Kustomize 转换器说明与使用

AUTHOR: 彭玲 TIME: 2021/11/9

简介

Kustomize 提供了一种自定义 Kubernetes 资源配置的解决方案，通过 kustomization 文件定制 K8s 对象。

从 1.14 版本开始，`kubectl` 也开始支持使用 kustomization 文件来管理 Kubernetes 对象。要查看包含 kustomization 文件的目录中的资源，执行下面的命令：

```
kubectl kustomize <kustomization_directory>
```

要应用这些资源，使用参数 `--kustomize` 或 `-k` 标志来执行 `kubectl apply`：

```
kubectl apply -k <kustomization_directory>
```

功能特性

Kustomize 提供以下`功能特性`来管理 应用配置文件：

- 从其他来源生成资源。
- 为资源设置贯穿性（Cross-Cutting）字段。
- 组织和定制资源集合。

特性列表

字段	类型	解释
namespace	string	为所有资源添加名字空间。
namePrefix	string	此字段的值将被添加到所有资源名称前面。
nameSuffix	string	此字段的值将被添加到所有资源名称后面。
commonLabels	map[string]string	要添加到所有资源和选择算符的标签。
commonAnnotations	map[string]string	要添加到所有资源的注解。
resources	[] string	列表中的每个条目都必须能够解析为现有的资源配置文件。
configMapGenerator	[] ConfigMapArgs	列表中的每个条目都会生成一个 ConfigMap。
secretGenerator	[] SecretArgs	列表中的每个条目都会生成一个 Secret。
generatorOptions	GeneratorOptions	更改所有 ConfigMap 和 Secret 生成器的行为。
bases	[] string	列表中每个条目都应能解析为一个包含 kustomization.yaml 文件的目录。
patchesStrategicMerge	[] string	列表中每个条目都能解析为某 Kubernetes 对象的策略性合并补丁。
patchesJson6902	[] Patch	列表中每个条目都能解析为一个 Kubernetes 对象和一个 JSON 补丁。
vars	[] Var	每个条目用来从某资源的字段来析取文字。
images	[] Image	每个条目都用来更改镜像的名称、标记与/或摘要，不必生成补丁。
configurations	[] string	列表中每个条目都应能解析为一个包含 Kustomize 转换器配置 的文件。
crds	[] string	列表中每个条目都赢能够解析为 Kubernetes 类别的 OpenAPI 定义文件。

configMapGenerator

要基于文件来生成 ConfigMap，可以在`configMapGenerator`的`files`列表中添加表项。

可以从`.env`文件 或 `literals`(基于`字面量`的键值对) 生成 ConfigMap。下面是一个根据`.properties`文件中的数据条目来生成 ConfigMap 的示例：

```
# 创建 application.properties
anxin@node38:~/pengling/k8s/kustomize$ cat <<EOF >application.properties
> FOO=Bar
> EOF

# 创建 kustomization.yaml
anxin@node38:~/pengling/k8s/kustomize$ cat <<EOF >./kustomization.yaml
> configMapGenerator:
> - name: example-configmap-1
>   files:
>   - application.properties
> EOF
```

所生成的 ConfigMap 可以使用`kubectl kustomize ./`命令来检查：

```
# 查看生成的 ConfigMap
anxin@node38:~/pengling/k8s/kustomize$ kubectl kustomize ./
apiVersion: v1
data:
  application.properties: |
    FOO=Bar
kind: ConfigMap
metadata:
  name: example-configmap-1-8mbdf7882g
```

configurations 转换器

configurations 列表中每个条目都应能解析为一个包含 Kustomize 转换器配置 的文件。Kustomize 根据转换器创建新的资源。

默认转换器

- annotations
- images
- labels
- name reference
- namespace
- prefix/suffix
- variable reference

Labels 转换器

labels 转换器会为所有资源添加`metadata/labels`字段。也会为所有 Service 资源添加`spec/selector`字段、为所有 Deployment 资源添加`spec/selector/matchLabels`字段。

```
commonLabels:
- path: spec/selectors
  create: true
  kind: MyKind
```

