拉勾教育
—互联网人实战大学—

《Kubernetes 原理剖析与实战应用》

正范

— 拉勾教育出品 —

08 | 配置管理：Kubernetes 管理业务配置方式有哪些？

前言

使用过程中，常常需要对 Pod 进行一些配置管理

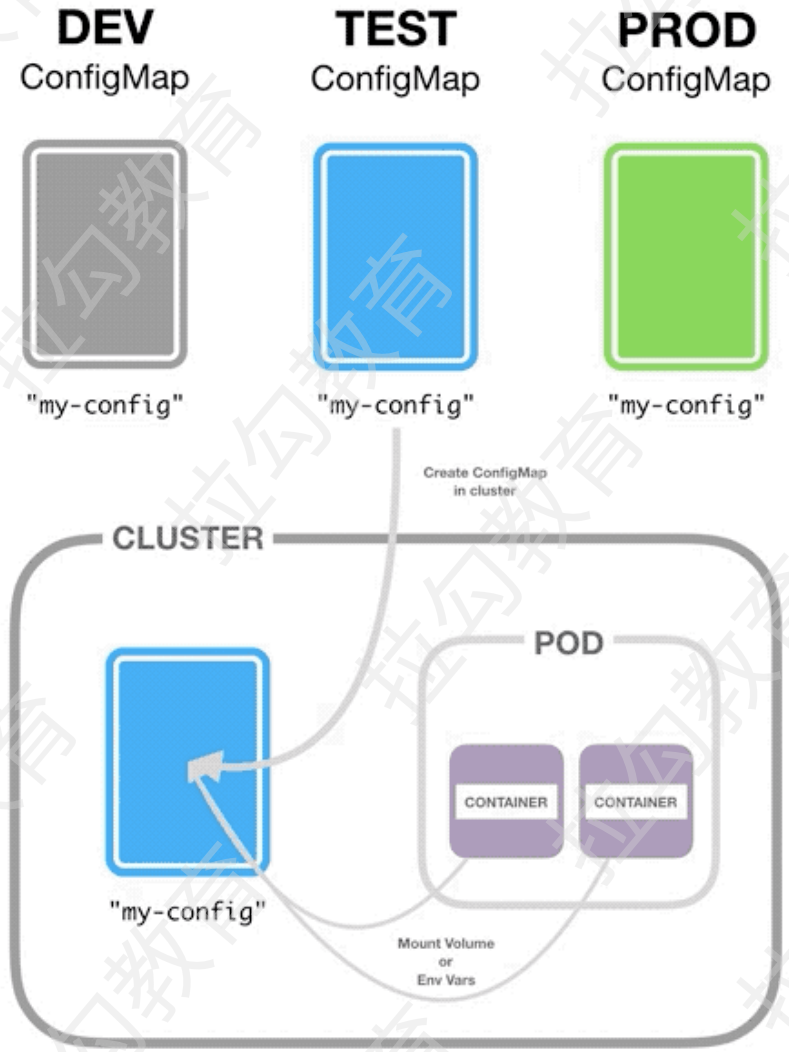
比如参数配置文件怎么使用，敏感数据怎么保存传递等等

前言

- 有些不变的配置是可以打包到镜像中的，那可变的配置呢？
- 信息泄漏，很容易引发安全风险，尤其是一些敏感信息，比如密码、密钥等
- 每次配置更新后，都要重新打包一次，升级应用

  镜像版本过多，也给镜像管理和镜像中心存储带来很大的负担
- 定制化太严重，可扩展能力差，且不容易复用

ConfigMap


DEV
ConfigMap
TEST
ConfigMap
PROD
ConfigMap
"my-config"
"my-config"
"my-config"
Create ConfigMap
in cluster
CLUSTER
POD
CONTAINER
CONTAINER
"my-config"
Mount Volume
or
Env Vars

ConfigMap


```
$ cat cm-demo-mix.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: cm-demo-mix #对象名字
  namespace: demo #所在的命名空间
data: #这是跟其他对象不太一样的地方，其他对象这里都是spec
  #每一个键都映射到一个简单的值
  player_initial_lives: "3" #注意这里的值如果数字的话，必须用字符串来表示
  ui_properties_file_name: "user-interface.properties"

  #也可以来保存多行的文本
  game.properties: |
    enemy.types=aliens,monsters
    player.maximum-lives=5
  user-interface.properties: |
    color.good=purple
    color.bad=yellow
    allow.textmode=true
$ cat cm-demo-all-env.yaml
apiVersion: v1
```

