Despliegue de aplicaciones contenerizadas: un caso de implementación basado en Crane

José Miguel Silva Pavon [1], Franco Bellino [1], Patricia Bazán [3], Alejandra B. Lliteras [2,4], Jose Arcidiacono [1,] Nicolás del Rio[1]

UNLP, Facultad de Informática [1], LINTI [2] UNLP, Facultad de Informática, LIFIA [3] UNLP, Facultad de Informática, LINTI, [4] CICPBA

js.silva.010@gmail.com, fran85bellino@gmail.com, pbaz@info.unlp.edu.ar, alejandra.lliteras@lifia.info.unlp.edu.ar, jarcidiacono@linti.unlp.edu.ar, ndelrio@info.unlp.edu.ar

RESUMEN

Crane es una herramienta para el despliegue local de aplicaciones en contenedores, liviana, de propósito general y con capacidades de escalado automático, lo que la diferencia, por ejemplo, de la herramienta Minikube, la que permite algunas pruebas locales de la API de Kubernetes y se utiliza principalmente para el desarrollo de nuevas funcionalidades para K8s.
El diseño de Crane presentado en [Arcidiacono et al., 2022] contempla la implementación de la herramienta con un componente de back-end para la creación y despliegue de contenedores y un componente de front-end orientado a un usuario final que debe desempeñar el rol de devop dentro de un proyecto [Httermann M, 2012].
El objetivo de este trabajo es proporcionar un servicio REST [Fielding & Taylor, 2002] para crear y desplegar servicios docker[1] con sus correspondientes herramientas de medición, definición de políticas de escalado y gestión de alertas para la toma de decisiones.

CONTEXTO

Este proyecto surge con el fin de implementar una herramienta para el despliegue local de aplicaciones en contenedores y que sea liviana, de propósito general y con capacidades de escalado automático.

DEHIA es un administrador de flujo de trabajo para la recopilación de datos con intervención humana [Arcidiacono et al., 2020]. Su arquitectura se basa en microservicios. En un primer intento de uso de DEHIA para la entrega en un servidor local, resultó ser complejo y difícil de replicar debido a sus diversos componentes y tecnologías.
Una posible solución a este problema es la contenerización [Bullington-McGuire et. al, 2020]. Para comenzar, se eligió el componente más simple, una puerta de enlace. Este componente no tiene dependencias funcionales con los otros componentes y no tiene estado interno.
El componente de puerta de enlace sólo necesita un puerto abierto (para recibir las solicitudes que tiene que redirigir) y espera un pequeño conjunto de parámetros. Por eso, era viable implementarlo automáticamente. Para ello se decidió desarrollar una herramienta de despliegue automático de Docker (denominada “Crane”), con el añadido de escalar la aplicación bajo demanda, creando nuevas instancias.

1- INTRODUCCIÓN

En la actualidad la separación entre la infraestructura y el desarrollo se ha reducido considerablemente con el uso de servicios

[1] https://www.docker.com/

PaaS [Rani & Ranjan, 2014] los cuales son entornos completos de desarrollo y despliegue en la nube, con recursos que permiten construir y escalar aplicaciones sin necesidad de preocuparse por la infraestructura.

Esta posibilidad hizo que muchos desarrolladores optaran por construir y desplegar aplicaciones directamente en la nube.

Un problema que surge con el despliegue en la nube, es que los costos que genera el alto rendimiento de estos servicios alojados en un PaaS imposibilitan a los desarrolladores hacer pruebas de rendimiento en etapas tempranas, construir y probar aplicaciones escalables o analizar el comportamiento de la misma como servicio distribuido.

Los desarrolladores se terminan "chocando" contra una gran pared al intentar simular esta infraestructura en un entorno local. Ya sea por el hecho de no poseer recursos para alojar los servicios o por el tiempo y la complejidad que lleva toda la implementación.

