Practical operation Declarative Object Configuration

Knowledge

管理对象

- **命令行指令**

例如，使用`kubectl`命令来创建和管理 Kubernetes 对象。

命令行就好比口头传达，简单、快速、高效。

但它功能有限，不适合复杂场景，操作不容易追溯，多用于开发和调试。

- **声明式配置**

kubernetes使用yaml文件来描述 Kubernetes 对象。

声明式配置就好比申请表，学习难度大且配置麻烦。

好处是操作留痕，适合操作复杂的对象，多用于生产。

常用命令缩写

名称	缩写	Kind
namespaces	ns	Namespace
nodes	no	Node
pods	po	Pod
services	svc	Service
deployments	deploy	Deployment
replicasets	rs	ReplicaSet
statefulsets	sts	StatefulSet

YAML规范

-
  - 缩进代表上下级关系
  - **缩进时不允许使用Tab键，只允许使用空格，通常缩进2个空格**
  - `:` 键值对，后面必须有空格
  - `-`列表，后面必须有空格
  - `[ ]`数组
  - `#`注释
  - `|` 多行文本块
  - `---`表示文档的开始，多用于分割多个资源对象

```yaml
group:
  name: group-1
  members:
    - name: "Jack Ma"
      UID: 10001
    - name: "Lei Jun"
      UID: 10002
  words:
    ["I don't care money","R U OK"]
  # comments
  test: |
    line
    new line
    3rd line
```

配置对象

在创建的 Kubernetes 对象所对应的 `yaml` 文件中，需要配置的字段如下：

- `apiVersion` - Kubernetes API 的版本
- `kind` - 对象类别，例如 `Pod`、`Deployment`、`Service`、`ReplicaSet` 等
- `metadata` - 描述对象的元数据，包括一个 name 字符串、UID 和可选的 namespace
- `spec` - 对象的配置

标签

标签（Labels） 是附加到对象（比如 Pod）上的键值对，用于补充对象的描述信息。

标签使用户能够以松散的方式管理对象映射，而无需客户端存储这些映射。

由于一个集群中可能管理成千上万个容器，我们可以使用标签高效的进行选择和操作容器集合。

- **键的格式：**
- - **前缀**(可选)**/名称**(必须)。
- **有效名称和值：**
- - 必须为 63 个字符或更少（可以为空）
  - 如果不为空，必须以字母数字字符（[a-z0-9A-Z]）开头和结尾
  - 包含破折号 `**-**`、下划线 `**_**`、点 `**.**` 和字母或数字

选择器

标签选择器 可以识别一组对象。标签不支持唯一性。

标签选择器最常见的用法是为Service选择一组Pod作为后端。

目前支持两种类型的选择运算：**基于等值的**和**基于集合的。**

多个选择条件使用逗号分隔，相当于**And(**&&)**运算。

Practical Operation

my-pod.yaml 示例

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: my-nginx
spec:
  containers:
  - name: my-nginx
    image: nginx:1.22
    ports:
    - containerPort: 80
```

```
# 查看创建的pod
controlplane $ kubectl get pod/my-nginx
NAME       READY   STATUS    RESTARTS   AGE
my-nginx   1/1     Running   0          105s
# 查看创建的pod 并且以yaml 输出
controlplane $ kubectl get pod/my-nginx -o yaml
apiVersion: v1
kind: Pod
metadata:
  annotations:
    cni.projectcalico.org/containerID:
b0d43188dffb12027d42794a560e77445a15a0ebbe17f388df7ad614c37830bb
    cni.projectcalico.org/podIP: 192.168.1.5/32
    cni.projectcalico.org/podIPs: 192.168.1.5/32
    kubectl.kubernetes.io/last-applied-configuration: |
      {"apiVersion":"v1","kind":"Pod","metadata":{"annotations":{},"name":"my-
nginx","namespace":"default"},"spec":{"containers":
[{"image":"nginx:1.22","name":"my-nginx","ports":[{"containerPort":80}]}]}}
  creationTimestamp: "2023-09-04T03:21:16Z"
  name: my-nginx
  namespace: default
  resourceVersion: "2527"
  uid: dedae2c1-1c13-409c-a4c1-e018cc38d46e
spec:
  containers:
  - image: nginx:1.22
    imagePullPolicy: IfNotPresent
    name: my-nginx
    ports:
    - containerPort: 80
      protocol: TCP
    resources: {}
    terminationMessagePath: /dev/termination-log
    terminationMessagePolicy: File
    volumeMounts:
    - mountPath: /var/run/secrets/kubernetes.io/serviceaccount
      name: kube-api-access-m4w8l
      readOnly: true
  dnsPolicy: ClusterFirst
  enableServiceLinks: true
  nodeName: node01
  preemptionPolicy: PreemptLowerPriority
  priority: 0
  restartPolicy: Always
  schedulerName: default-scheduler
```

