IMPLEMENTACIÓN DE LA ARQUITECTURA DE MICROSERVICIOS EN LA UNIVERSIDAD TÉCNICA NACIONAL DE COSTA RICA - UTN

IMPLEMENTATION OF THE MICROSERVICES ARCHITECTURE AT UNIVERSIDAD TÉCNICA NACIONAL DE COSTA RICA - UTN

Resumen

En el presente artículo se describe el proceso de transformación de la arquitectura de desarrollo de software de la Universidad Técnica Nacional, pasando del enfoque monolítico al de microservicios, aprovechando herramientas tecnológicas robustas y vanguardistas de código abierto como Docker, Kubernetes, Gitlab, entre otras, disponibles para el uso de las instituciones y empresas que lo necesiten.

Palabras clave: Arquitectura de Software, microservicios, Kubernetes, Docker, Gitlab.

Abstract

In this article, the process of transforming the software development architecture of UTN is described, transitioning from a monolithic approach to microservices, leveraging robust and cutting-edge open-source technological tools such as Docker, Kubernetes, GitLab, among others, available for use by institutions and companies as needed.

Keywords: Software architecture, microservices, Kubernetes, Docker, Gitlab.

Introducción

La UTN es la quinta universidad pública de Costa Rica, creada en 2008 para atender las exigencias de formación técnica superior en el país. Su necesidad natural de soluciones tecnológicas, trajo consigo el desarrollo interno de sistemas informáticos que fueron automatizando paulatinamente procesos institucionales, entre ellos la gestión de oferta académica, gestión presupuestaria, etc. Para esto se usó Drupal, un gestor de contenido web de código abierto que permite la creación de nuevo software mediante un lenguaje basado en PHP, y con MariaDB como motor de base de datos.

Drupal tiene inherente una arquitectura monolítica que fuerza la interdependencia entre los sistemas que se van anexando(módulos), por lo que cualquier problema en alguno detiene el funcionamiento de todos. Sumado a esta desventaja, se depende de la comunidad de Drupal para garantizar su seguridad y se mantiene una incertidumbre por posibles incompatibilidades entre actualizaciones de su núcleo y el software creado con anteriores versiones.

Alrededor del año 2017, la Dirección de Gestión de Tecnologías de Información(DGTI) tenía más de 10 proyectos desplegados en un único sitio Drupal en ambiente de producción, además del sistema de becas en un sitio aparte. Esto representaba un riesgo elevado de sufrir tiempos de baja generalizados en los servicios digitales que se brindan a usuarios

internos de la UTN, y a decenas de miles de estudiantes y personas aspirantes en procesos sustantivos de la institución, como solicitudes de beca o admisión. Con todo este panorama, la búsqueda de una arquitectura de software que permitiera gestionar de forma ágil y resiliente el despliegue de los numerosos desarrollos era imperativa, teniendo la transformación tecnológica como eje transversal de esta necesidad.

Objetivos

- Estandarizar el flujo de trabajo del desarrollo de aplicaciones institucionales para eliminar incompatibilidades entre los servicios digitalizados.

- Implementar una arquitectura flexible para el despliegue continuo de mejoras en ambientes de pruebas y producción, para usuarios expertos y finales, integrando automáticamente las nuevas soluciones informáticas a las existentes.

- Eliminar la dependencia de funcionamiento entre los sistemas de información mediante la segmentación de los procesos institucionales, donde cada parte será convertida en un microservicio que integrará aplicaciones finales modularizadas.

- Gestionar el ciclo de vida e interacciones del software institucional de forma ágil y efectiva mediante herramientas tecnológicas que tengan amplio historial de éxito y que sean respaldadas por la comunidad informática.

- Implementar técnicas de escalamiento horizontal automático para la correcta distribución de cargas de trabajo variables, minimizando tiempos de interrupción de los servicios digitales de la universidad provocados por una gestión ineficiente de los recursos en servidores.

- Modernizar las técnicas de creación de software para aprovechar las ventajas de las nuevas tecnologías, emulando casos de éxito en instituciones y empresas donde la experiencia de usuarios y administradores de servicios digitales haya mejorado sustancialmente.

