

加我微信可进技术群学习交流:

微信号:

han421302

也可通过扫描下面二维码添加



长按或者扫描上面二维码图片,  
备注K8S即可进入技术交流群

微信号: han421302

课程更新的知识点会通过微信公众号免费分享给大家, 可以关注我的公众  
号



## 配置管理中心 configmap

### 1.1 Configmap 概述

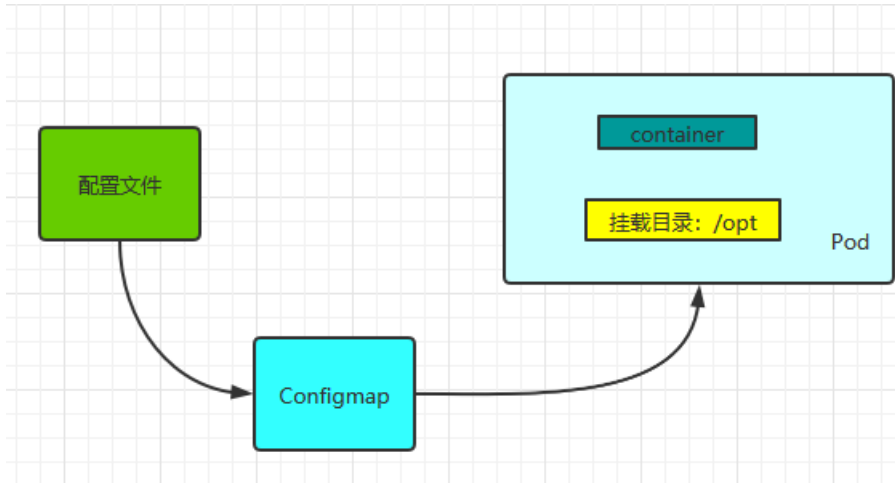
#### 1.1.1 什么是 Configmap?

Configmap 是 k8s 中的资源对象, 用于保存非机密性的配置的, 数据可以用 key/value 键值对的形式保存, 也可通过文件的形式保存。

#### 1.1.2 Configmap 能解决哪些问题?

版权声明, 本文档全部内容及版权归韩先超所有, 只可用于自己学习使用, **禁止私自传阅, 违者依法追责。**

我们在部署服务的时候, 每个服务都有自己的配置文件, 如果一台服务器上部署多个服务: nginx、tomcat、apache 等, 那么这些配置都存在这个节点上, 假如一台服务器不能满足线上高并发的要求, 需要对服务器扩容, 扩容之后的服务器还是需要部署多个服务: nginx、tomcat、apache, 新增加的服务器上还是要管理这些服务的配置, 如果有一个服务出现问题, 需要修改配置文件, 每台物理节点上的配置都需要修改, 这种方式肯定满足不了线上大批量的配置变更要求。所以, k8s 中引入了 Configmap 资源对象, 可以当成 volume 挂载到 pod 中, 实现统一的配置管理。



- 1、Configmap 是 k8s 中的资源, 相当于配置文件, 可以有一个或者多个 Configmap;
- 2、Configmap 可以做成 Volume, k8s pod 启动之后, 通过 volume 形式映射到容器内部指定目录上;
- 3、容器中应用程序按照原有方式读取容器特定目录上的配置文件。
- 4、在容器看来, 配置文件就像是打包在容器内部特定目录, 整个过程对应用没有任何侵入。

#### 1.1.3 Configmap 应用场景

1、使用 k8s 部署应用, 当你将应用配置写进代码中, 更新配置时也需要打包镜像, configmap 可以将配置信息和 docker 镜像解耦, 以便实现镜像的可移植性和可复用性, 因为一个 configMap 其实就是一系列配置信息的集合, 可直接注入到 Pod 中给容器使用。configmap 注入方式有两种, 一种将 configMap 做为存储卷, 一种是将 configMap 通过 env 中 configMapKeyRef 注入到容器中。

2、使用微服务架构的话, 存在多个服务共用配置的情况, 如果每个服务中单独一份配置的话, 那么更新配置就很麻烦, 使用 configmap 可以友好的进行配置共享。

