Redes-IC-7602

Apuntes Clase 2023-02-10

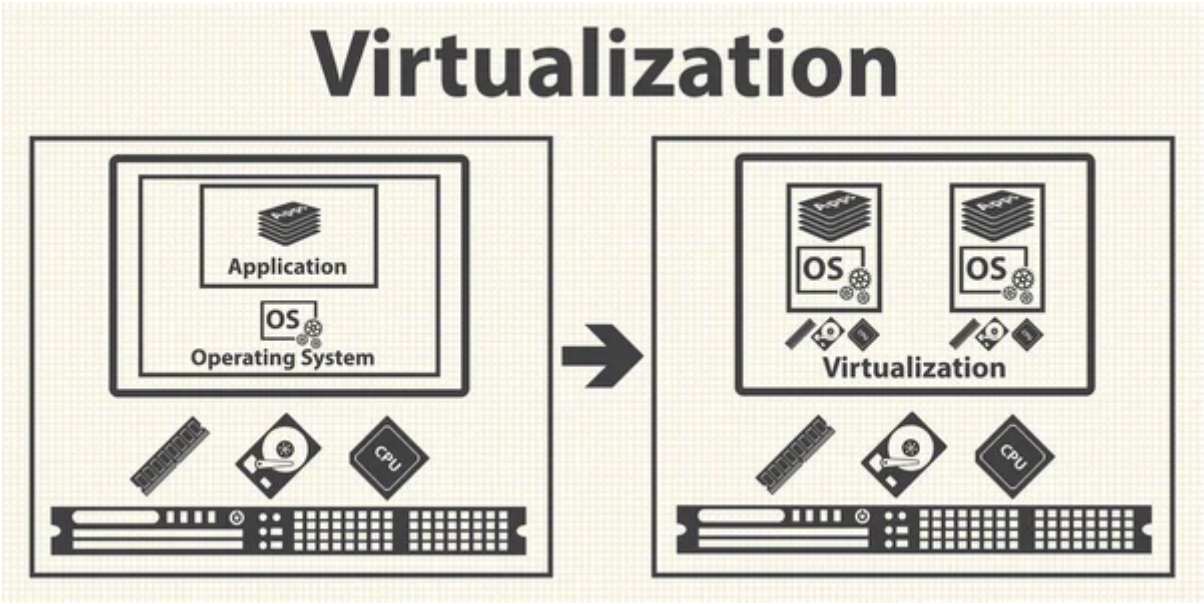
Leonardo David Fariña Ozamis - 2020045272

En este documento podremos observar de manera general los temas relacionados a Kubernets (contenedores, namespaces, comandos, conexiones, etc.).

Introducción

Docker es una plataforma de virtualización que permite la creación de entornos aislados y portables para la ejecución de aplicaciones. Esta tecnología utiliza contenedores, que son unidades de software que incluyen todo lo necesario para que una aplicación funcione de manera autónoma. De esta manera, Docker facilita el despliegue y la gestión de aplicaciones en cualquier infraestructura, sin importar el sistema operativo o las dependencias de software necesarias.

En la siguiente imagen se puede observar cómo pasamos de usar los recursos de manera directamente en nuestro S.O. a empezar a administrar los recursos de manera "virtual" (por eso hablamos de virtualización) para poder "simular" estos recursos dentro de nuestras máquinas, esto haciendo posible poder tener un mejor control y manejo sobre los recursos que tenemos(memoria, CPU y almacenamiento).



Al empezar a utilizar virtualización se dieron cuenta que esta capa de virtualización empezaba a hacer muy lento todo el entorno de virtualización, lo cual no era eficiente, por lo que para poder mejorar la eficiencia de este hemos de quitar todo lo que no nos sirva directamente para esa tarea, ya que los S.O. traen un montón de procesos que no son necesarios, por lo que solo queremos virtualizar lo necesario para cumplir nuestra tarea por lo que reducimos nuestros S.O. a S.O. con lo mínimo y necesario, a estos se les llama "HiperVisors", S.O. muy chiquitos. Esto con el fin de poder optimizar la eficiencia con los recursos de nuestro hardware.

Kubernetes es una plataforma de orquestación de contenedores que permite desplegar, gestionar y escalar aplicaciones en contenedores de manera eficiente y automatizada en un entorno de infraestructura en la nube o local. Kubernetes se encarga de administrar la distribución de recursos, la tolerancia a fallos, la escalabilidad

y la disponibilidad de las aplicaciones, lo que permite a los desarrolladores enfocarse en la creación de aplicaciones y no en la infraestructura subyacente. Esto usando Docker. La virtualización empezó alrededor de hace 20 años. Al hablar de contenedores se trata de la virtualización de procesos y no del S.O.

Redes Virtuales

Las redes virtuales han cambiado mucho estos últimos años por temas de protocolos, seguritdad, etc. A la hora de virtualizar contenedores y querer que estos se comuniquen entre si necesitamos definir como se hará. Este enfoque es sobre como Kubernets lo hace.

