Con la aparición de las aplicaciones ejecutando dentro de los contenedores, es indispensable

observar la cantidad de logs que se generan constantemente para verificar el correcto funcionamiento de distintos componentes: sistemas, objetos de Kubernetes, contenedores y almacenar todos sus logs en un disco **persistente** y poder filtrar por nombre de espacio, fecha o nombre del objeto más tarde.

Se propone crear un posible entorno de laboratorio o en torno controla la cual se pudiera dividir en dos etapas

* Primera etapa entrono local o en la nube pero bajo el esquema de IaaS
* Segunda etapa replicar lo obtenido en la primera etapa ahora en un entorno de Cloud bajo PaaS o SaaS

Para ello, en primer lugar, se desplegaría un clúster de Kubernetes en el entorno local o en nube (IaaS) utilizando máquinas virtuales (VMs). Se tiene la posibilidad de analizar de obtener los (logs) de los contenedores. Para ello existen distintas herramientas que permiten analizar los (logs), sin embargo, cuando el contenedor se detiene, sus logs desaparecen. La conservación de dichos logs es importante para llegar al origen del problema. Tras analizar carencias de los comandos y funcionalidades que ofrece Kubernetes de forma nativa, se podría instalar las herramientas **(Elasticsearch, Logstash y Kibana)** sobre el clúster.

En un clúster K8S formado por miles de pods, se hace indispensable la implantación de un sistema que sea capaz de registrar y almacenar logs en un lugar seguro fuera del clúster. El reto del registro de logs reside en que tras desaparecer el pod, su log deja de estar accesible por el administrador del clúster. La complejidad reside en que los contenedores que se ejecutan dentro de los pods poseen su propio sistema de ficheros, por tanto, cuando se escala el número de pods, es posible que algunos contenedores dejen de existir y por tanto desaparezca su sistema de ficheros (incluyendo logs).

Una de las ventajas que tiene Kubernetes es que las aplicaciones son fácilmente escalables aumentando o reduciendo el número de pods, por tanto, disponer de un sistema de persistencia de logs es fundamental.

Los logs pueden dar información concreta y detallada acerca de los eventos que tantos usuarios individuales, administradores, motor de contenedores (docker), kubelet y otros componentes del sistema ha realizado en el sistema. Además, proporciona un conjunto de registros ordenados de forma cronológica y permite responder a cuestiones como:

• ¿Qué ha pasado? ¿cuándo ha pasado? ¿quién lo ha iniciado? ¿sobre qué ha pasado?¿dónde se ha observado?¿desde dónde se ha iniciado? ¿Por qué un pod muere?

**Plan de trabajo**

En este apartado trato de explicar los pasos a seguir para implementar la solución, se distinguirá la solución para el entorno local y para el entorno de nube.

**Entorno local**

En primer lugar: se crearán tres máquinas virtuales con cualquiera de estos hipervisores: (VirtualBox o Hyper-V o VMware) se instalará Kubernetes mediante Ansible. Se instalarán dos nodos trabajadores y un nodo máster. Para verificar la correcta instalación se comprobará el estado de los nodos del clúster.

En segundo lugar: se investigaria sobre los comandos que se pueden ejecutar con kubectl para obtención de logs de un pod y se investigaria la perdurabilidad en el tiempo de dichos logs cuando el pod esté caído o no se encuentre disponible. Si tras la muerte de un pod, los logs desaparecen del nodo que ejecutaba dicho pod, se analizaría la posibilidad de extraer dichos logs fuera del pod.

Además, se estudiaría la posibilidad de filtrar los logs por cada contenedor, se estudiará el cuadro de mando ofrecido por la API de Kubernetes como alternativa para obtener logs.

En tercer lugar: Si hubieran las posibilidades de las soluciones nativas, se instalará la pila de herramientas ELK. Las tres herramientas se instalarán como pods dentro del nombre de espacio Ej.: (kube-estorage-log-system), sobre esos pods, se despliegan servicios que permiten acceder a las interfaces gráficas de dichas herramientas (Elasticsearch y Kibana). Todas las imágenes se descargarán del repositorio de imágenes Docker de Elastic.

**Entorno nube**

Para la solución en la nube, se puede utilizar cualquiera de los tres grandes proveedores de Cloud (Google Cloud, AWS o Microsoft Azure) Pero puede ser desplegado en cualquier otro que cumpla con los requerimientos técnicos según sea el caso.

**Por una parte**: se crean tres máquinas virtuales, luego se accederá a ellas vía SSH y se instalará Kubernetes vía Ansible. Estará formado por un nodo máster y dos nodos trabajadores al igual que en el modo local. A continuación, se instalará la pila ELK sobre el clúster montado y se estudiaria la posible gestión para centralizar los (logs).

**Por otra parte**: se desplegará Kubernetes como PaaS o SaaS (funcionalidad nativa) y se estudiará la herramienta nativa para obtención de logs. Esta opción no requiere configuración ni instalación ya que está integrada de forma nativa

Elasticsearch, Logstash y Kibana serían un conjunto de herramientas de código abierto que combinados ofrecen una solución al problema de persistencia de logs. Permite la monitorización y el análisis de los logs. Mediante estas herramientas se pretende recopilar toda la información de los logs, procesarla y almacenarla de forma distribuida para posteriormente realizar consultas sobre los logs. A continuación, se comenta que hace cada herramienta de forma individual.

**Elasticsearch** se trata de una base de datos distribuida que se instala en todos los

nodos, haciendo que sea tolerante a fallos y tenga alta disponibilidad.

**Logstash** realiza el pre procesamiento de los logs y envía la información a la base de datos. Realiza tres etapas: Parte de entrada, filtro y salida.

• Entada: Recoge información de los agentes emisores de beat. Un beat es un elemento que produce información.

• Filtro: Agrega o elimina metadatos a los logs para facilitar posterior filtrado y procesado por Kibana.

• Salida: Envía los logs procesado a Elasticsearch.

**Kibana:** se trata de una herramienta visual que realiza consultas a Elasticsearch para filtrar y obtener información en forma de cuadros de mando.