Configuración de pods en Kubernetes

KEEPCODING
Tech School

Kubernetes Workloads

- En esta sección:
  - Gestión de la configuración
  - Command + args
  - Variables de entorno
  - ConfigMaps
  - Secrets

¿Qué es la gestión de configuración?

- Reglas no escritas de diseño de contenedores:
  - Una misma aplicación puede funcionar de distintas maneras dependiendo:
    - El propósito de la aplicación y quién la use.
    - El entorno en el que se esté ejecutando.
    - Dependencias con servicios de terceros.
  - Por ello las aplicaciones deben permitir ser configurables desde fuera y no llevar la configuración “hardcoded” o “a fuego”.
  - Aún así, la aplicación debe de poder funcionar con valores **por defecto**.

https://kubernetes.io/docs/concepts/configuration/overview/

¿Qué es es la gestión de Configuración?

- En Kubernetes podemos suministrar configuraciones de distintas formas:
  - Como argumentos del comando a ejecutar (entrypoint / cmd)
  - Variables de entorno
  - ConfigMaps
  - Secrets

- Todo esto es similar a lo visto anteriormente con Docker.

https://kubernetes.io/docs/concepts/configuration/overview/

Configuración por línea de argumentos

- Nuestra aplicación soporta distintos parámetros por línea de comandos.
- Podemos especificarlo directamente mediante la opción “**command**” o “**args**” de un POD.
- SIMILAR A ENTRYPOINT y CMD en Docker

```
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
  labels:
    app: myapp
spec:
  containers:
  - name: myapp-container
    image: busybox

    command: ['sh', '-c']
    args:
      - |
        while true; do
          echo "Hello Kuberntes"
          sleep 1
        done
```

https://kubernetes.io/docs/concepts/configuration/overview/

Configuración por variable de entorno

- Además de las variables de entorno que se inyectan de forma automática podemos definir nuestras propias variables de entorno
- Cuidado con declarar variables que contengan **información sensible**.

```
apiVersion: v1
kind: Pod
metadata:
  name: myvar
  labels:
    app: myvar
spec:
  containers:
  - name: myapp-container
    image: busybox
    env:
     - name: MY_VAR
       value: "Prueba"
    command: ['sh', '-c']
    args:
      - |
        while true; do
          echo "Hello $MY_VAR"
          sleep 1
        done
```

- Definiremos nuetras variables dentro de: `spec.containers[].env`

https://kubernetes.io/docs/concepts/configuration/overview/

ConfigMaps

KEEPCODING
Tech School

ConfigMap

- Son objetos utilizados para almacenar datos no confidenciales en el formato clave-valor (se pueden almacenar ficheros completos como valor de una clave).
- Permite desacoplar la configuración de una imagen de contenedor, para mejorar la portabilidad.
- No proporciona encriptación.
- Tipos de consumo de ConfigMaps:
  - **Argumento** de CMD o **entrypoint** de un contenedor.
  - **Variable de entorno** de contenedor
  - Directorio o fichero montado a través de **volúmen**.

ConfigMap

- Creación de ConfigMaps:
  - A través de fichero yaml, como el resto de recursos
  - Directamente desde la línea de comandos, usando ficheros o directorios existentes como fuente de datos.
- Ejemplo de ConfigMap:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: my-config
data:
  conf.properties: |
    color=blue
    country=Spain
  conf.json: |
    {"color": "blue", "country": "Spain"}
  otra-clave: |
    Este es el contenido de la clave
    en múltiples líneas.
```

Montar ConfigMap como directorio

- Podemos consumir un ConfigMap directamente como si fuera un directorio dentro del contenedor. Cada clave del ConfigMap será un fichero en el punto de montaje.
- Se define como un volúmen y luego se configura dónde ha de ser montado.

```
apiVersion: v1
kind: Pod
metadata:
  name: dapi-test-pod
spec:
  containers:
    - name: test-container
      image: k8s.gcr.io/busybox
      command: [ "/bin/sh", "-c", "ls /etc/config/" ]
      volumeMounts:
      - name: config-volume
        mountPath: /etc/config
  volumes:
    - name: config-volume
      configMap:
        # Provide the name of the ConfigMap containing the files you
want
        # to add to the container
        name: my-config
```

https://kubernetes.io/docs/tasks/configure-pod-container/configure-pod-configmap/

Variables de entorno desde ConfigMap

- Para ello el dato del ConfigMap debe de estar en formato key-value.

```
apiVersion: v1
kind: Pod
metadata:
  name: cm-env-value
spec:
  containers:
    - name: test-container
      image: k8s.gcr.io/busybox
      command: [ "/bin/sh", "-c", "env" ]
      env:
        # Define the environment variable
        - name: MY_COLOR
          valueFrom:
            configMapKeyRef:
              # The ConfigMap containing the value you want to
assign to SPECIAL_LEVEL_KEY
              name: colors
              # Specify the key associated with the value
              key: primary
  restartPolicy: Never
```

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: colors
data:
  primary: green
  second: blue
```

Configurar todas las Keys de CM como Env

- Podemos directamente inyectar como variables de entorno todos los valores de un ConfigMap

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: my-data
data:
  SPECIAL_LEVEL: very
  SPECIAL_TYPE: charm
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: cm-env-value
spec:
  containers:
    - name: test-container
      image: k8s.gcr.io/busybox
      command: [ "/bin/sh", "-c", "env" ]
      envFrom:
      - configMapRef:
          name: my-data
  restartPolicy: Never
```

```
$ kubectl apply -f configmap.yaml
$ kubectl apply -f pod.yaml
$ kubectl logs cm-env-value
```

ConfigMap

- Demo!

Secrets

KEEPCODING
Tech School

Secrets

- Los secretos son muy similares a los configmaps, pero los usaremos para almacenar y manejar información sensible, como contraseñas, tokens, claves SSH, etc.
- La información va codificada en Base64, así que para trabajar con ellos cómodamente utilizaremos kubectl.

