



Universidad Nacional Mayor de San Marcos

Universidad del Perú. Decana de América

Facultad de Ingeniería de Sistemas e Informática

Escuela Profesional de Ingeniería de Sistemas

**Implementación de la metodología DevOps para
mejorar el proceso de despliegue continuo en una
empresa dedicada al área de seguros**

TRABAJO DE SUFICIENCIA PROFESIONAL

Para optar el Título Profesional de Ingeniero de Sistemas

AUTOR

Jhonatán Paúl URBINA TOLENTINO

ASESOR

Jorge Luis ZAVALA CAMPOS

Lima, Perú

2026



Reconocimiento - No Comercial - Compartir Igual - Sin restricciones adicionales

<https://creativecommons.org/licenses/by-nc-sa/4.0/>

Usted puede distribuir, remezclar, retocar, y crear a partir del documento original de modo no comercial, siempre y cuando se dé crédito al autor del documento y se licencien las nuevas creaciones bajo las mismas condiciones. No se permite aplicar términos legales o medidas tecnológicas que restrinjan legalmente a otros a hacer cualquier cosa que permita esta licencia.

Referencia bibliográfica

Urbina, J. (2026). *Implementación de la metodología DevOps para mejorar el proceso de despliegue continuo en una empresa dedicada al área de seguros*. [Trabajo de suficiencia profesional de pregrado, Universidad Nacional Mayor de San Marcos, Facultad de Ingeniería de Sistemas e Informática, Escuela Profesional de Ingeniería de Sistemas]. Repositorio institucional Cybertesis UNMSM.

Metadatos complementarios

Datos de autor	
Nombres y apellidos	Jhonatán Paúl Urbina Tolentino
Tipo de documento de identidad	DNI
Número de documento de identidad	48358454
URL de ORCID	https://orcid.org/0009-0005-4805-9583
Datos de asesor	
Nombres y apellidos	Jorge Luis Zavaleta Campos
Tipo de documento de identidad	DNI
Número de documento de identidad	09696217
URL de ORCID	https://orcid.org/0000-0002-3923-1725
Datos del jurado	
Presidente del jurado	
Nombres y apellidos	Iván Carlo Petrlik Azabache
Tipo de documento	DNI
Número de documento de identidad	10140461
Miembro del jurado 1	
Nombres y apellidos	Armando David Espinoza Robles
Tipo de documento	DNI
Número de documento de identidad	08633326
Datos de investigación	
Línea de investigación	No aplica

Grupo de investigación	No aplica
Agencia de financiamiento	Sin financiamiento
Ubicación geográfica de la investigación	País: Perú Departamento: Lima Provincia: Lima
	Distrito: Cercado de Lima Jr. Carlos Amezaga No. 375 Universidad Nacional Mayor de San Marcos Latitud: -12.0564232 Longitud: -77.0843327
Año o rango de años en que se realizó la investigación	2025
URL de disciplinas OCDE	2.02.04 -- Ingeniería de sistemas y comunicaciones https://purl.org/perepo/ocde/ford#2.02.04



UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS
FACULTAD DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
Escuela Profesional de Ingeniería de Sistemas

Acta Virtual de Sustentación
del Trabajo de Suficiencia Profesional

Siendo las 20:00 horas del día 01 de abril del año 2026, se reunieron virtualmente los docentes designados como Miembros de Jurado del Trabajo de Suficiencia Profesional, presidido por el Dr. Petrlik Azabache Iván Carlo (Presidente), Mg. Espinoza Robles Armando David (Miembro) y el Ing. Zavaleta Campos Jorge Luis (Miembro Asesor), usando la plataforma Meet (<http://meet.google.com/hot-tmmg-afm>), para la sustentación virtual del Trabajo de Suficiencia Profesional intitulado: **“IMPLEMENTACIÓN DE LA METODOLOGÍA DEVOPS PARA MEJORAR EL PROCESO DE DESPLIEGUE CONTINUO EN UNA EMPRESA DEDICADA AL ÁREA DE SEGUROS”**, por el Bachiller Urbina Tolentino Jhonatán Paúl; para obtener el Título Profesional de Ingeniero de Sistemas.

Acto seguido de la exposición del Trabajo de Suficiencia Profesional, el Presidente invitó al Bachiller a dar las respuestas a las preguntas establecidas por los miembros del Jurado.

El Bachiller en el curso de sus intervenciones demostró pleno dominio del tema, al responder con acierto y fluidez a las observaciones y preguntas formuladas por los señores miembros del Jurado.

Finalmente habiéndose efectuado la calificación correspondiente por los miembros del Jurado, el Bachiller obtuvo la nota de 18 (**DIECIOCHO**).

A continuación el Presidente de Jurados el Dr. Petrlik Azabache Iván Carlo, declara al Bachiller **Ingeniero de Sistemas**.

Siendo las 20:55 horas, se levantó la sesión.

Presidente

Dr. Petrlik Azabache Iván Carlo

Miembro

Mg. Espinoza Robles Armando David

Miembro Asesor

Ing. Zavaleta Campos Jorge Luis



UNIVERSIDAD NACIONAL MAYOR DE
SAN MARCOS
Universidad del Perú, Decana de América

CERTIFICADO DE SIMILITUD

Yo, Jorge Luis Zavaleta Campos en mi condición de asesor acreditado mediante Resolución Decanal N° 001268-2025-D-FISI/UNMSM y OFICIO N° 0035-PTTSP-FISI-2025 del Trabajo de Suficiencia Profesional, cuyo título es Implementación de la metodología DevOps para mejorar el proceso de despliegue continuo en una empresa dedicada al área de seguros, presentado por el bachiller Urbina Tolentino Jhonatán Paúl para optar el Título Profesional de Ingeniero de Sistemas, CERTIFICO que se ha cumplido con lo establecido en la Directiva de Originalidad y de Similitud de Trabajos Académicos, de Investigación y Producción Intelectual. Según la revisión, análisis y evaluación mediante la plataforma PIDS, el documento evaluado cuenta con el 4% de similitud, nivel PERMITIDO para continuar con los trámites correspondientes y para su **publicación en el repositorio institucional.**

Se emite el presente certificado en cumplimiento de lo establecido en las normas vigentes, como uno de los requisitos para la obtención del **TITULO PROFESIONAL DE INGENIERO DE SISTEMAS.**

Lima, 4 de junio de 2026

Firma del Asesor:

Nombres y Apellidos del asesor: Jorge Luis Zavaleta Campos

DNI: 09696217

FICHA CATALOGRÁFICA

**IMPLEMENTACIÓN DE LA METODOLOGÍA DEVOPS PARA MEJORAR
EL PROCESO DE DESPLIEGUE CONTINUO EN UNA EMPRESA
DEDICADA AL ÁREA DE SEGUROS**

AUTOR: URBINA TOLENTINO, JHONATÁN PAÚL

ASESOR: ZAVALETA CAMPOS, JORGE LUIS

LIMA – PERÚ, 2025

TÍTULO PROFESIONAL: Título Profesional de Ingeniero de Sistemas

ÁREA / PROGRAMA / LÍNEA DE INVESTIGACIÓN:

**Ingenierías / ODS9: Industria, Innovación e Infraestructura / Ingeniería de
Software**

**PREGRADO: Universidad Nacional Mayor de San Marcos - Facultad de
Ingeniería de Sistemas e Informática - Escuela Profesional de Ingeniería de
Sistemas**

AGRADECIMIENTOS

A mi familia por su absoluto y devoto apoyo.

A mi asesor por su responsabilidad y compromiso brindado para la elaboración del Trabajo de Suficiencia Profesional.

A la institución por la inmejorable oportunidad de enriquecer mis habilidades profesionales, adquiriendo valiosos conocimientos.

DEDICATORIA

Dedico este trabajo a mi familia, cuyo apoyo y amor incondicional me han acompañado en cada paso de este largo sendero.

UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS

FACULTAD DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

**IMPLEMENTACIÓN DE LA METODOLOGÍA DEVOPS PARA
MEJORAR EL PROCESO DE DESPLIEGUE CONTINUO EN UNA
EMPRESA DEDICADA AL ÁREA DE SEGUROS**

Autor: Urbina Tolentino, Jhonatán Paúl

Asesor: Zavaleta Campos, Jorge Luis

Título: Trabajo de Suficiencia Profesional para optar el Título Profesional de Ingeniero de
Sistemas

Fecha: Enero 2026

RESUMEN

El presente informe se centra en la implementación de la metodología DevOps a lo largo de un proceso de despliegue continuo de una empresa en el dominio de los seguros. El problema central radica en los procesos manuales y tecnológicamente ineficientes que disminuyen la ventaja competitiva de las compañías de seguros en un mercado en digitalización que requiere transformación digital y automatización de procesos en la entrega de software.

El objetivo principal del proyecto es reingenierizar fundamentalmente los procesos tecnológicos subyacentes utilizando DevOps con la intención de aplicar métricas DORA, marcos de DevOps mejorados DORA, pipelines de entrega y despliegue automático optimizada y reducir el tiempo del ciclo de desarrollo, los errores de producción y mejorar los procesos de recuperación ante fallos.

El enfoque de la metodología es la integración del desarrollo de software con las operaciones de TI, con un enfoque en un alto grado de automatización colaborativa e interacción. Esto incluye la construcción de tuberías CI/CD con Jenkins, JFrog, GitLab, GitHub, SVN, Redgate SQL Change Automation, Ansible y otros. Este enfoque incluye el desarrollo de recetas de automatización Ansible para pruebas continuas, automatización de servidores en la nube, implementación de métricas DORA para medir el rendimiento del desarrollo y despliegue, y más.

Palabras clave: Entrega Continua, Despliegue Continuo, Seguros, Devops, Ansible AWX, AWS Cloud Development Kit (CDK).

MAJOR NATIONAL UNIVERSITY OF SAN MARCOS

FACULTY OF SYSTEMS AND INFORMATICS ENGINEERING

PROFESSIONAL SCHOOL OF SYSTEMS ENGINEERING

**IMPLEMENTATION OF THE DEVOPS METHODOLOGY TO
IMPROVE THE CONTINUOUS DEPLOYMENT PROCESS IN AN
INSURANCE COMPANY**

Author: Urbina Tolentino, Jhonatán Paúl

Advisor: Zavaleta Campos, Jorge Luis

Title: Professional Proficiency Work to obtain the Professional Title of Systems Engineer

Date: January 2026

ABSTRACT

This report focuses on the implementation of DevOps methodology throughout a continuous deployment process at a company in the insurance industry. The central problem lies in manual and technologically inefficient processes that diminish the competitive advantage of insurance companies in a digitizing market that requires digital transformation and process automation in software delivery.

The main objective of the project is to fundamentally reengineer the underlying technological processes using DevOps with the intention of applying DORA metrics, DORA-enhanced DevOps frameworks, an optimized automated deployment delivery pipeline, and reducing development cycle time, production errors, and improving failover processes.

The methodology's approach is to integrate software development with IT operations, with a focus on a high degree of collaborative automation and interaction. This includes building CI/CD pipelines with Jenkins, JFrog, GitLab, GitHub, SVN, Redgate SQL Change Automation, Ansible, and others. This approach includes developing Ansible automation

playbooks for continuous testing, automating cloud servers, implementing DORA metrics to measure development and deployment performance, and more.

Keywords: Continuous Integration, Continuous Deployment, Insurance, DevOps, Ansible AWX, AWS Cloud Development Kit (CDK).

