## 前言

在第10章中，我们学习了如何上手 `Kubernetes Secret` 。我们都知道， `Kubernetes Secret` 的主要作用是来存放密码，密钥等机密信息。

但是在日常开发部署时，我们还会遇到一些环境变量的配置：例如你的数据库地址，负载均衡要转发的服务地址等等信息。这部分内容使用 `Secret` 显然不合适，打包在镜像内耦合又太严重。这里，我们可以借助 `Kubernetes ConfigMap` 来配置这项事情

## 什么是 ConfigMap

`ConfigMap` 是 `Kubernetes` 的一种资源类型，我们可以使用它存放一些环境变量和配置文件。信息存入后，我们可以使用挂载卷的方式挂载进我们的 Pod 内，也可以通过环境变量注入。和 `Secret` 类型最大的不同是，存在 `ConfigMap` 内的内容不会加密。

## 创建方式

和 `Secret` 一样， `ConfigMap` 也支持多种创建方式

### 命令行直接创建

第一种是使用命令行直接创建。我们直接使用 `kubectl create configmap [config_name]` 命令创建即可。格式如下：

```shell
kubectl create configmap [config_name] --from-literal=[key]=[value]
```

在这里， `--from-literal` 对应一条信息。如果想创建多个 `key value` 组合，向后重复 `--from-literal=[key]=[value]` 即可。

例如我创建一个 `mysql` 的配置文件，其中包含了服务地址，端口。则可以下面这种格式创建：

itjc8.com搜集整理

```
3   --from-literal=MYSQL_PORT=3306
```

> 这里需要注意，configmap 的名称必须是全小写，特殊符号只能包含 '-' 和 '.'。
> 可以用下面的这个正则表达式校验下看看符不符合规则：

> a-z0-9?(.a-z0-9?)*')

创建成功后，我们可以使用 `kubectl get cm` 查看我们创建过的 `configmap`：

```
[root@master ~]# kubectl get cm
NAME           DATA   AGE
mysql-config   2      4m33s
[root@master ~]#
```

@稀土掘金技术社区

可以看到，上面的就是我们刚创建的 `ConfigMap`。里面的 `DATA` 为 `2`，代表有 `2` 条数据存在。我们直接使用 `kubectl describe cm mysql-config` 即可查看下这个 `ConfigMap` 的具体信息：

```
[root@master ~]# kubectl describe cm mysql-config
Name:         mysql-config
Namespace:    default
Labels:       <none>
Annotations:  <none>

Data
====
MYSQL_HOST:
----
192.168.1.172
MYSQL_PORT:
----
3306
Events:  <none>
```

@稀土掘金技术社区

这里可以看到刚才我们存放的数据，代表该 `configmap` 创建成功。

### 配置清单创建

itjc8.com搜集整理

我们新建一个文件，名称为 `mysql-config-file.yaml` ，填入以下内容：

yaml 复制代码

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: mysql-config-file
data:
  MYSQL_HOST: '192.168.1.172'
  MYSQL_PORT: 3306
```

> 字符串不要忘记加引号

在这里，相信大部分字段大家已经都非常熟练了。 `kind` 的值为 `ConfigMap` ，代表声明一个 `ConfigMap` 类型的资源； `metadata.name` 代表是该 `configmap` 的名称； `data` 是存放数据的地方，数据格式为 `key:value` 。

按照惯例，我们保存后使用 `kubectl apply` 命令即可使配置生效：

shell 复制代码

```shell
kubectl apply -f ./mysql-config-file.yaml
```

生效后，我们直接使用 `kubectl describe cm mysql-config-file` 查看下配置结果

```
[root@master configmap]# kubectl describe cm mysql-config-file
Name:         mysql-config-file
Namespace:    default
Labels:       <none>
Annotations:  <none>

Data
====
MYSQL_PORT:
----
3306
MYSQL_HOST:
----
192.168.1.172
Events:  <none>
```

@稀土掘金技术社区

可以看到，要保存创建的内容成功存入。

### 文件创建

itjc8.com搜集整理

shell 复制代码

```shell
kubectl create configmap [configname] --from-file=[key]=[file_path]
```

这里每一条 `--from-file` 都代表一个文件。key是文件在 `configmap` 内的 `key` ，`file_path` 是文件的路径。

我们创建一个文件，然后将文件内容存入 `configmap` 中。创建一个名为 `env.config` 的文件，输入以下内容：

javascript 复制代码

```javascript
URL: 172.168.81.111
PATH: /root/abcd/efg
```

保存后，我们使用 `kubectl create configmap` 命令将其保存至 `configmap` 内：

shell 复制代码

