Como crear un elemento persistente en Kubernetes

Contenido

[Introducción 1](#_Toc69205103)

[Docker Compose 2](#_Toc69205104)

# Introducción

A la hora de crear nuestras aplicaciones FiL, es posible que necesitemos crear componentes que sean persistentes, es decir, que no pierdan los datos que contienen aunque no puedan seguir con su funcionamiento normal.

El mejor ejemplo es una base de datos. Si tenemos una BBDD en nuestro cluster Kubernetes, estará desplegada en un pod. Estos elementos se destruyen y se crean y, por lo tanto, pierden la información cada vez que esto ocurre. Para solucionar este problema Kubernetes nos ofrece el concepto de **volumen**. Los volúmenes son considerados otro recurso de Kubernetes y sirven para administrar el almacenamiento duradero en un clúster.

Por lo tanto, si necesitamos que no se pierda la información de algún componente crearemos un volumen y enlazaremos el componente con él. Así, toda la información se guardará en ese volumen, y si hay algún problema y se cae el componente, cuando se vuelva a crear cogerá los datos del volumen.

# Docker Compose

El primer paso si queremos crear un elemento persistente es definirlo en el archivo ***docker-compose.yml***. En él definiremos los volúmenes que queramos para cada componente. El ejemplo con el que vamos a trabajar es la base de datos eXist. Necesitaremos dos volúmenes, uno para guardar los datos y otro para la configuración del componente.

Este será el contenido del archivo comentado:

version: "3.6"

services:

exist:

image: existdb/existdb:latest

container\_name: existdb

ports:

- 8080:8080

- 8443:8443

volumes:

- type: volume

source: exist-data

target: exist/data

- type: volume

source: exist-config

target: exist/config

volumes:

data:

driver: labels

name: exist-data

config:

driver: local

name: exist-config

Como se puede apreciar, en el apartado *volumes* hemos definido lo volúmenes deseados. Hay dos formas de definirlos. La primera es como se ve en el documento anterior; la segunda es un poco más simple:

volumes:

* exist-data:/exist/data
* exist-config:/exist/config

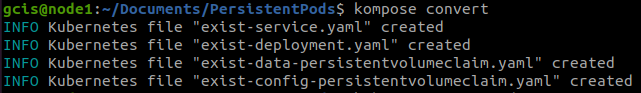
Además, en la parte de abajo, hemos definido esos volúmenes.

# Kompose

En la parte de Kompose tendremos que realizar varios cambios. Recordemos que esta herramienta sirve para traducir las definiciones de Docker Compose a archivos de despliegue de Kubernetes. Aun así, en este caso no crea los archivos del todo correctamente. Primero ejecutaremos el comando de traducción en el mismo directorio donde tengamos el archivo *docker-compose.yml*:

> kompose convert

Así, nos creará varios archivos de despliegue para Kubernetes:



Como podemos apreciar, crea un archivo para crear un Deployment, un Service y dos PersitentVolumeClaim. Los archivos de Deployment y Service no los tendríamos que cambiar, a no ser que queramos exponer el servicio fuera del cluster (como ya hemos visto en anteriores manuales de usuario).

Antes de desplegar nada, tendremos que realizar más cambios.

# Persistent Volume

Antes de desplegar nada, tendremos que crear unos volúmenes persistentes. Los recursos ***PersistentVolume*** se usan para administrar el almacenamiento duradero en un clúster. Los recursos *PersistentVolume* son recursos de clúster que existen de forma independiente de los pods. Esto significa que el disco y los datos representados por un *PersistentVolume* continúan existiendo a medida que el clúster cambia y los pods se borran y se vuelven a crear. Los recursos *PersistentVolume* se pueden aprovisionar de manera dinámica a través de *PersistentVolumeClaims*.

Una ***PersistentVolumeClaim*** es una solicitud y una reclamación de un recurso *PersistentVolume*. Los objetos *PersistentVolumeClaim* solicitan un tamaño específico, un modo de acceso y una *StorageClass* para *PersistentVolume*. Si existe un *PersistentVolume* que satisface la solicitud o se puede aprovisionar, *PersistentVolumeClaim* se vincula a ese *PersistentVolume*.

Como hemos visto en el anterior apartado, nos ha creado dos archivos de PersistenVolumeClaim. Esto quiere decir que cuando despleguemos el componente, se va a solicitar un tamaño especifico de un volumen persistente ya creado. Por eso tendremos que crear un volumen persistente por cada *claim* que se vaya a crear.

