
Universidad
Tecmilenio®

Universidad Tecmilenio

Contenerización y servicios con Kubernetes

Introducción a Kubernetes

Semana 9

Un miembro de un equipo de desarrollo de software había escuchado sobre las grandes ventajas de utilizar la plataforma de Kubernetes. Su inquietud era mejorar y optimizar sus procesos.

Al presentársela a su líder y ver los beneficios que se tienen no lo pensaron más y tomaron la decisión de moverse a la brevedad posible hacia esta plataforma.

Una gestión más adecuada de costos, mejor capacidad de respuesta a incidentes, mayor control administrativo, conversión más sencilla de requerimientos de infraestructura, son algunos de los beneficios de conocer y utilizar dicha plataforma para la optimización de procesos de desarrollo de software.

Explicación

¿Qué es Kubernetes?

Kubernetes es una plataforma de código abierto que trabaja en Linux, gestiona aplicaciones en contenedores, de tal manera que permite hacerlo en los diferentes ambientes de desarrollo, como en la nube, físico, virtuales e híbridos. Su funcionalidad principal es la capacidad para programar cargas de trabajo en contenedores en toda su infraestructura.

Explicación

¿Qué problemas resuelve Kubernetes?

Existen grandes compañías que han cambiado su forma de trabajo, pues al realizar una comparación de antes y después, es posible ver el ahorro de los recursos en general, así como una gran optimización en sus procesos (Cuemby, 2019).

Los desarrolladores reconocidos y las grandes empresas cambiaron de tener aplicaciones por separado y solas, a la tecnología de microservicios y, por tanto, se vio la necesidad del uso de contenedores y se incrementó su uso.

Kubernetes ofrece alta disponibilidad y rendimiento. Gracias a su desarrollo, puede replicarse de manera muy sencilla, cuenta con respaldos y restauración constante de la información de una manera inteligente y adecuada, que nos permite estar preparados y tener una recuperación ante desastres.

Explicación

Arquitectura de Kubernetes

Kubernetes, en su arquitectura, tiene una agrupación de aplicaciones ejecutándose simultáneamente, preparadas para lanzar microservicios de manera organizada y estructurada. Una de las funcionalidades clave de Kubernetes es la organización de los contenedores, lo cual significa que debe ejecutar varias cargas de trabajo programadas para correr en máquinas virtuales o físicas. Kubernetes se encarga de vigilar y reemplazar aquellos contenedores que no funcionan.

Clúster	Un clúster es una colección de cómputo y almacenamiento.
Nodo	Es un hospedador singular, ya sea máquina virtual o física. Su trabajo es ejecutar pods.
Master	Es el control plano de Kubernetes. Contiene varios elementos como un servidor API.
Pod	Es una unidad de trabajo en Kubernetes. Cada uno obtiene un identificador único para distinguirlo.
etcd	Contiene la configuración de cada uno de los nodos del clúster.
Programador	Es el servicio maestro responsable de administrar la carga de trabajo.
Estibador	Su primer requisito de cada nodo es Docker y ejecuta los contenedores.
Kubelet	Transmite la información de ida y vuelta del servicio plano de control e interactúa con el etcd.
Servicio de proxy	Se ejecuta en cada nodo y da la disponibilidad de servicios para el exterior.