Soluciones tecnológicas implementadas

Docker

Los problemas de integración y compatibilidad entre sistemas creados en hardware distinto, generaban retrasos y malestar en el equipo de desarrollo de la UTN. Con Docker se abstraen los entornos de programación, permitiendo la utilización de cualquier sistema operativo y evitando la instalación de dependencias, al ser estas parte de las imágenes utilizadas para trabajar. Su implementación requirió la creación de dos archivos Dockerfile(Imagen 1) en el repositorio de cada microservicio, uno para programación en la máquina local (Dockerfile.dev) con docker-compose, y otro para el despliegue en kubernetes (Dockerfile.prod). La mayoría de microservicios en la UTN utilizan nodeJS como lenguaje de programación, por lo que los Dockerfile de estos proyectos son idénticos

teniendo como imagen base la etiquetada como *node:current-alpine* en el registro de contenedores público de docker.

Algunas aplicaciones necesitan una base de datos propia, para la cual se crea un proyecto de Gitlab y se despliega como microservicio en los ambientes de pruebas que se requiera. Se suele utilizar *mariadb* como motor de base de datos, por lo que se escogió como imagen base *mariadb-galera* y así crear nodos multi-maestro, aprovechando la distribución de cargas en kubernetes con el objeto *Service*.

Para el ambiente de producción se mantienen las bases de datos en servidores dedicados para este fin.