A su vez un despliegue local tiene dependencias, como por ejemplo versiones de librerías y sistemas operativos donde puede funcionar, esto supone una grave limitación a la hora de migrar el sistema a otro host, ya que se tendría que adaptar una solución local a cada uno de los entornos donde se vaya a ejecutar.

Con el uso de un ambiente de virtualización como Docker [Rad et al., 2017] podemos desplegar la solución en contenedores los cuales se podrán migrar a otro host independientemente de sus características, siempre y cuando tenga Docker instalado.

Una vez realizado el despliegue, surge la necesidad de orquestar, medir y escalar estos contenedores bajo demanda. Existen soluciones como Kubernetes [Burns et al., 2022] pero requiere una gran cantidad de recursos para funcionar, por lo que se hace necesario que además de ser portable, use pocos recursos.

También se encuentra otro inconveniente y es que muchas veces para configurar contenedores es necesario crear previamente archivos de configuración estáticos, que son usados a la hora de desplegar los mismos y que dependen de rutas locales que no son iguales en todos los hosts.

Es por esto que se toma como base lo propuesto por [Arcidiacono et al., 2022] donde se habla sobre Crane una herramienta que permite optimizar el proceso de despliegue y control de servicios, implementado una API REST que siga esa pauta y permita interactuar y administrar aplicaciones completas de una manera flexible, simple y rápida, pudiendo así disminuir la distancia entre el desarrollo y el despliegue.

2- LÍNEAS DE INVESTIGACIÓN Y DESARROLLO

La línea de investigación presentada apunta a la creación de aplicaciones seguras, con reglas claramente definidas y en donde toda la complejidad del despliegue sea transparente al desarrollador, proponiendo una serie de configuraciones básicas, las cuales permitirán ganar tiempo y seguridad a la hora de crear entornos de desarrollo.

Como base de este proyecto, se ha tomado lo propuesto en [Arcidiacono et al., 2022] donde se habla sobre Crane, una herramienta que permite optimizar el proceso de despliegue y control de servicios, implementando una API REST para interactuar con el usuario.

El desarrollador enviará en el cuerpo de la solicitud los nombres de las imágenes del repositorio oficial de docker que desea utilizar y Crane API se encargará de crear, almacenar y desplegar un stack de servicios a partir de un archivo docker compose [Reis, D. et al., 2021].

El mismo será generado y ejecutado utilizando herramientas y librerías que proveen el lenguaje de programación utilizado. Luego se guardarán en memoria las aplicaciones creadas por el usuario con el fin de que puedan ser controladas, escaladas y monitorizadas.
En paralelo se le permitirá, al desarrollador, definir reglas y políticas para la toma de decisiones pudiendo así ajustar el consumo que realiza el servicio, sin desperdiciar/sobrecargar la infraestructura cuando el rendimiento baja/sube.
El proyecto incluye abordar la creación de contenedores mediante un servicio REST, la implementación de un Proxy reverso [Sommerlad et al., 2003] la integración con un sistema de gestión de métricas, la integración con un sistema de gestión de alertas, la integración con un sistema de gestión de políticas, la creación de archivos de configuración dinámicos y estáticos para despliegue de contenedores, la administración de las configuraciones de seguridad de los contenedores, la implementación de base de datos para la gestión de aplicaciones y el modelo RBAC [Ferraiolo et al.,1995] para la autenticación y autorización sobre el sistema