ConfigMap


```
  #每一个键都映射到一个简单的值
  player_initial_lives: "3" #注意这里的值如果数字的话，必须用字符串来表示
  ui_properties_file_name: "user-interface.properties"

  #也可以来保存多行的文本
  game.properties: |
    enemy.types=aliens,monsters
    player.maximum-lives=5
  user-interface.properties: |
    color.good=purple
    color.bad=yellow
    allow.textmode=true
$ cat cm-demo-all-env.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: cm-demo-all-env
  namespace: demo
data:
  SPECIAL_LEVEL: very
  SPECIAL_TYPE: charm
```

ConfigMap


```
$ kubectl create -f cm-demo-mix.yaml
configmap/cm-demo-mix created
$ kubectl create -f cm-demo-all-env.yaml
configmap/cm-demo-all-env created
```

https://kubernetes.io/zh/docs/tasks/configure-pod-container/configure-pod-configmap/#%E4%BD%BF%E7%94%A8-kubectl-create-configmap-%E5%88%9B%E5%BB%BA-configmap

ConfigMap


```
$ kubectl get cm -n demo
NAME              DATA   AGE
cm-demo-all-env   2      30s
cm-demo-mix       4      2s
$ kubectl describe cm cm-demo-all-env -n demo
Name:         cm-demo-all-env
Namespace:    demo
Labels:       <none>
Annotations:  <none>


Data
====
SPECIAL_LEVEL:
----
very
SPECIAL_TYPE:
----
charm
Events:  <none>
```

ConfigMap


```
Events:  <none>
$ kubectl describe cm cm-demo-mix -n demo
Name:         cm-demo-mix
Namespace:    demo
Labels:       <none>
Annotations:  <none>


Data
====
user-interface.properties:
----
color.good=purple
color.bad=yellow
allow.textmode=true


game.properties:
----
enemy.types=aliens,monsters
```

ConfigMap


```
user-interface.properties:
----
color.good=purple
color.bad=yellow
allow.textmode=true


game.properties:
----
enemy.types=aliens,monsters
player.maximum-lives=5


player_initial_lives:
----
3
ui_properties_file_name:
----
user-interface.properties
Events: <none>
```

ConfigMap


```
$ cat cm-demo-pod.yaml
apiVersion: v1
kind: Pod
metadata:
  name: cm-demo-pod
  namespace: demo
spec:
  containers:
    - name: demo
      image: busybox:1.28
      command:
        - "bin/sh"
        - "-c"
        - "echo PLAYER_INITIAL_LIVES=$PLAYER_INITIAL_LIVES && sleep 10000"
      env:
        #定义环境变量
        - name: PLAYER_INITIAL_LIVES #请注意这里和 ConfigMap 中的键名是不一样的
          valueFrom:
            configMapKeyRef:
```

ConfigMap


```
    - configMapRef:
        name: cm-demo-all-env
    volumeMounts:
    - name: full-config #这里是下面定义的 volume 名字
      mountPath: "/config" #挂载的目标路径
      readOnly: true
    - name: part-config
      mountPath: /etc/game/
      readOnly: true
  volumes: #您可以在 Pod 级别设置卷，然后将其挂载到 Pod 内的容器中
  - name: full-config #这是 volume 的名字
    configMap:
      name: cm-demo-mix #提供你想要挂载的 ConfigMap 的名字
  - name: part-config
    configMap:
      name: cm-demo-mix
      items: #我们也可以只挂载部分的配置
      - key: game.properties
        path: properties
```

ConfigMap


```
$ kubectl create -f cm-demo-pod.yaml
pod/cm-demo-pod created
```

ConfigMap


```
$ kubectl exec -it cm-demo-pod -n demo sh
kubectl exec [POD][COMMAND] is DEPRECATED and will be removed in a future
version. Use kubectl kubectl exec [POD] -- [COMMAND] instead.
/ # env
KUBERNETES_SERVICE_PORT=443
KUBERNETES_PORT=tcp://10.96.0.1:443
UI_PROPERTIES_FILE_NAME=user-interface.properties
HOSTNAME=cm-demo-pod
SHLVL=1
HOME=/root
SPECIAL_LEVEL=very
TERM=xterm
KUBERNETES_PORT_443_TCP_ADDR=10.96.0.1
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
KUBERNETES_PORT_443_TCP_PORT=443
KUBERNETES_PORT_443_TCP_PROTO=tcp
KUBERNETES_SERVICE_PORT_HTTPS=443
KUBERNETES_PORT_443_TCP=tcp://10.96.0.1:443
PLAYER_INITIAL_LIVES=3
KUBERNETES_SERVICE_HOST=10.96.0.1
```

