Contenedores

Un entorno aislado dentro de un SO con kernel Linux donde ejecutar procesos. Aislado, en cuanto a qué?

- El contenedor tiene su propia conf de red -> sus propias IP
- Tiene sus propias variables de entorno
- Tiene su propio filesystem (sistema de archivos)
- Puede tener limitaciones en cuanto al acceso al hardware del host

Los contenedores los creamos desde Imágenes de contenedores. Los contenedores se crean y operan a través de un gestor de contenedores, como: Docker, Podman, ContainerD, CRIO

Cual es la gran diferencia entre un contenedor y una máquina virtual? Las máquinas virtuales deben llevar su propio Sistema operativo, mientras que los contenedores comparten el kernel del host. En un contenedor NO ES POSIBLE levantar un SO.

Tienen los contenedores persistencia de la información? SI Igual que las máquinas virtuales. Será si borro un contenedor cuando pierda los datos... igual que si borro una máquina virtual o si reformateo una máquina física!

El problema es que normalmente no estamos creando y borrando máquinas virtuales... pero sí que estamos creando y borrando contenedores como churros.

Si trabajo con máquinas virtuales... y tengo una con el mariadb... y quiero actualizar el mariadb... qué hacía? Arrancaba la máquina virtual del mariadb, entraba y actualizaba el mariadb. Y con contenedores? Me lo crujo y creo uno nuevo con la nueva versión (usando una imagen nueva de mariadb = RAPIDITO) y fácil. Y los datos... ah! los he perdido... Pues uso un volumen... guardo los datos en un sitio FUERA del sistema de archivos del contenedor Si borro el contenedor, ese sitio, que está fuera del contenedor, no se borra. Creo luego uno nuevo y le digo que use ese volumen... y ya tengo los datos de nuevo.

Para qué sirven los volúmenes?

- Persistencia de datos tras la eliminación de un contenedor
- Compartir datos entre contenedores (a lo mejor no quiero que tengan persistencia)
- Inyectar archivos/carpetas al contenedor

Imagen de contenedor

Una imagen de contenedor es un triste fichero comprimido (ZIP-> tar.gz) que:

- Suele contener una estructura de carpetas compatible con POSIX / bin/ etc/ opt/ home/ var/ tmp/ ...
- En la que ya se ha instalado de antemano un software
- Junto con su configuración... y otros archivos, librerías, comandos que puedqan necesitarse para su ejecución.

Adicionalmente ese archivo (imagen de contenedor) lleva adjuntos una serie de metadatos.,

> Ejemplo: Quiero montar un SQLServer en mi máquina Windows. Qué tengo que hacer?

0. Tener instalados y configurados los requisitos previos (librerías)
1. Descargar el instalador de SQLServer
2. Ejecuto el instalador --> c:\Archivos de programa\SQLServer >>> ZIP >> EMAIL
3. Arranco el SQLServer

Ese ZIP es la imagen de contenedor.... En una imagen viene un software PREINSTALADO de antemano por alguien.

Con contenedores no instalamos software... desplegamos software.

Adicionalmente las operaciones habituales sobre cualquier software:

- Arrancarlo
- Pararlo
- Ver los logs
- ... en el mundo de los contenedores están ESTANDARIZADAS. Da igual el software que tenga dentro de un contenedor... da igual el gestor de contenedores que utilice... Cualquier contenedor (y por ende el software que contiene) se opera de la misma manera.

Las imágenes de contenedor las descargo de un REGISTRO DE REPOSITORIOS DE IMAGENES DE CONTENEDOR:

- DockerHub
- Quay.io (el de redhat)
- Microsoft Container Registry tiene el suyo
- Las empresas suelen montar su propio registro de repositorios de imágenes de contenedor:
    - NEXUS
    - Artifactory
    - GitLab
    - GitHub
    - Quay.io

Las imágenes de contenedores se identifican por: registro/repo:tag

Muchas veces, cuando trabajamos con registros públicos omitimos la parte del registro: mariadb:latest Y en este caso, se busca ese repo/tag en el registro por defecto que tenga configurad en mi gestor de contenedores (docker, podman, ...) que por defecto es docker hub.

El tag

Los tag se usan para informar de ciertos aspectos del software que contiene la imagen de contenedor... y de cómo está configurado:

- version nginx:1.21.4
- la imagen base que se ha utilizado nginx:1-alpine
- Software adicional que se ha instalado nginx:1.21.4-php

`latest` NO ES NINGUNA PALABRA MAGICA que existe en el registro de imágenes de contenedor. Es un tag como otro cualquiera, que ha tenido que ser creado de antemano

```
mariadb -> Por defecto se busca el tag etiquetado como latest... que puede
no existir
```

Latest es un tag variable, que puede apuntar a una imagen concreta en un momento dado... y a otra imagen distinta en otro momento. De hecho es muy común usar este tipo de tags variables... LATEST es una MUY MALA PRÁCTICA.