Consejo **NO HACER DEPENDECIAS POR IP**, ya que estas van cambiando a medida que los contenedores se levantan o mueren. No hay una fuerte dependencia entre contenedores, esto para poder aislar los contenedores con sus tareas específicas y poder tener un mejor control sobre estás.

En Docker se maneja un servicio D.N.S.(Domain Name System) que este maneja una tabla con las IPs de los contenedores.

A continuación vemos los tres tipos de redes que podemos configurar en Kubernets.

1. ClusterIP: Ip fija dentro del cluster que hace el balance dentro entre los pods Ip privada que usa el selector.
2. NodePort: Puerto que dirije el trafico (lo exponemos a la red externa(internet)). Atado por la IP de cada nodo, tambien usa un selector pero crea un puerto en cada nodo que es publico, al que yo puedo entrar siempre que tenga un IP publica.
3. LoadBalancer: Nube, balanceador de carga para la nube, tambien un selector.

Conceptos de Docker

1. Layers: Es basicamente una instrucción que agrega un componente a la imagen.
2. Imagen: Una imagen de Docker es un paquete que contiene todos los componentes necesarios para ejecutar una aplicación, incluyendo el código, las bibliotecas y las dependencias.
3. DockerHub: En este sitio podemos encontrar imagenes ya subidas.
4. DockerFile: Es un pequeño archivo con instrucciones que cada una de estás va a tener una repercusión directa en nuestro DockerImage. 1.Algunas instrucciones son CMD,COPY,RUN y WORKDIR.

Conceptos de Kubernets

1. Contenedor: Es una instancia en tiempo de ejecución de una imagen de Docker.
2. Pod: Es la unidad básica de despliegue en Kubernetes. Es un grupo de uno o varios contenedores que comparten un espacio de red y un volumen de almacenamiento. Los pods permiten que los contenedores se comuniquen entre sí y con el exterior, y facilitan la gestión de su escalado y su tolerancia a fallos.
3. Deployment: Set de Pods que pueden correr en un solo nodo,es un objeto de Kubernetes que define cómo se deben crear y actualizar los pods y los contenedores en un cluster. Permite que los usuarios especifiquen el número de replicas que se deben crear, las estrategias de actualización y los parámetros de salud del despliegue.
4. DaemonSets: Set de Pods que corren cada uno en un nodo diferente, es un objeto de Kubernetes que asegura que un pod se ejecute en cada nodo del cluster. Esto es útil para tareas que requieren ser ejecutadas en todos los nodos, como la recolección de logs o la monitorización de los recursos.

5. StatefulSets: Es un objeto de Kubernetes que permite la gestión de aplicaciones con estado, como bases de datos, servidores web y sistemas de ficheros. Permite que los pods tengan identidades únicas, persistentes y estables, lo que garantiza su escalado y su recuperación ante fallos.
6. Jobs: Es un objeto de Kubernetes que se utiliza para ejecutar tareas únicas, como procesos batch o trabajos programados. Permite que los usuarios definan el número de replicas que se deben crear y la duración máxima de la tarea.
7. CronJobs: Es un objeto de Kubernetes que se utiliza para programar tareas que se deben ejecutar de manera periódica, como limpieza de logs o copias de seguridad. Permite que los usuarios definan el horario de ejecución de la tarea.
8. Namespaces: División logica de datos en kubernetes.

Algunos Comandos de ayuda

1. kubectl get nodes // ver nodos
2. kubectl config get-contexts // ver contextos
3. kubectl get ns // ver namespaces
4. kubectl -n kube-system get pods // ver pods de kube-system
5. kubectl -n kube-system get pods -o wide // ver más información de pods de kube-system
6. kubectl -n kube-system delete pod "name-pod" // eliminar pod de kube-system
7. kubectl apply -f "name-pod" // manifiesto de pod
8. kubectl delete pod "name-pod" // eliminar pod
9. kubectl exec -it "name-pod" -- sh // entramos dentro de nuestro contenedor del pod - ctrl+d salimos
10. kubectl get pod "name-pod" -o yaml // nos muestra la info del yaml de nuestro pod más las variables por defecto si no la definimos
11. kubectl describe pod "name-pod" // ver los estados del pod y sus eventos
12. kubectl get statefulsets // nos muestra los statefulsets
13. kubectl get all // Ver todo nuestros objetos
14. kubectl describe svc "namesvc" // Ver los endpoints de los pods e información del svc
15. kubectl exec -it "namecontainer" -- bash // entramos dentro de nuestro contenedor del pod - ctrl+d salimos

Programa de ayuda

Lens es un programa donde podemos visualizar el estado de nuestros objetos de Kubernets. Podemos descargarlo desde aquí.

Recomendación para ver que tenemos nuestros objetos corriendo en buen estado podemos ir a **WorkLoads** y despues en **OverView**, ahí podremo visualizar errores, advertencias y de manera general el

estado de nuestros objetos.