Universidad Tecmilenio

Explicación

Arquitectura de Kubernetes

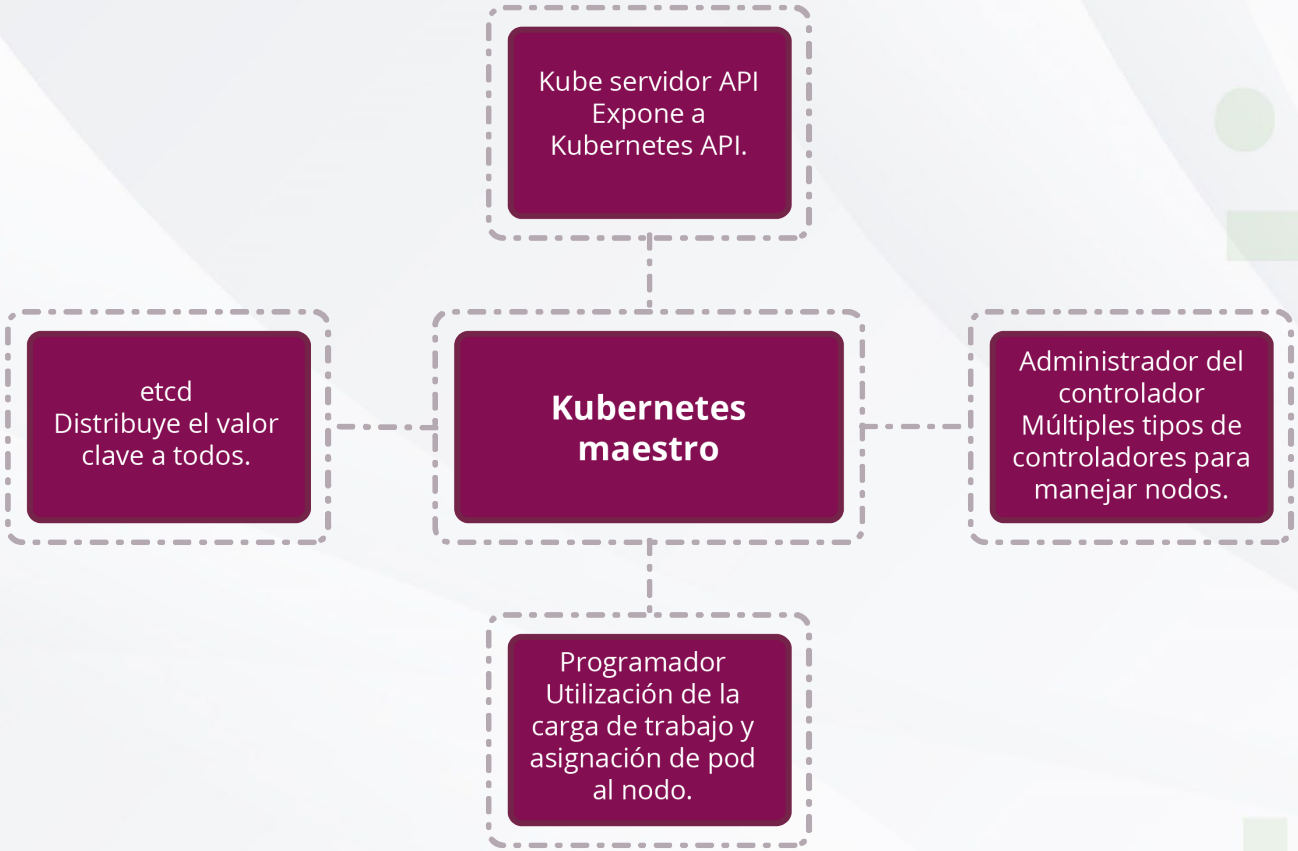


Figura 1. Kubernetes maestro y sus componentes principales: el servidor API Kube, etcd, administrador del controlador y el programador (Ramos, 2021).