Imagen base de contenedor

Las uso para montar mis propias imágenes: ubuntu, alpine, fedora, debian Lo que me dan es:

- Una estructura de carpetas posix
- Unos cuantos comandos dentro del /bin ls mv cp mkdir chmod cat tail

Alpine: MUY REDUCIDA ls mv cp mkdir chmod cat tail sh wget

Ubuntu: ls mv cp mkdir chmod cat tail sh apt apt-get bash

Debian: ls mv cp mkdir chmod cat tail sh apt bash

Fedora: ls mv cp mkdir chmod cat tail sh yum dnf bash

Esquema de versionado estandar de software

Versión: a.b.c

```
            Cuándo cambian?
```

a: MAYOR Breaking changes: Cambios que no soportan retrocompatibilidad b: MINOR Nueva funcionalidad Funcionalidades marcadas como deprecated + arreglo de bugs c: PATCH Arreglo de bugs

En los entornos de producción, que versión de software debo usar?

- nginx:latest NUNCA... ni puñetera idea de que se está montando... en un momento dado me puede pasar de la versión 1 a la 2... y que todo deje de funcionar.
- nginx:1 Tampoco... El software requerirá de una funcionalidad... la funcionalidad la marca el MINOR. Si con un MINOR me vale, no metas más... más funcionalidad puede implicar MAS BUGS
- nginx:1.21 ESTA ES LA GUAY ! Este tag apuntará en cadq momento a la última 1.21 que exista... 1.21.1 1.21.2 1.21.8
- nginx:1.21.4 NASTI DE PLASTI: Si hay una versión que ofrezca la misma funcionalidad pero con más bugs arreglados, coño, esa es la que quiero

Kubernetes

Orquestador de gestores de contenedores que posibilita el despliegue de software en un entorno de producción.

Hablamos de otros conceptos:

- Balanceadores de carga
- Proxies reversos
- Volúmenes de almacenamiento
- Reglas de firewall de red

Yo no hago nada en el cluster más que decirla a Kubernetes cómo debe operarlo. Con Kubernetes hablamos usando un lenguaje DECLARATIVO !!!!

Todas las herramientas, librerías, etc... que hoy en día se ponen de moda es por usar lenguajes declarativos.

El lenguaje declarativo no es sino otro PARADIGMA DE PROGRAMACION:

- Imperativo
- Procedural
- Funcional
- Orientado a objetos
- Declarativo

Imperativo (le doy una orden a felipe)

Felipe, si hay algo que no sea una silla debajo de la venta Felipe, lo quitas Felipe... si no hay una silla debajo de la ventana (IF) Felipe, si no hay sillas, vete al ikea a por una silla Felipe, pon una silla debajo de la ventana

ODIAMOS LOS LENGUAJES IMPERATIVOS... son lo peor de lo peor de lo peor.... mucho curro

Lenguaje DECLARATIVO

Felipe, debajo de la ventana tiene que haber una silla

La diferencia es sustancial... ya que en este caso, delego la responsabilidad de consieguir lo que quiero en FELIPE. ME CENTRO EN LO QUE ME INTERESA... y no en el cómo se consigue.

Herramientas que usan lenguaje declarativo: Kubernetes, Terraform, Ansible, Springboot, Spring, Angular

Cluster típico de kubernetes (producción)

```
nodo1 - plano de control    \
nodo2 - plano de control     > Aqui van los programas de kubernetes
nodo3 - plano de control    /
    kubelet
        v
    gestor de contenedor: ContainerD, CRIO

nodo A - nodos de trabajo
    kubelet
        v
    gestor de contenedor: ContainerD, CRIO
```

```
nodo B - nodos de trabajo
    kubelet
        v
    gestor de contenedor: ContainerD, CRIO
...
nodo Z - nodos de trabajo
    kubelet
        v
    gestor de contenedor: ContainerD, CRIO
```

Plano de control de kubernetes

- Kubelet realmente es un servicio que se instala a hierro en todas las máquinas del cluster
- Etcd (BBDD de kubernetes)
- Kubeproxy (balanceo / exposición de puertos)
- Api server (para posibilitar la comunicación de herramientas extras a kubernetes con el cluster) Si monto un operador de kubernetes, necesitaré interlocutar con ese API SERVER Si uso kubectl, dashboard web, helm
- Scheduler (decide en qué nodo de trabajo se despliega cada pod)
- CoreDNS (resolución de nombres de red)

Kustomize

Nos permite generar archivos de despliegue desde plantillas de despliegues

HELM

Nos permite generar archivos de despliegue desde plantillas de despliegues + desplegar ese software... y ocuparse del ciclo de vida del software desplegado (UPGRADES)

Entorno de producción. Características clave

- Alta disponibilidad (HA) \ REDUNDANCIA: Mediante clusters (grupo)
- Escalabilidad /

- Recuperación ante desastres (DR)
- Auditoría
- Monitorización

Objetos en Kubernetes

A Kubernetes le pasamos configuraciones mediante objetos (principalmente en documentos YAML). Y por defecto en Kubernetes (especificación) se habla de un montón de objetos:

