIP" : {
   "Type" : "AWS::EC2::EIP",
   "Properties" : {
      "InstanceId" : { "Ref" : "MyEC2Instance" }
   }
}
```

YAML

```
MyEIP:
```

```
  Type: "AWS::EC2::EIP"
  Properties:
    InstanceId: !Ref MyEC2Instance
```

Funções compatíveis

Não use quaisquer funções na função `Ref`. É necessário especificar uma string que seja um ID lógico do recurso.

Referência de pseudoparâmetros

Os pseudoparâmetros são parâmetros que são predefinidos pelo AWS CloudFormation. Você não os declará-las em seu modelo. Use-os da mesma forma como usaria um parâmetro, como o argumento para a função `Ref`.

Exemplo

O trecho a seguir atribui o valor do pseudoparâmetro `AWS::Region` a um valor de saída:

JSON

```
"Outputs" : {
   "MyStacksRegion" : { "Value" : { "Ref" : "AWS::Region" } }
}
```

YAML

```
Outputs:
  MyStacksRegion:
    Value: !Ref "AWS::Region"
```

AWS::AccountId

Apresenta o ID da conta da AWSna qual a pilha está sendo criada, como `123456789012`.

AWS::NotificationARNs

Retorna a lista de Nomes de recursos da Amazon (ARNs) de notificação para a pilha atual.

Para obter um único ARN da lista, use Fn::Select (p. 6743).

JSON

```
"myASGrpOne" : {
   "Type" : "AWS::AutoScaling::AutoScalingGroup",
   "Version" : "2009-05-15",
   "Properties" : {
      "AvailabilityZones" : [ "us-east-1a" ],
```