Por lo tanto, crearemos un volumen persistente para los datos y otro para la configuración de eXist. Al igual que cualquier recurso de Kubernetes se crearán con archivos YAML, y esta es su definición:

apiVersion: v1

kind: PersistentVolume

metadata:

name: exist-data

labels:

type: local

spec:

storageClassName: exist-data

capacity:

storage: 500Mi

accessModes:

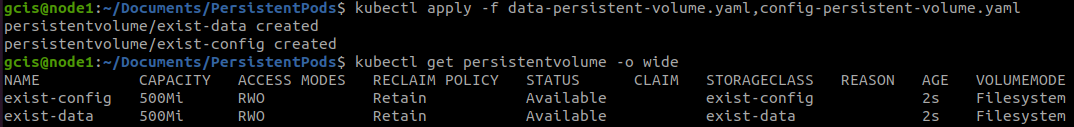
- ReadWriteOnce

hostPath:

path: "/exist/data"

También, para desplegarlos en Kubernetes, seguiremos el mismo proceso:

> kubectl apply -f data-persistent-volume.yaml,config-persistent-volume.yaml



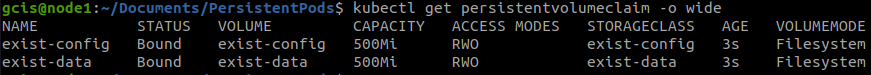
# Despliegue

Ahora, ya podremos desplegar nuestro componente en el cluster. Teniendo todos los archivos YAML (deployment, service, data volume claim, config volume claim) en un directorio, los desplegaremos:

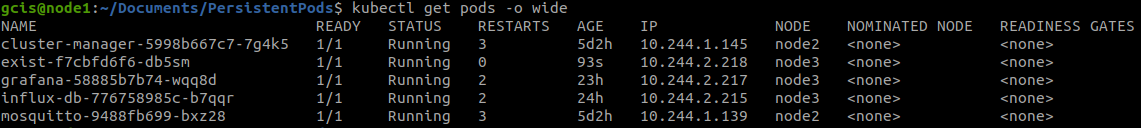
> kubectl apply –f /Directorio con YAMLs/

Si queremos ver los PersistentVolumeClaims:

> kubectl get persistentvolumeclaim -o wide

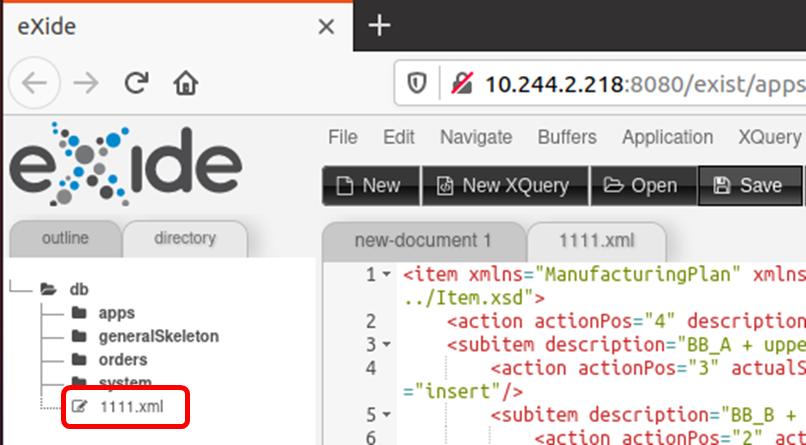


Como podremos comprobar, tendremos nuestro elemento en el cluster, en este caso, eXist:

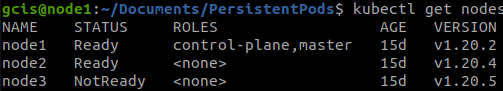


# Prueba de fallo

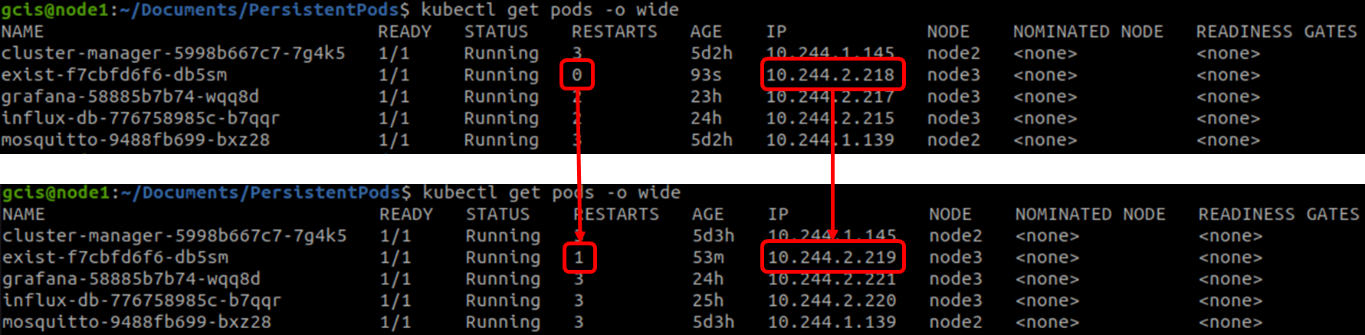
Ahora vamos a hacer una prueba de fallo, para ver si los datos no se pierden. Primero, almacenaremos un XML en la base de datos:



Para hacer la prueba de un fallo, apagaremos el nodo donde está desplegado, en este caso el *node3*.



Ahora, volveremos a arrancarlo y podremos ver como su dirección IP han cambiado, y también que se ha reconstruido una vez.



Si vamos a ver la base de datos, podremos comprobar que el archivo XML sigue estando ahí.