- Node

- Namespace
- Pod
- Deployment
    - Replicaset
- Statefulset
- DaemonSet
- Secret Para rellenar volumenes y variables de entorno
- Configmap Para rellenar volumenes y variables de entorno
- Service
    - ClusterIP
    - NodePort
    - LoadBalancer
- Ingress
- PV
- PVC
- Network Policy
- ResourceQuota
- LimitRange
- ServiceAccount

Pod

Conjunto de contenedores que:

- Comparten configuración de red:
    - Tienen la misma IP
    - Pueden hablar entre si mediante "localhost"
- Tengo garantizado que se despliegan en el mismo nodo
    - Pueden compartir volúmenes de almacenamiento locales (NO PERSISTENTES)
- Escalan juntos, a la vez

> Quiero montar un Wordpress

Para montar mi gran web y hacerme rico:

- Servidor web (APACHE)
- PHP + WP
- MariaDB

> > Cuántos contenedores?

OPCION 1:

1. contenedor Apache + php + wp
2. contenedor MariaDB

OPCION 2:

1. contenedor Apache
2. contenedor PHP + WP
3. contenedor MariaDB

OPCION 3: RUINA

1. contenedor Apache + php + wp + MariaDB

- No es escalable. No puedo escalar la BBDD de forma independiente al servidor web
- Quiero tener procesos aislados... coño separo la BBDD / servidor web
- Si quiero actualizar la BBDD a una nueva versión... tengo que actualizar todo !

Lo ponemos fácil... OPCIÓN 1: 2 contenedores

> 1 pod o 2 pods?

x Necesito que los contenedores Compartan configuración de red x Necesito asegurar que se despliegan en el mismo nodo x Necesito que compartan volúmenes de almacenamiento locales (NO PERSISTENTES) x Es requerido y obligatorio que escalen juntos

2 pods.

> Quiero centralizar los logs de los apaches... ElasticSearch << Kibana

Para ello necesito de un programita que me permita llevar (evacuar) los logs de los contenedores de apache a ElasticSearch: filebeat | fluentd

Apache + Filebeat... 1 contenedor o 2? 2!!!! Pa' que lo voy a juntar... quiero entornos aislados para cada proceso... por qué? por qué no? si me sale gratis!

1 pod o 2 pods? 1 pod

x Necesito que los contenedores Compartan configuración de red √ Necesito asegurar que se despliegan en el mismo nodo. LO QUIERO !!!! √ Necesito que compartan volúmenes de almacenamiento locales (NO PERSISTENTES) El apache va a escribir un fichero Ese fichero va a ser consumido por Filebeat. Tengo 2 opciones: OPCION 1: Guardar el fichero en un volumen compartido en RED OPCION 2: Guardar el fichero en un volumen local **** Los logs no tienen que tener persistencia en los apaches Si los guardo en RED... agarrete los de comunicaciones... lo flipan... me están buscando por toda la empresa con un kalashnikov en la mano.. por petarles la red. Además, vamos a tardar la hueva en escribir por red cada dato... cooooño que es un log, que debe de ir rapidito que me frena el sistema!!!! `emptyDir` De hecho una configuración muy habitual en este caso sería montar un volumen de tipo: emptyDir con soporte en RAM. - Tomo un trozo de la RAM: 100Kbs y los monto como un volumen en una carpeta del contenedor /logs del contenedor apache apunta a 100Kbs de la RAM - Monto el fichero de logs en Apache rotado, con 2 ficheros de máximo 50Kbs - Y en el contenedor de filebeat monto esa misma zona de ram en la carpeta: /fichero/a/leer Os imaginais el rendimiento de esto? y que me cuesta 100kbs? GUAY !!!! Y si un dia a año se cae un pod... y hay 3 datos del access log que los pierdo... son 3 datos de un log... prefiero no cargarme la red... y que todo vaya rápido... y si pierdo 3 datos... no son tan críticos √ Es requerido y obligatorio que escalen juntos Si creo un contenedor nuevo con Apache en otra máquina, necesito pegarle al culo un contenedor de filebeat... y que se desplieguen juntos Tendría un pod con 2 contenedores. El contenedor de filebeat es lo que llamaríamos un contenedor SIDECAR

Cuantos pod voy a crear en un cluster de Kubernetes? NINGUNO !

Yo no quiero crear pods... QUE PUEDO HACERLO !!!! Que lo haga Kubernetes que pa' eso le hemos contratado.