Explicación

Arquitectura de Kubernetes

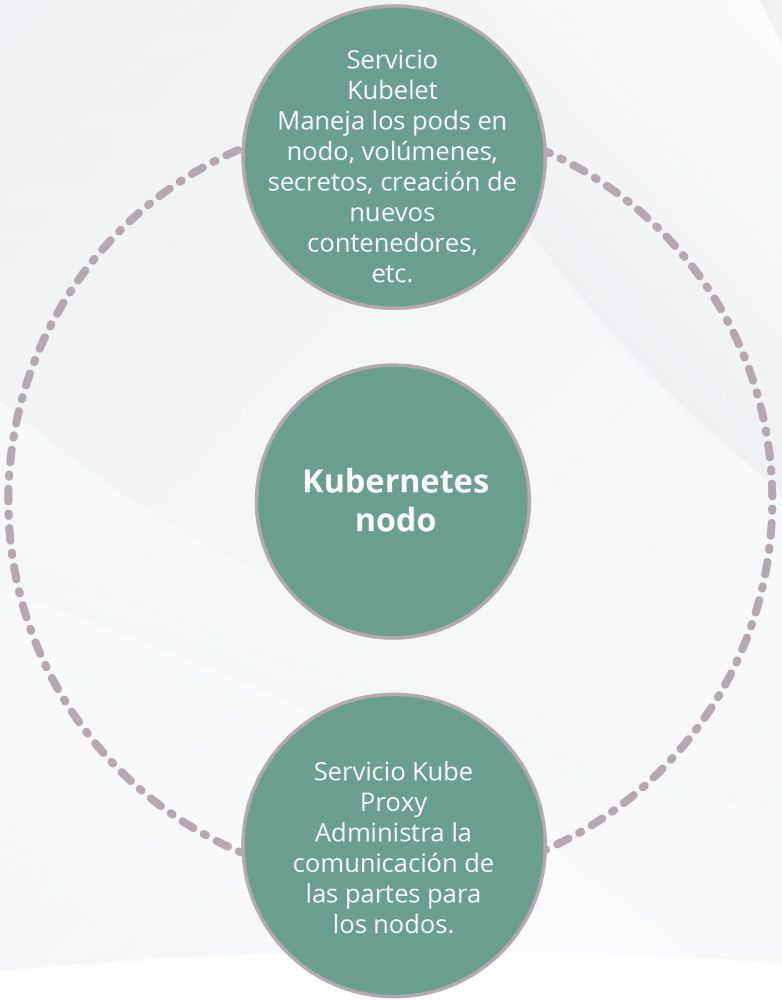


Figura 2. Kubernetes nodo y sus componentes: servicio Kubelet y servicio Kube Proxy (Ramos, 2021).

Ejercicio

Con base en lo descrito en el tema, realiza la siguiente actividad.

1. Realiza un resumen (con tus propias palabras) en el que incluyas los conceptos de la arquitectura de Kubernetes.

2. Revisa un video en donde expliquen Kubernetes, luego elabora un mapa conceptual sobre la arquitectura de Kubernetes.

¿Se podrán migrar todos los procesos de desarrollo de software de una empresa a Kubernetes?

¿Por qué Kubernetes se ha convertido en la plataforma líder?

Cierre

Existen grandes compañías que mejoraron considerablemente sus procesos, ahorrando tiempo, dinero y esfuerzo. Por ejemplo, la compañía Denso ahora lanza 10 aplicaciones al año con un ciclo de desarrollo de dos meses para aplicaciones no críticas, cuando previamente lo hacía de dos a tres años.

Si pudiste contestar las preguntas anteriores, sabes que Kubernetes realmente abre un camino de oportunidades para aquellos que trabajan con esta plataforma tan completa de contenedores orquestados, usando las buenas prácticas, nos darán cada vez mejores resultados en nuestros desarrollos tanto personales como organizacionales.

Bibliografía

Cuemby. (2019). *Grandes Empresas que usan Kubernetes*. Recuperado de https://medium.com/@cuemby/grandes-empresas-que-usan-kubernetes-d73d56334243

Ramos, C. (2021). *APRENDE KUBERNETES DE PRINCIPIANTE A EXPERTOS EN 2021: COMPRENDE LA ADMINISTRACIÓN DE CONTENEDORES MAS AVANZADA DEL 2021 (Edición En Español).* Estados Unidos: Kindle.

Universidad Tecmilenio

Contenerización y servicios con Kubernetes

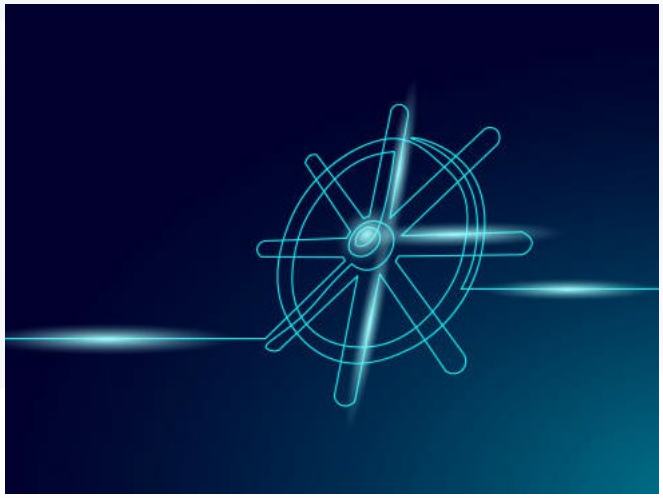
Conceptos básicos de Kubernetes

Semana 9

La relevancia que Kubernetes ha podido lograr es gracias a sus grandes ventajas y a su estructura, al desarrollar conceptos básicos, como *Pod, Secret, Ingress Controler, Service* y *Deployment.*

Por ejemplo, cuando vamos a preparar la comida, los primeros pasos que necesitamos realizar es saber el platillo o platillos que vamos a hacer, es decir, sabemos qué es lo que queremos lograr.

Lo mismo sucede con el desarrollo de aplicaciones, dependiendo de la aplicación o desarrollo específico, necesitaremos diversos tipos de definiciones de la información. Si cometemos un error, cómo podemos corregirlo, qué tipos de elementos existen y cómo podemos usarlos de manera óptima.