#### 1.1.4 局限性

ConfigMap 在设计上不是用来保存大量数据的。在 ConfigMap 中保存的数据不可超过 1 MiB。如果你需要保存超出此尺寸限制的数据, 可以考虑挂载存储卷或者使用独立的数据库或者文件服务。

#### 1.2 Configmap 创建方法

版权声明, 本文档全部内容及版权归韩先超所有, 只可用于自己学习使用, **禁止私自传阅, 违者依法追责。**

### 1.2.1 命令行直接创建

直接在命令行中指定 configmap 参数创建, 通过--from-literal 指定参数

```
[root@xianchaomaster1 ~]# kubectl create configmap tomcat-config --from-literal=tomcat_port=8080 --from-literal=server_name=myapp.tomcat.com
[root@xianchaomaster1 ~]# kubectl describe configmap tomcat-config
Name:          tomcat-config
Namespace:     default
Labels:        <none>
Annotations:   <none>
Data
====
server_name:
-----
myapp.tomcat.com
tomcat_port:
-----
8080
Events:      <none>
```

### 1.2.2 通过文件创建

通过指定文件创建一个 configmap, --from-file=<文件>

```
[root@xianchaomaster1 ~]# vim nginx.conf
```

```
server {
    server_name www.nginx.com;
    listen 80;
    root /home/nginx/www/
}
```

#定义一个 key 是 www, 值是 nginx.conf 中的内容

```
[root@xianchaomaster1 ~]# kubectl create configmap www-nginx --from-file=www=./nginx.conf
```

```
[root@xianchaomaster1 ~]# kubectl describe configmap www-nginx
```

```
Name:          www-nginx
Namespace:     default
Labels:        <none>
Annotations:   <none>
```

Data

====

www:

-----

```
server {
    server_name www.nginx.com;
```

版权声明, 本文档全部内容及版权归韩先超所有, 只可用于自己学习使用, 禁止私自传阅, 违者依法追究。

```
listen 80;
root /home/nginx/www/
}
```

### 1.2.3 指定目录创建 configmap

```
[root@xianchaomaster1 ~]# mkdir test-a
[root@xianchaomaster1 ~]# cd test-a/
[root@xianchaomaster1 test-a]# cat my-server.cnf
server-id=1
[root@xianchaomaster1 test-a]# cat my-slave.cnf
server-id=2
#指定目录创建 configmap
[root@xianchaomaster1 test-a]# kubectl create configmap mysql-config --
from-file=/root/test-a/
#查看 configmap 详细信息
[root@xianchaomaster1 test-a]# kubectl describe configmap mysql-config
Name:          mysql-config
Namespace:     default
Labels:        <none>
Annotations:   <none>
Data
====
my-server.cnf:
-----
server-id=1
my-slave.cnf:
-----
server-id=2
Events:  <none>
```

### 1.2.4 编写 configmap 资源清单 YAML 文件

```
[root@xianchaomaster1 mysql]# cat mysql-configmap.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: mysql
  labels:
    app: mysql
data:
  master.cnf: |
    [mysqld]
    log-bin
```

版权声明, 本文档全部内容及版权归韩先超所有, 只可用于自己学习使用, **禁止私自传阅, 违者依法追责。**

---

```

log_bin_trust_function_creators=1
lower_case_table_names=1
slave.cnf: |
[mysqld]
super-read-only
log_bin_trust_function_creators=1

```

### 1.3 使用 Configmap

#### 1.3.1 通过环境变量引入: 使用 configMapKeyRef

```

#创建一个存储 mysql 配置的 configmap
[root@xianchaomaster1 ~]# cat mysql-configmap.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: mysql
  labels:
    app: mysql
data:
  log: "1"
  lower: "1"
[root@xianchaomaster1 ~]# kubectl apply -f mysql-configmap.yaml
#创建 pod, 引用 Configmap 中的内容
[root@xianchaomaster1 ~]# cat mysql-pod.yaml
apiVersion: v1
kind: Pod
metadata:
  name: mysql-pod
spec:
  containers:
  - name: mysql
    image: busybox
    command: [ "/bin/sh", "-c", "sleep 3600" ]
    env:
    - name: log_bin    #定义环境变量 log_bin
      valueFrom:
        configMapKeyRef:
          name: mysql    #指定 configmap 的名字
          key: log #指定 configmap 中的 key
    - name: lower    #定义环境变量 lower
      valueFrom:
        configMapKeyRef:
          name: mysql