Y Kubernetes para crear pods, necesita una PLANTILLA DE POD:

- Deployment/ReplicaSet Una plantilla de pod + número de réplicas Todas las réplicas comparten volumen de almacenamiento
- StatefulSet Una plantilla de pod + Una plantilla de PVC + número de réplicas Cada réplica tiene su propio volumen de almacenamiento
- DaemonSet Una plantilla de pod, de la que Kubernetes creará tantas replicas como nodos tengo en el cluster (algo configurable) Suele ser raro que usemos este tipo de objeto: - Sistema de monitorización: En cada nodo un programa que lea métricas - Sistema de logs: En cada nodo un programa que lea logs del kubelet - Programa que gestione la red virtual entre los nodos

> Tenemos el WP con el MariaDB

Queríamos 17 pod con Apache + PHP + WP (PLANTILLA 1) Los WP... guardan datos? Todos los archivos que subo (imagen, pdf) Si tengo 17 réplicas, quiero que todas usen la misma carpeta (volumen) o volúmenes independientes? El mismo... si subo una imagen en un WP... quiero que se vea en todos los WP wp1 wp2 Balanceador Cliente1 wp17 Cliente2

```
Lo configuraré como un DEPLOYMENT
```

Queríamos 3 pod con MariaDB (PLANTILLA 2)

```
Tengo 3 instancias de MariaDB en cluster (MARIADB-GALERA)
Quiero que guarden todas la misma información... o cada una la suya
independiente? Cada uno el suyo SIEMPRE EN TODO ESCENARIO. TODO SISTEMA
QUE ALMACENA DATOS FUNCIONA ASI

mariadb1    dato1   dato2
mariadb2    dato1   dato3
mariadb3    dato2   dato3

    En 2 unidades de tiempo he guardado 3 datos.
    Si tengo 1 instancia, en 2 unidades de tiempo he guardado 2 datos.
    Con 3 máquinas he conseguido una mejora del 50% en el almacenamiento
de datos. (máxima teórica)

Las BBDD todas asi: MARIADB-GALERA, COCKROACHDB, COUCHBASE, COUCHDB,
MONGODB, ORACLE, POSTGRESQL, ...
Los indexadores: Elastic, SOLR, ...
Los sistemas de mensajería: RabbitMQ, Kafka, ...
```

```
Todo este tipo de sistemas lo configuramos con un STATEFULSET
Si se cae un pod, no quiero cualquier otro pod en su reemplazo... quiero
un pod igual que el que tenía, con su misma personalidad (coño, los mismos
datos!!)
Para eso, los nombres de los pods no pueden ser aleatorios... tienen que
ser fijos... y eso lo hace el STATEFULSET
```

Kubernetes y los contenedores... y los pods...

Cómo sabe kubernetes que un contenedor de un pod está haciendo bien su trabajo?

- Si el proceso principal del contenedor está corriendo

QUE SON ESTAS?

- StartupProbes
- LivenessProbes
- ReadinessProbes

StartupProbes: Es una prueba que hacemos para ver si el contenedor ha terminado de arrancar o no Si no acaba de arrancar... me lo crujo (KUBERNETES LO REINICIA) Una vez superada la strartupProbe, se pasa a la livenessProbe: LiveNessProbe: Es una prueba que hacemos para ver si el contenedor está en un estado SALUDABLE... aunque no listo para prestar servicio Si no está haciendo bien su trabajo... me lo crujo (KUBERNETES LO REINICIA) Una vez superada la livenessProbe, se pasa a la readinessProbe:

Tipos de software

- Driver
- Aplicación (Chrome)
- Demonio/Servicio
- Librería
- Script
- Comando

Hay una diferencia muy grande entre Aplicación/Demonio/servicio con Script/Comando.

- Las apps, demonios y servicios se quedan corriendo en el sistema hasta que los paramos
- Los scripts no se quedan de perpetua... acaban su trabajo

Pregunta.... Todos esos tipos de software son contenedorizables... Puedo poner en un container de un POD de Kubernetes un SCRIPT? NI DE COÑA !

Los containers de los pods están pensados para ejecutar aplicaciones/demonios/servicios... que se quedan corriendo indefinidamente. Si un proceso de un contenedor de un pod acaba... el contenedor se para. Y

KUBERNETES SE VUELVE LOCO... ENTRA EN PANICO ! y se poner a reiniciar el pod como si no hubiera mañana !

Eso si... los pods, tienen además de containers: INIT CONTAINERS Los init están pensados para ejecutar scripts o comandos... que deben acabar en algún momento... si no acaban, KUBERNETES SE VUELVE LOCO... ENTRA EN PANICO ! y se poner a reiniciar el pod como si no hubiera mañana !

Los containers de un pod se ejecutan en paralelo. Los init containers se ejecutan secuencialmente, según el orden de definición en el pod. Y no es hasta que todos acaban que se arrancan los containers del pod.

Si lo que quiero es ejecutar SOLO un script en el cluster, dentro de un contenedor, tenemos otro tipo de objeto en Kubernetes: JOB, que si es capaz de ejecutar scripts o comandos...

> Cuántos jobs voy a crear en kubernetes? POCOS

Para tareas repetitivas, que se ejecutan cada cierto tiempo, que los cree kubernetes, y para eso tenemos tenemos otro tipo de objeto: CRONJOB

- CronJob: Plantilla de job + CRON

En ocasiones hay jobs puntuales que ejecuto en el cluster (por ejemplo si monto un operador...que la primera vez que se ejecuta debe hacer unas tareas de inicialización... y luego ya no hace nada más)... podría ser que yo cree el job.