Explicación

Conceptos básicos de Kubernetes: Pod

Pod en inglés significa “vaina”, de acuerdo con Kubernetes (2020); en México sería como un ejote que contiene varias bolitas. De acuerdo con Morrejón (2022), un pod es la unidad básica de ejecución en Kubernetes; es el objeto más pequeño y simple que podrá encontrar dentro del clúster. Las principales características son las siguientes:

- Encapsula a uno o varios contenedores en ejecución.
- Tiene asignada una IP, todos los contenedores dentro del pod comparten la misma IP y puertos.
- Los contenedores dentro de un pod se pueden comunicar entre sí utilizando localhost.

Dentro de un pod se encuentran aplicaciones, las cuales tienen acceso a volúmenes compartidos, que son parte del pod y están disponibles para cargarse en el sistema de archivos de la aplicación. Desde el punto de vista de un Docker, un pod se presenta como un grupo de contenedores Docker con *namespaces* y volúmenes de sistemas de archivos compartidos.

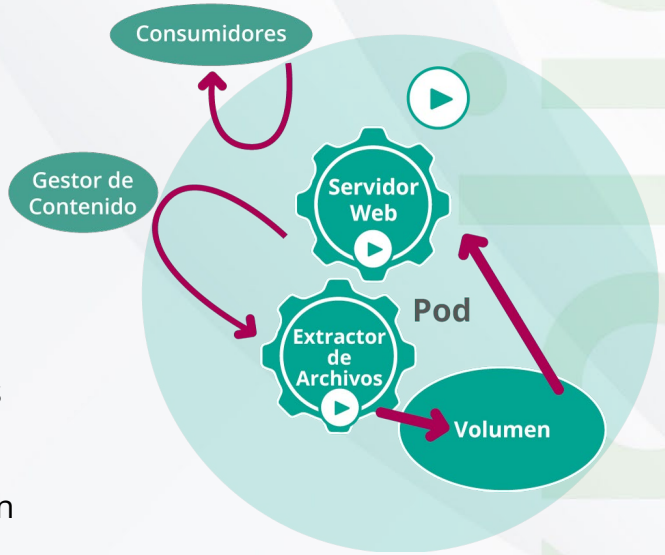


Figura 1. Pod Basado de Pods.
Fuente: Kubernetes. (2022). *Pods*. Recuperado de https://Kubernetes.io/es/docs/concepts/workloads/pods/pod/

Explicación

Conceptos básicos de Kubernetes: Secret

Son para gestionar y guardar información sensible, como pueden ser contraseñas, fichas OAuth y llaves SSH.

En Kubernetes encontramos la manera de proteger cierta información a la que no cualquiera debería tener acceso, entre este tipo de elementos podemos encontrar contraseñas, tokens o llaves.

Esta información confidencial, una vez ya salvada en un volumen secreto, se puede pasar a los pods.

La forma para declarar un objeto Secret es la siguiente:

```
apiVersion: v1
kind: Secret
metadata:
name: ejemplo1
```

Tabla 1. Declaración de objeto Secret en Kubernetes.

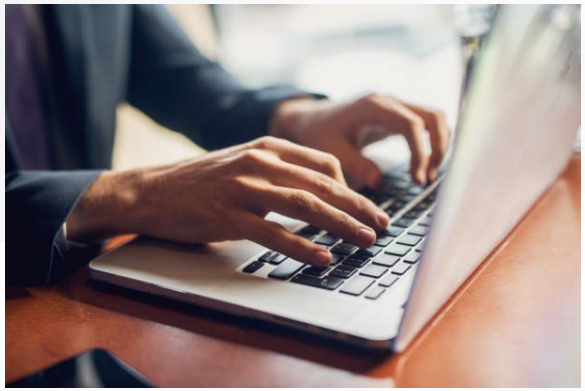


Figura 2. Secret interactuando con el pod.

Explicación

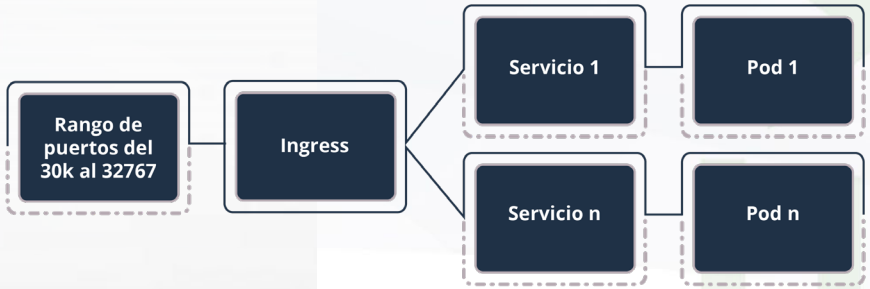
Conceptos básicos de Kubernetes: Ingress Controller

Significa “ingreso”. Este objeto va a manejar el acceso externo a los servicios, típicamente HTTP y HTTPS. Normalmente lo conocemos por acceder a las páginas web.v. Un ingress controller debe estar activado y en ejecución para que el recurso Ingress pueda funcionar.