ConfigMap


```
PWD=/
SPECIAL_TYPE=charm
/ # ls /config/
game.properties         ui_properties_file_name
player_initial_lives    user-interface.properties
/ # ls -alh /config/
total 12
drwxrwxrwx    3 root     root        4.0K Aug 27 09:54 .
drwxr-xr-x    1 root     root        4.0K Aug 27 09:54 ..
drwxr-xr-x    2 root     root        4.0K Aug 27
09:54 ..2020_08_27_09_54_31.007551221
lrwxrwxrwx    1 root     root          31 Aug 27 09:54 ..data -
> ..2020_08_27_09_54_31.007551221
lrwxrwxrwx    1 root     root          22 Aug 27 09:54 game.properties -
> ..data/game.properties
lrwxrwxrwx    1 root     root          27 Aug 27 09:54 player_initial_lives -
> ..data/player_initial_lives
lrwxrwxrwx    1 root     root          30 Aug 27 09:54 ui_properties_file_name -
> ..data/ui_properties_file_name
lrwxrwxrwx    1 root     root          32 Aug 27 09:54 user-interface.properties -
```

ConfigMap


```
drwxrwxrwx    3 root     root        4.0K Aug 27 09:54 .
drwxr-xr-x    1 root     root        4.0K Aug 27 09:54 ..
drwxr-xr-x    2 root     root        4.0K Aug 27
09:54 ..2020_08_27_09_54_31.007551221
lrwxrwxrwx    1 root     root          31 Aug 27 09:54 ..data -
> ..2020_08_27_09_54_31.007551221
lrwxrwxrwx    1 root     root          22 Aug 27 09:54 game.properties -
> ..data/game.properties
lrwxrwxrwx    1 root     root          27 Aug 27 09:54 player_initial_lives -
> ..data/player_initial_lives
lrwxrwxrwx    1 root     root          30 Aug 27 09:54 ui_properties_file_name -
> ..data/ui_properties_file_name
lrwxrwxrwx    1 root     root          32 Aug 27 09:54 user-interface.properties -
> ..data/user-interface.properties
/ # cat /config/game.properties
enemy.types=aliens,monsters
player.maximum-lives=5
/ # cat /etc/game/properties
enemy.types=aliens,monsters
player.maximum-lives=5
```

Secret

可以用 Secret 来保存一些敏感的数据信息，比如密码、密钥、token 等

跟 ConfigMap 的用法基本保持一致，都可以用来作为环境变量或者文件挂载

Secret


```
$ kubectl create secret  -h
 Create a secret using specified subcommand.
Available Commands:
  docker-registry Create a secret for use with a Docker registry
  generic         Create a secret from a local file, directory or literal value
  tls             Create a TLS secret
Usage:
  kubectl create secret [flags] [options]
Use "kubectl  --help" for more information about a given command.
 Use "kubectl options" for a list of global command-line options (applies to all commands).
```

Secret


```
$ kubectl create secret -n demo docker-registry regcred \
  --docker-server=yourprivateregistry.com \
  --docker-username=allen \
  --docker-password=mypassw0rd \
--docker-email=allen@example.com
secret/regcred created
$ kubectl get secret -n demo regcred
NAME    TYPE                                DATA  AGE
regcred  kubernetes.io/dockerconfigjson  1     28s
```

Secret


```
$ kubectl describe secret -n demo regcred
Name:         regcred
Namespace:    demo
Labels:       <none>
Annotations:  <none>

Type:  kubernetes.io/dockerconfigjson

Data
====
.dockerconfigjson:  144 bytes
```

Secret


```
$ kubectl get secret -n demo regcred -o yaml
apiVersion: v1
data: #跟 configmap 一样，这块用于保存数据信息
  .dockerconfigjson:
eyJhdXRocyI6eyJ5b3VycHJpdmF0ZXJlZ2lzdHJ5LmNvbSI6eyJ1c2
VybmFtZSI6ImFsbGVuIiwicGFzc3dvcmQiOiJteXBhc3N3MHJkIiwiZ
W1haWwiOiJhbGxlbkBleGFtcGxlLmNvbSIsImF1dGgiOiJZV3hzWlc
0NmJYbHdZWE56ZHpCeVpBPT0ifX19
kind: Secret
metadata:
  creationTimestamp: "2020-08-27T12:18:35Z"
  managedFields:
  - apiVersion: v1
    fieldsType: FieldsV1
    fieldsV1:
      f:data:
        .: {}
        f:.dockerconfigjson: {}
```