Comunicaciones dentro de un cluster

Service

Cluster IP

```
Es una IP de balanceo de carga autogestionada por kubernetes junto con una
entrada en el DNS interno de Kubernetes.

    Pod 1 mariadb
    Pod 2 mariadb       Balanceo de carga (Service)    Cliente1
    Pod 3 mariadb
```

NodePort

```
Es un servicio clusterIP + una exposición de puertos en cada nodo del host
(a partir del 30000)
```

LoadBalancer

```
Es un servicio NodePort + con una configuración AUTOMATIZADA De un
balanceador de carga externo al cluster compatible con kubernetes (que
debo haber montado)
Si trabajo con clouds (AWS, AZURE, GCP,...) ellos me "dan" (previo paso
por caja €€€) ya un balanceador de carga compatible con kubernetes.
Si trabajo con un cluster on premise, tendré que montar mi propio
balanceador de carga compatible con kubernetes:
    METALLB
```

Cluster de kubernetes http://mi-web 192.168.0.200:80 -> 192.168.0.1 o 192.168.0.2 o 192.168.0.3 en el puerto 30080 Navegador de menchu: Balanceador de carga Menchu PC mi-web-> 192.168.0.200 | 192.168.0.101 DNS externo | | | +---------- red de mi empresa (192.168.0.0/16)----------+----------+ | ||= 192.168.0.1 -Nodo CP 1 || | NETFILTER (Pregunta... sabeis quien da de alta esas reglas en cada host? KUBEPROXY) || | Si alguien pide ir a 10.0.1.101:3307 -> 10.0.0.103:3306 || | Si alguien pide ir a 10.0.1.102:81 -> 10.0.0.101:80 | 10.0.0.104:80 || | Si alguien pide ir a 10.0.1.103:80 -> 10.0.0.106:80 || | Si alguien toca la puerta de 192.168.0.1:30080 -> mi-ingress:81 || |- Kubeproxy || |- CoreDNS || mi-bbdd-wp -> 10.0.1.101 || mi-wp -> 10.0.1.102 || mi-ingress -> 10.0.1.103 ||= 192.168.0.2 -Nodo1 || | NETFILTER || | Si alguien pide ir a 10.0.1.101:3307 -> 10.0.0.103:3306 || | Si alguien pide ir a 10.0.1.102:81 -> 10.0.0.101:80 | 10.0.0.104:80 || | Si alguien pide ir a 10.0.1.103:80 -> 10.0.0.106:80 || | Si alguien toca la puerta de 192.168.0.2:30080 -> mi-ingress:81 || |- Kubeproxy || |- 10.0.0.101-Pod1- Apache || |- contenedor apache+wp: 443 (https) || config.php: db_host: 10.0.0.102:3306 ??? FUNCIONARIA? SI || Pero sería una MIERDA GIGANTE ! || Si el pod se cae o mueve de máquina pilla otra IP || A priori conozco la IP del pod mariadb? || config.php: db_host: mi-bbdd-wp:3307 ||= 192.168.0.3 -Nodo2 || | NETFILTER || | Si alguien pide ir a 10.0.1.101:3307 -> 10.0.0.103:3306 || | Si alguien pide ir a 10.0.1.102:81 -> 10.0.0.101:80 | 10.0.0.104:80 || | Si alguien pide ir a 10.0.1.103:80 -> 10.0.0.106:80 || | Si alguien toca la puerta de 192.168.0.3:30080 -> mi-ingress:81 || |- Kubeproxy || |- 10.0.0.103-Pod2- MariaDB || | |- contenedor mariadb: 3306 || |- 10.0.0.106-Pod6- IngressController: NGINX || |- contenedor nginx: 80 || --- https://mi-web -> mi-wp:80 <<<< ESTA REGLA DE CONFIGURACION PARA EL INGRESS CONTROLER = INGRESS ||= 192.168.0.4 -Nodo3 || | NETFILTER || | Si alguien pide ir a 10.0.1.101:3307 -> 10.0.0.103:3306 || | Si alguien pide ir a 10.0.1.102:81 -> 10.0.0.101:80 | 10.0.0.104:80 (por defecto se hace round robin)... algo así || | eso es configurable... pero no en kubernetes || | a nivel del kernel de linux || | Si alguien pide ir a 10.0.1.103:80 -> 10.0.0.106:80 || | Si alguien toca la puerta de 192.168.0.4:30080 -> mi-ingress:81 || |- Kubeproxy || |- 10.0.0.104-Pod3- Apache || |- contenedor apache+wp: 80 | Red virtual de kubernetes... donde se pinchan los pods: 10.0.0.0/16... 10.0.0.1 - 10.0.254.254

Creo un servicio clusterIP para el pod de mariaDB: Una IP de balanceo + entrada de DNS en Kubernetes 10.0.1.101 mi-bbdd-wp Esa IP de balanceo me ofrece HA. Si el pod se mueve a otra máquina o recrea, su IP cambia... y kubernetes reasigna la entrada en NET FILTER

