Série de certificação GCP: 3.7 Implantando um aplicativo usando o Deployment Manager

Prashanta Paudel

26 de outubro de 2018 · 6 minutos de leitura

O gerenciador de implementação é um local central para a implementação de máquinas e instâncias usando o mercado ou usando qualquer outra instância do modelo usando um arquivo .yaml.

O Deployment Manager permite gerenciar sua infraestrutura usando configurações declarativas.

Neste codelab, você usará o Deployment Manager para configurar regras de firewall de rede e iniciar uma instância do Google Compute Engine. Você configurará um script de inicialização que iniciará um servidor da Web e gerará uma página da Web contendo metadados da instância.

Colocar a infraestrutura em um modelo e usá-las repetidamente vem do conceito de “infraestrutura como um código”.

É ainda melhor ter um código e uma infraestrutura juntos para que qualquer um possa construí-lo novamente.

Algumas das ferramentas para usar a infraestrutura como código são

1. Ansible
2. Terraform

Mas a definição no modelo pode não corresponder ao que está disponível na plataforma de provedores de nuvem.

À medida que os produtos de cada fornecedor de nuvem estão mudando rapidamente, eles fornecem sua própria ferramenta para definir a infraestrutura de nuvem usando uma linguagem declarativa que pode ser armazenada externamente.

O gerente do Google Cloud Deployment pode automatizar a configuração de todos os seus recursos do GCP.

Noções básicas sobre o Gerenciador de Implantação (ref: https://medium.com/google-cloud/2018-google-deployment-manager-5ebb8759a122)

O Deployment Manager tem o conceito de uma *implantação* , que é uma coleção de recursos do GCP que formam uma unidade lógica e que são implantados juntos. Os recursos de uma implantação podem ser qualquer coisa disponível no GCP: VMs, endereços IP, servidores de banco de dados, clusters do Kubernetes, etc.

Para criar uma implantação, você precisa de uma *configuração de implantação* , que é um arquivo YAML contendo a definição dos recursos. Para se ter uma ideia, poderia ser o seguinte:

```
resources:
- name: example-vm
  type: compute.v1.instance
  properties:
    zone: europe-west1-b
    machineType: zones/europe-west1-b/machineTypes/n1-
standard-1
    disks:
       ...
```

Cada recurso listado sempre tem um *tipo* , que é o tipo de recurso que será criado (uma VM, um endereço IP, etc.), um *nome* e *propriedades que* descrevem com quais parâmetros o recurso deve ser criado.

Um conceito importante, em comparação com o Ansible, é que, quando você cria uma implantação, ela existe como um recurso em si dentro do GCP. Se você alterar posteriormente a configuração da implementação (modificando o arquivo YAML, por exemplo) e executar o comando update, o gerenciador de implementação compara a nova configuração com o que foi implementado antes e fará apenas as mudanças necessárias. Não apenas menos trabalho precisa ser feito, mas, mais importante, também garante que todos os recursos removidos da configuração também sejam removidos da infraestrutura do GCP.

Você pode encontrar muitos exemplos de configuração de implantação no projeto de amostras do Gerenciador de implantação do Google no GitHub. Geralmente é mais útil do que a documentação oficial , descobrir exatamente quais parâmetros você deve definir.

O arquivo cluster.yaml no repositório do GitHub especifica mais alguns parâmetros, como distribuir os nós em duas zonas e ativar a "atualização automática" para os nós do cluster.

Você pode instanciar esta implantação usando o comando "create":

```
$ gcloud deployment-manager deployments create example-
cluster --config cluster-1/cluster.yaml
```

Isso cria uma implantação chamada "cluster de exemplo", usando a definição encontrada no cluster 1 / cluster.yaml. Posteriormente, você pode atualizar o arquivo yaml e atualizar a implantação implantada com o comando "update":

```
$ gcloud deployment-manager deployments update example-
cluster --config cluster-1/cluster.yaml
```

Esse comando compara a implantação implantada com a nova definição e executa somente as alterações necessárias (por exemplo, cria um novo recurso adicionado).

Por enquanto, exclua o cluster. Vamos recriá-lo novamente mais tarde.

```
$ gcloud deployment-manager deployments delete example-
cluster
```

Modelo de Infraestrutura

Arquivos Yaml criados em formato padrão realmente codificam sua infraestrutura.

O Deployment Manager suporta

- Scripts em Python
- Jinja 2 como linguagem de templates.

Fundamentos do Deployment Manager

Configuração

Uma configuração descreve todos os recursos que você deseja para uma única implantação. Uma configuração é um arquivo escrito na sintaxe YAML que lista cada um dos recursos que você deseja criar e suas respectivas propriedades de recurso. Uma configuração deve conter uma `resources:` seção seguida da lista de recursos a serem criados.