Entre sus funciones está la administración de servicios basados en DNS (nombres de dominio) y equilibrar las cargas y terminación segura (SSL).

Azure Kubernetes Services (AKS), Contour y Istio son ejemplos de controladores Ingress. Son diferentes opciones a considerar, dependiendo de la que se adapte más al ambiente en el que estamos trabajando.

Es importante entender que es la puerta de seguridad por su exposición a recibir ataques. Como podemos ver, al utilizar la conexión mediante el rango de puertos de 30 mil al 32,767 y por medio del controlador Ingress conectar con los servicios para después llegar al pod. El tráfico es controlado por las reglas definidas en el recurso Ingress.



Explicación

Conceptos básicos de Kubernetes: Service

Este es el objeto de la API de Kubernetes, con él podemos conocer la sintaxis para acceder a las aplicaciones. El servicio (Service) determina un grupo de pods y la regla de cómo alcanzarlos. Cuando se definen, se les asigna una IP a la que después de que el pod ya no sea útil se le asignará a otro.	
Código	**Breve explicación (parte no necesaria de escribir en el código)**
apiVersion: v1	Versión de la API.
kind: Service	Versión de la API.
metadata:	
name: servicio1	Nombre del servicio que será utilizado por las aplicaciones para llamar a los pods.
spec:	
selector:	
app: App1	Conjunto de etiquetas para agrupar pods.
ports:	Puertos a usar.
port: 8080	Valor numérico del puerto usado por el servicio.
- protocol: TCP	Protocolo usado para IP (HTTP, UDP o TCP).
targetPort: 9000	Puerto del pod donde viajará la información.

Tabla 3. Declaración del objeto Service en Kubernetes.

Explicación

Conceptos básicos de Kubernetes: Deployment

El controlador de despliegues (Deployment) nos permite realizar actualizaciones de una aplicación sin interrupciones. Las actualizaciones para los pods y los replicasets son realizadas por los controladores de Deployment. Se puede usar a los deployments para definir los estados del pod, ir a una versión anterior del Deployment, escalarlo para aguantar más cargas, para pausarlo y después continuar para indicar el estado de este; para desplegar un replicaset y limpiar los antiguos.

Se muestran ejemplos de implementación (Ramos, 2021):

```
apiVersion: v1
kind: Deployment
metadata:
name: Tomcat-
ReplicaSet
spec:
replicas: 3
template:
metadata:
lables:
app: Tomcat-ReplicaSet
tier:Backend
spec:
containers:
-name: Tomcatimage:
tomcat: 8.0
ports:
-containerPort: 7474
```

Tabla 4. Implementación de Deployment

```
$kubectl créate -f Deployment.yaml –record
deployment “Deployment” created Successfully.
```

Tabla 5. Crear despliegue.

```
$kubectl get deployments
NAME DESIRED CURRECNT UP-TO-DATE
AVILAVLE AGE
Deplyment 3 3 3 33 20s
```

Tabla 6. Obtener la implementación.

```
$kubectl rollout status
deployment/Deployment
```

Tabla 7. Verificar el estado de la implementación.

```
$kubectl set image
deployment/Deployment
tomcat=tomcat:6.0
```

Tabla 8. Actualización de la implementación.

Ejercicio

Explica con tus propias palabras los conceptos básicos de Kubernetes.

Realiza un resumen (con tus propias palabras) en el que incluyas los conceptos de los objetos de Kubernetes: Pod, Secret, Ingress Controller y Deployment.

Revisa el siguiente video (haz clic aquí) en donde explican Kubernetes en Pod y Servicios. Después elabora un mapa conceptual.

Con base en la estructura de Kubernetes, ¿cuáles son los componentes principales?

Cierre

Ahora conoces los elementos básicos de la plataforma Kubernetes, como los pods y su uso para las aplicaciones, también el objeto Secret que sirve para almacenar los elementos de contraseñas y demás información sensible.

Por otro lado, aprendiste sobre el controlador Ingress, así como la importancia de la protección de la información, a través de las buenas prácticas y, finalmente, las diferentes funcionalidades de los deployments para lanzar versiones, regresar a ellas y más.