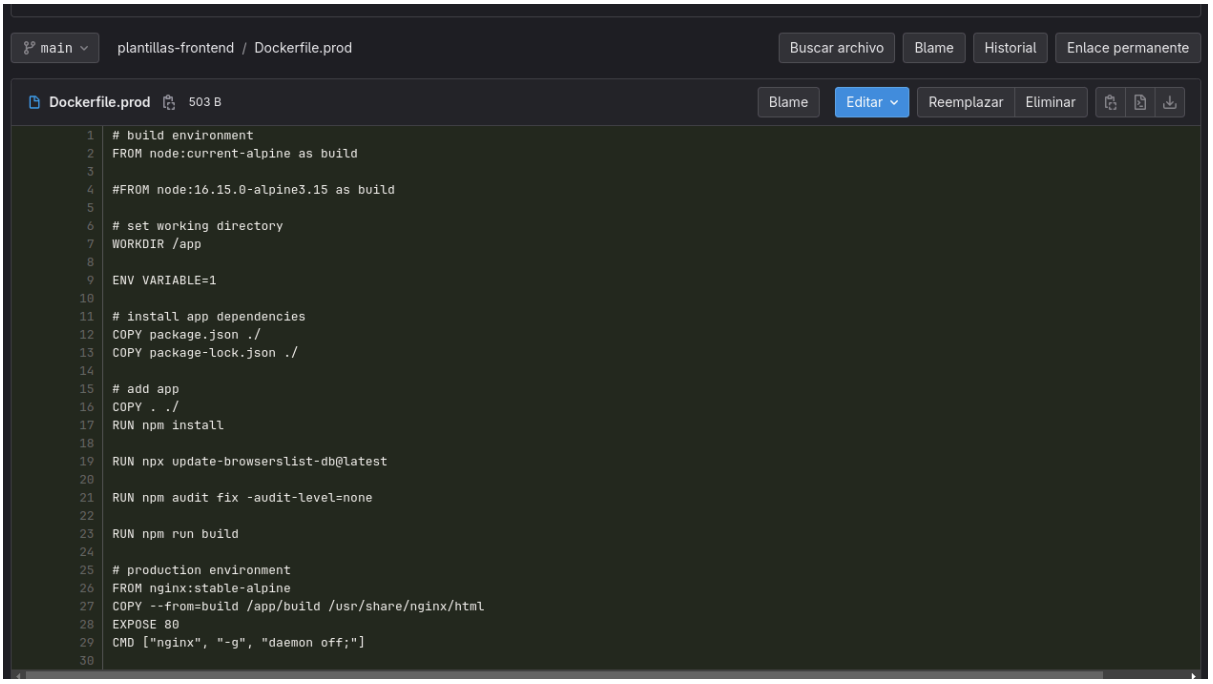

Imagen 1. Archivo Dockerfile.prod en proyecto de Gitlab

Gitlab

Gitlab cuenta con control de versiones de código fuente y las herramientas necesarias para la integración y entrega continuas (CI/CD) de aplicaciones, significando esto que, al registrarse cambios en los repositorios(*"commit"*), se pueden ejecutar tareas que automaticen la instalación del producto en un servidor con acceso para los usuarios respectivos según el ambiente en el que se esté trabajando(calidad, pruebas, producción, etc). Para su uso se requiere la creación de un archivo *.gitlab-ci.yaml*, en donde se agreguen los comandos a ejecutar en cada *"commit"*(Imagen 2).

Se implementaron las secuencias(*"pipelines"*) de tareas(*"jobs"*) necesarias para CI/CD de la siguiente manera:

- Creación de imagen: Se ejecutan los comandos *"docker build"* y *"docker push"* que generarán y almacenarán la imagen de docker de la aplicación en el registro de

contenedores privado previamente instalado en el mismo Gitlab, utilizando el archivo Dockerfile contenido en cada repositorio.