Creo un servicio clusterIP para el pod de wp: Una IP de balanceo + entrada de DNS en Kubernetes 10.0.1.102 mi-wp

Creo un servicio LoadBalancer para el pod de nginx-ingressController: Una IP de balanceo + entrada de DNS en Kubernetes + expon en el puerto 30080 de cada nodo 10.0.1.103 mi-ingress + Kubernetes configura el balanceador externo pata que balancee entre mis máquinas, al puerto NodePort que se haya abierto

Habeis oido hablar de NETFILTER? Eso es un componente (programa) del kernel de Linux. Cualquier paquete que pisa una red en una máquina Linux se manda a NETFILTER. Os suena IPTABLES? IP Tables solo es una forma de configurar NETFILTER.

---

apiVersion: v1 kind: Service

metadata: name: mi-bbdd-wp # este es el nombre del DNS

spec: selector: app: mariadb # Estas son las etiquetas que tienen que tener los pods a los que se balancea el tráfico ports: - protocol: TCP port: 3307 # Puerto de la IP de balanceo targetPort: 3306 # Puerto del pod type: ClusterIP

No hay nada en Kubernetes (de forma estandar) que me permita configurar automáticamente un DNS externo a Kubernetes.

Openshift (la distro de kubernetes de la gente de Redhat), si tiene un objeto para ello: ROUTE

> En un cluster típico de kubernetes, que proporción o cantida de servicios creo de cada tipo:

```
                    %       Absoluto
ClusterIP                   Resto               Comunicaciones internas
NodePort            0       0
LoadBalancer                1(2)                Exponer servicios al
público

Y si tengo 2 o 2000 webs, app
```

> Nos hace aquí un concepto típico de entornos de producción: PROXY REVERSO

```
Menchu ------------------>
Felipe ----> proxy   ---->   proxy reverso -->  Servidor web (app)
                               nginx
                               httpd
                               haproxy
                               envoy

El proxy se pone para proteger la identidad de Felipe... y para que el
servidor web no sepa que Felipe existe.
    De paso... en las empresas aprovechan y en el proxy capan el acceso a
muchas web.

El proxy reverso protege a mi servidor web... de forma que los cliente no
lleguen a él. No sepán ni quien es.

En Kubernetes al proxy reverso se le lama INGRESS CONTROLLER = PROXY
REVERSO
```

Ingress

Regla de configuración de proxy reverso en kubernetes. Tengo un ficherito YAML, donde defino un INGRESS (REGLA DE PROXY REVERSO): miapp.es ---> mi-wp:80 El ingress controller se encarga de escribir esa regla en la sintaxis adecuada del proxy reverso concreto que se esté usando... sea nginx, httpd, traeffic, envoy

---

Network Policy

Reglas de firewall de red en kubernetes. Cuidado... depende del driver de red virtual que use para los pods, se soportan o no.

Ejemplo: Al servicio bbdd:3306 del namespace miapp-prod solo pueden atacarlo desde el mismo namespace (determinados pods... IPs)

Es una locura... Cuando acabo con un huevo de servicios, el mnto de esto se torna imposible.

ISTIO

ResourceQuota -> Trabaja con datos acumulados a nivel de un ns

```
En tal ns no se pueden crear pods que en total requieran más de 4 cores
o de 32 Gbs de RAM
En tal ns no se pueden hacer más de 3 pvc.
```

LimitRange -> Trabaja con datos a nivel de un pod de un ns

```
En tal ns cada pod no puede pedir más de 2 cores
o más de 10Gbs de RAM
```

Permiten limitar el consumo de recursos de un determinado ns

Recursos asignados a los pods:

Al definir un pod, detallamos los recursos:

```
resources:
    requests:               # Lo que se solicita garantizado al cluster
        cpu: 100m
        memory: 256Mi
    limits:                 # Hasta cuanto puedo usar si hay hueco
        cpu: 8000m
        memory: 256Mi       # Se configura IGUAL que la del request (LA
LIO A LA MINIMA!)

                            garantizado
                            (request)
            Capacidad       comprometido        USO
            Cores   RAM     Cores     RAM       Cores   RAM
Nodo1        4      10                           0       0
    Pod1                        3      6         3       6  El pod 1 pasa
de 4 a 3 cores... va más lento
    Pod3                        1      3         1       3
Nodo2        4      10
    Pod2                        2      8         2       2

                       REQUEST            LIMITS
                       Cores  RAM      Cores  RAM
Pod1 - wp(apache)       3      6        6      10
Pod2 - mariadb          2      8
Pod3 - nginx            1      3
El kubernetes se carga el pod1 ... lo RENICIA POR EL ARTICULO 33
```

Quién decide en que máquina se despliega un pod? El scheduler de kubernetes SOLO MIRA EL GARANTIZADO/COMPROMETIDO DEL NODO

En JAVA, cuando levanto la JVM le asigno también memoria RAM -Xms512m -Xmx512m La recomendación en JAVA que sean iguales.

