Anexo V

Universidad Nacional del
Noroeste de la Provincia de Buenos Aires

Título: Reestructuración CI/CD
Hydrolix
Carrera: Ingeniería en Informática
Práctica Profesional Supervisada

Alumnos: Claudio Gomez Luengo
Supervisor Docente: Federico Nasso
Tutor de Empresa: Francisco Vives
Fecha de Presentación: 30 de Noviembre del 2023

UNNOBA
Universidad Nacional
Noroeste | Buenos Aires

Índice

UNNOBA
Universidad Nacional
Noroeste | Buenos Aires

Presentación del Informe

Glosario

1. Introducción

Durante la Práctica Profesional Supervisada (PPS) realizada en Clarolab, una empresa de consultoría de software con sede en Junín, Buenos Aires.

Clarolab, se dedica a brindar servicios de consultoría de software para empresas y organizaciones desde su fundación en 2008. Con más de sesenta empleados calificados, la empresa trabaja en colaboración con sus clientes para asegurarse de que el software desarrollado cumpla con sus necesidades y requisitos.

Una de las empresas que Clarolab le presta sus servicios es Hydrolix, esta se dedica al desarrollo de una innovadora plataforma de base de datos en la nube. Su objetivo principal es transformar la gestión y aprovechamiento de grandes volúmenes de datos. Aspiran a proporcionar a las empresas una solución única que permita desbloquear el máximo potencial de sus datos y también reduzca significativamente los costos asociados a su manejo.

Hydrolix posee una integración de CI/CD en GitLab, que le permite realizar despliegues automáticos y continuos de sus desarrollos, la Práctica Profesional Supervisada (PPS) se enfocó en optimizar y reestructurar estos pipelines. Durante este período, se llevó a cabo una revisión y una reingeniería de los procesos para mejorar aún más su eficiencia y rendimiento. Específicamente diseñados para las necesidades de Hydrolix, estos pipelines de CI/CD se convirtieron en un punto crítico de enfoque.

La optimización de estos pipelines buscaba reducir tanto los costos asociados como el tiempo necesario para su ejecución. Al lograr esto, se mejoró significativamente la eficiencia del desarrollo y despliegue de software para Hydrolix.

Esta iniciativa no solo buscaba mejorar la infraestructura existente, sino también garantizar un flujo de trabajo más eficaz y económico para la empresa.

Es importante destacar que la colaboración entre Clarolab y Hydrolix durante la PPS demuestra la importancia de la sinergia entre diferentes empresas para abordar desafíos tecnológicos.

UNNOBA
Universidad Nacional
Noroeste | Buenos Aires

2. Objetivos

Objetivos Generales:

1. Optimizar los pipelines de CI/CD: Mejorar y optimizar los pipelines de integración continua y entrega continua.
2. Reducir costos y tiempo de ejecución: Lograr una disminución significativa en los costos asociados y el tiempo requerido para completar los procesos de integración y despliegue de software, aumentando así la eficiencia operativa.
3. Satisfacer las necesidades de Hydrolix: Garantizar la satisfacción de las necesidades de Hydrolix en cuanto al desarrollo de software.
4. Potenciar la sinergia entre Clarolab y Hydrolix: Fortalecer la colaboración y cooperación entre ambas empresas para abordar desafíos tecnológicos complejos, desarrollar soluciones innovadoras y alcanzar resultados exitosos en el ámbito del desarrollo de software.