3- RESULTADOS ESPERADOS/OBTENIDOS

Se espera obtener un servicio que permita a los usuarios crear aplicaciones de forma fácil y rápida, manteniendo las mismas configuraciones y políticas en cualquier equipo donde se ejecute. El usuario mediante una API REST podrá crear contenedores que expondrán métricas y ajustar las reglas de escalado dependiendo de su propio análisis y/o reglas previamente definidas. Tendrá el control de los puertos expuestos de la aplicación, tráfico de los contenedores y podrá clonar las configuraciones de aplicaciones para generar nuevas instancias de forma más rápida.
Además, al ser portable y de fácil acceso, el desarrollador podrá desplegarlo en cualquier servidor ejecutando un comando.
A su vez el usuario tendrá a su disposición los archivos de docker compose generados para poder llevarlos a cualquier lugar sin necesidad de tener que sacar a Crane del entorno local de desarrollo.
En la Figura 1 se representa el flujo de trabajo completo para el despliegue y monitoreo de las aplicaciones contenerizadas creadas a través de la API de Crane. Se observa que se usa Traefik[2] para el balanceo de carga entre las instancias del servicio, Prometheus[3] para recopilar y almacenar métricas sobre el rendimiento y comportamiento. Y luego, la información obtenida será evaluada por Alert Manager y Open Policy Agent[4] para tomar medidas en caso de que se cumpla alguna regla previamente definida.

4- FORMACIÓN DE RECURSOS HUMANOS

La diversidad de líneas de investigación enunciadas, así como la gran variedad de aplicación de los resultados esperados, propone un foco de investigación claro y concreto para la formación de recursos humanos, de una manera directa en el desarrollo de conocimientos en Docker y el manejo de contenedores, así como, conceptos de seguridad, comunicación entre contenedores o simplemente ser capaz de

[2] https://traefik.io/
[3] https://prometheus.io/
[4] https://www.openpolicyagent.org/

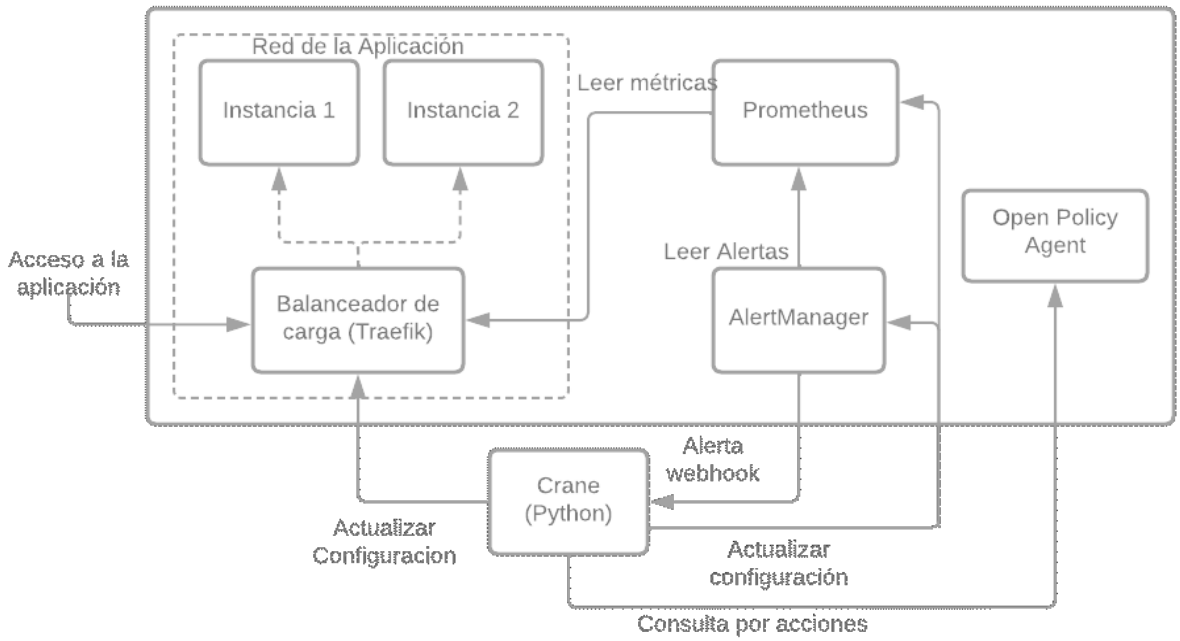

Fig. 1.- Arquitectura instanciada en Crane

analizar las métricas obtenidas por un determinado contenedor y tomar decisiones fundamentadas.