Bibliografía

Kubernetes. (2020). *Pods*. Recuperado de https://Kubernetes.io/es/docs/concepts/workloads/pods/pod/

Morrejón, M. (2022). *Érase una vez Kubernetes: Plataforma para gestionar Contenedores (Edición enero 2022)*. Estados Unidos: Kindle.

Ramos, C. (2021). *APRENDE KUBERNETES DE PRINCIPIANTE A EXPERTOS EN 2021: COMPRENDE LA ADMINISTRACIÓN DE CONTENEDORES MAS AVANZADA DEL 2021 (Edición En español)*. Estados Unidos: Kindle.

Universidad Tecmilenio

Contenerización y servicios con Kubernetes

Recursos de Kubernetes

Semana 9

Una vez que se decide trabajar con Kubernetes, es importante saber que puede derivarse un tipo de “desorden” si no se tiene cuidado en contar con una buena gestión de este, por lo que será necesario establecer un orden de trabajo para aprovechar al máximo el clúster.

Es importante conocer la forma en cómo Kubernetes permite establecer la cantidad de recursos que necesita un pod, los límites, los valores predeterminados por defectos, entre otros y, de esta manera, asegurar el éxito de la ejecución de las aplicaciones programadas.
Será importante tener en cuenta cuáles serán los sistemas que tendrán prioridad sobre otros para realizar las configuraciones adecuadas sin afectar su rendimiento al usarse por el cliente final.

Explicación

Tipos de recursos de Kubernetes

Uno de los temas más importantes al comenzar a utilizar Kubernetes es la configuración de los recursos del contenedor para asegurarse de que esté en buen funcionamiento la imagen del Docker y que pueda implementarse en el clúster de Kubernetes. Una aplicación que se esté desarrollando deberá ser implementada en el clúster de producción donde estará con otras aplicaciones, por lo que será necesario asignar recursos para el contenedor.

Según Tech (2022), los recursos informáticos son los siguientes:

- Unidad central de procesamiento (CPU) - núcleos.
- Memoria (MEM) - bytes.

Recursos de pods solicitados.	Es igual a la suma de todos los recursos que han sido solicitados por todos los pods.
Limitar recursos de pod.	Es igual a la suma de los recursos limitantes de todos los contenedores.

Explicación

Tipos de recursos de Kubernetes

Al ejecutar request.memory no hay una correlación con el Docker, ya que no tiene identificado este campo, pero sí puede preguntar si se necesita, dado que sí es un campo importante para Kubernetes y, de acuerdo con la información que contenga, kube-Scheduler decidirá en qué nodo programar el Pod.

En caso de que el contenedor alcance los límites de la memoria, el Pod se colocará en el grupo Pod que se detendrá cuando no hay memoria en el nodo. Y si el contenedor llegara a sobrepasar los límites de memoria, se hará una terminación debido a OOM-killer y se reiniciará en caso de que así esté configurado en Restart Policy.

Kubernetes cuenta con varios componentes, como el nodo principal en el que se pueden encontrar varios procesos, por ejemplo, kube-apiserver kube-controller-manager kube Scheduler.

El CPU es flexible cuando los contenedores alcanzan el límite, se inicia una aceleración del CPU. Si es alcanzado el límite del MEM, se detendrá el contenedor y se hará un reinicio, siempre y cuando esté configurado.

Explicación

Creación de un recurso en Kubernetes

En **resources** tenemos la sección *request* que va a definir la cantidad de memoria y de CPU que se necesitará como mínimo.

Una cuestión muy útil es saber cómo los nodos están utilizando los recursos. El comando “describe” puede acceder a una sección llamada Non-terminated Pods, visualiza el número total de pods y la cantidad de recursos que están solicitando cada uno de ellos.

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-requests
spec:
  containers:
  - name: web
    image: nginx
    resources:
        requests:
          cpu: "500m"
          memory: "10Mi"
```

Imagen 3. Establecimiento de recursos.