Objetivos Específicos:

1. Reestructurar los pipelines de CI/CD para Hydrolix:
    a. Realizar modificaciones de deployment de kubernetes en una VM de AWS en los pipelines para disminuir costos y mejorar su rendimiento.
    b. Buscar una herramienta de despliegue más ágil y en una plataforma económica.
2. Mejorar la eficiencia de los pipelines:
    a. Definición de métricas claras: Establecimiento de indicadores para evaluar la eficiencia, incluyendo tiempos de ejecución y frecuencia.
    b. Análisis exhaustivo de procesos: Revisión detallada de flujos de trabajo.
    c. Estrategias de optimización: Implementación de mejoras específicas para reducir tiempos.
    d. Revisión continua: Ciclo constante de revisión para ajustes continuos y evolución de la eficiencia de los pipelines.
3. Implementar soluciones tecnológicas adecuadas para Hydrolix:
    a. Colaboración estrecha con el equipo de Hydrolix: Trabajo directo y cercano con el equipo de Control **(CTL)** de Hydrolix para comprender sus necesidades específicas.
4. Fomentar la comunicación y cooperación entre los equipos de ambas empresas:
    a. Interacción directa entre equipos: Promoción de la interacción directa y la colaboración entre los equipos.
    b. Alineación con expectativas del cliente: Garantizar que el desarrollo de software se alinee con las expectativas de Hydrolix.
    c. Cooperación para metas compartidas: Trabajo conjunto para cumplir metas, asegurando una ejecución óptima y resultados satisfactorios.

UNNOBA
Universidad Nacional
Noroeste | Buenos Aires

3. Metodología de Trabajo

La metodología ágil Scrum se empleó para dirigir el proyecto. Se organizaron sprints de 4 semanas, donde el equipo de CI y operaciones se reunía al inicio de cada ciclo para identificar y abordar los problemas existentes en los pipelines. Durante estas reuniones, se definían objetivos claros y específicos para el siguiente sprint, asegurando así una planificación efectiva y una ejecución orientada a resultados.

Usando Jira como herramienta de gestión, se crearon Epics detallados específicamente destinados a la reestructuración de los pipelines. Estos Epics se dividieron en Stories, cada una dirigida a reescribir y mejorar una pipeline específica que debía ejecutarse en CI/CD de GitLab.

Para cada Story, se desglosaron en varias tasks, detallando las modificaciones necesarias en cada job dentro de la pipeline. Por ejemplo, si una Story abordaba la optimización de la pipeline de pruebas, se crearon tasks individuales para mejorar la configuración de las pruebas unitarias, revisar los flujos de integración y asegurar la compatibilidad con diferentes entornos.

Se aplicaron métodos de estimación de esfuerzo adquiridos en la materia Métricas de Software para calcular con precisión el tiempo y recursos requeridos en cada tarea dentro de los Stories.

Durante los sprints, se llevaron a cabo reuniones diarias de seguimiento, conocidas como "standups", donde el equipo compartió avances, discutió posibles obstáculos y tomó decisiones para garantizar una ejecución efectiva del trabajo. Se aplicaron técnicas de planificación y asignación de esfuerzos, acorde con los conocimientos adquiridos en ingeniería, para asegurar la resolución efectiva de los problemas identificados en los pipelines.

Al final de cada ciclo, se realizó una revisión para demostrar las mejoras logradas en los pipelines. Se llevaron a cabo retrospectivas para analizar qué aspectos funcionaron bien y cuáles requieren mejoras, aplicando un enfoque de aprendizaje continuo basado en la retroalimentación recibida.

UNNOBA
Universidad Nacional Noroeste | Buenos Aires

4. Plan de Trabajo y Carga Horaria

			Agosto			Septiembre	
Tarea	**Responsable**	**Duracion**	**S1**	**S2**	**S3**	**S4**	**S5**
Investigacion y lectura de documentacion	**Claudio Gomez Luengo**	**1S**					
Lectura de documentacion	**Claudio Gomez Luengo**	1S					
Instalacion de sofware	**Claudio Gomez Luengo**	1S					
Investigacion de la Infraestructura CI CD	**Claudio Gomez Luengo**	**3S**					
Investigacion sobre Gitlab CI CD	**Claudio Gomez Luengo**	1S					
Investigacion sobre Kubernetes	**Claudio Gomez Luengo**	1S					
Investigacion sobre AWS - Google Cloud - Linode	**Claudio Gomez Luengo**	1S					
Investigacion alternativas (Terraform, Pulumi)	**Claudio Gomez Luengo**	1S					
Creacion de tickets en Jira	**Claudio Gomez Luengo**	**1S**					
Creacion de propuesta	**Claudio Gomez Luengo**	1S					
Creacion de Tickets en Jira	**Claudio Gomez Luengo**	1S					
Modificacion de Pipeline	**Claudio Gomez Luengo**	**2S**					
Creacion de VM con Kubernetes	**Claudio Gomez Luengo**	1S					
Pruebas de campo	**Claudio Gomez Luengo**	1S					
Reestructuracion de Pipeline	**Claudio Gomez Luengo**	1S					

