Centro de Procesamiento de Datos

Práctica 7. Kubernetes 101

Objetivo:
Introducción a Kubernetes como entono para desplegar un sistema escalable de contenedores.

Presentar un documento pdf en SWAD→Actividades→ Práctica 7 con la siguiente información:

-(obligatorio): Realizar diversas capturas donde se muestren de los apartados:

II.a) Captura del Dashboard.
II.b) Captura donde aparece el puerto modificado.
II.c) Captura donde se muestran los Deployment y Services.
II.d) Captura donde aparece personalizado el NodePort.
II.e) Captura donde personalizamos el enrutamiento de nivel 4.
II.f) Captura donde personalizamos el mensaje de salida en el almacenamiento persistente.
II.g) Captura de la ejecución de la automatización con *forge*.

Desarrollo:

Esta práctica nos familiarizamos con diversos recursos que utiliza Kubernetes.

I) Kubernetes

Kubernetes permite automatizar el despliegue y escalado de contenedores. Es una alternativa más potente y completa a Docker Swarm. La información completa de Kubernetes la podemos encontrar en: https://kubernetes.io/docs/

Veamos algunos conceptos básicos:

- **Cluster:** Conjunto de máquinas físicas o virtuales y otros recursos utilizados por Kubernetes.
- **Nodo:** Una máquina física o virtual ejecutándose en Kubernetes donde los Pods pueden ser ejecutados.
- **Pod:** Conjunto de contenedores y volúmenes. Son la unidad más pequeña desplegable. Todos los contenedores de un Pod se ven ya que virtualmente se encuentran en el mismo host (ej. localhost)
- **Deployment o ReplicaSet: (Controllers):** Gestor de Pods que asegura que están funcionando las réplicas y permite escalar de forma fácil. Una réplica es una copia exacta de un Pod. Levanta Pods en caso de fallos o reinicios. Deployment permite actualizar de una forma más controlada las aplicaciones
- **Services:** Define como acceder a un grupo de Pods.
- **Namespaces:** Establece un nivel adicional de separación entre los contenedores que comparten los recursos de un clúster.

- **Configmap:** Servicio para gestionar la configuración de nuestras aplicaciones (Una buena práctica es separar el código de nuestra aplicación de la configuración).
- **Secrets:** Servicio para gestionar los secretos de nuestras aplicaciones.
- **Volumes:** Servicio para gestionar la persistencia de los contenedores.

II) Primeros pasos con Kubernetes

Iniciamos el entorno:
https://www.katacoda.com/courses/kubernetes

II.a) Lanzamos: "Launch a multi-node cluster using Kubeadm"

En este apartado aprenderemos a utilizar Kubernetes con 2 nodos. Básicamente:
- Iniciamos los nodos de Kubernetes con Kubeadm
- Activamos la capa de red que permite que se vean entre sí de forma transparente todos los contenedores situados en diversas máquinas, Container Network Interface (CNI). En este ejemplo utilizamos WeaveWorks, aunque existen diversos entornos de red para Kubernetes (Calico, Flannel, ...)
-Iniciamos un Pod
-Activamos el Dashboard: Entorno Web para mostrar los diversos recursos activos en Kubernetes.

II.b) Lanzamos "Deploy Containers Using Kubectl"

En este apartado:
-Lanzamos minikube
-Ejecutamos un despliegue y exportamos el puerto para acceder
-Escalamos

II.c) Lanzamos "Deploy Containers Using YAML"

En este apartado:
-Creamos un Deployment y un Servicio con ficheros YAML.
Un Deployment se asegura de que grupo de uno o más pods esté siempre disponible. Un Service es una abstracción que define un grupo lógico de Pods y una política de acceso a los mismos.

II.d) Lanzamos "Kubernetes - Networking Introduction"

En este apartado evaluamos distintos elementos de comunicación:

- **Cluster IP**: Modelo por defecto cuando se crean Servicios. Se asigna un IP interna visible por los Pods.
- **Target Ports**: Nos permite asignar otro puerto distinto al puerto donde donde escucha la aplicación. En el fichero YAML, *TargetPort* es el puerto de escucha de la aplicación, y *Port* es el puerto que será accesible externamente.

- **NodePort**: los modelos anteriores TargetPort y ClusterIP están visibles sólo en la red interna. NodePort expone el servicio en el puerto especificado y en la IP del nodo.
- **External IPs**: De esta forma el servicio queda visible externamente.
- **Load Balancer**: Permite que el balanceo de carga se realice externamente por el proveedor de Cloud.

II.e) Lanzamos "Create Ingress Routing"

En este apartado estudiamos la capacidad de Kubernetes para hacer enrutamiento de Nivel 4. En este ejemplo, en función de la url que solicita el cliente, la solicitud http puede derivarse a distintos Deployments.

II.f) Lanzamos "Running Stateful Services on Kubernetes"

En esta apartado:
-Activamos un servidor NFS.
-Desplegamos los volúmenes persistentes (PV)
-Desplegamos los recursos que demandan volúmenes persistentes (PVC)
-Utilizamos los volúmenes
-Borramos y reactivamos otro Pod para verificar el almacenamiento persistente.

II.g) Lanzamos "Deploying a service from source onto Kubernetes"

En este apartado:
- Creamos un contenedor Docker y a lanzarlo dentro de Kubernetes, definiendo límites de uso de CPU y memoria.
-Utilizamos un registro de contenedores. Aunque utilizamos el propio de Katakoda, existen otros servicios de almacenamiento de contenedores de pago (Google, Azure, AWS, …) o gratuitos: (Docker hub, https://www.canister.io/ , https://treescale.com/).
-Utilizar forge para automatizar el lanzamiento.

Para ampliar podemos completar el resto de apartados de Katakoda/Kubernetes. También podemos utilizar el laboratorio la introducción a Kubernetes:
https://training.play-with-kubernetes.com/