101Kubernetes 系列（九十四）K8s Operator — Annotations

现在我们知道如何管理资源状态和条件，是时候查看注释了。

所有注释均来自 Kubebuilder(https://book.kubebuilder.io/)。

有很多不同的注释，我们已经看到了其中一些，因为我们使用注释为 Operator 定义了 **RBAC 权限** 。

```
// +kubebuilder:rbac:groups=apps,resources=deployments,verbs=get;list;watch;create;update;patch;delete
// +kubebuilder:rbac:groups=core,resources=services,verbs=get;list;watch;create;update;patch;delete
```

今天，我们将看到 4 个注释，它们将帮助我们在使用自定义资源或调试它们时更加高效。

1. 资源 - Resource

该注释必须位于资源结构定义之上。（在 `api/…/xxx_types.go` 中）

它们由 3 个主要部分组成，用于：

- 定义 Operator “范围”（如果 Operator 是命名空间范围或集群范围）
- 定义可用于访问资源的备用名称列表（当您 kubectl get … ）。每个名称必须用 ; 分隔
- 定义资源名称的单数（也可以在执行时使用它 kubectl get … ）

```
//+kubebuilder:resource:scope=Cluster,shortName=myproxies;mp,singular=myproxy
type MyProxy struct {
   metav1.TypeMeta   `json:",inline"`
   metav1.ObjectMeta `json:"metadata,omitempty"`
   Spec   MyProxySpec   `json:"spec,omitempty"`
   Status MyProxyStatus `json:"status,omitempty"`
}
```

2.打印列

拥有自定义资源固然很好，但能够使用 `kubectl get` 显示重要信息就更好了！

仍然在资源结构定义之上（在 `api/…/xxx_types.go` 中），您可以使用以下模式定义自定义列

```
// +kubebuilder:printcolumn:name="NAME",type=TYPE,JSONPath=JSONPATH
```

- NAME ：自定义列的名称。
- TYPE ：定义要显示的数据类型。（例如：字符串、数字……）,检查 OpenAPI 规范(https://github.com/OAI/OpenAPI-Specification/blob/main/versions/2.0.md#data-types)以了解可以使用哪种类型。

- JSONPATH ：定义用于访问自定义资源中的信息的 JSONPath。如果资源中没有该信息，您将无法显示它。检查 Kubernetes 文档(https://kubernetes.io/docs/reference/kubectl/jsonpath/)以了解如何编写 JSONPath 值。

例如：

```
// +kubebuilder:printcolumn:name="Name",type=string,JSONPath=`.spec.name`
```

这里有3个关于这个注解的更具体的案例！

2.1 Priority

当我们想要获取资源的信息时，我们可以使用选项 `-o wide` 来获取比默认更多的信息。

要使用 Operator 执行此操作，您只需在只想使用选项 -o wide 显示的值的 printcolumn 定义中添加 ,priority=10 （或任何严格大于 0 的值）。

例如：

```
// +kubebuilder:printcolumn:name="Name",type=string,JSONPath=`.spec.name`,priority=10
```

2.2 Age

默认情况下，当您对自定义资源执行 `kubectl get ...` 时，会显示它们的年龄。但一旦添加了一个打印列，该信息就不再显示。

因此，如果您想保留这些信息，这里是您必须添加的注释。

```
// +kubebuilder:printcolumn:name="Age",type=date,JSONPath=.metadata.creationTimestamp
```

2.3 Ready

在本系列之前，我们讨论了条件，并表示了解一切是否顺利会很有用。通过下面的列，我们将能够轻松查看一切是否顺利。

```
// +kubebuilder:printcolumn:name="Ready",type=string,JSONPath=`.status.conditions[?(@.type=="Ready")].status`
```

3. Validation

以下注释只是在您的规范字段上添加验证。

以下是其中的一些：

- 强制输入字段： //+kubebuilder:validation:Required
- 将字段设置为可选： // +kubebuilder:validation:Optional
- 定义字符串的最小长度： //+kubebuilder:validation:MinLength=8
- 定义字符串的最大长度： //+kubebuilder:validation:MaxLength=20
- 检查该值是否包含在枚举列表中： // +kubebuilder:validation:Enum=earth;wind;fire
- 定义数组中元素的最小数量： //+kubebuilder:validation:MinItems=2
- 对于数字，检查该值是否是特定值的倍数： //+kubebuilder:validation:MultipleOf=15

还有很多，如果你有兴趣，可以去查看 KubeBuilder 文档(https://book.kubebuilder.io/reference/markers/crd-validation.html)。但通过所有这些示例，您对可以做什么或不可以做什么有一个很好的想法。

4. Default

我们今天将看到的最后一个注释是定义规范字段的默认值的注释。

它在可选字段的情况下非常有用。

例如：

```
// +kubebuilder:default:=test
// +kubebuilder:default:=0
```

现在您已经了解了所有这些注释，您将能够轻松地使用 Operator 执行更多操作。

在本系列的下一部分中，我们将离开教程部分来讨论一个已经存在的 Operator：Crossplane！

我希望它能对您有所帮助，如果您有任何问题（不存在愚蠢的问题）或有些问题您不清楚，请随时在评论中添加您的问题。