```

版权声明, 本文档全部内容及版权归韩先超所有, 只可用于自己学习使用, 禁止私自传阅, 违者依法追责。

```
    key: lower
    restartPolicy: Never
#更新资源清单文件
[root@xianchaomaster1 ~]# kubectl apply -f mysql-pod.yaml
[root@xianchaomaster1 ~]# kubectl exec -it mysql-pod -- /bin/sh
/ # printenv
log_bin=1
lower=1
```

### 1.3.2 通过环境变量引入: 使用 envfrom

```
[root@xianchaomaster1 ~]# cat mysql-pod-envfrom.yaml
apiVersion: v1
kind: Pod
metadata:
  name: mysql-pod-envfrom
spec:
  containers:
  - name: mysql
    image: busybox
    imagePullPolicy: IfNotPresent
    command: [ "/bin/sh", "-c", "sleep 3600" ]
    envFrom:
    - configMapRef:
        name: mysql      #指定 configmap 的名字
    restartPolicy: Never
```

```
#更新资源清单文件
[root@xianchaomaster1 ~]# kubectl apply -f mysql-pod-envfrom.yaml
[root@xianchaomaster1 ~]# kubectl exec -it mysql-pod-envfrom -- /bin/sh
/ # printenv
lower=1
log=1
```

### 1.3.3 把 configmap 做成 volume, 挂载到 pod

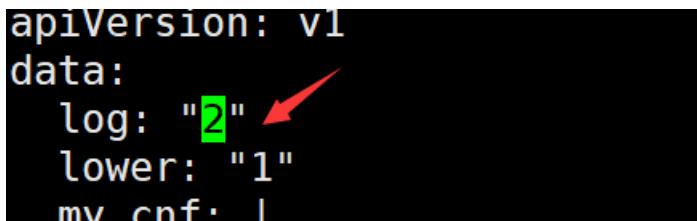
```
[root@xianchaomaster1 ~]# cat mysql-configmap.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: mysql
  labels:
    app: mysql
data:
  log: "1"
  lower: "1"
```

版权声明, 本文档全部内容及版权归韩先超所有, 只可用于自己学习使用, 禁止私自传阅, 违者依法追责。

```
my.cnf: |
  [mysqld]
  Welcome=xianchao
[root@xianchaomaster1 ~]# kubectl apply -f mysql-configmap.yaml
[root@xianchaomaster1 ~]# cat mysql-pod-volume.yaml
apiVersion: v1
kind: Pod
metadata:
  name: mysql-pod-volume
spec:
  containers:
  - name: mysql
    image: busybox
    command: [ "/bin/sh", "-c", "sleep 3600" ]
    volumeMounts:
    - name: mysql-config
      mountPath: /tmp/config
  volumes:
  - name: mysql-config
    configMap:
      name: mysql
  restartPolicy: Never
[root@xianchaomaster1 ~]# kubectl apply -f mysql-pod-volume.yaml
[root@xianchaomaster1 ~]# kubectl exec -it mysql-pod-volume -- /bin/sh
/ # cd /tmp/config/
/tmp/config # ls
log    lower  my.cnf
```

#### 1.4 Configmap 热更新

```
[root@xianchaomaster1 ~]# kubectl edit configmap mysql
把 logs: "1" 变成 log: "2"
```



```
apiVersion: v1
data:
  log: "2"
  lower: "1"
my.cnf: |
```

保存退出

```
[root@xianchaomaster1 ~]# kubectl exec -it mysql-pod-volume -- /bin/sh
/ # cat /tmp/config/log
2
#发现 log 值变成了 2, 更新生效了
```

注意:

更新 ConfigMap 后:

使用该 ConfigMap 挂载的 Env 不会同步更新

使用该 ConfigMap 挂载的 Volume 中的数据需要一段时间（实测大概 10 秒）才能同步更新