a. Tareas

- Investigación y lectura:
  - Se investigó sobre las herramientas a usar, en este caso la herramienta más usada para el trabajo fue Kubernetes.
- Lectura de documentación:
  - Se leyó la documentación de Hydrolix para poder instalar el software y realizar tareas de pruebas
- Instalación de software.
  - Se instaló hydrolix en un entorno de Kubernetes en AWS y Google Cloud para realizar el onboarding.
- Investigación sobre la Infraestructura de CI/CD:
  - Se tomaron cursos de las distintas herramientas:
    - Gitlab CI CD.
    - Kubernetes
    - AWS, Google Cloud, Linode
    - Pulumi
    - Terraform

- Creación de tickets en Jira
    - Creación de propuesta:
        - Se creó una Epic en Jira con la propuesta de la reestructuración del pipeline
    - Creación de tickets
        - Se crearon stories y tasks para la epic
- Modificación de Pipeline
    - Creación de VM con Pulumi:
        - Se creó un proof of concept de cómo se usa Pulumi para crear una VM con Kubernetes
    - Pruebas de campo
        - Se realizó un deploy de una VM con pulumi en el entorno de CI/CD
    - Reestructuración del pipeline
        - Se reescribieron los jobs correspondientes del pipeline para utilizar la VM deployada con Pulumi.

5. Descripción de la Práctica Profesional Efectuada

La principal tarea de la designación del plan de trabajo fue investigar que es Hydrolix y cómo funciona y subsiguiente la lógica de la infraestructura de CI CD, logrando así conocer las herramientas utilizadas dentro de la misma, como posterior tarea se realizó capacitación sobre como funciona GitLab CI CD y las herramientas donde se instala Hydrolix en la nube (AWS, Google Cloud, Linode).

Subsiguiente a la investigación y capacitación se planificó una manera de optimizar la velocidad de la ejecución del pipeline de CI CD y los gastos que deriva cuando se ejecuta.

Se investigaron las alternativas fueron Terraform y Pulumi para poder crear los clusters en una máquina virtual de Amazon Web Service con código en vez de las herramientas Kubernetes de Amazon Web Service, Google Cloud o Linode.

Analizando las alternativas, se decidió quedarse con Pulumi, dado que permitió instalar K3S en las máquinas virtuales de AWS de manera satisfactoria.

Una vez definidas las herramientas a utilizar se definió a crear tickets en la herramienta Jira con el objetivo de modificar los Pipelines de Gitlab CI CD.

UNNOBA
Universidad Nacional
Noroeste | Buenos Aires

6. Herramientas y tecnología

En el ámbito de la Ingeniería Informática, el uso de herramientas tecnológicas especializadas es fundamental para el desarrollo eficiente de aplicaciones y sistemas. Estas soluciones, como Pulumi, Kubernetes y GitLab CI/CD, transformaron la gestión y el despliegue de software, permitiendo una automatización efectiva en áreas clave del ciclo de vida de los proyectos.

Pulumi

Pulumi es una herramienta de código abierto que permite la gestión de la infraestructura como código (IaC, por sus siglas en inglés). Con Pulumi se puede definir y desplegar recursos de infraestructura, como servidores, bases de datos y redes, utilizando lenguajes de programación familiares como Python, JavaScript, TypeScript y otros. Esto facilita la automatización y la gestión de la infraestructura de manera programática, lo que mejora la reproducibilidad y la escalabilidad de las aplicaciones y la infraestructura en la nube.