Índice de Contenido

ÍNDICE DE TABLAS	xi
ÍNDICE DE FIGURAS	xiii
INTRODUCCIÓN	1
CAPÍTULO I - TRAYECTORIA PROFESIONAL	3
1.1. Presentación Profesional	3
1.2. Experiencia profesional	3
1.2.1. Experiencia profesional después del Bachiller	3
1.2.2. Experiencia profesional antes del Bachiller	6
1.3. Formación académica	8
CAPÍTULO II – CONTEXTO EN EL QUE SE DESARROLLA LA EXPERIENCIA	9
2.1. Empresa – Actividad que realiza	9
2.2. Valores	10
2.3. Enfoque en innovación y transformación digital	11
2.4. Visión	12
2.5. Misión	12
2.6. Metodologías ágiles y cultura de innovación	12
2.7. Responsabilidad social y sostenibilidad	13
2.8. Estrategias de crecimiento y expansión regional	13
2.9. Colaboración y alianzas estratégicas	14
2.10. Capacitación y desarrollo del talento	14
2.11. Organización de la Empresa	15
2.12. Área, cargo y funciones desempeñadas	19
2.13. Experiencia Profesional realizada en la organización	20
CAPÍTULO III – ACTIVIDADES DESARROLLADAS	22
3.1. Situación problemática	22
3.2. Solución	24
3.2.1 Objetivos	26
3.2.2 Alcance	27
3.2.3 Etapas y Metodología	30
3.2.4 Fundamentos Utilizados	32
3.2.5 Implementación de las áreas, procesos, sistemas y buenas prácticas	38
3.2.5.2 Diseño	40

3.2.5.3 Implementación.....	42
3.2.5.4 Resultados esperados y entregables	59
3.2.5.5. Métricas del proyecto	59
3.3. Evaluación.....	70
3.3.1. Evaluación económica	70
3.3.2. Interpretación del VAN y TIR.....	84
Capítulo IV - REFLEXIÓN CRÍTICA DE LA EXPERIENCIA	88
Capítulo V – Conclusiones y recomendaciones	91
5.1 Conclusiones.....	91
5.2 Recomendaciones.....	92
BIBLIOGRAFÍA.....	94

ÍNDICE DE TABLAS

Tabla 1	<i>Empresa 1</i>	3
Tabla 2	<i>Empresa 2</i>	4
Tabla 3	<i>Empresa 3</i>	5
Tabla 4	<i>Empresa 4</i>	5
Tabla 5	<i>Empresa 5</i>	6
Tabla 6	<i>Empresa 6</i>	6
Tabla 7	<i>Empresa 7</i>	7
Tabla 8	<i>Formación</i>	8
Tabla 9	<i>Idiomas</i>	8
Tabla 10	<i>Arquitectura del sistema de entrega continua implementado Elaboración Propia</i>	25
Tabla 11	<i>Herramientas empleadas en la metodología DevOps y su alineación teórica</i>	32
Tabla 12	<i>Roles de los participantes en el proyecto</i>	39
Tabla 13	<i>Objetivo y rol de los participantes en las reuniones</i>	40
Tabla 14	<i>Jenkins Stabilization PRD</i>	43
Tabla 15	<i>DevOps Charter</i>	43
Tabla 16	<i>DevOps Metrics</i>	46
Tabla 17	<i>Fortify</i>	48
Tabla 18	<i>AWS Pipelines</i>	49
Tabla 19	<i>Database Pipeline Standards</i>	54
Tabla 20	<i>Pipeline Standards</i>	54
Tabla 21	<i>Jenkins & qTest Integration</i>	55
Tabla 22	<i>Indicador AWS antes y después</i>	59
Tabla 23	<i>Indicador Jenkins antes y después</i>	61
Tabla 24	<i>Lead time for Changes (días)</i>	63
Tabla 25	<i>Comparativa Mean Time To Recovery</i>	65
Tabla 26	<i>Change Failure Rate indicador(en miles)</i>	66
Tabla 27	<i>Resumen de las métricas</i>	67
Tabla 28	<i>Diagrama de Gantt</i>	69
Tabla 29	<i>Recursos Humanos</i>	70
Tabla 30	<i>Servicios Cloud y Licencias SaaS</i>	71
Tabla 31	<i>Licencias On Premise</i>	72
Tabla 32	<i>Infraestructura Local y Virtualización</i>	73
Tabla 33	<i>Herramientas Open Source</i>	74
Tabla 34	<i>Consolidado Global de Inversión</i>	75
Tabla 35	<i>Escenario On Premise</i>	77
Tabla 36	<i>Comparación Global de Costos</i>	80
Tabla 37	<i>Flujo Económico Proyectado a 5 años</i>	81
Tabla 38	<i>Indicadores Financieros</i>	82
Tabla 39	<i>Supuestos clave de cálculo</i>	85

Tabla 40	<i>Cálculo del VAN (10%, 5 años)</i>	85
Tabla 41	<i>Detalle del VAN: flujos descontados año a año (10%)</i>	86
Tabla 42	<i>Relación Beneficio/Costo</i>	86
Tabla 43	<i>TIR</i>	87
Tabla 44	<i>TIR Equipo de Migración de repositorios para el año 2025</i>	90

ÍNDICE DE FIGURAS

Figura 1	<i>Valores Organizacionales</i>	11
Figura 2	<i>Organización de la empresa</i>	15
Figura 3	<i>Organización de los recursos de la empresa en el cliente</i>	16
Figura 4	<i>Organización del equipo DevOps en el cliente</i>	17
Figura 5	<i>Flujo de despliegue para aplicación web en .NET utilizando Jenkins, Git, Artifactory.</i>	41
Figura 6	<i>Arquitectura de Jenkins Java 17</i>	42
Figura 7	<i>Herramientas empleadas</i>	45
Figura 8	<i>Fortify Dashboard</i>	48
Figura 9	<i>Gráfico de Docker Host</i>	50
Figura 10	<i>Gráfico AWS</i>	51
Figura 11	<i>Qtest Dashboards</i>	56
Figura 12	<i>As IS del Pipeline de Base de Datos</i>	57
Figura 13	<i>To be del Pipeline de Base de Datos</i>	58
Figura 14	<i>Comparativa año 2023 y 2024</i>	60
Figura 15	<i>Comparativa 2023 vs 2024 Jenkins</i>	62
Figura 16	<i>Comparativa 2023 vs 2024</i>	64
Figura 17	<i>Comparativa 2023 - 2024</i>	66
Figura 18	<i>Migracion de Gitlab, Subversion, AWS CodeCommit</i>	90

INTRODUCCIÓN

Las compañías de seguros se enfrentan actualmente al desafío crítico de acelerar sus procesos tecnológicos para seguir siendo competitivas en un mercado cada vez más dinámico. El cambio hacia metodologías ágiles y automatizadas es más que una tendencia: es un imperativo que está definiendo el éxito organizacional en la industria de seguros.

En este sentido, hay dos variables fundamentales que determinan el éxito de la transformación digital en la industria de seguros. La primera variable es la adopción del nuevo paradigma DevOps, que se refiere a la unificación del Desarrollo de Software con las operaciones de TI a través de la colaboración automatizada, la interactividad continua y una cultura de optimización, que permite a los equipos de ventas, operaciones y servicio al cliente unirse a los equipos de desarrollo (Nash, 2022). La segunda variable se refiere a transformar el proceso de despliegue continuo es adoptando una estrategia centrada en la optimización de las canalizaciones de entrega automatizadas, la reducción del tiempo de ciclo, la minimización de errores en producción y procesos de recuperación eficientes ante fallos (Istifarulah y Tiaharyadini, 2023).

La experiencia profesional trabajada en este contexto consiste en transformar los procesos tecnológicos de una aseguradora mediante la incorporación de la metodología DevOps con las métricas dignas de mención DORA (DevOps Research and Assessment) como KPI. Estas métricas miden objetivamente el éxito de una implementación en términos de frecuencia de despliegue, entrega, recuperación y fallo de cambios. La investigación actual sostiene que dentro de los paradigmas DevOps, las pruebas continuas migradas a la nube son un avance para los aseguradores con capacidad de escalabilidad y pruebas que antes no eran posibles dentro de los modelos tradicionales. Además, los modelos predictivos impulsados por IA aumentan la eficiencia de las canalizaciones DevOps (Enemosah, 2025) mientras que los modelos predictivos mejoran considerablemente la eficiencia de las ejecuciones DevOps (Gollapudi y Subbian, 2025).

A continuación, se detalla la estructura del trabajo:

En el CAPÍTULO I, la experiencia del candidato en el sector asegurador aplicando DevOps, automatización de procesos y tecnología contemporánea de infraestructura como servicios.

En el CAPÍTULO II, la experiencia del candidato, en conjunción con la compañía de seguros, el trabajo profesional llevado a cabo y los desafíos tecnológicos particulares de la industria.

En el CAPÍTULO III, se describe el trabajo del candidato, incluyendo la situación problemática, los objetivos, la solución del problema y la metodología del ciclo de vida de DevOps.

En el CAPÍTULO IV, la experiencia del candidato refleja una derivada subyacente. Se atendieron el problema, la experiencia y la formación de planificadores técnicos en el cambio organizacional y cultural de DevOps.

En el CAPÍTULO V, la evidencia del candidato en el trabajo de Suficiencia Profesional, el impacto medible en indicadores clave de rendimiento y otras lecciones aplicables en sectores similares en la industria de seguros.

CAPÍTULO I - TRAYECTORIA PROFESIONAL

1.1. Presentación Profesional

Profesional con grado académico de Bachiller en Ingeniería de Sistemas de la Universidad Nacional Mayor de San Marcos. Actualmente el autor desempeña labores en la empresa la cual es objetivo de este trabajo liderando proyectos de Automatización Devops con clientes de la compañía además cuenta con experiencia en previas implementaciones Devops en on premise, nube de Azure y AWS.

1.2. Experiencia profesional

1.2.1. Experiencia profesional después del Bachiller

Tabla 1

Empresa 1

Consultora – Perú	
Diciembre 2023 a la fecha	
Cargo: Devops Developer	
Funciones	- Diseñar y construir pipelines CI/CD óptimos de aplicaciones e infraestructura.

Nota. Empresa Consultora donde soy actualmente Devops desarrollador.
Elaboración propia

Tabla 2

Empresa 2

Ensitech México	
Octubre 2022 a octubre 2023	
Cargo: Site Reliability Engineer	
Funciones	- Gestión y ampliación de la plataforma de microservicios siguiendo SOPs (Procedimientos operativos estándar para implementación, configuración y resolución de problemas). - Ampliación de conocimientos en Azure, Kubernetes y tecnologías relacionadas.
Logros	- Creación de nuevas automatizaciones powershell. - Creación de Automatización en Python. - Genero visibilidad por medio de Grafana - Generar visibilidad por medio de Azure Monitor - Despliegue continuo por Pipeline de Elastic Contianer Service AWS

Nota. Empresa Ensitech Mexico donde fui Site Reliability Engineer.

Elaboración propia

Tabla 3*Empresa 3*

Globant	
Agosto 2021 a septiembre 2022	
Cargo: Devops	
Funciones	- Trabajé como DevOps en el equipo de infraestructura de Globant, creando pipelines CI/CD para microservicios y apps móviles (Android/iOS).
Logros	- Implementación de plataforma de microservicios en Azure Kubernetes Service - Generación de Pipelines en Elastic Container Service AWS tanto integración continua como despliegue

Nota. Empresa Globant donde fui Devops. Elaboración propia.

Tabla 4*Empresa 4*

NTT Data Perú	
Agosto 2020 a febrero 2021	
Cargo: Ingeniero de Operaciones De Ti/Devops Engineer	
Funciones	- Brindar soporte a aplicaciones utilizando Docker y Kubernetes, y administrar entornos en OpenShift.
Logros	- Documentar los microservicios desplegados en Azure y Openshift. - Implementar Azure WAF en los 3 ambientes dev/qa/prod. - Implementacion de Dynatrace como herramienta de monitoreo en Infraestructura On Premise/cloud.

Nota. Empresa NTT Data Perú donde fui Ingeniero de Operaciones De Ti/Devops Engineer. Elaboración propia

Tabla 5

Empresa 5

SERVICIOS EXTERNOS SAC	
Octubre 2019 a abril 2020	
Cargo: Devops Engineer	
Funciones	- Crear y mantener pipelines en GitLab CI, utilizando Ansible para la configuración como código. - Automatizar copias de seguridad de GitLab. - Gestionar servidores on-premise en distintos entornos y desplegar servicios con Docker y Kubernetes
Logros	- Diseño, desarrollo y prueba de un flujo de CI/CD a extremo para aplicaciones en Java, JavaScript y Python.

Nota. Empresa SERVICIOS EXTERNOS SAC donde fui Devops Engineer.

Elaboración propia

1.2.2. Experiencia profesional antes del Bachiller

Tabla 6

Empresa 6

Tata Consultancy Service TCS	
Febrero 2018 a marzo 2019	
Cargo: Trainee de Sistemas	
Funciones	- Crear nuevos componentes Software en Angular y APIS de Spring Boot. - Migración de componentes front de Vue 2 a Vue 3. - Documentar APIS con Swagger
Logros	- Migrar la página web de la empresa de un stack antiguo a uno nuevo usando mejores tecnologías

Nota. Empresa Tata Consultancy Service TCS donde fui Trainee de Sistemas.
Elaboración propia

Tabla 7

Empresa 7

Peru Software Factory S.A.	
Julio 2017 a enero 2018	
Cargo: Programador de sistemas	
Funciones	- Crear nuevos módulos en Angular 2 de una aplicación dedicada a la gestión de proyectos. - Desplegar la aplicación en ambientes de pruebas.
Logros	- Entregar aplicación de gestión de proyectos creada desde cero con documentación.

Nota. Empresa Peru Software Factory S.A. donde fui Programador de Sistemas. Elaboración propia

1.3. Formación académica

Tabla 8

Formación

Grado Académico de Bachiller en Ingeniería de Sistemas Escuela Académico Profesional de Ingeniería de Sistemas Facultad de Ingeniería de Sistemas e Informática Universidad Nacional Mayor de San Marcos
- Certificación SAFE Devops - SAFE - Certificación GitHub Foundations – Github - Certification Prometheus Certified Associate - The Linux Foundation (LF) - Certification Security, Compliance, and Identity Fundamentals – Microsoft - Certificación Poder Plataforma Fundamentals – Microsoft - Certificación Azure Data Fundamentals – Microsoft - Certificación Azure AI Fundamentals – Microsoft - Certificación Azure Fundamentals – Microsoft - Certification Certified Kubernetes Administrator - The Linux Foundation (LF) - Certificación Azure Administrador Associate – Microsoft - Certificación Cloud Practitioner – AWS - AWS Solutions Architect Associate – Bootcamp Institute - FinOps Certified Practitioner – The Linux Foundation(LF) - Mastering Devops en la Era del AI-Driven Cloud – Desafío Latam

Nota. Elaboración propia

Tabla 9

Idiomas

Español	Nativo
Ingles	Intermedio

Nota. Elaboración propia

CAPÍTULO II – CONTEXTO EN EL QUE SE DESARROLLA LA EXPERIENCIA

2.1. Empresa – Actividad que realiza

Esta empresa tecnológica, con raíces que datan de 1986 y parte de un conglomerado diversificado de origen norteamericano, ha evolucionado hasta convertirse en un referente en la transformación digital. Actualmente, cuenta con una plantilla superior a 12,000 especialistas altamente capacitados distribuidos en al menos 10 países, incluidos México, Brasil, Chile, Colombia, Perú, Argentina, Ecuador, Estados Unidos y España.