- Revisiones de seguridad: Se ejecutan pruebas de seguridad de aplicaciones estáticas y dinámicas (*SAST* y *DAST*) sobre la imagen creada utilizando herramientas de seguridad informática como Trivy.
- Despliegue de aplicación: Se ejecutan comandos de *"kubectl"* o *"helm"* usando manifiestos escritos por la DGTI para la creación de todos los objetos de kubernetes(*"configmaps"*, *"ingress"*, *"deployment"*, *"persistent volume claim"*, etc.) necesarios para el despliegue de la imagen almacenada, esto en el cluster de pruebas o producción según la rama del repositorio sobre la que se han realizado los cambios.

Imagen 2. *"Pipeline"* desencadenado por *"commit"*

Cada una de estas tareas son ejecutadas en secuencia mediante *gitlab-runners* (Imagen 3) previamente instalados y configurados para tener acceso al cluster de kubernetes y al registro de contenedores de Gitlab.

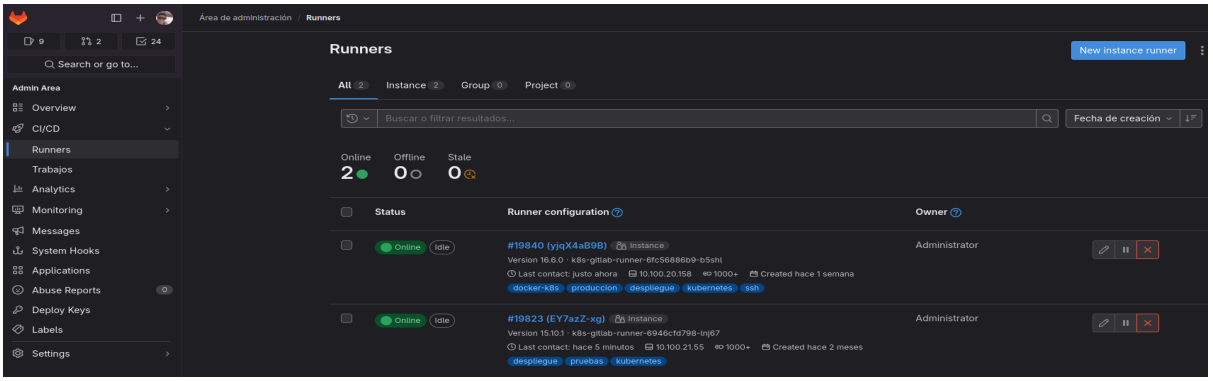

Imagen 3. *Gitlab-runners* para ejecución de *"pipelines"* instalados en Gitlab

Kubernetes

La orquestación de contenedores en la UTN comenzó en Docker Swarm, sin embargo, se cambió a Kubernetes por ser esta la plataforma más utilizada, además de que la integración

con los repositorios de código se facilita por las herramientas de CI/CD integradas en Gitlab.