```
  securityContext: {}
  serviceAccount: default
  serviceAccountName: default
  terminationGracePeriodSeconds: 30
  tolerations:
  - effect: NoExecute
    key: node.kubernetes.io/not-ready
    operator: Exists
    tolerationSeconds: 300
  - effect: NoExecute
    key: node.kubernetes.io/unreachable
    operator: Exists
    tolerationSeconds: 300
  volumes:
  - name: kube-api-access-m4w8l
    projected:
      defaultMode: 420
      sources:
      - serviceAccountToken:
          expirationSeconds: 3607
          path: token
      - configMap:
          items:
          - key: ca.crt
            path: ca.crt
          name: kube-root-ca.crt
      - downwardAPI:
          items:
          - fieldRef:
              apiVersion: v1
              fieldPath: metadata.namespace
            path: namespace
status:
  conditions:
  - lastProbeTime: null
    lastTransitionTime: "2023-09-04T03:21:16Z"
    status: "True"
    type: Initialized
  - lastProbeTime: null
    lastTransitionTime: "2023-09-04T03:21:17Z"
    status: "True"
    type: Ready
  - lastProbeTime: null
    lastTransitionTime: "2023-09-04T03:21:17Z"
    status: "True"
    type: ContainersReady
  - lastProbeTime: null
    lastTransitionTime: "2023-09-04T03:21:16Z"
    status: "True"
    type: PodScheduled
  containerStatuses:
  - containerID:
containerd://9349ad6b80efc09491c02cfd1739ee095907c2a153c7300b540bc3575a219a48
    image: docker.io/library/nginx:1.22
    imageID:
docker.io/library/nginx@sha256:fc5f5fb7574755c306aaf88456ebfbe0b006420a184d52b92
3d2f0197108f6b7
    lastState: {}
```

```
    name: my-nginx
    ready: true
    restartCount: 0
    started: true
    state:
      running:
        startedAt: "2023-09-04T03:21:16Z"
  hostIP: 172.30.2.2
  phase: Running
  podIP: 192.168.1.5
  podIPs:
  - ip: 192.168.1.5
  qosClass: BestEffort
  startTime: "2023-09-04T03:21:16Z"
controlplane $
```

my-service.yaml 示例以及操作

```
# 执行label-demo yaml
controlplane $ kubectl apply -f label-demo
pod/label-demo created
# 查看文件内容
controlplane $ cat label-demo
apiVersion: v1
kind: Pod
metadata:
  name: label-demo
  labels: #定义Pod标签
    environment: test
    app: nginx
spec:
  containers:
  - name: nginx
    image: nginx:1.22
    ports:
    - containerPort: 80controlplane $ kubectl get pod --show-labels
NAME         READY   STATUS    RESTARTS   AGE     LABELS
label-demo   1/1     Running   0          2m21s   app=nginx,environment=test
controlplane $ kubectl get pod -l app=nginx
NAME         READY   STATUS    RESTARTS   AGE
label-demo   1/1     Running   0          4m9s
controlplane $ kubectl get pod -l environment test
error: name cannot be provided when a selector is specified
controlplane $ kubectl get pod -l environment=test
NAME         READY   STATUS    RESTARTS   AGE
label-demo   1/1     Running   0          4m34s
controlplane $ kubectl get pod -l environment=test,app=nginx
NAME         READY   STATUS    RESTARTS   AGE
label-demo   1/1     Running   0          4m43s
controlplane $ touch my-service
controlplane $ kubectl apply -f my-service
service/my-service created
controlplane $ curl localhost:3007
curl: (7) Failed to connect to localhost port 3007: Connection refused
# 访问开通的 svc 端口 发现可 证明 svc 和 pod 通过lable 关联
controlplane $ curl localhost:30007
<!DOCTYPE html>
```

```
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
# 通过lable 标签查看对应的命中情况
controlplane $ kubectl get all -l environment=test,app=nginx
NAME             READY   STATUS    RESTARTS   AGE
pod/label-demo   1/1     Running   0          9m27s
controlplane $ kubectl get svc -l environment=test,app=nginx
No resources found in default namespace.
controlplane $ kubectl get svc -l environment=test
No resources found in default namespace.
# 查看svc 文件内容
controlplane $ cat my-service
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  type: NodePort
  selector: #与Pod的标签一致
    environment: test
    app: nginx
  ports:
      # 默认情况下，为了方便起见，`targetPort` 被设置为与 `port` 字段相同的值。
    - port: 80
      targetPort: 80
      # 可选字段
      # 默认情况下，为了方便起见，Kubernetes 控制平面会从某个范围内分配一个端口号（默认：
30000-32767）
      nodePort: 30007controlplane $
controlplane $
```