Su capacidad tecnológica y operativa se refleja en un modelo organizacional basado en prácticas horizontales especializadas, que incluyen desarrollo de software, testing, omnicanalidad, infraestructura, hiperautomatización y servicios en la nube, y prácticas verticales orientadas a sectores estratégicos como salud, manufactura, agroindustria, finanzas, minería, smart cities, retail y transporte. Este enfoque les permite ofrecer soluciones integrales, agnósticas a tecnología, adaptando herramientas y tecnologías líderes como Google Cloud, Microsoft Azure y Oracle.

Su portafolio incluye todo, desde consultoría a través de metodologías ágiles, hasta el desarrollo e implementación de soluciones paquetizadas para problemas reales en diversas industrias. En el área financiera, por ejemplo, implementan tecnologías innovadoras como grabación inteligente y análisis de voz.

La empresa opera centros regionales de excelencia, como su Centro de Excelencia de Cloud en Argentina, que ofrece servicios con alto estándar para toda la región. Con la integración en la empresa de unidades especializadas en ciberseguridad, buscan brindar un portafolio completo que mezcla conectividad, seguridad y transformación digital, alineándose a las demandas cambiantes del mercado corporativo.

2.2. Valores

Su cultura corporativa gira en torno a un propósito general de “hacer posible un mundo mejor”, que se concreta en una atención especial hacia la innovación sostenible, la equidad e inclusión y el desarrollo humano. Los valores clave son: sustentabilidad, integridad, eficiencia, colaboración, innovación y atención a la experiencia de clientes.

Del mismo modo, la organización se ha comprometido con la responsabilidad social y la creación de impactos positivos en las comunidades donde tiene presencia, participando en activos en proyectos destinados a disminuir el gasto energético y a la contaminación ambiental generada por el uso de la inteligencia artificial y otras tecnologías, reafirmando cuanto contribuye al objetivo de ser sostenibles y de sostener operaciones eficientes de su enfoque sostenible.

Figura 1

Valores Organizacionales



Nota. Los principios fundamentales que guían a la organización constituyen los valores esenciales que cada miembro debe tener siempre presentes en su labor diaria, siendo estos elaborados internamente. Tomado del portal público de la Empresa. Elaboración propia

2.3. Enfoque en innovación y transformación digital

La empresa ha logrado destacar por la forma en que se atreve a innovar. No se limita a seguir las tendencias: las adopta y las transforma. Aprovecha tecnologías como la hiperautomatización, la inteligencia artificial, el análisis de datos y los servicios en la nube para hacer que sus clientes avancen con mayor rapidez en su transformación digital. Su meta no es solo modernizar procesos, sino también ayudarlos a competir con fuerza en un mercado que no deja de cambiar. Además, participa activamente en importantes eventos tecnológicos, convirtiéndose en uno de los referentes en la digitalización de sectores clave para el crecimiento sostenible de Latinoamérica, como la agricultura y la minería. Al trabajar de la mano con otras divisiones del grupo enfocadas en conectividad y ciberseguridad, logra

ofrecer soluciones completas, pensadas para los retos de un mundo empresarial cada vez más global e interconectado.

2.4. Visión

La consultora tiene la intención de ser identificada como la líder global en la movilidad de la Sociedad Digital, alterando el sector tecnológico con las soluciones innovadoras que generen la mejor de las experiencias de los diferentes usuarios. Nos posicionamos a nivel internacional como agentes de cambio, la transformación digital, con más de 30 años de experiencia en la creación de valor sostenido en el ecosistema tecnológico norteamericano de un conglomerado industrial de capital mexicano de clase mundial.

2.5. Misión

La misión de la entidad en cuestión, la de integrar de forma interdependiente las TIC, las comunicaciones y los servicios digitales con el fin de proporcionar soluciones de soporte adecuadas para el logro de los objetivos estratégicos de los clientes.

Se orienta hacia la gestión proactiva del riesgo tecnológico, hacia la gestión de incidentes adecuada, así como hacia la gestión de servicios antifraude para proveer productos sólidos y seguros. Integra acuerdos estratégicos, tanto internacionales como locales, para garantizar que las aplicaciones y sistemas empresariales sean funcionales, seguros y en constante evolución.

2.6. Metodologías ágiles y cultura de innovación

La compañía ha conseguido también un entorno en la que las metodologías ágiles son las que predominan en la gestión diaria. No se trata solamente de hacer las cosas rápido, sino de hacerlas bien, con calidad y en constante coordinación con los clientes. Esa cercanía nos ayuda a adaptarnos fácilmente a los cambios y a responder con soltura a los desafíos que surgen en un entorno tan cambiante. Este modo de trabajar nos motiva a seguir creciendo, a buscar mejoras continuas y a enfocar nuestros esfuerzos en resultados reales, medibles y

rentables. Es justamente eso lo que nos diferencia de los demás. Además, fomentar una cultura donde se pueda hablar abiertamente, aprender de los errores y probar nuevas ideas hace que la innovación deje de ser una meta y se convierta en una costumbre. Todos estos elementos fortalecen nuestra posición y nos mantienen un paso adelante en un mercado cada vez más competitivo.

2.7. Responsabilidad social y sostenibilidad

El compromiso social de la empresa no se queda solo en lo que hace dentro del mundo comercial; forma parte de su esencia y de todo lo que representa. La sostenibilidad ambiental ocupa un lugar clave dentro de su estrategia, y cada acción está pensada para cuidar el planeta. Se trabaja constantemente en mejorar la eficiencia energética de tecnologías como la inteligencia artificial, con el objetivo de reducir al máximo la huella de carbono. Al mismo tiempo, la empresa impulsa políticas que promueven la inclusión, la equidad y la diversidad, garantizando que todas las personas, sin importar el país en el que estén, tengan las mismas oportunidades. La convicción detrás de todo esto es simple: el crecimiento real no se mide solo en números, sino en el impacto positivo que se genera en la sociedad y en el entorno.

2.8. Estrategias de crecimiento y expansión regional

De este modo, nuestra manera de ver el crecimiento se traduce, por un lado, en seguir expandiéndonos en la zona, y por otro lado, en la apertura de nuevas sucursales y nuevas oficinas de desarrollo de software, las cuales nos aportan la capacidad de acercarnos más a nuestros clientes y nos permiten, por lo tanto, conocerlos más y adaptarnos mucho más rápidamente a la demanda del mercado. La finalidad, visto desde una óptica académica, está clara: incrementar el peso de la empresa en sectores que tienen un especial interés: las finanzas, las smart cities, el transporte, la energía, etc.; y ser capaz de convertirse en un motor de innovación y de cambio. Este crecimiento se produce a partir de la inversión continuada, sistemática y mantenida en talento especializado, en infraestructuras modernas y en tecnologías de última generación, de tal manera que nuestras soluciones sean siempre eficientes, creativas y adecuadas a los retos del momento presente.

2.9. Colaboración y alianzas estratégicas

Basado en todo lo anterior, podemos afirmar que la compañía estructura su modelo de negocio en una sólida red de partners que representan auténticos referentes tecnológicos de gran renombre, a modo de ejemplos, como pueden ser los grandes del cloud: Microsoft Azure, Google Cloud, Oracle, etcétera. Gracias a lo cual somos capaces de edificar soluciones fiables, personalizadas, flexibles y apropiadas a los diferentes sectores en los que nuestros clientes desarrollan su negocio. Asimismo, contribuyen a devolver el modo de trabajar tanto en la compañía como en los partners, donde emergen valores como la confianza, la transparencia o el intercambio de ideas, por el hecho de ser la conexión constante entre ellos la que permite aprender del uno al otro y evolucionar en los proyectos generando valor y también construir juntos un ecosistema innovador en el cual ambos crecen.

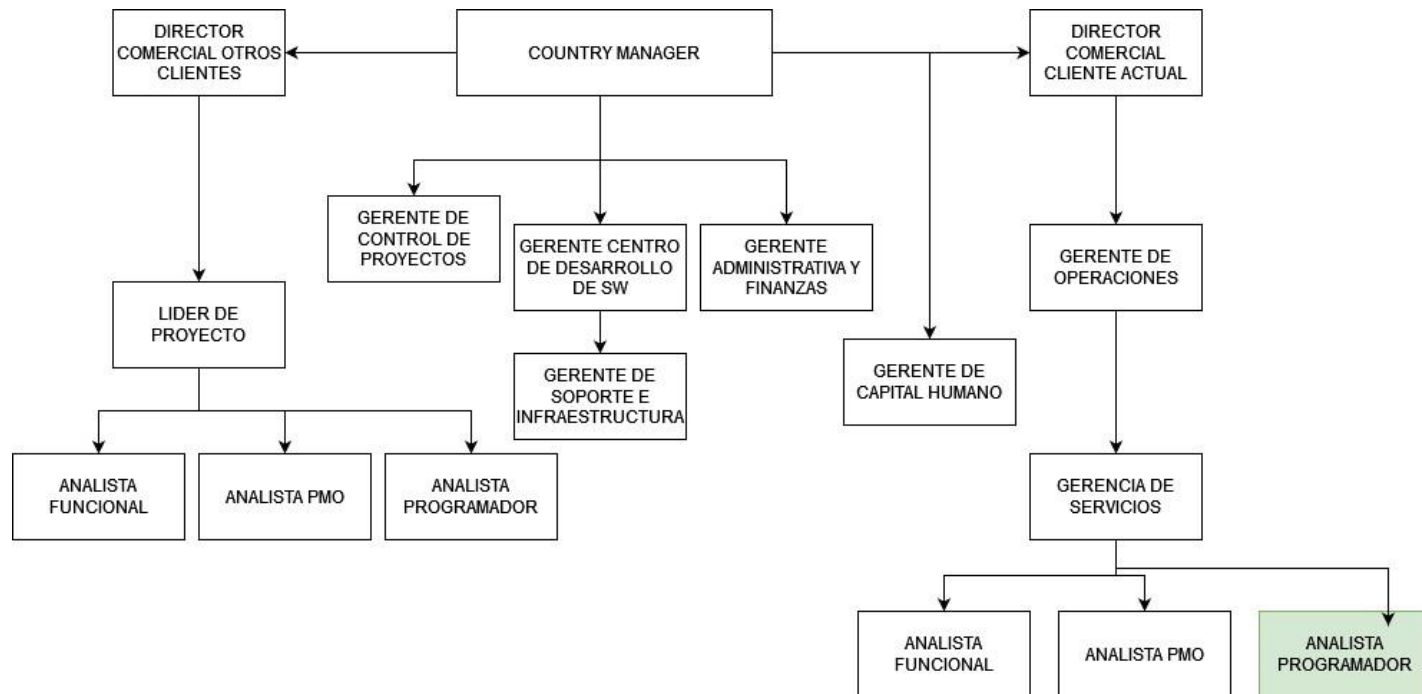
2.10. Capacitación y desarrollo del talento

La empresa apuesta de forma constante por el crecimiento de su equipo, invirtiendo en la formación y desarrollo de cada persona. Lo hace a través de distintos programas de capacitación, plataformas digitales de aprendizaje y evaluaciones que ayudan a medir el progreso y las oportunidades de mejora. El personal recibe entrenamiento en temas técnicos, ética profesional, sostenibilidad, gestión de riesgos y habilidades interpersonales, buscando que todos estén listos para enfrentar los retos de un entorno tecnológico en constante cambio. Además, la organización se preocupa por el bienestar y la experiencia de sus colaboradores, entendiendo que ambos son clave para conservar un ambiente de trabajo positivo, motivador y altamente productivo.