Secret


```
  creationTimestamp: "2020-08-27T12:18:35Z"
  managedFields:
  - apiVersion: v1
    fieldsType: FieldsV1
    fieldsV1:
      f:data:
        .: {}
        f:.dockerconfigjson: {}
      f:type: {}
    manager: kubectl
    operation: Update
    time: "2020-08-27T12:18:35Z"
  name: regcred
  namespace: demo
  resourceVersion: "1419452"
  selfLink: /api/v1/namespaces/demo/secrets/regcred
  uid: 6d34123e-4d79-406b-9556-409cfb4db2e7
type: kubernetes.io/dockerconfigjson
```

Secret


```
$ kubectl get secret regcred -n demo --
output="jsonpath={.data.\.dockerconfigjson}" | base64 --decode
{"auths":{"yourprivateregistry.com":{"username":"allen","passwor
d":"mypassw0rd","email":"allen@example.com","auth":"YWxsZW4
6bXlwYXNzdzByZA=="}}}
```

Secret


```
$ cat secret-demo.yaml
apiVersion: v1
kind: Secret
metadata:
  name: dev-db-secret
  namespace: demo
type: Opaque
data: #这里的值都是 base64 加密后的
  password: UyFCXCpkJHpEc2I9
  username: ZGV2dXNlcg==
```

Secret


```
$ kubectl create secret generic dev-db-secret -n demo \
  --from-literal=username=devuser \
  --from-literal=password='S!B\*d$zDsb='
```

Secret


```
$ echo -n 'username=devuser' > ./db_secret.txt
$ echo -n 'password=S!B\*d$zDsb=' >> ./db_secret.txt
$ kubectl create secret generic dev-db-secret -n demo \
--from-file=./db_secret.txt
```

Secret


```
$ cat secret-demo-stringdata.yaml
apiVersion: v1
kind: Secret
metadata:
  name: dev-db-secret
  namespace: demo
type: Opaque
stringData:
  password: devuser
  username: S!B\*d$zDsb=
```

Secret


```
$ cat pod-secret.yaml
apiVersion: v1
kind: Pod
metadata:
  name: secret-test-pod
  namespace: demo
spec:
  containers:
    - name: demo-container
      image: busybox:1.28
      command: [ "/bin/sh", "-c", "env" ]
      envFrom:
      - secretRef:
          name: dev-db-secret
  restartPolicy: Never
$ kubectl create -f pod-secret.yaml
pod/secret-test-pod created
```

Secret

```
$ kubectl get pod -n demo secret-test-pod
NAME              READY   STATUS      RESTARTS   AGE
secret-test-pod   0/1     Completed   0          14s
$ kubectl logs -f -n demo secret-test-pod
KUBERNETES_SERVICE_PORT=443
KUBERNETES_PORT=tcp://10.96.0.1:443
HOSTNAME=secret-test-pod
SHLVL=1
username=devuser
HOME=/root
KUBERNETES_PORT_443_TCP_ADDR=10.96.0.1
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
KUBERNETES_PORT_443_TCP_PORT=443
password=S!B\*d$zDsb=
KUBERNETES_PORT_443_TCP_PROTO=tcp
KUBERNETES_SERVICE_PORT_HTTPS=443
KUBERNETES_PORT_443_TCP=tcp://10.96.0.1:443
KUBERNETES_SERVICE_HOST=10.96.0.1
PWD=/
```

写在最后

ConfigMap 和 Secret 是 Kubernetes 常用的保存配置数据的对象

可以根据需要选择合适的对象存储数据

写在最后

- 如果业务自身支持 reload 配置的话，比如nginx -s reload

  可以通过 inotify 感知到文件更新，或者直接定期进行 reload

- Reloader 通过 watch ConfigMap 和 Secret，一旦发现对象更新，就自动触发对 Deployment 或 StatefulSet 等工作负载对象进行滚动升级

Next：《09 | 存储类型：如何挑选合适的存储插件？》

拉勾教育

—互联网人实战大学—

关注拉勾「教育公众号」
获取更多课程信息

L / A / G / O / U