Kubernetes

Kubernetes es una herramienta de código abierto para la administración de aplicaciones en contenedores. Ofrece un entorno de orquestación que automatiza tareas como el despliegue, la escalabilidad y la recuperación de aplicaciones en contenedores, permitiendo que los desarrolladores se enfoquen en escribir código sin preocuparse por la infraestructura subyacente. Kubernetes es ampliamente utilizado en la industria para crear y administrar aplicaciones escalables y altamente disponibles en entornos de nube o locales.

Kubernetes Cloud (AWS, Google Cloud, Linode)

Kubernetes es una plataforma de orquestación de contenedores que se puede utilizar en múltiples proveedores de servicios en la nube, como Amazon Web Services (AWS), Google Cloud y Linode, para gestionar y escalar aplicaciones en contenedores de manera

eficiente y automatizada.

Gitlab CI CD

GitLab CI/CD es un conjunto de herramientas y prácticas que automatizan la integración continua (CI) y el despliegue continuo (CD) de software. Ayuda a los equipos de desarrollo a automatizar tareas como pruebas, construcción y despliegue de aplicaciones, lo que mejora la calidad del software, reduce los errores y acelera el proceso de entrega. GitLab CI/CD se integra con GitLab, una plataforma de gestión de repositorios de código fuente, para proporcionar un flujo de trabajo completo de desarrollo y entrega de software.

7. Composición del Pipeline

En el caso de Hydrolix, el despliegue de un pipeline tiene el fin de deployar una versión de la aplicación. El cual consiste de:

A. La construcción de las imágenes de sistemas operativos para buildear la aplicación y configurar el entorno de testing.
B. Buildear la aplicación.
    i. Se utiliza un script de build que crea la aplicación para que pueda ser usada en el siguiente job del pipeline.
C. Deployar el entorno de testing
    i. Se deploya la aplicación en un entorno de Kubernetes en una VM de AWS con pulumi, y luego se actualiza a la nueva versión generada en el job de build.
D. Realizar testeo automatizado correspondiente (smoke test y tests automatizados).
    i. Se realizan los testeos con un framework para realizar los tests correspondientes de la aplicación.
E. Cleanup
    i. Borrado del entorno de Kubernetes con la aplicación.

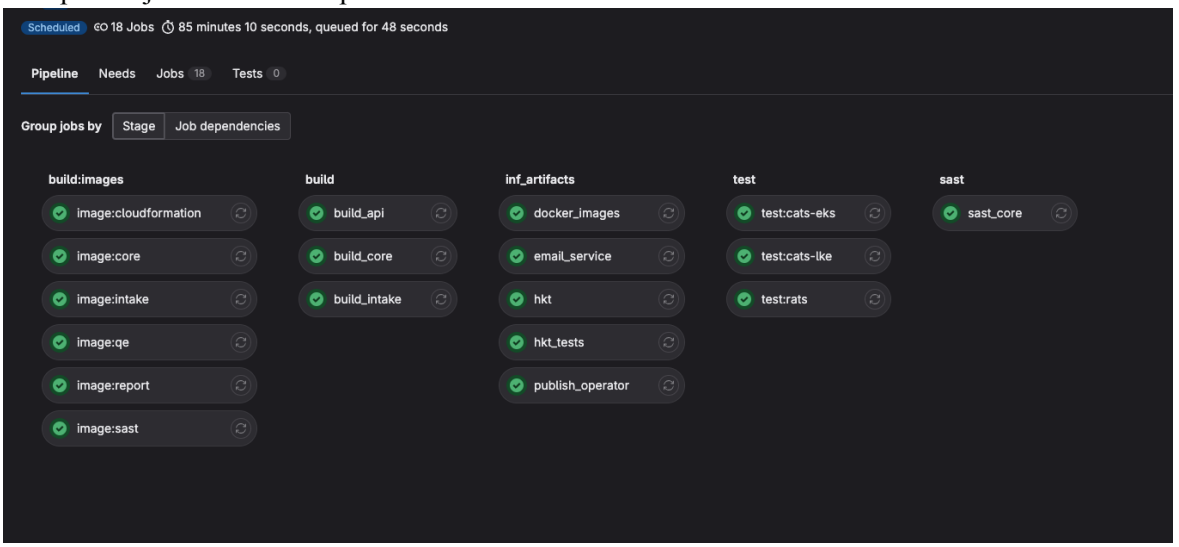
UNNOBA
Universidad Nacional
Noroeste | Buenos Aires