Cada recurso deve conter três componentes:

- `name` - A cadeia definida pelo usuário para identificar este recurso, como `my-vm` , `project-data-disk` , `the-test-network` .
- `type` - O tipo de recurso a ser implantado, tal como `compute.v1.instance` , `compute.v1.disk` . Os tipos de recurso base são descritos e listados na documentação Tipos de Recursos Suportados .
- `properties` - Os parâmetros para este tipo de recurso. Eles devem corresponder às propriedades para o tipo tais como `zone: asia-east1-a` , `boot: true` .

```
resources:
- name: the-first-vm
  type: compute.v1.instance
  properties:
    zone: us-central1-a
    machineType:
https://www.googleapis.com/compute/v1/projects/myproject/zon
es/us-central1-a/machineTypes/f1-micro
    disks:
    - deviceName: boot
      type: PERSISTENT
      boot: true
      autoDelete: true
      initializeParams:
```

```
        sourceImage:
https://www.googleapis.com/compute/v1/projects/debian-
cloud/global/images/debian-7-wheezy-v20150423
    networkInterfaces:
    - network:
https://www.googleapis.com/compute/v1/projects/myproject/glo
bal/networks/default
      accessConfigs:
      - name: External NAT
        type: ONE_TO_ONE_NAT
```

Modelos

Uma configuração pode conter modelos, que são partes essenciais do arquivo de configuração que foi abstraído em blocos de construção individuais. Depois de criar um modelo, você poderá reutilizá-lo nas implantações conforme necessário. Da mesma forma, se você se encontrar reconfigurando configurações que compartilham propriedades muito semelhantes, você poderá abstrair as partes compartilhadas em modelos. Os modelos são muito mais flexíveis do que os arquivos de configuração individuais e destinam-se a oferecer suporte à fácil portabilidade entre as implantações.

Um modelo é um arquivo escrito em Python ou Jinja2. O sistema do Deployment Manager interpretará cada modelo recursivamente e embutirá os resultados com o arquivo de configuração. Como tal, a interpretação de cada modelo deve resultar na mesma sintaxe YAML para os recursos definidos acima para o próprio arquivo de configuração.

Para criar um modelo simples, leia Criando um modelo básico .

As configurações são descritas como totalmente expandidas ou não expandidas. Uma configuração totalmente expandida descreve todos os recursos e propriedades da implantação, incluindo qualquer conteúdo de arquivos de modelo importados. Por exemplo, você forneceria uma configuração não expandida que usa um modelo da seguinte forma:

```
imports:
- path: vm_template.jinja

resources:
- name: vm-instance
```

```
    type: vm_template.jinja
    properties:
      zone: us-central1-a
      project: myproject
```

Uma vez expandido, seu arquivo de configuração conteria o conteúdo de todos os seus modelos, assim:

```
resources:
- name: the-first-vm
  type: compute.v1.instance
  properties:
    zone: us-central1-a
    machineType:
https://www.googleapis.com/compute/v1/projects/myproject/zon
es/us-central1-a/machineTypes/f1-micro
    disks:
    - deviceName: boot
      type: PERSISTENT
      boot: true
      autoDelete: true
      initializeParams:
        sourceImage:
https://www.googleapis.com/compute/v1/projects/debian-
cloud/global/images/debian-7-wheezy-v20150423
        networkInterfaces:
        - network:
https://www.googleapis.com/compute/v1/projects/myproject/glo
bal/networks/default
    accessConfigs:
    - name: External NAT
      type: ONE_TO_ONE_NAT
```

Desdobramento, desenvolvimento

Uma implantação é uma coleção de recursos que são implantados e gerenciados juntos, usando uma configuração.

Criando um modelo básico

Um arquivo de configuração básico pode ser suficiente para cargas de trabalho simples, mas para arquiteturas mais complexas ou para configurações que você pretende reutilizar, é possível dividir sua configuração em modelos.

Um modelo é um arquivo separado que é importado e usado como um tipo em uma configuração. Você pode usar quantos modelos desejar em uma configuração e o Deployment Manager expandirá recursivamente todos os modelos importados para criar sua configuração completa.

Os modelos permitem separar sua configuração em partes diferentes que você pode usar e reutilizar em diferentes implantações. Os modelos podem ser tão generalizados ou específicos quanto você precisar. Com modelos, você também pode aproveitar recursos como propriedades de modelo, variáveis de ambiente, módulos e outras funcionalidades de modelo para criar arquivos dinâmicos de configuração e modelo.

Criando uma configuração básica

Um arquivo de configuração define todos os recursos do Google Cloud Platform que formam uma implantação. Você deve ter um arquivo de configuração para criar uma implementação. Um arquivo de configuração deve ser escrito na sintaxe YAML.

Estrutura do arquivo de configuração

Um arquivo de configuração é gravado no formato YAML e possui a seguinte estrutura:

```
imports:
- path: path/to/template.jinja
  name: my-template
- path: path/to/another/template.py


resources:
  - name: NAME_OF_RESOURCE
    type: TYPE_OF_RESOURCE
    properties:
      property-a: value
      property-b: value
      ...
      property-z: value
  - name: NAME_OF_RESOURCE
    type: TYPE_OF_RESOURCE
    properties:
      property-a: value
      property-b: value
      ...
      property-z: value
```

Cada uma das seções define uma parte diferente da implementação:

- As `imports` seções são uma lista de arquivos de modelo que serão usados pela configuração. O Deployment Manager expande recursivamente todos os modelos importados para formar sua configuração final.
- A `resources` seção é uma lista de recursos que compõem essa implantação. Um recurso pode ser:
- Um tipo base gerenciado pelo Google, como uma instância de VM do Compute Engine.
- Um modelo importado
- Um tipo composto .
- Um provedor de tipo .

Você também pode incluir outras seções opcionais, como as seções `outputs` e `metadata` . A `outputs` seção permite expor dados de seus modelos e configurações como saídas de outros modelos na mesma implantação para consumir ou saídas para os usuários finais, enquanto a `metadata` seção permite usar outros recursos, como definir dependências explícitas entre recursos.

No mínimo, uma configuração deve sempre declarar a `resources` seção, seguida por uma lista de recursos. Outras seções são opcionais.

Recursos:

- nome: [nome do recurso]

tipo: compute.v1.instance

Propriedades:

- - - - -

- - - - - -

- - - - -