El ecosistema de los contenedores en general es:

- Desconocido.
- Inestable.
- Repleto de tecnologías bastante inmaduras.

El Reto: entender y aprender patrones en los que los contenedores fracasan.

De ahí que el Monitoring se vea como un mecanismo para aprender y anticipar a fallos.

Infraestructura para monitoring de contenedores

Una infraestructura para monitoring contenedores requiere:

- Un sistema de monitoring basado en contenedores.

Estos sistemas presentan una arquitectura jerárquica:

- Un sistema central monitoreo los clusters.
- Cada host del cluster monitoreo sus contenedores.

Las características de una infraestructura para monitorear contenedores:

- Ligera.
- Alta disponibilidad.
- Escalable.
- Contenerizada.
- Distribuida.
- Capaz de reducir el ruido para evitar distracción.
- Que tenga más código para analizar métricas que para coleccionarlas.

Las partes de una infraestructura de este tipo son:

- Logs
- Colección y almacenamiento de datos.
- Exploración del sistema.
    - Alertas o Notificaciones.
    - Análisis de los datos.

Logs

Es necesario visualizar los logs para saber que esta pasando en mí aplicación.

- Coleccionarlos.
- Procesarlos.
- Almacenarlos.

Docker ofrece un comando muy sencillo para mostrar los logs en stdout:

```
$ docker logs CONTAINER_NAME
```

Docker ofrece la posibilidad de utilizar diferentes plugins para redireccionar los logs:

- Json-file
- Fluentd.
- Journald.
- Syslog.
- Awslog.
- Gelf.

Este plugin hay que configurarlo cuando arranquemos el Docker daemon:

```
$ docker daemon --logs-driver=DRIVER_NAME
```

Logs

Una de las arquitecturas tradicionales y efectivas es el uso y almacenamiento de logs es:

- Elasticsearch.
- Logstash.
- Logstash-forwarder.
- Kibana.

Logs – Consejos

Una de las arquitecturas tradicionales y efectivas es el uso y almacenamiento de logs es:

- Descubrir lo deconocido a través de los logs.
- Minimizar el uso de **docker logs** ya que es CPU intensive.
- Usar containers para los componentes de ELK.
- Use logging levels (DEBUG, INFO, ERROR, etc…).

Colecciona y almacena métricas

Tres niveles de métricas son almacenados y analizados.

Uso de contenedores para instalar y correr estas aplicaciones:

- Influx DB
- Grafana
- cAdvisor

Colecciona y almacena métricas – Consejos –

- El uso de InfluxDB puede ser problemático, uso de CPU, no estable.
- cAdvisor puede ser CPU intensive.
- Distinguir entre métricas a nivel de cluster, host y contenedor.
- Limitar los recursos de los contenedores a nivel de CPU y Memoria.

Exploración del Sistema

- Prevenir y identificar patrones de errores antes de tiempo.
- Definir alertas de una manera jerárquica.
  - Cluster => Hosts => Contenedor

Principales herramientas:

- Riemann (stream processing system)
- Sensu (monitor router) - Host and cluster alert clients
- Comunicación:
  - Slack
  - Mailgun
  - OpsGenie

Otras herramientas usadas para explorar tu sistema (a nivel de host):

- Sysdig
- htop
- strace
- tcpdump
- sysstat
- tcpflow
- perf_events
- iotop
- …

Exploración del Sistema – Consejos –

Otras herramientas usadas para explorar tu sistema (a nivel de host):

- Uso de Riemann (cadvisor plugin for riemann)
- Evalua la latencia/ancho de banda
- Docker tareas de limpieza:
  - Images se apilan en el sistema de ficheros.
  - Contenedores se apilan en el sistema de ficheros.
- Uso del canal de comunicaión más adecuado para cada alerta (los emails no se leen).
- Healthchecks para contenedores de usuario o los nuestros propios.

- El ancho de banda difiere entre hosts y contenedores.
- Errores en el almacenamiento (AWS vols, quotas, movimiento de datos).
- Corrupción del sistema de ficheros.
- Mapeo de puertos con Docker (port binding, OOM, btfrs/overlay (readonly)).
- ICMP, SNMP pings.

Conclusiones

- Usa soluciones que esten contenerizadas. :D
- Usa las herramientas que mejor se adaptan a tu problema.
- Colecciona métricas para prevenir y entender los desconocido.
- Analiza los datos coleccionados para detectar patrones en los fallos.