8. Resultados obtenidos

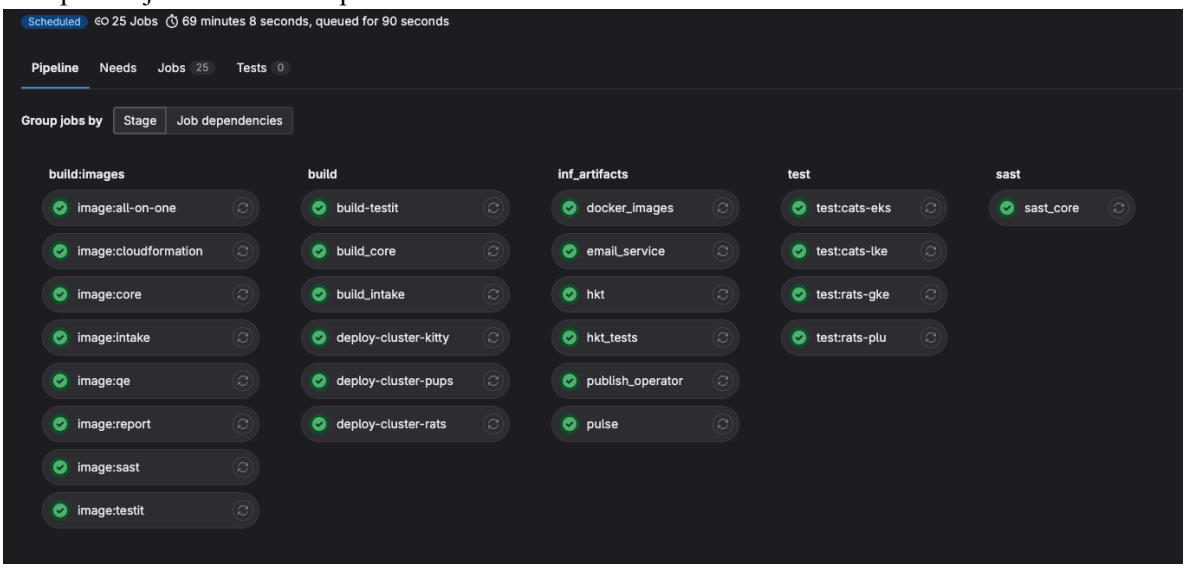
Notablemente las mejoras de CI/CD han impactado en el correr de los pipelines de tests.

El tiempo de ejecución de los tests se vio reducido de 85 minutos a 70 minutos.

Tiempo de ejecución en el Pipeline anterior: 85 minutos



Tiempo de ejecución en el Pipeline final: 70 minutos



En términos económicos la compañía por temas de confidencialidad no permite el acceso a esos datos.

UNNOBA
Universidad Nacional
Noroeste | Buenos Aires

9. Planes a Futuro

A partir de los cambios que se han hecho los jobs del Pipeline utilizando Pulumi y deployando un Kubernetes con una versión específica (1.27).

Se determinó que se deben realizar coverage en diferentes versiones de Kubernetes.

- El coverage debe cubrir tanto versiones anteriores como posteriores, dado que los clientes pueden usar cualquiera de estas versiones.