Se desarrollaron plantillas de archivos *".yaml"* (Imagen 4) que son utilizados en las tareas de integración y entrega contínuas (*"pipelines"*) de Gitlab para crear cada uno de los objetos de kubernetes necesarios y así lograr el funcionamiento correcto de los microservicios. Entre estos se encuentran los *"deployment"*, los *"services"* de tipo *"ClusterIP"*(Imagen 5) que distribuyen las cargas de trabajo entre *"pods"*, los recursos *"ingress"* que enrutan el tráfico desde el exterior hasta los servicios a través del controlador de ingreso *nginx* previamente instalado en el cluster, los *"configmaps"* que almacenan variables y configuraciones, los *"secrets"* para datos encriptados, entre otros.

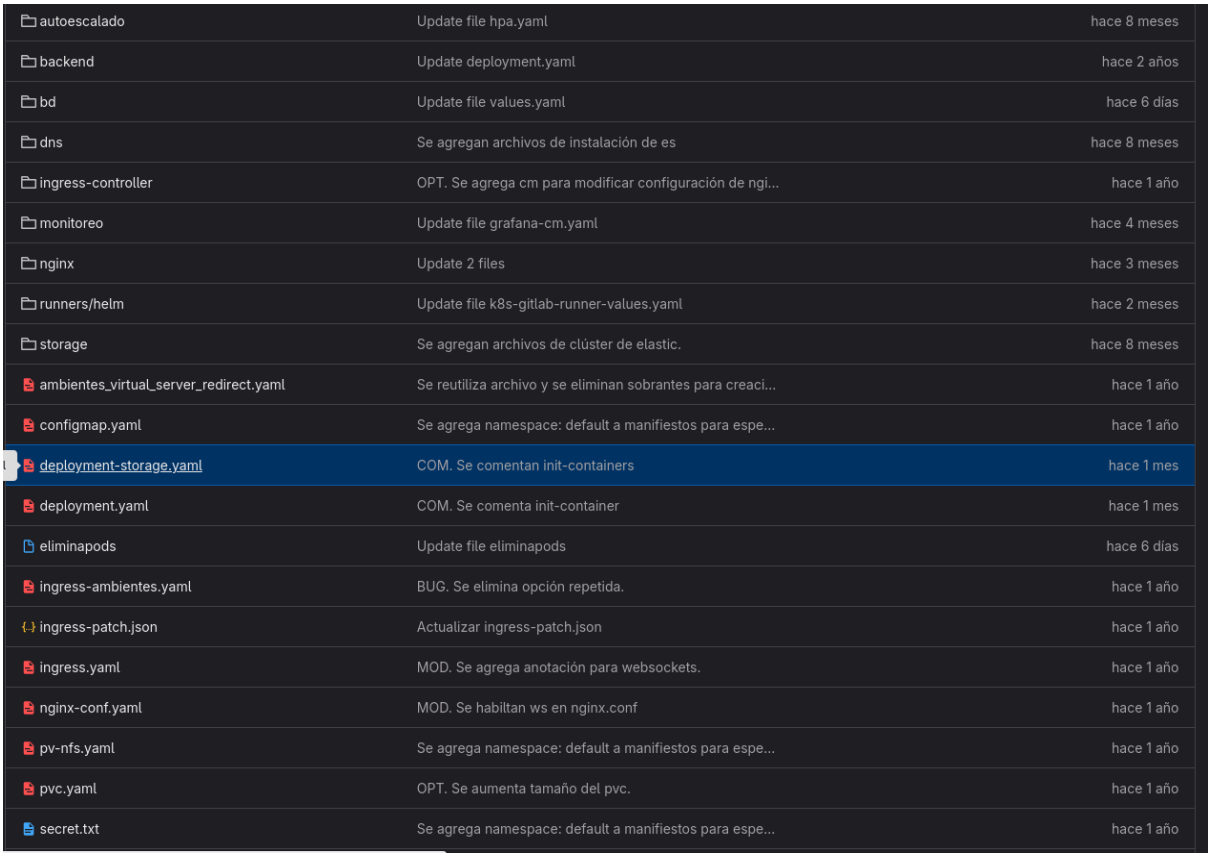

Imagen 4. Plantillas de manifiestos para creación de objetos en k8s

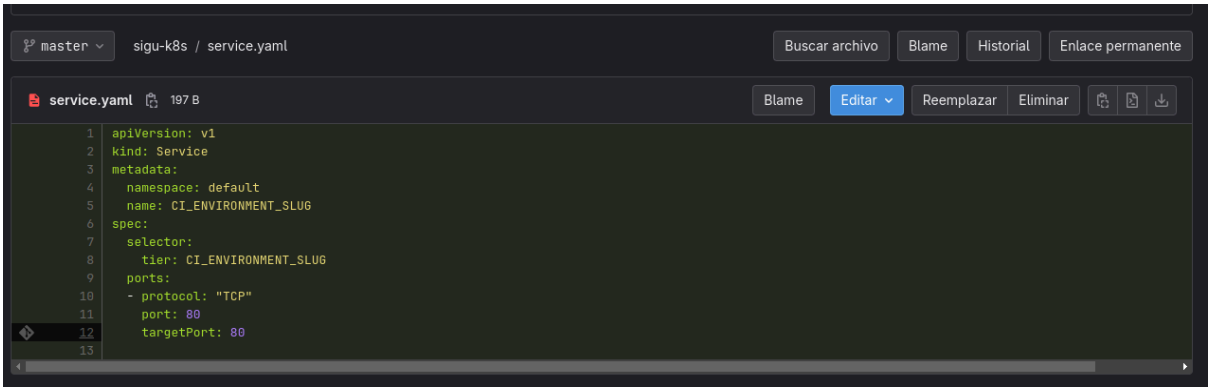

Imagen 5. Plantilla para creación de recurso *"Service"*

También, se implementó el escalado horizontal automático mediante la creación del recurso *“hpa”*(Imagen 6) por cada aplicación que crea o elimina réplicas del microservicio de acuerdo a un porcentaje preestablecido de consumo de cpu y memoria, aumentando la capacidad de respuesta cuando se aumenta el tráfico o procesamiento y liberando los recursos cuando disminuyen.