ServiceAccount

Es una cuenta de servicio. Me sirven para interactuar con el APISERVER de kubernetes

Imaginad que quiero montar un programa que provea volumenes en automático cuando se hace una petión de volumen. Ese programa monitorizará WATCH de las pvc de TODOS los NAMESPACES del cluster... Eso se lo tiene que permitir Kubernetes. Para identificarme (como programa) ante kubernetes, cuando llame a su APISERVER, uso una cuenta de servicio (~USUARIO) = SERVICE ACCOUNT Los service accounts tiene asignados ROLES (permisos) en kubernetes. Esos permisos se configuran mediante combinaciones de: VERBOS + RECURSOS + NAMESPACE ROLE: CREADOR DE VOLUMENES - VERBOS: WATCH, UPDATE, GET RECURSOS: PVC NAMESPACE: * - VERBOS: CREATE, GET, DELETE RECURSOS: PV A la cuenta de servicio le asigno roles: ROLE BINDING El service Account: provisionador-de-volumenes <-> ROLE: CREADOR DE VOLUMENES

Ese programa cuando interactue con el API SERVER de kubernetes, lo hará con la cuenta de servicio provisionador-de-volumenes Kubernetes le pedirá AUTENTICACIÓN... y el programa le pasará el TOKEN de seguridad asociado la cuenta de servicio provisionador-de-volumenes, que está guardado en un SECRET de kubernetes.

Volumenes en kubernetes

Los volumenes en el mundo de los contenedores los usamos para:

- Hacer accesible a un contenedor de un pod ficheros del host (Monitorizar, acceder a servicios del host-socket, ...) hostPath, doy una ruta del host que monto en el poc/contenedor
- Compartir datos entre contenedores de un mismo pod emptyDir (hdd, memoria ram)
- Inyectar archivos/configuraciones/Carpetas a un pod configMap, secret, emptyDir Imaginad que tengo u archivo de configuración de un programa X... en GIT... y lo quiero meter en mi contenedor. Monto un initContainer que rellene un fichero/carpeta que comparte con un contenedor que arranca luego
- Guardar datos de forma persistente accesibles tras la eliminación de un pod persitentVolumeClaim nfs iscsi fibre channel azure disk aws ebs gce pd

qué es un PV?

Es una REFERENCIA a un volumen de almacenamiento que puedo usar dentro de un cluster de kubernetes para asignarlo a un pod, al que le pongo unos metadatos.

Si defino un pv de tipo azure disk, el pv es una referencia a un volumen que he creado previamente en azure.

```
apiVersion: v1
kind: PersistentVolume
metadata:
    name: mivolumen1-amazon
spec:
    # Metadatos:
    capacity:
        storage: 3Gi # Este es el tamaño real del volumen? NO... es lo que
doy de alta en Kubernetes
                           # Kubernetes no mira esto para nada... a la
hora del espacio que hay en el volumen... ni lo comprueba... lo lo
chequea... ni lo asegura..#
                           # He puesto yo a manita lo que me ha salido de
las narices
                           # Más me vale que ande con cuidado y ponga
algo dsensato (lo mismo que en AWS,... lo disponible de verdad)
    accessModes:
        - ReadWriteOnce         Es un volumen de lectura y escritura que
será usado como mucho en un NODO (estando en él accesible para todos los
```

```
pods que quiera)
        - ReadOnlyMany          Es un volumen de solo lectura que será
usado por varios nodos (y por sus pods)
        - ReadWriteMany         Es un volumen de lectura y escritura que
será usado por varios nodos (y por sus pods)
        - ReadWriteOncePod      Es un volumen de lectura y escritura que
será usado por un solo pod
    storageClassName: repidito-redundante
    # Es una referencia al volumen REAL donde dios quiera que esté
precreado
    awsElasticBlockStore:
        volumeID: "vol-0e3fbaeee540b4611"
        fsType: ext4
```

3Gi -> Gibibyte 1 gibibyte = 1024 mebibytes 1 Mibibyte = 1024 kibibytes 1 Kibibyte = 1024 bytes

```
1 Gb = 1000 Mb
1 Mb = 1000 Kb
1 Kb = 1000 bytes

Hace 25 años, 1 Gb eran 1024 Mb... pero se cambio HACE 25 años YA ... para
seguir los prefijos del SI (Sistema Internacional de Unidades)
```

A priori, NO NECESITO USAR PVS para nada en kubernetes y aún así tener persistencia en mis pods: VEASE :

```
apiVersion: v1
kind: Pod
metadata:
    name: test-ebs
spec:
    containers:
        -   image: registry.k8s.io/test-webserver
            name: test-container
            volumeMounts:
                - mountPath: /test-ebs
                  name: test-volume
    volumes:
        - name: test-volume
          awsElasticBlockStore:
            volumeID: "vol-0e3fbaeee540b4611"
            fsType: ext4
          persitentVolumeClaim
          nfs
          iscsi
          fibre channel
          azure disk
```

```
aws ebs
gce pd
emptyDir
configMap
secret
hostPath
AQUI NO EXISTE pv... ni persistentVolume
```