```shell
kubectl create configmap env-from-file --from-file=env=./env.config
```

接着，我们直接使用 `kubectl get cm env-from-file -o yaml` 来查看下保存进入的内容

```
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this file will be
# reopened with the relevant failures.
#
apiVersion: v1
data:
  env: |
    URL: 172.168.81.111
    PATH: /root/abcd/efg
kind: ConfigMap
metadata:
  creationTimestamp: "2020-11-20T08:46:51Z"
  name: env-from-file
  namespace: default
  resourceVersion: "1560270"
  selfLink: /api/v1/namespaces/default/configmaps/env-from-file
  uid: 47a82e62-fa26-4499-b439-6cbd1c5ee08f
~
```

@稀土掘金技术社区

这里我们看到， `configmap` 直接将我们整个文件内容都保存了进去。 `env` 则是这个文件的 `key` 值。

## 目录创建

当然，可以将单个文件存入，也可以直接将一个目录下的文件整个存入进去。

itjc8.com搜集整理

shell 复制代码

```shell
kubectl create configmap [configname] --from-file=[dir_path]
```

我们创建一个文件夹，下面存放几个文件来测试下。这里我们创建了三个文件，分别是 `env1.config` ， `env2.config` ， `env3.config` 。内容也和其文件名对应。

shell 复制代码

```shell
mkdir env && cd ./env
echo 'env1' > env1.config
echo 'env2' > env2.config
echo 'env3' > env3.config
```

这样我们使用创建命令，将内容批量存入到 `configmap` 内：

shell 复制代码

```shell
kubectl create configmap env-from-dir --from-file=./
```

创建完成后，我们使用 `kubectl get cm env-from-dir -o yaml` 查看下保存进去的文件内容：

itjc8.com搜集整理

```
  env1.config: |
    env1
  env2.config: |
    env2
  env3.config: |
    env3
kind: ConfigMap
metadata:
  creationTimestamp: "2020-11-20T08:57:49Z"
  managedFields:
  - apiVersion: v1
    fieldsType: FieldsV1
    fieldsV1:
      f:data:
        .: {}
        f:env1.config: {}
        f:env2.config: {}
        f:env3.config: {}
    manager: kubectl-create
    operation: Update
    time: "2020-11-20T08:57:49Z"
  name: env-from-dir
  namespace: default
  resourceVersion: "1561858"
  selfLink: /api/v1/namespaces/default/configmaps/env-from-dir
  uid: de95c719-3f07-473f-a5da-ec380c77133d
[root@master env]#
```

@稀土掘金技术社区

我们看到，文件夹下的文件内容被批量存放了进去。

## 使用方式

在了解了其创建方式后，我们来看看如何使用 `ConfigMap`

### 环境变量注入

注入到环境变量是一种比较常见的方式。在这里，我们编辑下 `front-v1` 的 `deployment` 配置文件，来将 `configmap` 注入进环境变量内：

yaml 复制代码

```yaml
env:
- name: MYSQL_HOST
  valueFrom:
    configMapKeyRef:
```

itjc8.com搜集整理

configmap 的环境变量注入，其实和 Secret 的环境变量注入方式差别不大，只是字段换成了 configMapKeyRef 。 name 为要选择注入的 configmap 名称； key 则为 configmap 内的其中一个 key 。

编辑完后，保存并退出。使用 kubectl apply -f 命令生效下配置文件，此时旧 Pod 会被杀死重启创建。

shell 复制代码

```
kubectl apply -f ./v1.yaml
```

生效后，在最新的 Pod 内使用 kubectl exec 命令来看看环境变量注入结果：

shell 复制代码

```
kubectl exec -it [POD_NAME] -- env | grep MYSQL_HOST
```

```
[root@master deployment]# kubectl exec -it front-v1-79f7d7f66c-6b6qn -- env | grep MYSQL_HOST
MYSQL_HOST=192.168.1.172
[root@master deployment]#
```

@稀土掘金技术社区

此时可以看到，我们的环境变量成功的注入了进去。

---

可是，如果一条一条地注入环境配置，是不是太麻烦了。怎样才能一次性将整个 ConfigMap 都注入进去呢?

在这里，我们可以借助 containers.envFrom 字段去一次性批量导入我们的 configmap ：

yaml 复制代码

```yaml
envFrom:
- configMapRef:
    name: mysql-config
    optional: true
```

> 如果你的configmap中的key含有 "-"，会自动转换为 "_"

这里我们的 name 值为已配置好的 configmap ， optional 代表如果没有该 configmap ，容器是否能够正常启动。

itjc8.com搜集整理