2.11. Organización de la Empresa

Figura 2

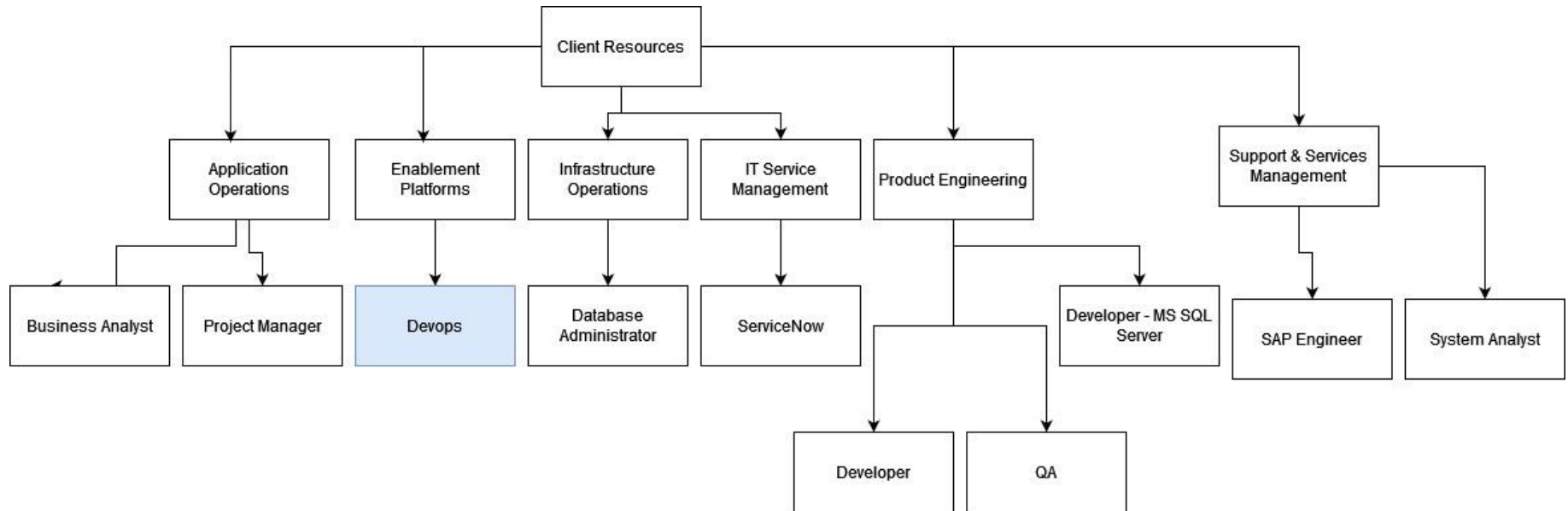
Organización de la empresa



Nota. El diagrama representa la estructura jerárquica y funcional de la empresa, donde se visualizan las principales áreas de gestión y los flujos de dependencia entre las distintas posiciones. Elaboración propia

Figura 3

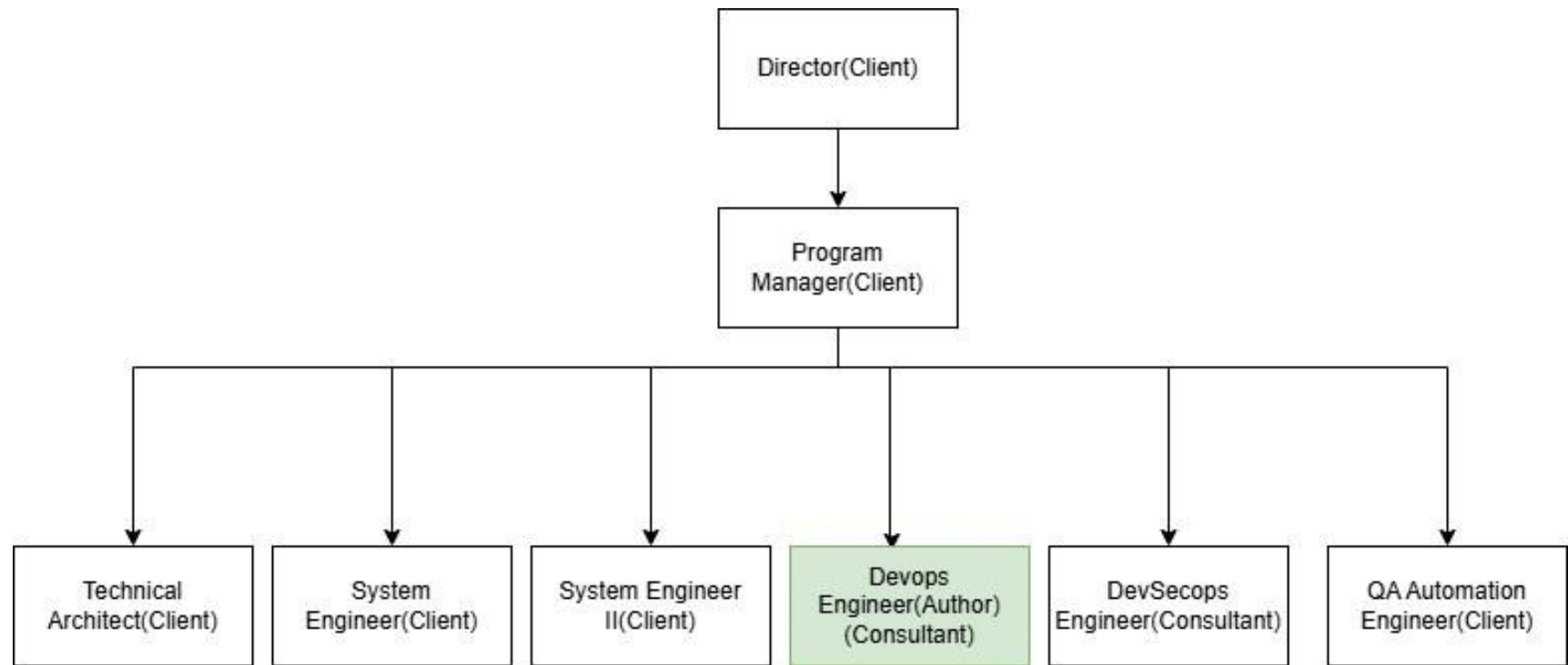
Organización de los recursos de la empresa en el cliente



Nota. La figura refleja cómo están agrupados los recursos en el cliente según áreas funcionales como desarrollo, gestión de servicios y plataformas, ubicando al rol de *DevOps* bajo el dominio de Enablement Platforms. Elaboración propia

Figura 4

Organización del equipo DevOps en el cliente



Nota. La figura muestra la estructura jerárquica entre los integrantes del equipo del cliente y los consultores externos, resaltando la posición del autor como DevOps Engineer en calidad de consultor. Elaboración propia

La estructura de la empresa consultora (Figura2) sigue un modelo de organización jerárquico pensado para mantener una dirección estratégica clara. En la parte superior se encuentra el Country Manager, encargado de marcar el rumbo y asegurar que la visión corporativa se cumpla. La compañía se divide en dos grandes áreas: la comercial, dirigida por los responsables de negocio, y la operativa, que cuenta con cuatro gerencias clave: Control de Proyectos, Soporte e Infraestructura, Administración y Finanzas, y Capital Humano. En el ámbito operativo, el director de Operaciones coordina equipos técnicos a través de una estructura matricial que permite una distribución flexible de los recursos adecuada a cada proyecto.

Área de DevOps y DevSecOps Engineers

El área de DevOps (Figura 3) es un área importante dentro de la estructura operativa, puesto que forma parte del equipo de Enablement Platforms. La actividad que se recoge en torno al área de DevOps establece claramente ese cambio que ya se ha producido desde las antiguas formas de desarrollar y operar hacia una forma de desarrollo que resalta por ser mucho más cíclica, continua, unificada y automatizable. El área de DevOps se presenta como la forma de llegar a ser un puente entre las diferentes áreas de la empresa para permitir el proceso de la colaboración y de la automatización tecnológica con el fin de permitir llegar a una gestión más ágil.

Responsabilidades y funciones del área DevOps

El organigrama (Figura 4) representa la estructura jerárquica de un equipo técnico en el que el director del lado del Cliente que es la aseguradora lidera la organización y delega la gestión operativa al Program Manager, también del cliente. Bajo su supervisión se encuentra un grupo mixto conformado por empleados directos del cliente (como arquitectos técnicos, ingenieros de sistemas y QA) y consultores externos, entre los cuales se encuentra el autor del trabajo desempeñando el rol de DevOps Engineer. Como consultor, se trabaja en colaboración con el equipo interno, pero no se forma parte directa de la nómina del cliente, lo que indica una relación contractual diferente. Este esquema refleja claramente la distribución de roles y la dependencia organizacional dentro del proyecto

Los expertos en DevOps desarrollan funciones a lo largo del ciclo de vida del desarrollo de software. En la gestión de los pipelines de entrega continua (CI/CD), desarrollan pipelines automáticos con herramientas como Jenkins, GitLab CI/CD, GitHub Actions y Azure DevOps , integrando actividades de build, pruebas automáticas y análisis de calidad, gestión de artefactos con JFrog Artifactory y despliegue automatizado en múltiples entornos.

El área aplica las métricas DORA para evaluar de forma objetiva y cuantificable el rendimiento del proceso de entrega (frecuencia de despliegue, tiempos de entrega de cambios, tiempo medio de recuperación (MTTR), tasa de fallos, etc.). Usan herramientas de monitorización como Grafana, Prometheus, Azure Monitor y Dynatrace para garantizar la visibilidad. En relación a la arquitectura cloud desarrollan soluciones en AWS y Azure usando contenedores (Docker), orquestación con Kubernetes y servicios gestionados, para facilitar la escalabilidad y eficiencia.

DevSecOps Engineers: Seguridad Integrada

Los especialistas en DevSecOps se encargan de integrar herramientas de análisis automatizado en los pipelines CI/CD, utilizando herramientas para SAST (análisis estático del código con SonarQube, Checkmarx, Fortify), DAST (pruebas dinámicas con OWASP ZAP), SCA (escaneo de dependencias con Snyk, WhiteSource) y Container Security Scanning (análisis de imágenes Docker con Trivy, Aqua Security).

Además, configuran sistemas de monitoreo de seguridad, incluyendo IDS/IPS y herramientas SIEM, para la detección continua de amenazas y respuesta automatizada. Implementan controles automatizados para asegurar el cumplimiento de regulaciones como PCI-DSS, GDPR, SOC 2 e ISO 27001, lo cual es especialmente crítico en la industria de seguros, donde se maneja información altamente sensible.

2.12. Área, cargo y funciones desempeñadas

Área de Devops, en una empresa consultora para un cliente de Seguros.

Cargo de Devops Sr. En el equipo de Infraestructura del cliente bajo la supervisión del Equipo de Plataforma Empresarial encargados del desarrollo de los componentes software de la empresa.

Tuve funciones globales como:

- La de gestión de accesos a las herramientas Jfrog Gitlab Github, Mantenimiento y configuración de infraestructura On premise con Ansible, Mantenimiento de Pipelines tanto en On premise y AWS
- Definición, diseño de los modelos implementados de Integración Continua y Despliegue Automatizado
- Establecimiento de lineamientos, buenas prácticas y mejora de procesos relacionados a DevOps.
- Coordinación con los equipos de desarrollo, calidad y Operaciones con el objetivo de identificar las necesidades de cada frente para la implementación de DevOps
- Identificación de oportunidades de mejora en las implementaciones de DevOps
- Construcción de pipelines de Integración Continua y Despliegue Automatizado del proyecto

2.13. Experiencia Profesional realizada en la organización

El autor realizó un trabajo destacado en la transformación de los procesos de desarrollo y de despliegue de software, enfrentándose a los grandes retos de la implementación manual y la ineficiencia operativa. Se abordó este reto mediante la implementación de soluciones DevOps que automatizaban completamente el ciclo de vida del software. La implementación incluyó la definición de pipelines para bases de datos mediante Redgate SQL Change Automation, así como la automatización de herramientas DevOps como Jenkins, JFrog, GitLab, GitHub, SVN y Redgate. También se desarrollaron Playbooks de Ansible para automatizar la estandarización de los servidores heterogéneos. La actividad desplegada por el autor ha sido de tipo sistémico y multinivel ya que los procesos de garantizar, modernizar las infraestructuras y optimizar procesos de trabajo tecnológicos y

procesos de automatización se ejecutaban concurrentemente a medida que el trabajo se iniciaba.

Las tácticas planteadas por el autor lograron mejorar las implementaciones manuales que se estaban llevando a cabo, y originaron importantes mejoras en los plazos de entrega de software. Este traslado laboral se daba sobre la automatización de pruebas, la configuración de entornos y la ejecución de pipelines, y tuvo una repercusión en la productividad operacional y en la fiabilidad de los despliegues y el apoyo a la resolución de incidencias (por ejemplo, en la configuración del entorno local; en la generación de las métricas DORA...) se convirtió en un paso clave para establecer una base sólida, adecuada y robusta de unas prácticas DevOps codificadas. La consecuencia última fue una optimización de los procesos tecnológicos que acarrea una mejora en los plazos de implementación, un bajo número de errores operativos y un brote de arquitectura ágil, al mismo tiempo que segura y escalable. El poder aplicar la propuesta descrita permite mejorar diferentes aspectos de la empresa tales como la productividad, asimismo establecer de forma ordenada las respectivas actividades y por último permite ser más competitivos frente a otras entidades.

CAPÍTULO III – ACTIVIDADES DESARROLLADAS

3.1. Situación problemática

El entorno global relacionado con DevOps se ha ido forjando a partir de una labor de investigación sostenida que, en un periodo de nueve años, ha recogido evidencias de más de 36 000 profesionales de desarrollo y operaciones. Este conjunto de datos, considerado uno de los más robustos en su tipo, ha permitido comprender cómo los equipos de alto desempeño combinan prácticas técnicas, optimización de procesos y cultura colaborativa para mejorar su eficiencia organizacional. Tales hallazgos constituyen una base adecuada para entender los retos actuales del ciclo DevOps en organizaciones donde la continuidad operativa resulta crítica para mantener competitividad (Harvey y DeBellis, 2023).

El análisis histórico muestra además que el liderazgo técnico y la automatización temprana generan diferencias significativas en el rendimiento. El reporte de 2017 evidenció que los equipos guiados por líderes con visión clara, comunicación efectiva y capacidad de innovación alcanzaban mejores niveles de desempeño. Paralelamente, la automatización se consolidó como eje estratégico: los equipos de alto rendimiento automatizaban configuración, pruebas, liberaciones y aprobaciones, reduciendo tiempos críticos y eliminando cuellos de botella. Este enfoque explica por qué procesos de despliegue pueden presentar fricción operativa cuando no alcanzan niveles equivalentes de madurez en automatización (Forsgren et al., 2018).