PROBLEMA DE ESCRIBIR COSAS COMO ESTA:

- Quién configura este archivo... quién lo escribe? DESARROLLO Por qué? joder, porque es el que sabe como hay que desplegar su app... mis variables de entorno que necesito... mis puertos... los recursos que necesito, la imagen que uso
- Tengo yo npi de donde están creando en esta empresita los volumenes? en amazon, en azure, en fb? NPI... es más NO QUIERO SABERLO.
- Como desarrollador yo, lo que sé es:
    - Necesito 100Gbs de espacio en disco según se ha estimado en el proyecto
    - Y rapiditos
    - Y redundantes
    - Y pa mi solo... no se lo quiero compartir a nadie

      Y para esto están los PVC... para hablar mi IDIOMA :

      ```
      apiVersion: v1
      kind: PersistentVolumeClaim
      metadata:
          name: mi-pvc
      spec:
          accessModes:
              - ReadWriteOnce
          resources:
              requests:
                  storage: 50Gi
          storageClassName: rapidito-redundante
      ```

      El administrador del cluster configura un PV:

      ```
          apiVersion: v1
          kind: PersistentVolume
          metadata:
              name: mivolumen1-amazon
          spec:
              # Metadatos:
              capacity:
      ```

```
            storage: 100Gi # Este es el tamaño real del volumen? NO... es lo que doy de alta en Kubernetes
                                # Kubernetes no mira esto para nada... a la hora del espacio que hay en el volumen... ni lo comprueba... lo lo chequea... ni lo asegura..#
                                # He puesto yo a manita lo que me ha salido de las narices
                                # Más me vale que ande con cuidado y ponga algo dsensato (lo mismo que en AWS,... lo disponible de verdad)
        accessModes:
            - ReadWriteOnce         Es un volumen de lectura y escritura que será usado como mucho en un NODO (estando en él accesible para todos los pods que quiera)
            - ReadOnlyMany          Es un volumen de solo lectura que será usado por varios nodos (y por sus pods)
            - ReadWriteMany         Es un volumen de lectura y escritura que será usado por varios nodos (y por sus pods)
            - ReadWriteOncePod      Es un volumen de lectura y escritura que será usado por un solo pod
        storageClassName: rapidito-redundante
        # Es una referencia al volumen REAL donde dios quiera que esté precreado
        awsElasticBlockStore:
            volumeID: "vol-0e3fbaeee540b4611"
            fsType: ext4
    Y entonces  Kubernetes hace magia: MATCH entre PVC y PV... comparando metadatos.

        mi-pvc <-> mivolumen1-amazon SURGE EL AMOR !
    El desarrollador, su volumen en su pod(configurao con una plantilla: Deployment/StatefulSet) pone de tipo persistentVolumeClaim:


            apiVersion: v1
            kind: Pod
            metadata:
                name: test-ebs
            spec:
                containers:
                    -   image: registry.k8s.io/test-webserver
                        name: test-container
                        volumeMounts:
                            - mountPath: /test-ebs
                            name: test-volume
                volumes:
                    - name: test-volume
                      persistentVolumeClaim:
                            claimName: mi-pvc
```

```
                                    KUBERNETES SUSTITUYE ESTO POR:
(SACANDOLO DEL PV ASOCIADO AL PVC)
                                    awsElasticBlockStore:
                                        volumeID: "vol-
0e3fbaeee540b4611"
                                        fsType: ext4
```

Antiguamente, hace 8-10 años, los administradores de sistemas del cluster de kubernetes, tenía precreados 100 volumenes Hoy en día, lo que hacemos es montar dentro del cluster PROVISIONADORES AUTOMATICOS DE VOLUMENES PERSISTENTES

Cuando un desarrollador solicita (crea) un pvc, el provisionar (un programa que está corriendo por ahí) se da cuenta (está todo el puñetero rqato preguntando al kubernetes si hay un pvc nuevo?) y bajo demanda, se conecta con el BACKEND REAL DE ALMACEMAMIENTO:

- AWS
- AZURE
- CABINA DE DISCOS EMC2 Crea allí un volumen REAL Y genera una PV en kubernetes (una referencia a ese volumen real) y la asocia al PVC que ha creado el desarrollador.

Cuando trabajo con un cluster contratado a un cloud, el cloud ya me da preinstalado en el cluster un provisionador de volumenes que trabaja con SUS volumenes. Si me monto yo mi cluster on premise, tengo que montar mi propio provisionador de volumenes que trabaje con mis volumenes.

Si se crea un PV ... queda como disponible hasta que es vinculado a una PVC Si se crea una PVC ... queda como desasignada hasta que es vinculada a un PV

Hoy en día la tendencia es a crear primero los PVC ... y que los PV se generen en automático por un provisionador de volumenes bajo demanda. Antiguamente los admin creaban 50 pvs... y los dejaban ahí hasta que se consumían.

PVC

Una petición para que un pod solicite donde guardar sus datos.