Debido a que este trabajo está cimentado sobre un proxy reverso, esto brinda muchos beneficios en la gestión y optimización de la infraestructura de la aplicación, tales como: balanceos de carga para optimizar al máximo los recursos disponibles, gestión de DNS, escalabilidad y también monitoreo.

También, al tener integrado un sistema de alertas, el recurso podrá recopilar métricas de rendimiento de los sistemas y configurar manualmente alertas personalizadas que, en caso de ser activadas, responderán dependiendo de unas políticas definidas.

Todo esto aporta conocimientos amplios sobre cómo se crean contenedores, cómo se comportan en cada momento y qué se puede hacer para mejorar su rendimiento o qué hacer en caso de fallo.

De una manera indirecta, es posible apoyarse en CRANE para el desarrollo de aplicaciones, que facilita un despliegue rápido del entorno mediante una simple petición http, lo cual permitirá a los recursos humanos involucrados, formarse de una manera más rápida y efectiva, sin necesidad de pasar tiempo configurando un entorno que, en general, nada tiene que ver con el fin de su trabajo específico de desarrollador.

5- BIBLIOGRAFÍA

[Arcidiacono et al., 2022] Arcidiacono, J., Bazán, P., del Río, N., & Lliteras, A. B. (2022). Crane: A Local Deployment Tool for Containerized Applications. In Conference on Cloud Computing, Big Data & Emerging Topics (pp. 58-71). Springer, Cham.

[Arcidiacono et al., 2020] Arcidiacono, J., Lliteras, A. B., & Bazán, P. A. (2020). DEHIA, una plataforma para la generación y ejecución de actividades de recolección de datos con intervención humana aplicada en el Programa E-Basura. In VI Simposio Argentino de Ciencia de Datos y GRANdes DAtos (AGRANDA 2020)-JAIIO 49 (Modalidad virtual).

[Bullington et al., 2020] Bullington-McGuire, R. and Dennis, A.K. and Schwartz, M. (2020). Docker for Developers: Develop and run your application with Docker containers using DevOps tools for continuous delivery. Packt Publishing.

[Burns et al., 2022] Burns, B., Beda, J., Hightower, K., & Evenson, L. (2022). Kubernetes: up and running. " O'Reilly Media, Inc.".

[Ferraiolo et al.,1995] Ferraiolo, D., Cugini, J., & Kuhn, D. R. (1995, December). Role-based access control (RBAC): Features and motivations. In Proceedings of 11th annual computer security application conference (pp. 241-48).

[Fielding & Taylor, 2002] Fielding, R. T., & Taylor, R. N. (2002). Principled design of the modern web architecture. ACM Transactions on Internet Technology (TOIT), 2(2), 115-150.

[Httermann, 2012] Httermann, M. (2012). DevOps for developers. Apress: delivers a practical, thorough introduction to approaches, processes and tools to foster collaboration between software development and operations

[Rad et al., 2017] Rad, B. B., Bhatti, H. J., & Ahmadi, M. (2017). An introduction to docker and analysis of its performance. International Journal of Computer Science and Network Security (IJCSNS), 17(3), 228.

[Rani & Ranjan, 2014] Rani, D., & Ranjan, R. K. (2014). A comparative study of SaaS, PaaS and IaaS in cloud computing. International Journal of Advanced Research in Computer Science and Software Engineering, 4(6).

[Reis et al., 2021] Reis, D., Piedade, B., Correia, F. F., Dias, J. P., & Aguiar, A. (2021). Developing docker and docker-compose specifications: A developers' survey. IEEE Access, 10, 2318-2329.

[Sommerlad et al., 2003] Sommerlad, P. (2003, June). Reverse Proxy Patterns. In EuroPLoP (pp. 431-458).