```
$ kubectl create secret generic my-secret --from-literal=password=verysecret
$ kubectl create secret generic my-secret --from-file=./password.txt
```

```
apiVersion: v1
kind: Secret
metadata:
  name: my-secret
type: Opaque
data:
  password: YWRtaW4=
```

https://kubernetes.io/es/docs/concepts/configuration/secret/

Secrets

- Todo lo aprendido en ConfigMaps es aplicable a Secrets
  - La forma de usarlos / consumirlos es similar:
    - Como variables de entorno
    - Cargándolos en volúmenes y montándolos en directorios de los pods (o como ficheros).
- Al ir codificados nos vamos a centrar en aprender a crearlos y manejarlos.
- En el yaml del secret podemos usar **stringData** en lugar de **data** para dejar los datos en claro.

https://kubernetes.io/es/docs/concepts/configuration/secret/

Secrets

- Demo!

https://kubernetes.io/docs/concepts/configuration/overview/

Montar Secret como directorio

- Podemos consumir un Secret directamente como si fuera un directorio dentro del contenedor. Todo el contenido del secret estará disponible en el directorio montado.
- Se define como un volúmen y luego se configura dónde ha de ser montado.

```
apiVersion: v1
kind: Pod
metadata:
  name: dapi-test-pod
spec:
  containers:
    - name: test-container
      image: k8s.gcr.io/busybox
      command: [ "/bin/sh", "-c", "ls /etc/config/" ]
      volumeMounts:
      - name: secret-volume
        mountPath: /etc/config
  volumes:
    - name: secret-volume
      secret:
        # Provide the name of the Secret containing the files you want
        # to add to the container
        name: my-secret
    restartPolicy: Never
```

https://kubernetes.io/docs/tasks/configure-pod-container/configure-pod-configmap/

Variables de entorno desde Secret

- Para ello los datos del secreto deben ser clave / valor también.

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: s-env-value
spec:
  containers:
    - name: test-container
      image: k8s.gcr.io/busybox
      command: [ "/bin/sh", "-c", "env" ]
      env:
        # Define the environment variable
        - name: PASSWORD
          valueFrom:
            secretKeyRef:
              # The Secret containing the value you want to assign to PASSWORD
              name: my-secret
              # Specify the key associated with the value
              key: password
  restartPolicy: Never
```

https://kubernetes.io/docs/concepts/configuration/secret/

Secrets especiales

- Tipos de secrets:
  - generic / Opaque (el tipo por defecto)
  - kubernetes.io/tls (tls.key & tls.crt)
  - kubernetes.io/basic-auth (username & password)
  - kubernetes.io/ssh-auth
  - …

https://kubernetes.io/docs/concepts/configuration/overview/

Ejercicios ConfigMaps y Secrets

KEEPCODING
Tech School

Ejercicios ConfigMap y Secret

1. Crea un ConfigMap llamado “config” con los valores foo=lala,foo2=lolo
2. Muestra el ConfigMap en YAML
3. Crea el mismo configmap, pero desde un fichero.
4. Crea un configmap desde un fichero config.env: `echo -e "var1=val1\n# un comentario\n\nvar2=val2\n#otrocomentario" > config.env`
5. Crea un configmap desde fichero, pero que la key sea distinta al nombre del fichero.
6. Crea un configMap con valores "var6=val6", "var7=val7". Carga este configmap como variables de entorno en un POD de nginx.
7. Crea un configmap con valores "var8=val8","var9=val9". Que se cargue dentro de un POD de nginx en el path /etc/data
8. Crea un secreto llamado "secreto" con "usuario=pepito" y "password=muysecreto". Inténtalo desde un fichero YAML (usando 'data') y comprueba que el secret final tenga los datos correctos.
9. Crea un POD que con variables de entorno ES_USER y ES_PASSWORD obtenidas desde el secreto anterior.
10. Crea un secreto llamado "secreto2" que obtenga datos desde un fichero de texto

5.1

Configura un Deployment de nginx con 2 réplicas.
El nginx debe hacer de proxy inverso hacia https://www.google.com
Genera todos los YAML.
Créalo todo a partir de los YAML.

Ayuda: puedes usar de referencia la configuración que usamos en el proxy inverso lanzado en el módulo de docker-compose.

5.2

Configura el nginx anterior para que escriba los logs en JSON vía ConfigMap. Requiere un poco de investigación.

Posible ayuda: https://github.com/rccrdpccl/nginx-jsonlog

5.3

Configura un Stack de wordpress + MySQL.
Piensa y argumenta los tipos de recursos seleccionados.
Genera los YAML.
En un namespace específico (no en default).
Nos podemos olvidar de la persistencia de momento.

KEEPCODING

Tech School

Madrid | Barcelona | Bogotá