Actualmente, la discusión internacional se centra en la integración total de CI/CD, la seguridad continua y la incorporación de inteligencia artificial. El State of CI/CD Report 2024 confirma una alta adopción de integración continua, pero advierte que la falta de acoplamiento entre herramientas puede reducir la velocidad de despliegue en producción, obligando a restauraciones del proceso y generando menor valor en la entrega (Sanyika, 2024). Investigaciones recientes también evidencian que la inteligencia artificial no elimina problemas del pipeline, sino que expone deficiencias estructurales existentes en el flujo de

trabajo; el 74 % de equipos que usan IA reconoce la necesidad de consolidar su toolchain frente al 57 % de quienes no la utilizan (Steer, 2024).

Sin embargo, pese a la amplia adopción de DevOps, la literatura muestra una brecha importante en la medición objetiva de su impacto en calidad. La mayoría de estudios evalúa seguridad, pero pocos consideran otras características del modelo ISO/IEC 25010, dificultando validar mejoras reales en mantenibilidad y rendimiento. Asimismo, muchas organizaciones carecen de métricas operativas confiables para tomar decisiones sobre sus ciclos de despliegue (Korkmaz y Aydin, 2024).

Este panorama teórico se reflejaba en la empresa aseguradora objeto de estudio, donde el proceso de entrega de software presentaba limitaciones operativas concretas. El despliegue de componentes era manual y no versionado, dependía exclusivamente de personal SysOps y DBA con alta carga laboral, y estaba condicionado por estricta segregación de roles que impedía a los desarrolladores intervenir directamente. Las solicitudes debían gestionarse mediante un sistema de tickets con múltiples aprobaciones jerárquicas, generando tiempos prolongados de atención.

Las pruebas de concepto y primeros pipelines de integración continua presentaban cuellos de botella técnicos: las pruebas unitarias, la compilación y el escaneo de vulnerabilidades se ejecutaban de manera secuencial y no paralela, incrementando los tiempos de ejecución. Además, las dependencias se descargaban nuevamente en cada ejecución por ausencia de mecanismos de cacheo, alargando innecesariamente la construcción de artefactos.

Los ambientes de despliegue On premise eran complejos de configurar. Requerían instalación manual de paquetes en versiones compatibles, certificados de seguridad, configuración de servicios Windows, servidores IIS y coexistencia de múltiples tecnologías (Java, .NET containerizado y no containerizado, automatizaciones) así como también la publicación de APIs en ambientes de despliegue en AWS. Esto incrementaba la probabilidad de error humano durante configuraciones productivas. Los archivos de configuración

contenían secretos sensibles administrados manualmente por desarrolladores, sin una bóveda centralizada, elevando el riesgo de exposición de credenciales.

La organización tampoco disponía de entornos replicables ni infraestructura como código para crear ambientes en On-Premise o AWS, ni nodos dedicados para pruebas automatizadas (Selenium) o para la validación de nuevos plugins Jenkins. En consecuencia, los despliegues resultaban lentos, inconsistentes y propensos a fallos, generando incidentes productivos.

Desde el punto de vista del negocio, estas deficiencias impedían lanzar promociones y nuevas funcionalidades en tiempos competitivos, afectando el time-to-market. La empresa no podía experimentar rápidamente nuevas iniciativas ni recibir retroalimentación temprana, reduciendo la captación de clientes potenciales y deteriorando la experiencia del usuario debido a inestabilidad en los sistemas productivos. Esto ocasionaba pérdidas económicas asociadas a oportunidades comerciales no aprovechadas y a la insatisfacción del cliente final.

En síntesis, la aseguradora presentaba un proceso de entrega de software manual, dependiente de especialistas, sin estandarización ni automatización, con alta probabilidad de error humano y baja velocidad de despliegue, lo que impactaba directamente en la competitividad del negocio.

3.2. Solución

La solución consistió en el diseño e implementación de una arquitectura de entrega continua basada en DevOps que permitió transformar el proceso manual de despliegue en un sistema automatizado de liberación controlada de software. La propuesta no se limitó a la adopción de herramientas, sino a la definición de un flujo operativo reproducible, medible y seguro para la construcción, validación y liberación de artefactos en un entorno On-Premise y AWS.

Se implementó un modelo de control de versiones basado en estrategia Trunk-Based Development, donde las ramas de trabajo cortas convergen continuamente hacia la rama principal, evitando divergencias prolongadas y reduciendo conflictos de integración. Sobre este esquema se configuró un pipeline multibranch que detecta automáticamente nuevas ramas o cambios, ejecutando procesos de validación independientes por cada contribución.

Tabla 10

Arquitectura del sistema de entrega continua implementado Elaboración Propia

Capa del proceso	Componente implementado	Estrategia técnica aplicada	Objetivo operativo
Control de código fuente	Repositorio Git multibranch	Trunk-Based Development	Reducir conflictos de integración y permitir integración continua
Integración continua	Pipeline automático por rama	Validación por commit	Detectar errores tempranos en cada contribución
Construcción	Compilación paralela	Paralelización de etapas	Disminuir tiempo de build
Calidad de software	Pruebas unitarias automatizadas	Ejecución simultánea	Validar funcionamiento funcional inmediato
Seguridad	Escaneo de vulnerabilidades	DevSecOps Shift-Left	Detectar vulnerabilidades antes del despliegue
Optimización	Cacheo de dependencias	Reutilización de artefactos	Reducir tiempos repetitivos de descarga
Gestión de configuración	Infraestructura como código	Entornos declarativos replicables	Eliminar configuraciones manuales
Gestión de secretos	Bóveda centralizada	Externalización de credenciales	Evitar exposición de información sensible
Entrega continua	Despliegue Blue/Green	Publicación paralela controlada	Evitar interrupciones al usuario
Recuperación	Rollback automático	Reversión inmediata Patron Circuit Breaker	Minimizar impacto de fallos
Orquestación	Flujo automático de aprobación	Eliminación de tickets manuales	Reducir tiempos burocráticos
Operación	Entornos On-Premise + AWS	Arquitectura distribuida	Escalabilidad y continuidad operativa

Nota: Elaboración Propia

El pipeline de integración continua fue rediseñado bajo ejecución paralela. Las etapas de compilación, pruebas unitarias y análisis de vulnerabilidades se ejecutan simultáneamente,

incorporando además cacheo de dependencias para eliminar descargas repetitivas. Esto permitió transformar el proceso de construcción de artefactos en una operación determinística y repetible.

Posteriormente se implementó un esquema de infraestructura como código (IaC) para la creación de ambientes replicables tanto en servidores locales (Configuración as Code) como en AWS. La configuración de servicios, certificados, variables de entorno y servidores de aplicaciones pasó de ser manual a declarativa, permitiendo levantar entornos completos bajo la misma configuración en cualquier etapa del ciclo de vida.

El proceso de entrega continua se diseñó mediante despliegues controlados tipo Blue/Green Deployment, donde la nueva versión se publica en un entorno paralelo al productivo y el tráfico es redirigido únicamente tras validar su estabilidad. Se incorporó además un mecanismo de rollback automático, permitiendo restaurar la versión previa en caso de fallo sin intervención manual.

La liberación de software fue orquestada mediante flujos de aprobación automatizados, eliminando la dependencia de tickets manuales. Las credenciales y secretos fueron externalizados a una bóveda centralizada, gestionando su ciclo de vida de forma segura y evitando su exposición en archivos de configuración.

El resultado fue un sistema de entrega continua capaz de construir, probar, desplegar y revertir versiones de software de manera automática, reduciendo la intervención humana en operaciones repetitivas y disminuyendo la probabilidad de error operacional.

3.2.1 Objetivos

Objetivo general

Implementar una arquitectura DevOps para mejorar el desempeño del proceso de entrega de software, incrementando la frecuencia de despliegue, reduciendo el tiempo de entrega de cambios, disminuyendo el tiempo de recuperación ante fallos y reduciendo la tasa de fallos en producción, medidos mediante métricas DORA.

Objetivos específicos

- Incrementar la Deployment Frequency mediante automatización del pipeline CI/CD, pasando de un esquema manual a despliegues automatizados y repetibles.
- Reducir el Lead Time for Changes automatizando compilación, pruebas, seguridad y despliegue en paralelo dentro del pipeline.
- Disminuir el Mean Time To Recovery (MTTR) implementando despliegues Blue/Green y rollback automático.
- Reducir el Change Failure Rate incorporando pruebas automatizadas, escaneo de vulnerabilidades y entornos replicables.

3.2.2 Alcance

El alcance de la presente investigación se dividirá en fases interrelacionadas, como las que hacen referencia al análisis de los pipelines necesarios para la integración y a los aspectos específicos de su despliegue continuo en pipelines de servicios ya existentes. A continuación, se describieron de forma categórica los principales puntos de la investigación que estarán dentro del alcance del proyecto:

Determinación de herramientas críticas para la implementación de DevOps Se explica que, en esta fase se realizará la evaluación y selección de las herramientas necesarias para la implementación de la metodología DevOps. Las herramientas críticas para la integración continua (CI) y entrega continua (CD) del software serán:

- Jenkins para la automatización de la ejecución de los pipelines de integración y despliegue.
- Docker para la automatización de la gestión y la creación de entornos de ejecución consistentes.
- GitLab como plataforma para el control de versiones y para la integración con el repositorio de código.
- test para la gestión de pruebas y para la integración de los resultados de la ejecución de pruebas con los pipelines existentes.
- GitLab/Github/AWS Codecommit como plataforma para el control de versiones y para la integración con el repositorio de código.

- qTest/Selenium para la gestión de pruebas y para la integración de los resultados de la ejecución de pruebas con los pipelines existentes.

La evaluación de cada herramienta se realizará en base a su capacidad de integración con el entorno existente, a su facilidad de uso y a su adecuación a las necesidades del proyecto actual.

a) Diseño y despliegue de pipelines de integración y entrega continua

En la etapa que aboca al diseño y puesta en marcha de los pipelines, la investigación se centrará en construir el flujo automatizado que posibilitará mantener la integración y la entrega continua del software dentro del proyecto. Para ello se construirán pipelines declarativos en Jenkins que configuren, desde la organización, las actividades clave del ciclo. La automatización de compilaciones, pruebas y validaciones prontas permitirá detectar fallos antes de que se escalen, asegurando que cada modificación realizada por el equipo no suponga un riesgo a la estabilidad del sistema. Por otra parte, se establecerá un proceso de entrega que permitirá mover los cambios aprobados hacia los ambientes de staging o producción con mínima intervención humana, lo que irá a permitir reducir el tiempo que puede transcurrir entre la construcción de una mejora y su entrega al usuario final. El objetivo es que toda la cadena (desde la construcción hasta el despliegue) funcione con consistencia, trazabilidad y autonomía.

b) Refactorización y estandarización de los pipelines existentes

A la par, procederemos a realizar una exhaustiva identificación de los pipelines existentes con el objetivo de corregir deficiencias, homogenizar estructuras y mejorar prácticas que son determinantes para la continuidad del funcionamiento. Esta refactorización atenderá a la actualidad de los tiempos de ejecución y a los puntos donde se aprecien cuellos de botella, para volver a plantear etapas y mejorar el rendimiento. Junto a la mejora de la eficiencia, se robustecerá el proceso con mecanismos para control de la seguridad, incorporando herramientas de análisis estático y dinámico del código, así como controles que garanticen una gestión correcta de las credenciales, accesos y dependencias. La estandarización será un principio transversal: se desea que los pipelines sean homogéneos en la organización, el mantenimiento sea sencillo y sean fácilmente replicables en los nuevos servicios, sin unos elevados costes de adaptación. Para conseguirlo, se procederá a configurar adecuadamente

Jenkins, Docker y GitLab, lo que permitirá que las prácticas acordadas sean sostenidas en el tiempo.

Dando continuidad a un modelo DevSecOps, se incorporará la seguridad desde el comienzo del ciclo, siendo un paso más. En la ejecución se añadirán herramientas que identifiquen vulnerabilidades en el código y en los contenedores evitando que cualquier artefacto continúe su camino en el pipeline si no tiene un mínimo nivel de confianza. Se llevará a cabo en herramientas que protejan la información sensible y se establecerán reglas de acceso que determinen quién puede realizar despliegues o alteraciones en las configuraciones, mitigar la posibilidad de que cualquier persona ajena haga cambios. Durante todo el proceso, la investigación contempla un sistema de medición de la eficacia real del modelo DevOps que se adopte. Se recabarán indicadores que recojan el comportamiento del pipeline y métricas que recojan el estado de la seguridad a través de la recurrencia de las vulnerabilidades detectadas y el tiempo que se requiere para resolverlas; todo ello acompañado de la opinión del equipo y de los usuarios acerca de la calidad de las entregas y el flujo del desarrollo, ya que la aceptación de la solución y su impacto en el trabajo cotidiano son variables esenciales para medir el éxito de la implementación.

c) Configuración de herramientas DevOps bajo el enfoque DevSecOps

En el momento en el cual se establezcan todas las herramientas que soporten el ciclo DevOps ya se hará desde un punto de vista completamente alineado a DevSecOps, de manera que los elementos de seguridad acompañen a cada actividad del ciclo de vida del software. Siguiendo con esta idea, la seguridad del código o de los entornos no se ofrece como una revisión tardía, sino como algo que se irá activando desde las primeras etapas de integración continua y que irá tirando hasta el último despliegue de la aplicación ya acabada. Para ello se prevé la integración de soluciones que permiten un reconocimiento anticipado de riesgos, por ejemplo, herramientas de análisis que analicen vulnerabilidades del código fuente o también de los contenedores antes de que puedan ser autorizados y promovidos hacia entornos superiores. También la gestión de credenciales será trasladada a sistemas que asesoren la gestión de secretos para evitar que información sensible quede expuesta o que circule a lo largo del pipeline sin protección. Este marco se verá acompañado con políticas de accesos que delimiten perfectamente quiénes son los miembros del equipo autorizados para poder modificar configuraciones o poder ejecutar despliegues, reduciendo así las

posibilidades de que alguien pueda realizar intervenciones no esperadas o poder realizar alteraciones maliciosas.