```
- name: nginx
  imagePullPolicy: Always
  image: registry.cn-hangzhou.aliyuncs.com/janlay/k8s_test:v1
  envFrom:
  - configMapRef:
      name: mysql-config
      optional: true
```

@稀土掘金技术社区

添加后，保存并生效该 `deployment` ，此时 `Pod` 会杀死重建。新 `Pod` 启动后，我们使用 `kubectl exec` 命令看下 `Pod` 内环境变量注入情况：

yaml 复制代码

```
kubectl exec -it [POD_NAME] -- env | grep MYSQL
```

```
[root@master deployment]# kubectl exec -it front-v1-578dc96775-k95n6 -- env | grep MYSQL
MYSQL_HOST=192.168.1.172
MYSQL_PORT=3306
```

@稀土掘金技术社区

此时我们可以看到，环境变量被批量注入了进去。

### 存储卷挂载

第二种方式是存储卷挂载。这种方式会将 `configmap` 里内容中的每个 `key` 和 `value` ，以独立文件方式以外部挂载卷方式挂载进去（ `key` 是文件名， `value` 是文件内容）。这部分的用法和 `Secret` 的用法很像

我们编辑下 `front-v1` 的 `deployment` 配置文件，修改下配置：

**第一步：在 Pod 层面声明一个外部存储卷。 `name` 为存储卷名称； `configMap` 代表存储卷的文件来源为 `configMap` ； `configMap.name` 要填入要加载的 `configMap` 名称。位置如图所示：**

yaml 复制代码

```
volumes:
- name: envfiles
  configMap:
    name: env-from-dir
```

itjc8.com搜集整理


**第二步：在容器镜像层面配置存储卷。** `name` **的值来源于第一步配置的** `name` **值；** `mountPath` **为要挂载的目录；** `readonly` **则代表文件是不是只读。位置如图所示：**

yaml 复制代码

```yaml
volumeMounts:
- name: envfiles
  mountPath: /root/
  readOnly: true
```


编辑完后，保存并退出。使用 `kubectl apply -f` 命令生效下配置文件。

shell 复制代码

itjc8.com搜集整理

待 Pod 杀死重建后，我们来验证下文件是否已经挂载了进去。这里我们使用 kubectl exec 命令看下目录是否这个文件：

shell 复制代码

```
kubectl exec -it [POD_NAME] -- ls /root
```

```
[root@master deployment]# kubectl exec -it front-v1-f9b8c9798-h8wjd -- ls /root
env1.config  env2.config  env3.config
[root@master deployment]#
```

@稀土掘金技术社区

可以看到，三个文件都成功地挂载了进去。

---

但是，这种方式每次挂载都要将整个文件夹挂载进去，我们如何一次只挂载单个文件呢？这里我们可以借助 volumes.configMap.items[] 字段来配置多个 item 项：

yaml 复制代码

```yaml
volumes:
- name: envfiles
  configMap:
    name: env-from-dir
    items:
    - key: env1.config
      path: env1.config
    - key: env2.config
      path: env2.config
```

这里的 item 是个数组，每一项都是一条 ConfigMap 里的独立字段。

其中， key 是 ConfigMap 中的字段名称； path 则是要挂载的路径（相对于在容器镜像层面配置存储卷配置的 mountPath 字段）。填写保存后退出生效

接着我们用 kubectl exec 命令验证下挂载结果

```
[root@master deployment]# kubectl exec -it front-v1-6fc5d66885-pl878 -- ls /root
env1.config  env2.config
[root@master deployment]#
```

@稀土掘金技术社区

结果如我们所愿，只挂载进去我们配置的2个文件。

## 结束语

---

itjc8.com搜集整理

**污点**来更好地调度部署我们的 Pod

大家有什么问题，欢迎在评论区讨论提出 👏

上一章 下一章

留言

输入评论（Enter换行，Ctrl + Enter发送）

发表评论

全部评论（3）

沛公 JY.1 2年前

从配置单创建ConfigMap报错:
kubectl apply -f ./mysql-config-file.yaml
Error from server (BadRequest): error when creating "./mysql-config-file.yaml": ConfigMap in version "v1" cannot be handled as a ConfigMap: v1.ConfigMap.Data: ReadString: expects " or n, but found 3, error found in #10 byte of ...|QL_PORT":3306},"kind|..., bigger context ...|data":...

展开

点赞 2 ···

王圣松 （作者） 2年前

这个应该是缩进有点问题。根据yaml文件格式，不允许使用tab替代缩进。所以可以改成空格

点赞 回复 ···

用户256238... 1年前

因为不接收int的value

点赞 回复 ···

itjc8.com搜集整理

itjc8.com搜集整理