A raíz de realizar los nuevos cambios, hemos notado que no se encuentra unificado los logs que la aplicación entrega al final de su ejecución.

- Como solución a esto se planea agregar una plataforma que pueda consumir los logs y unificarlos.

10. Conclusiones

El cambio en la configuración de los pipelines reemplazando las tecnologías de Kubernetes en la nube (GCP, EKS, LKE) por un deployment en una máquina virtual ha resultado en mejoras de rendimiento y estandarización.
Estas mejoras resultaron en un impacto positivo en los costes de la compañía en utilizar máquinas virtuales a demanda en lugar de una plataforma en la nube.

La mejora más importante es la estabilización de los tests que ya dependen siempre de un entorno de cero sin intervenciones externas.

Los objetivo fueron cumplidos satisfactoriamente:

Optimización de los pipelines de CI/CD y Reducción de costos y tiempo de ejecución

Se han mejorado y optimizado los pipelines de integración continua y entrega continua (CI/CD) de GitLab y reducido significativamente los costos de la ejecución de cada pipeline conjunto con la reducción de tiempo.

Satisfacción de las necesidades de Hydrolix y Potenciar la sinergia entre Clarolab y Hydrolix

Con la mejora en la reducción de tiempo de ejecución de los Pipelines, los tiempos que le toman a los desarrolladores el despliegue de sus cambios en la aplicación se ve reducido, lo cual les permite enfocarse en el desarrollo.

Por consiguiente las releases y los tests que se realizan en estas se ven afectados por la misma mejora y hace que sea más dinámico el despliegue de una nueva versión del software.

UNNOBA
Universidad Nacional Noroeste | Buenos Aires

11. Bibliografía

[1] K. Morris, "Infrastructure as Code: Managing Servers in the Cloud."

[2] J. Chapin, "Programming AWS Lambda: Build and Deploy Serverless Applications with Java, Python, and Node.js."

[3] K. Hightower, B. Burns, and J. Beda, "Kubernetes: Up and Running: Dive into the Future of Infrastructure."

[4] N. Poulton, "The Kubernetes Book."

[5] G. Sayfan, "Mastering Kubernetes: Level up your container orchestration skills with Kubernetes to deploy, manage, and scale applications effectively."

[6] K. Jain, "GitLab CI/CD for Beginners: A hands-on guide to automating your DevOps workflow."

[7] J. M. Hogle, "GitLab Repository Management."

[8] K. Hightower, B. Burns, and J. Beda, "Kubernetes: Up and Running: Dive into the Future of Infrastructure" (también cubre Kubernetes en la nube, incluyendo GKE y EKS).

[9] V. SM, "Google Kubernetes Engine: Managing Secure and Scalable Container-Based Applications."

[10] Pulumi. "Pulumi Documentation." [En línea]. Disponible en: https://www.pulumi.com/docs/

[11] Kubernetes. "Kubernetes Documentation." [En línea]. Disponible en: https://kubernetes.io/docs/

[12] GitLab. "GitLab CI/CD Documentation." [En línea]. Disponible en: https://docs.gitlab.com/ee/ci/

[13] Hydrolix. "Hydrolix Documentation." [En línea]. Disponible en: https://docs.hydrolix.io/docs

[14] Google Cloud. "Google Kubernetes Engine Documentation." [En línea]. Disponible en: https://cloud.google.com/kubernetes-engine/docs

[15] Amazon Web Services. "Amazon Elastic Kubernetes Service (EKS) Documentation." [En línea]. Disponible en: https://docs.aws.amazon.com/eks/latest/userguide/what-is-eks.html

[16] Linode. "Linode Kubernetes Documentation." [En línea]. Disponible en: https://www.linode.com/docs/products/compute/kubernetes/

MORAN
Sebastian

Administrativo-
Depto.Informática
y Tecnología
UNNOBA
2023.12.05
14:06:29 -0300