En paralelo y de forma complementaria, en la medida en que sea posible la implementación de DevOps irá acompañada de un sistema de evaluación continua con la finalidad de determinar si la adopción de DevOps se puede evidenciar como una mejora estable en el trabajo. Para ello, se recogerá una serie de datos que pondrán de manifiesto el rendimiento del pipeline: el tiempo que tardan los trabajos de integración, la velocidad a la que se despliegan nuevas versiones, la frecuencia con la que se encuentran errores en las pruebas, etc. Se controlará también la evaluación de los resultados ofrecidos por las herramientas de seguridad (cuántas vulnerabilidades han sido encontradas, corregidas o prevenidas, el público cómo de profundo realizan las pruebas y cuáles son el código y los artefactos utilizadas). A este seguimiento técnico, se añadirá la percepción del equipo de trabajo y de los usuarios finales, pues la sensación de fluidez, fiabilidad y mejora de la calidad de las entregas es un indicador clave para determinar que los cambios llegaron a tener un impacto positivo en la experiencia general del desarrollo y en la estabilidad del servicio.

3.2.3 Etapas y Metodología

3.2.3.1. Planificación y Diseño del Pipeline de DevOps

La elaboración de un pipeline en un entorno DevOps necesita de un trabajo coordinado entre la parte que realiza el desarrollo y la que opera los sistemas, ya que su propósito más importante es construir un flujo que sea común y funcionante. Desde la parte teórica, esto puede ser visto como el hilo conductor para eliminar las intervenciones manuales innecesarias y evitar las etapas en las cuales el avance del software suele detenerse. La idea principal es que las etapas de programación, verificación o despliegue sigan produciendo de forma coherente y sin interrupciones, logrando entregas más veloces y con un margen menor de error. Tanto la entrega continua como la integración continua son vistas como dos prácticas centrales dentro del este esquema, ya que permiten aplicar cambios de forma habitual y favorecer su paso a producción bajo un esquema seguro y predecible. La colaboración interdepartamental es, en ese sentido, uno de los dos ejes centrales que permite mejorar notablemente el ritmo y la calidad del desarrollo de la práctica en la organización.

3.2.3.2. Implementación del Pipeline de Integración y Entrega Continua (CI/CD)

La implementación del pipeline de integración y entrega continua se fundamenta en esta misma premisa. De acuerdo con la idea de CI/CD, cada cambio que se incorpora al repositorio ha de ser revisado mediante pruebas automáticas que confirmen la estabilidad del sistema. La literatura clásica de DevOps sostiene que la automatización de ciclo total, desde la construcción a la entrega, no solo facilita la entrega pero también limita el riesgo de introducir fallos que comprometan el servicio. Revisión continua del código, seguimiento del estado de los entornos de desarrollo y feedback inmediato al equipo son principios que, en teoría, refuerzan la adecuada capacidad del sistema para adaptarse a los cambios sin interrupciones en la experiencia del usuario final.

3.2.3.3. Refactorización y Optimización de los Pipelines Existentes

En lo que respecta a los pipelines que ya se encuentran establecidas en un proyecto, la teoría viene a indicar exactamente lo contrario, afirmando que los pipelines deben ser revisados y mejorados de forma continua, lo que está alineado con la filosofía de mejora continua, aunque aquí aplicada al flujo de integración y despliegue. La refactorización de un pipeline, es decir, sus modificaciones estructurales, ha de ser repensado, en tanto que eliminar los pasos innecesarios, reducir el tiempo de ejecución de las pruebas realizadas y mejorar los mecanismos de seguridad que intervienen en el proceso de integración-despliegue. Esta optimización de los pipelines no sólo persigue que el trabajo diario del equipo sea más eficiente, sino que a su vez el pipeline debe ser capaz de dar respuesta a los estándares que sean los de actualidad y deberá ser suficientemente flexible para amoldarse a requisitos futuros. Adicionalmente, la integración de pruebas de seguridad dentro del flujo habitual del pipeline implica que los riesgos asociados a vulnerabilidades puedan ser detectados y corregidos antes de los entornos más críticos.

3.2.3.4. Integración de Seguridad con el Enfoque DevSecOps

Cuando se aplica el enfoque DevSecOps, la teoría va un paso más y afirma que la seguridad debe atravesar todas las etapas del desarrollo y no ser una revisión tardía. Esta idea defiende que los controles de seguridad deben ser introducidos dentro de la automatización del

pipeline, siendo analizados de manera continua, independientemente de parar la velocidad de entrega. Desde las últimas aportaciones de la literatura, DevSecOps no busca simplemente aumentar una capa de complejidad, sino buscar el establecimiento de una cultura de propiedad compartida, donde la seguridad está compartida. En este sentido, los equipos integran herramientas y prácticas que son oportunidades para detectar riesgos a través de los cuales se incrementa la resiliencia del software y aumentan las capacidades del proyecto para enfrentar amenazas sin disminución de la agilidad, aumentando las capacidades del proyecto para asumir amenazas sin sacrificar la agilidad.

Tabla 11

Herramientas empleadas en la metodología DevOps y su alineación teórica

Herramienta	Propósito	Relación teórica con DevOps
Jenkins	Automatización de integración continua y despliegue.	Es la herramienta clave para implementar CI/CD, facilitando la integración continua y mejorando la agilidad.
GitLab	Gestión de código fuente y repositorios.	Permite el control de versiones y facilita la colaboración continua entre desarrolladores y operaciones.
Docker	Contenedores para crear entornos consistentes.	Teóricamente, facilita la portabilidad y consistencia entre entornos de desarrollo, pruebas y producción.
qTest	Gestión de pruebas automatizadas.	Alineada con la calidad continua, asegura que el software se someta a pruebas automatizadas en cada etapa.
OWASP ZAP	Escaneo de vulnerabilidades.	Teóricamente, es esencial para la implementación de DevSecOps, integrando la seguridad en todas las fases.
SonarQube	Análisis de calidad de código.	Apoya la mejora continua, ayudando a mantener un código limpio y libre de vulnerabilidades.
Herramienta	Propósito	Relación teórica con DevOps
Ansible Vault	Gestión de secretos y credenciales.	Esencial para mantener la seguridad en el proceso de CI/CD, gestionando de manera segura credenciales y secretos.

Nota. Elaboración propia

3.2.4 Fundamentos Utilizados

3.2.4.1. Antecedentes

En el trabajo presentado por López (2022), se trabajó únicamente en adaptar las prácticas de DevOps al campo particular del IoT mediante el marco denominado "IoT DevOps". Los deseos de este autor estaban dirigidos a mejorar los ciclos de feedback entre el desarrollo y las operaciones, algo que es esencial en sistemas que necesitaban de la fiabilidad total debido a la interacción con cosas/dispositivos físicos. Con la finalidad de llevar a cabo dicha práctica, utilizando la metodología de la Ciencia del Diseño se desarrolló y verificó un modelo formal que permitía tener un flujo de información operativa para ayudar a tomar decisiones durante la fase de desarrollo. Algunos de los trabajos más relevantes de este estudio fueron la práctica “Powerful and Continuous Feedback of IoT Systems State”, descrita en SPEM 2.0, y la técnica denominada “Monitoring as Code”, que ayudaba a realizar el seguimiento del firmware de los dispositivos en producción. Los estudios de caso demostraron que dicha práctica ayudaba a recortar el tiempo de detección de las fallas y permitir una actualización continua, lo que llevaba a unas mejoras continuas en el software y a una mejora de la fiabilidad de las implementaciones de los sistemas IoT.

En contraste, el estudio llevado a cabo en la Universidad Técnica de Manabí fue de carácter más institucional, centrado en fortalecer la seguridad del ciclo de desarrollo en el departamento de TI. El diagnóstico realizó evidencias de la carencia de las prácticas de protección frente a las vulnerabilidades existentes en el ciclo de integración y en el ciclo de despliegue continuo. Con apoyo en un diseño no experimental amparado en la aplicación de encuestas al personal técnico se determinó que la mayor parte de los trabajadores tenían un escaso conocimiento sobre las herramientas propias de DevOps; no obstante, esto se determinó que había un interés considerable por incluir el enfoque. En base a estos resultados se determinó que una DevOps orientado por la seguridad permitiría acortar tiempos de desarrollo, mejorar la calidad del servicio tecnológico y garantizar respuestas más rápidas a

las necesidades de los usuarios, a la vez que aseguraría un control superior de los procesos operativos.

Por último, un estudio enfocado en comprobar la adopción de DevOps en la operación de servicios TI, evaluando sus ventajas y desventajas al ser puesto en práctica durante la operación en organizaciones, a partir de una metodología PRISMA para la revisión sistemática, identifica algunos factores técnicos y culturales que afectan su adopción. Las evidencias encontradas apuntan que DevOps permite aumentar la calidad y la performance de servicios debido a la automatización, entrega continua y una mejor interacción entre grupos de trabajo. No obstante, la revisión también pone de manifiesto importantes limitaciones como la falta de personal formado en esta práctica o la falta de claridad de cómo llevarlo a cabo. Pese a ello, la conclusión general es que, acompañado de marcos de gobernanza adecuados, DevOps puede cambiar significativamente la gestión de servicios TI, pero aún se necesita realizar estudios más amplios para entender su desarrollo en la práctica.

3.2.4.2. Definiciones

Dentro del ámbito de la Ingeniería de Sistemas, las metodologías de desarrollo funcionan como guías y ordenadoras del proceso de generación y evolución de las plataformas de software. Éste es el sentido que le da Marulanda (2022), quien añade que estos métodos y marcos de las metodologías de desarrollo tienen como objetivo el consultar de manera guiada y estructurada todo el proceso de la planificación inicial a la puesta en producción y el mantenimiento de sistemas de software, con el objetivo de que los resultados sean consistentes y satisfagan de manera coherente y acorde a los objetivos de la institución. Para los fines de este trabajo, resultan especialmente pertinentes tres metodologías: DevOps, Scrum e ITIL. En el caso de DevOps, se trata de una metodología, pero a la vez cultural y técnica que busca unir de una manera continua y permanente a quienes desarrollan y a quienes tienen que gestionar los sistemas. Cruz et al. (2023) añaden que el objetivo que persigue es el de acelerar la entrega de valor, y para ello, se basa en la automatización, integración continua y monitorización del sistema. En nuestro caso, su utilización ha permitido reducir la carga de las tareas manuales de volcado de la información a la plataforma de gestión y establecer un entorno de despliegue con un enfoque más controlado y seguro.

Por otro lado, el Scrum se encuentra dentro de las metodologías ágiles y está basado en un marco empírico que favorece la entrega gradual de funcionalidades a través de ciclos cortos, a los que se les llama sprints. De acuerdo a lo que dice Iñiguez (2024), este enfoque facilita la colaboración continua y la capacidad de adaptarse a requisitos cambiantes. Si bien la planificación del proyecto se llevó a cabo a partir de una lógica más tradicional, los principios de Scrum dieron cierta flexibilidad respecto a cómo organizar y llevar a cabo actividades.

Por último, ITIL 4 nos ofrece un marco con la finalidad de gestionar servicios de tecnología de información. Tal y como afirman Santosa y Mulyana (2023), dicho marco propone prácticas orientadas a la generación de valor centradas en la integración de los procesos técnicos ligados a las necesidades del negocio. En el desarrollo del presente trabajo, sus lineamientos resultaron útiles como puntos de referencia para ordenar los documentos, como definición de estándares y como guía para la capacitación de los usuarios que habrían de participar.

3.2.4.3. Ciclo DevOps

Se entiende el ciclo de vida de DevOps como un proceso que acontece de manera cíclica y en constante transformación, y que está formado por una secuencia de etapas que van desde la planificación, escritura del código, construcción, validación, liberación, despliegue, operación y supervisión de software. Angulo-Angulo (2023) refiere que esta estructura cíclica puede contribuir a mantener un flujo continuo de entrega, posibilitar la disminución de errores y fomentar la colaboración entre diferentes grupos de trabajo.

Estudios recientes, tales como el realizado por Valencia et al. (2024) también amplían dicho enfoque indicando que DevOps es más que un simple procedimiento, se trata, en realidad, de una transformación cultural dentro de una organización. Este enfoque también posibilita evaluar el desempeño mediante métricas ampliamente aceptadas como las métricas DORA. Estas métricas analizan la frecuencia de los despliegues, el tiempo del cambio, el tiempo del servicio tras incidentes y los cambios que provocan fallos. Esto fue de suma importancia en nuestro trabajo ya que sirvió para evidenciar avances cuantificables a nivel tanto de los tiempos de liberación como de la estabilidad de los sistemas en funcionamiento.