```
Non-terminated Pods:          (18 in total)
  Namespace                   Name                                CPU Requests  CPU Limits  Memory Requests  Memory Limits  AGE
  ---------                   ----                                ------------  ----------  ---------------  -------------  ---
  bookstore                   books-api-58cbc86667-g8nzk          0 (0%)        0 (0%)      0 (0%)           0 (0%)         11h
  bookstore                   books-api-58cbc86667-nnjhv          0 (0%)        0 (0%)      0 (0%)           0 (0%)         11h
  bookstore                   books-api-58cbc86667-rlncv          0 (0%)        0 (0%)      0 (0%)           0 (0%)         11h
  default                     booksapi-db4858f68-9gnhf            0 (0%)        0 (0%)      0 (0%)           0 (0%)         23h
  default                     pod-requests                        500m (26%)    0 (0%)      10Mi (0%)        0 (0%)         15m
  kings-landing               mongo-express-57cc9f88f8-b977n      0 (0%)        0 (0%)      0 (0%)           0 (0%)         23h
  kings-landing               mongo-express-57cc9f88f8-xjxj1      0 (0%)        0 (0%)      0 (0%)           0 (0%)         23h
  kings-landing               nginx-kings-landing                 0 (0%)        0 (0%)      0 (0%)           0 (0%)         7d20h
  kube-system                 coredns-698c77c5d7-bjd88            100m (5%)     0 (0%)      70Mi (1%)        170Mi (3%)     8d
  kube-system                 coredns-698c77c5d7-ldfrp            100m (5%)     0 (0%)      70Mi (1%)        170Mi (3%)     8d
  kube-system                 coredns-autoscaler-79b778686c-9jc4s 20m (1%)      0 (0%)      10Mi (0%)        0 (0%)         8d
  kube-system                 kube-proxy-2hjcb                    100m (5%)     0 (0%)      0 (0%)           0 (0%)         39h
  kube-system                 kubernetes-dashboard-74d8c675bc-p7trr 100m (5%)   100m (5%)   50Mi (1%)        500Mi (10%)    8d
  kube-system                 metrics-server-7d654ddc8b-69xbp     44m (2%)      0 (0%)      55Mi (1%)        0 (0%)         39h
  kube-system                 tunnelfront-59bc4f75f5-7mk4h        10m (0%)      0 (0%)      64Mi (1%)        0 (0%)         8d
  the-north-remembers         mongo-0                             0 (0%)        0 (0%)      0 (0%)           0 (0%)         40h
  the-north-remembers         mongo-1                             0 (0%)        0 (0%)      0 (0%)           0 (0%)         40h
  the-north-remembers         mongo-2                             0 (0%)        0 (0%)      0 (0%)           0 (0%)         40h
```

Imagen 4. Cantidad de recursos usados por pod.

Explicación

Creación de un recurso en Kubernetes

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-requests
spec:
  containers:
  - name: web
    image: nginx
    resources:
        limits:
          cpu: "1"
          memory: "50Mi"
```

Imagen 5. Establecimiento de limits.

La memoria es el tema más relevante al momento de establecer limits, dado que la memoria que puede llegar a utilizar un pod, por el contrario, a la del CPU, es más difícil de liberar. En caso de no realizarse este paso, un pod podría asegurar toda la memoria disponible y afectar a otros pods que se encuentren en el nodo incluyendo a los nuevos, porque kuber-Scheduler va a realizar las asignaciones con base en el espacio disponible, tomando como referencia la sección Allocatable del nodo, pero sin tomar en cuenta la memoria que se está utilizando.

Explicación

Creación de un recurso en Kubernetes

Kubernetes cuenta con un sistema de prioridades para atender dichos casos y categoriza a los pods de acuerdo con tres clases llamadas Quality of Service (QoS) (Torres, 2020):

Si se requiere que el pod tenga la categoría de Burstable, se necesitan poner solamente los requests, sin incluir los limits, o bien, estableciendo limits que no sean iguales a los requests. En caso de ser requerido, se puede contar con recursos extendidos, los cuales son nombres de recursos clasificados fuera del domin bbbio de Kubernetes.io.

Los dos pasos para utilizar dichos recursos según Kubernetes (2022), son los siguientes:

1. El operador del clúster deberá anunciar el recurso extendido.
2. Los usuarios deben solicitar el recurso extendido en los pods.

Ejercicio

1. Investiga ejemplos de códigos de configuración y describe la utilidad de cada uno de ellos:

- Unidad central de procesamiento (CPU) - núcleos.
- Memoria (MEM) - bytes.
- Recursos de pods solicitados.
- Limitar recursos de pod.

2. Con base en las tres clases de Quality of Service (QoS) mencionados a continuación, investiga y ejemplifica, para cada una de ellas, un caso real de aplicaciones y escribe el código usado para poder utilizarla en el desarrollo de tres sistemas (puedes referenciar a través de códigos de ejemplos usados en empresas que ya lo hayan implementado).

- BestEffort.
- Guaranteed.
- Burstable.