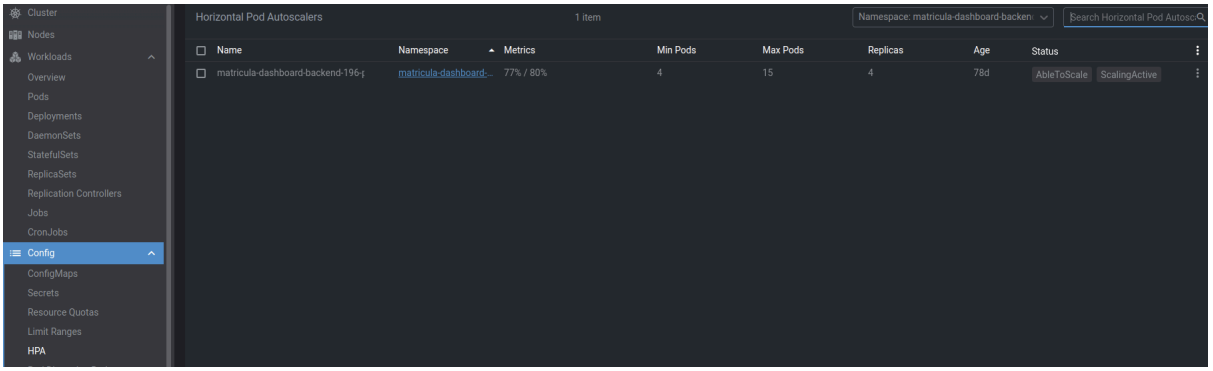

Imagen 6. Recurso *“hpa”* para escalado automático de réplicas

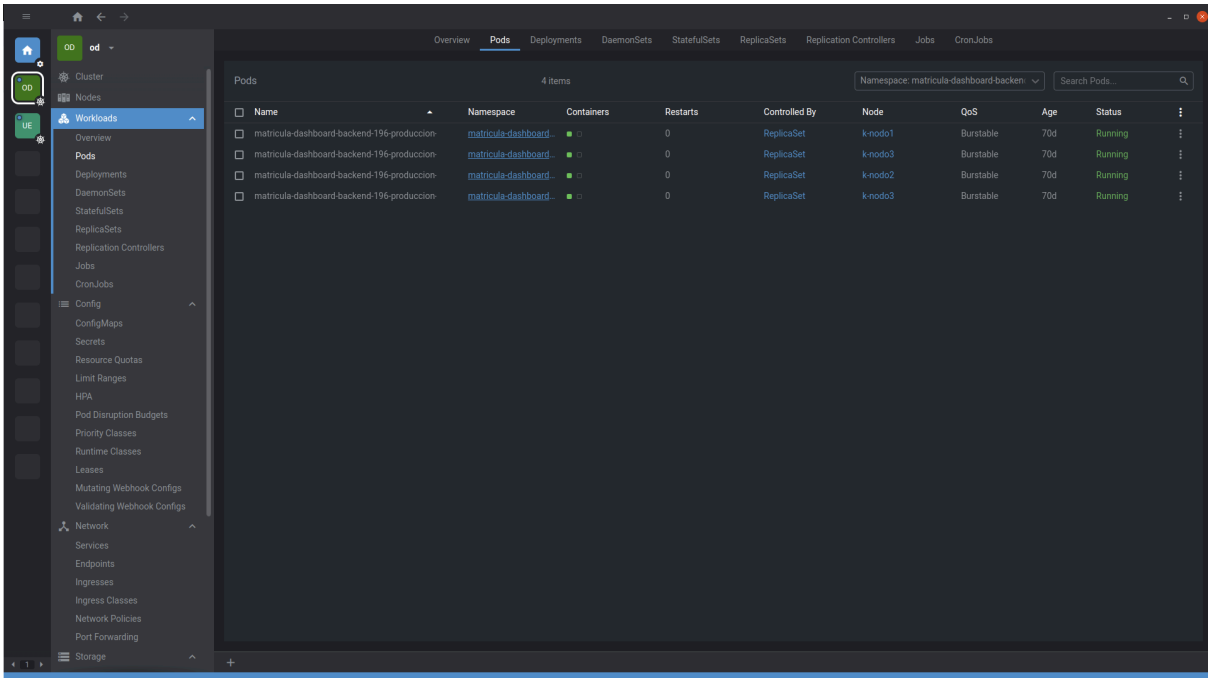

Imagen 7. Réplicas administradas por *“hpa”* al mínimo por cargas inferiores al porcentaje establecido para escalamiento horizontal automático

La ejecución de los comandos que crean los objetos en el cluster de kubernetes es realizada por *gitlab-runner*, previamente instalado en el clúster (Imagen 3).

Conclusiones

La transformación de la arquitectura de software de monolítica a microservicios es en la actualidad la solución a muchos problemas en el desarrollo de sistemas de información. Este cambio requirió en la UTN un esfuerzo de varios años en el que se fueron implementando cada uno de los componentes indispensables en esta nueva forma ágil y

colaborativa de creación de aplicaciones. También, el dominio del entramado de tecnologías y herramientas complejas requeridas en este proceso, han elevado la experticia dentro de la DGTI al incorporar conocimientos especializados de muchas áreas heterogéneas de tecnologías de información, desde manejo de *"iptables"* con redes virtuales hasta programación en lenguaje *"bash"*.

Resultados alcanzados

Actualmente se cuenta con alrededor de 30 aplicaciones desplegadas en ambiente de producción utilizando la arquitectura de microservicios, incluyendo las relacionadas con el proceso de admisión institucional que brindaron servicio a más de 16 mil personas en su primer uso, sin registrar ningún tiempo de interrupción generalizado y experimentándose tiempos de respuesta ágiles en las revisiones constantes que se realizaron durante el periodo que estuvo habilitado el acceso a la plataforma.

También se encuentran en desarrollo más de 150 nuevos microservicios relacionados con una gran variedad de procesos institucionales, siendo algunos para la migración de sistemas monolíticos que funcionan actualmente, y otros para la implementación de nuevas soluciones tecnológicas.