- Planificar entregas a través de lineamientos claros de branching y despliegue.
- Codificar con estándares de calidad y revisiones automatizadas mediante SonarQube y Fortify.
- Construir pipelines CI/CD en Jenkins y GitLab para 17 componentes principales.
- Probar mediante integración de herramientas SCA y SAST (Rapid7, Fortify).
- Liberar versiones de forma controlada con políticas en JFrog y GitLab.
- Desplegar aplicaciones en entornos cloud y on premise con AWS, ECS y Ansible.
- Operar mediante monitoreo continuo y soporte post-producción.
- Monitorear vulnerabilidades y métricas de rendimiento para retroalimentar el ciclo.

3.2.4.4. Servicios en la nube

La computación en la nube constituye otro de los pilares del presente proyecto. Ortiz et al. (2024) definen el cloud computing como un modelo de provisión de servicios de TI bajo demanda, escalables y medibles, accesibles de manera ubicua a través de la red. En este marco, se reconocen tres modelos de servicio:

- Infraestructura como Servicio (IaaS): el cliente gestiona aplicaciones y sistemas operativos sobre recursos virtualizados.
- Plataforma como Servicio (PaaS): el proveedor administra la infraestructura y el cliente se concentra en la gestión de aplicaciones y datos.
- Software como Servicio (SaaS): el proveedor ofrece aplicaciones completas listas para su consumo, garantizando disponibilidad y escalabilidad (Alejandro et al., 2023).

En el proyecto, el modelo PaaS se aplicó en la integración de pipelines y automatización de despliegues, mientras que el modelo SaaS estuvo presente en el consumo de herramientas como SonarQube o Fortify, utilizadas como servicios en la nube. La combinación de ambos enfoques otorgó flexibilidad y redujo los costos de mantenimiento.

3.2.4.5. Fundamentos técnicos del proyecto

Los fundamentos técnicos que respaldaron la solución se relacionan con las principales herramientas y lenguajes empleados. En esta sección se presentan en un formato de definición, autor y aplicación.

- Jenkins (CI/CD): Herramienta de automatización de código abierto que permite construir, probar y desplegar software de manera continua. En este proyecto se utilizó como núcleo para integrar pipelines de CI/CD en entornos cloud y on premise (Amgothu, 2024).
- GitLab y JFrog: Plataformas de control de versiones y gestión de artefactos. Según Karpoja (2024), GitLab centraliza la colaboración en código, mientras que JFrog optimiza la gestión de binarios. Se aplicaron en la gestión de repositorios y versionamiento de contenedores.
- SonarQube, Fortify : Herramientas de seguridad de código. Como destacan Makani y Jangampeta (2024), el uso de SAST, SCA y DAST es fundamental en el enfoque DevSecOps. Estas herramientas se integraron a los pipelines para detectar vulnerabilidades de forma temprana.
- Ansible y AWS CDK (IaC): Lenguajes de automatización declarativa. De acuerdo con Huedo (2021), Ansible permite la provisión de servidores de manera repetible. En este proyecto se usaron para estandarizar configuraciones y desplegar infraestructura bajo el paradigma Infrastructure as Code (Makani y Jangampeta, 2024).
- Docker y Orquestadores de Contenedores Docker: Tecnologías de contenedores y orquestación. Como sostienen Burns y Oppenheimer (2020), los contenedores permiten empaquetar aplicaciones de manera portable, y Kubernetes facilita su escalamiento y despliegue automatizado. En este proyecto se emplearon para crear entornos homogéneos y confiables (Makani y Jangampeta, 2024).

3.2.4.6. Lenguajes y frameworks

Los lenguajes y frameworks que se utilizaron para el desarrollo del proyecto fueron la base técnica que sustentaron los procesos de integración y despliegue continuos. Uno de los

elementos que resultó ser más importante fue YAML para las recetas de Ansible, YAML es un simple lenguaje de marcado que se ha convertido en el lenguaje de marcado de referencia junto con la sintaxis Groovy para los archivos Jenkinsfile en los pipelines de CI/CD. estas tecnologías facilitaron la definición de tareas, etapas y condiciones tanto en Jenkins como en GitLab, de modo que se volvió posible definir flujos de trabajo más ordenados y comprensibles. A ello se le sumó el uso de contenedores Docker, que hicieron posible el encapsulamiento de las aplicaciones más unas dependencias, logrando que los servicios se ejecutaran de la misma forma independientemente del entorno en el que se despliegan con lo que esto representó en términos de homogeneidad de entornos de desarrollo, pruebas y producción.

A ello se agregó la implementación de un sistema de orquestación de contenedores mediante Docker Compose para la arquitectura on-premise, así como la adopción de AWS Elastic Container Service (ECS) con Fargate para los entornos en la nube de AWS.

En el entorno on-premise, Docker Compose permitió la definición declarativa de servicios, redes y volúmenes, facilitando la estandarización de los despliegues, la replicabilidad de entornos y la simplificación de la gestión del ciclo de vida de las aplicaciones que usando Docker. Esto contribuyó a reducir errores de configuración, mejorar la trazabilidad de cambios y agilizar los procesos de puesta en producción y mantenimiento.

Por su parte, la implementación de AWS ECS Fargate eliminó la necesidad de administrar la infraestructura subyacente, delegando a AWS la gestión de servidores, capacidad de cómputo y escalado.

Gracias a estas capacidades, los servicios pudieron escalar de forma dinámica en función de la demanda, garantizando una respuesta oportuna ante picos de carga y optimizando el uso de recursos. Asimismo, se fortaleció la resiliencia del entorno de ejecución mediante balanceo de carga, lo que incrementó la tolerancia a errores y robusteció la estabilidad y continuidad operativa de la plataforma tecnológica en su conjunto.

3.2.5 Implementación de las áreas, procesos, sistemas y buenas prácticas

La ejecución del proyecto se inició con una fase de análisis cuyo objetivo fue comprender en profundidad la situación real del cliente y de las herramientas que ya existían en los procesos del propio cliente. Este primer contacto permitió identificar limitaciones, riesgos y oportunidades de

optimización, así como generar una visión común del punto de partida. Para ello se realizaron múltiples reuniones con los actores implicados donde se captaron, además de los requerimientos técnicos, las necesidades operativas y estratégicas que guiarían el posterior diseño de la solución. En esta fase se definieron también claramente las áreas que participarían en el proyecto y los procesos que serían afectados por la implementación.

Un aspecto clave del análisis fue la identificación de los perfiles que formarían el grupo de trabajo. La asignación correcta de los roles y previamente determinados las responsabilidades facilitaron la coordinación entre los diferentes grupos, asegurando de este modo que cada especialista aportase desde su área de conocimiento. Esta delimitación inicial de funciones facilitó organizar eficazmente las actividades y mantener un flujo de comunicación constante en todas las fases del proyecto.

Tabla 12
Roles de los participantes en el proyecto

Rol	Descripción
Project Manager	Encargado de la gestión global del proyecto, toma de decisiones y resolución de conflictos.
Líder Técnico	Líder del equipo de desarrollo y QA, responsable de coordinar las actividades técnicas y de asegurar la calidad del trabajo realizado.
Usuario de Procesos	Encargado de proporcionar información detallada sobre los procesos operativos y las funcionalidades necesarias para el sistema.
Desarrollador SaaS	Encargado de la implementación y configuración de las funcionalidades de la solución SaaS.
Analista QA	Responsable de realizar las pruebas funcionales y validar que los desarrollos cumplen con los requisitos establecidos.

Nota. Elaboración propia

Reuniones clave

Se realizaron reuniones de trabajo entre los participantes del proyecto y los stakeholders con el fin de alinear los objetivos, definir los entregables y establecer el alcance de cada fase. Los objetivos y los participantes de las reuniones más importantes del proyecto se resumen en la siguiente tabla:

Tabla 13

Objetivo y rol de los participantes en las reuniones

Nro.	Objetivo de la reunión	Rol de los Participantes
1	Definir las características y consideraciones en cada regla/promoción a implementar en la parametrización de la solución SaaS.	Project Manager, Líder técnico, Usuario de Procesos, Product Owner.

Nota. Elaboración propia

3.2.5.2 Diseño

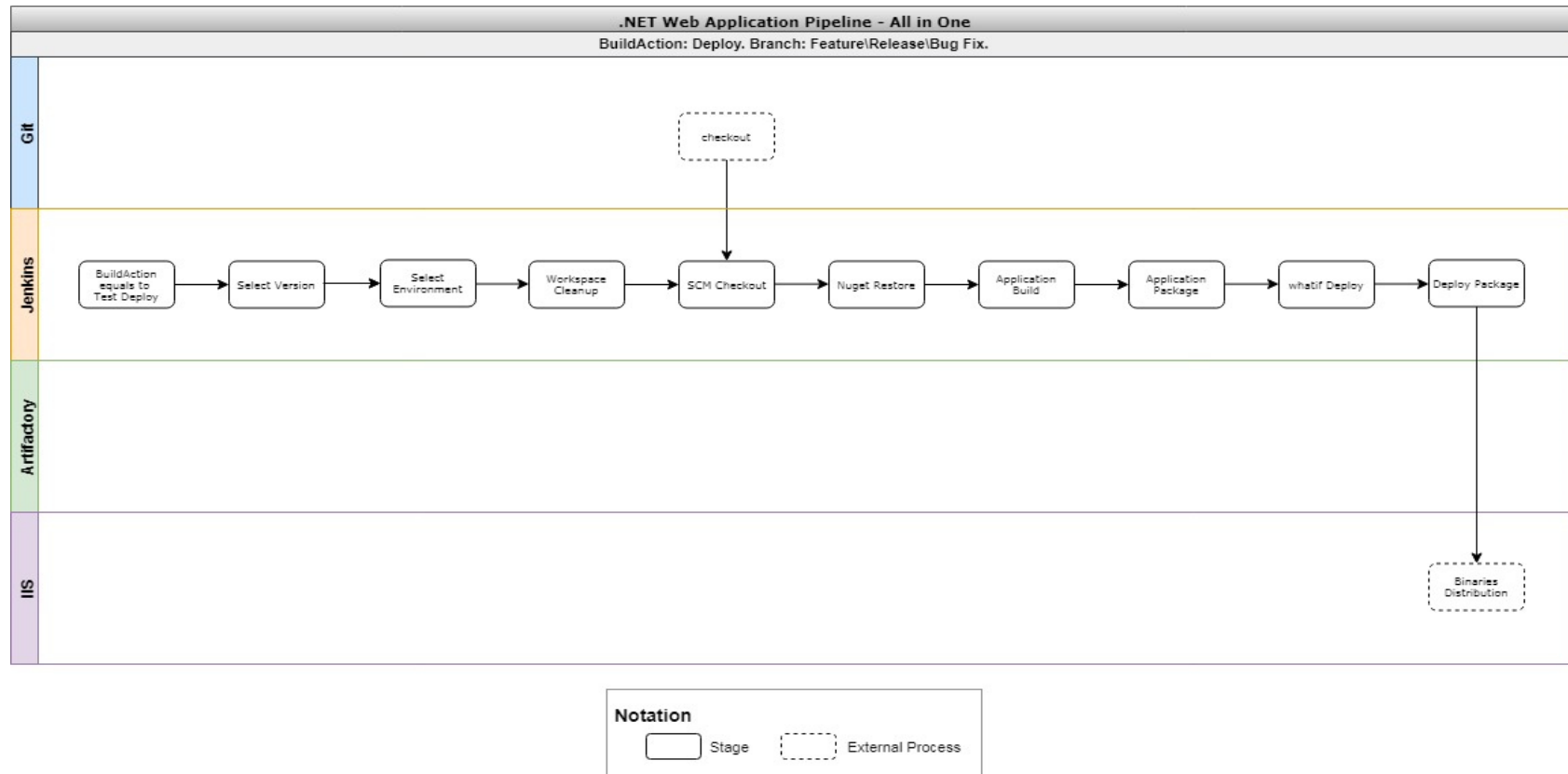
En la fase de diseño, se definieron los componentes técnicos y arquitectónicos de la solución, así como los procesos y flujos de trabajo necesarios para asegurar la correcta integración de los sistemas. Se desarrollaron diagramas clave que describen las interacciones entre la solución SaaS, y otros componentes de la infraestructura.

Diagrama de Arquitectura

El diagrama de arquitectura, como se muestra en la Figura 9. Gráfico de Docker Host, ilustra la relación entre los diferentes sistemas y plataformas involucradas, especificando cómo la información fluye entre los componentes. Además, en Figura 5. Flujo de despliegue se especifica como un nuevo artefacto se despliega en la Arquitectura On-premise

Figura 5

Flujo de despliegue para aplicación web en .NET utilizando Jenkins, Git, Artifactory.