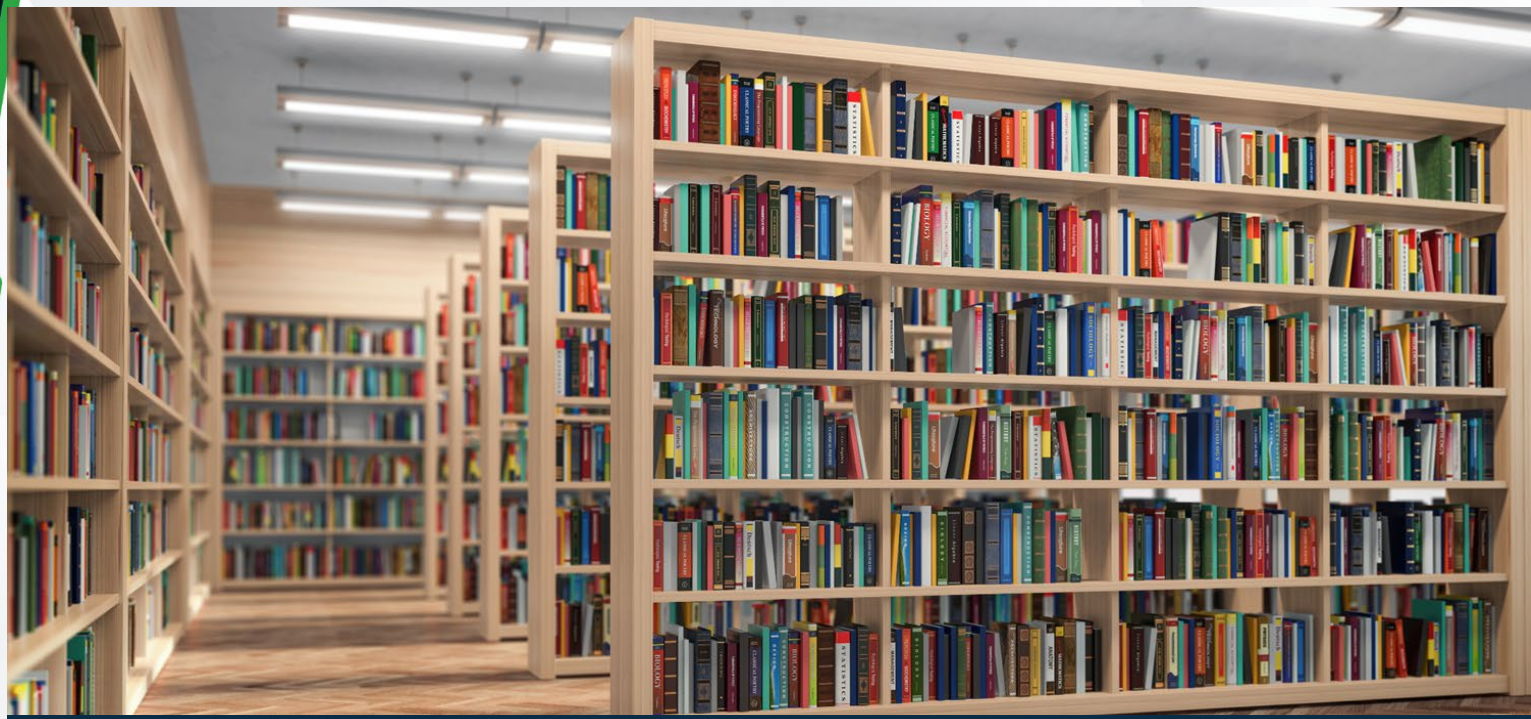
¿Cuál debería ser la configuración más adecuada para configurar dichos elementos en una aplicación?

¿Qué elementos son determinantes para elegir que una aplicación siga ejecutándose por encima de la otra?

Cierre

Como se ha visto a lo largo de este tema, existen diversos recursos que pueden usarse cuando se trabaja con Kubernetes para lograr que las aplicaciones y sistemas puedan funcionar de manera óptima. Al solicitar el uso de recursos compartidos, la determinación de los elementos de request y limits marcará la gran diferencia en el momento en que el procesador tenga elegir entre un pod y otro cuando se tenga una saturación en el uso de memoria y, de esta manera, se establecerán prioridades de acuerdo con el QoS que se lleve a cabo.

Bibliografía



Kubernetes. (2022). *Administrando los recursos de los contenedores.* Recuperado de https://Kubernetes.io/es/docs/concepts/configuration/manage-resources-containers/

Tech. (2022). *Cómo acceder a los recursos del pod de Kubernetes*. Recuperado de https://tech-es.netlify.app/articles/es516014/index.html

Torres, G. (2020). *Administrar los recursos para tus contenedores en Kubernetes*. Recuperado de https://www.returngis.net/2020/05/administrar-los-recursos-para-tus-contenedores-en-Kubernetes/