Name reference 转换器

名称引用转换器`nameReference`包括被引用资源的`kind`和引用资源的`fieldSpecs`2个字段。

```
kind: ConfigMap # reference to : 被引用的资源类型
version: v1
fieldSpecs:
- kind: Pod # reference from : 其他资源类型
  version: v1
  path: spec/volumes/configMap/name # path/to/the/field
- kind: Deployment
  path: spec/template/spec/volumes/configMap/name
- kind: Job
  path: spec/template/spec/volumes/configMap/name
```

自定义转换器

除了默认的转换器，还可以创建自定义 (CRD) 转换器。

CRD 项目结构

```
anxin@node38:~/pengling/k8s/kustomize/transformer-configs-crd$ tree
.
├── kustomization.yaml
├── mykind.yaml
└── resources.yaml
```

mykind.yaml

```
anxin@node38:~/pengling/k8s/kustomize/transformerconfigs-crd$ vi mykind.yaml

commonLabels: # labels transformer
- path: spec/selectors # adds labels to the spec/selector field
  create: true
  kind: MyKind # resource type

nameReference:
- kind: Bee # Bee 被引用
  fieldSpecs: # MyKind 引用 Bee 对象的 name
  - path: spec/beeRef/name
    kind: MyKind
- kind: Secret
  fieldSpecs: # MyKind 引用 Secret 对象的 name
  - path: spec/secretRef/name
    kind: MyKind

varReference:
- path: spec/containers/command # 变量引用
  kind: MyKind
- path: spec/beeRef/action
  kind: MyKind
```

resources.yaml

```
anxin@node38:~/pengling/k8s/kustomize/transformer-configs-crd$ vi resources.yaml

apiVersion: v1
kind: Secret
metadata:
  name: crdsecret
data:
  PATH: YmJiYmJiYmIK
---
apiVersion: v1beta1
kind: Bee
metadata:
  name: bee
spec:
  action: fly # Bee.spec.action
---
apiVersion: jingfang.example.com/v1beta1
kind: MyKind
metadata:
  name: mykind
spec:
  secretRef:
    name: crdsecret # secretRef.name
  beeRef:
    name: bee # beeRef.name
    action: $(BEE_ACTION) # 需要 varReference 转换器的配合，否则不做变量替换
  containers:
  - command:
    - "echo"
    - "$(BEE_ACTION)"
    image: myapp
```

kustomization.yaml

```
anxin@node38:~/pengling/k8s/kustomize/transformerconfigs-crd$ vi kustomization.yaml

resources:
- resources.yaml

namePrefix: test-

commonLabels:
  foo: bar

vars: # 变量定义
- name: BEE_ACTION # 从 Bee 类型的 spec.action 获取值
  objref:
    kind: Bee
    name: bee
    apiVersion: v1beta1
  fieldref:
    fieldpath: spec.action
```

```
19
20 configurations:
21   - mykind.yaml
```

查看资源

```
1  anxin@node38:~/pengling/k8s/kustomize/transformer-configs-crd$ kubectl
   kustomize .
2  apiVersion: v1
3  data:
4    PATH: YmJiYmJiYmIK
5  kind: Secret
6  metadata:
7    labels:
8      foo: bar
9    name: test-crdsecret
10 ---
11 apiVersion: jingfang.example.com/v1beta1
12 kind: MyKind
13 metadata:
14   labels:
15     foo: bar
16   name: test-mykind
17 spec:
18   beeRef:
19     action: fly
20     name: test-bee # Bee 对象的 name
21   containers:
22   - command:
23     - echo
24     - fly
25     image: myapp
26   secretRef:
27     name: test-crdsecret # Secret 对象的 name
28   selectors:
29     foo: bar
30 ---
31 apiVersion: v1beta1
32 kind: Bee
33 metadata:
34   labels:
35     foo: bar
36   name: test-bee
37 spec:
38   action: fly
```