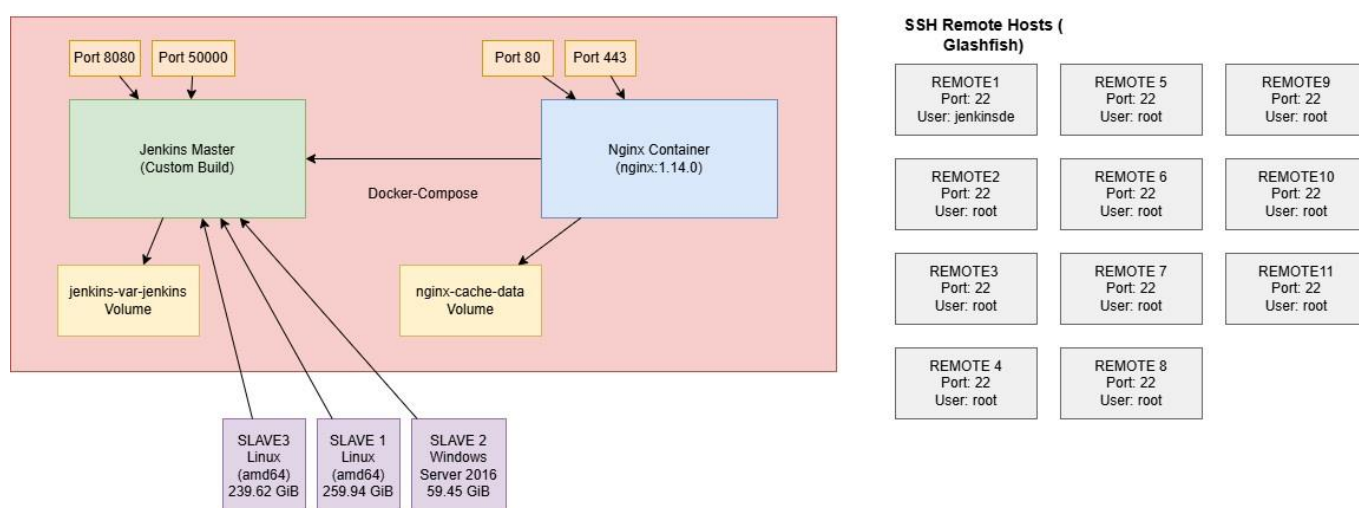
Nota. La figura representa el propósito del pipeline, herramientas implicadas (Git, Jenkins, Artifactory, IIS) y las etapas clave (checkout, build, package, deploy, distribución). Elaboración propia

3.2.5.3 Implementación

Este plan es una hoja de ruta estructurada a lo largo de todo el año, organizada en epics, que son grandes áreas de trabajo que permiten gestionar y coordinar los distintos aspectos de la implementación de DevOps. Cada epic tiene un programa, con objetivos y resultados clave que se distribuyen por mes, de forma que garantice la eficiente ejecución de las tareas. A continuación, se expone la descripción del plan de trabajo y a continuación se presentan tablas con los objetivos, los resultados, las dependencias y las fechas clave. El primer epic indica la estabilización de Jenkins, en el que el objetivo fundamental sea la migración de Java 7 a Java 17, la creación de un entorno de staging para las implementaciones de Jenkins. La migración es un paso esencial para mantener actualizada la infraestructura tecnológica y asegurar que las integraciones y los despliegues sean estables. La migración de Java a una versión más nueva ofrece más seguridad, más rendimiento y más compatibilidad con nuevas herramientas. Este aspecto tendrá que concluirse durante los primeros meses del año: enero y febrero.

Figura 6

Arquitectura de Jenkins Java 17



Nota. La imagen representa una arquitectura Jenkins personalizada con NGNIX como proxy inverso, integrando múltiples nodos esclavos Linux y Windows, así como hosts remotos configurados para despliegues mediante SSH. Elaboración propia

Tabla 14*Jenkins Stabilization PRD*

Mes	1	2	3	4	5	6	7	8	9	10	11	12
Actividades	X	X										
Objetivo	Migrar de Java 7 a Java 17 y crear Staging											

Nota. Elaboración propia

Uno de los primeros grupos de trabajo se corresponde con el que suele denominarse DevOps Charter, cuyo objetivo es fijar las bases de la adopción de DevOps en la organización. Este documento actúa como un documento de orientación que recoge los principios, las prácticas y las guías bajo los que se defina el pasaje hacia un modo colaborativo, automatizado, etc. Este epic se desembaraza desde enero a abril, y a lo largo de este episodio se busca que todo el equipo de TI entienda y se apropie de la estrategia que sustentará la mejora continua de la infraestructura tecnológica.

Tabla 15*DevOps Charter*

Mes	1	2	3	4	5	6	7	8	9	10	11	12
Actividades	X	X	X	X								
Objetivo	Establecer la fundación para la adopción de DevOps en Empresa aseguradora											
Resultados	1. Crear y revisar el DevOps Charter con el											
Clave	equipo de TI para el 31 de octubre.											
Dependencias	Ninguna											

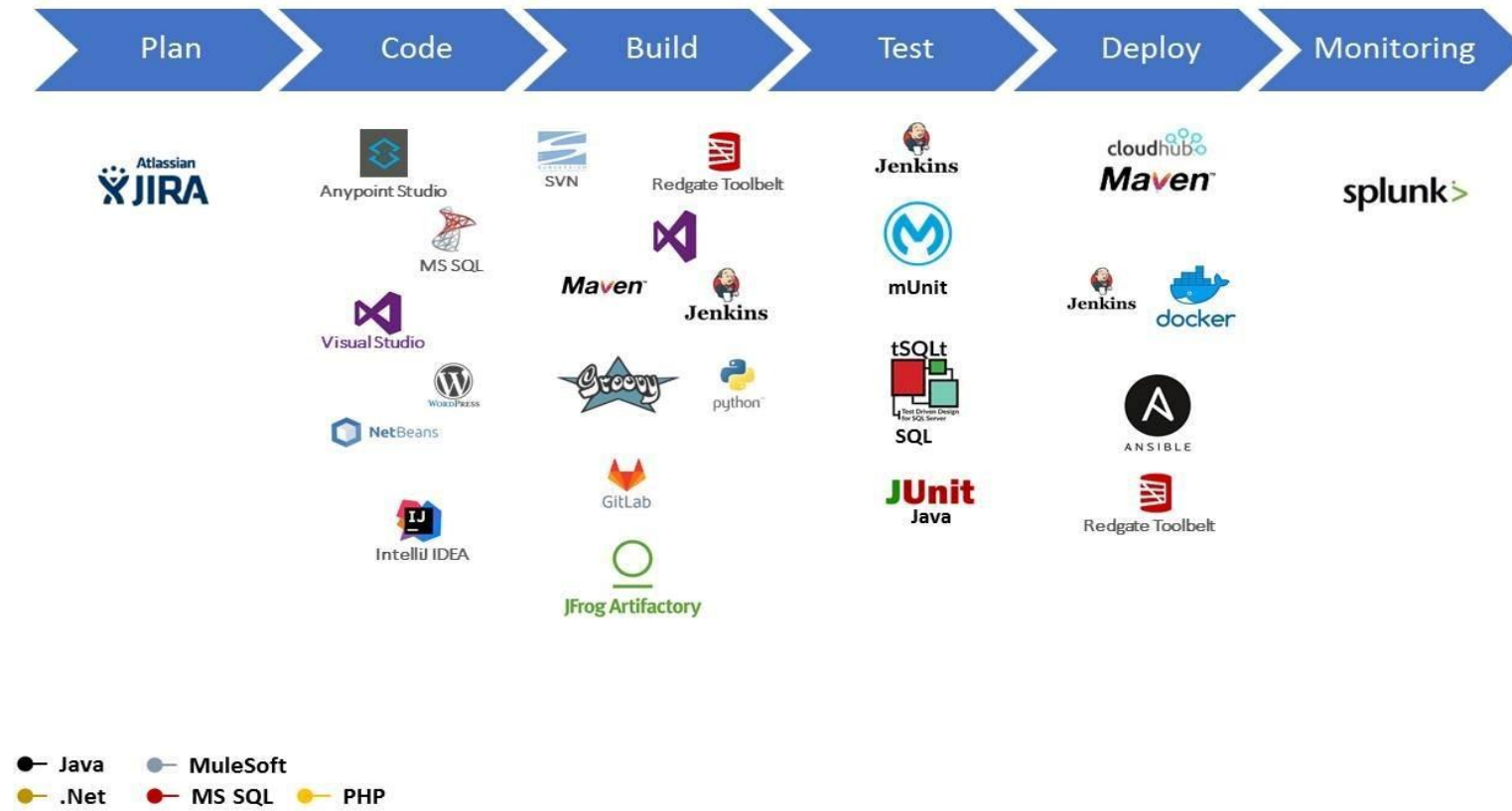
Nota. Elaboración propia

Otro epic muy importante es DevOps Metrics, que estudia la manera de establecer indicadores que nos ayuden a evaluar objetivamente el impacto y el nivel de madurez de las prácticas adoptadas. Este epic se desarrolla desde marzo hasta diciembre y persigue como

objetivo final hacer una selección de métricas que sean utilizables, que puedan ser recogidas y que sean consistentes con la realidad operativa, buscando de esta manera que la organización pueda medir su avance y demostrar los beneficios de la adopción de DevOps de una manera medible.

Figura 7

Herramientas empleadas



Nota. Esta figura muestra el flujo de herramientas DevOps en cada etapa del ciclo de vida del desarrollo: planificación, codificación, compilación, pruebas, despliegue y monitoreo, destacando herramientas como Jenkins, Docker, GitLab y Splunk. Elaboración propia

Las tecnologías DevOps proporcionan un conjunto estructurado de herramientas que acompañan todas las etapas del proyecto, desde la planificación inicial hasta la implementación final. En una primera etapa, se incorporaron repositorios de código con control de versiones, los cuales permitieron gestionar de manera ordenada y segura los cambios realizados durante el proceso de migración, el cual se desarrolló a través de múltiples iteraciones sucesivas.

Antes de la centralización del código, la gestión era fragmentada y altamente compleja: los cambios se realizaban de forma manual, con riesgo de inconsistencias, pérdida de versiones y dificultad para identificar errores o retroceder a estados estables del sistema. Esta situación incrementaba el tiempo de respuesta, la dependencia del personal clave y la probabilidad de fallos durante la implementación.

La adopción de repositorios centralizados permitió reducir significativamente dicha complejidad, facilitando el trabajo colaborativo del equipo, mejorando la trazabilidad de los cambios y asegurando la integridad del código. Como resultado, se logró una implementación más controlada, predecible y alineada con las buenas prácticas DevOps, optimizando tanto el tiempo de ejecución del proyecto como la calidad del resultado final.

Tabla 16
DevOps Metrics

Mes	1	2	3	4	5	6	7	8	9	10	11	12
Actividades			X	X								
Objetivo	Establecer métricas de DevOps para que su efectividad y valor puedan ser cuantificados.											

Mes	1	2	3	4	5	6	7	8	9	10	11	12
		2.										
		Identificar										
	1. Seleccionar	métricas		4. Socializar								
	al menos dos	adicionales	3. Establecer	el panel de								
	métricas	a través de	líneas base	métricas de								
Resultados	prácticas y	un ejercicio	para las	ServiceNow								
Clave	fáciles de	de mapeo	métricas	con los								
	implementar	de valor	seleccionadas	stakeholders								
	para el 31 de	(Value	para el 4T.	de TI y los								
	octubre.	Stream		equipos de								
		Mapping)		programa.								
		para el 4T.										
Dependencias	CTS (Sys											
	Admin)											

Nota. Elaboración propia

El epic de Fortify busca integrar herramientas de seguridad como *Fortify* para analizar el código durante el ciclo de desarrollo y asegurarse de que no existan vulnerabilidades en el software. Esto es clave en cualquier proceso DevOps, ya que garantiza que la seguridad se integre desde el principio en el ciclo de vida del software, no solo al final. Este epic comienza en abril y termina en mayo.

Tabla 17

Fortify

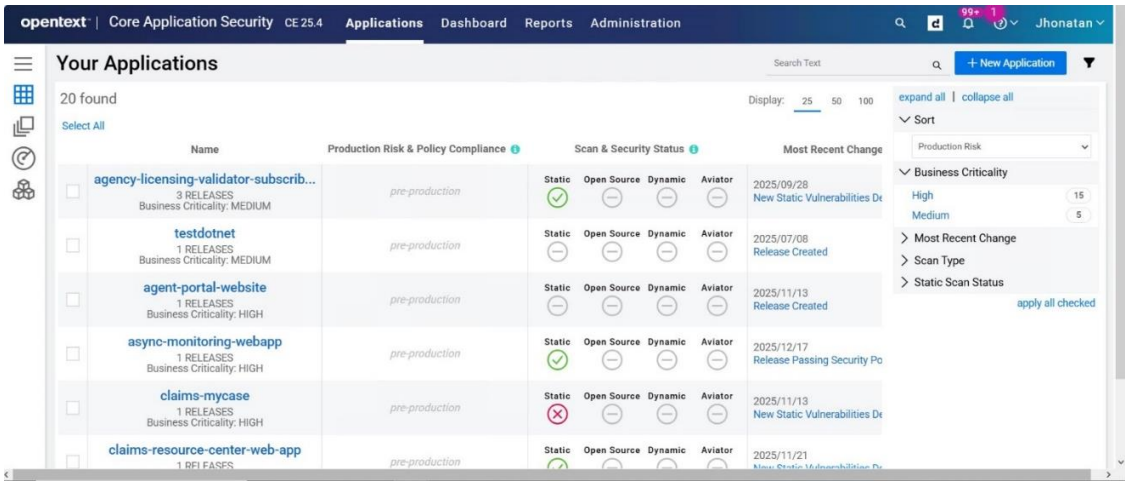
Mes	1	2	3	4	5	6	7	8	9	10	11	12
Actividades					X	X						
Dependencias	CTS (Sys Admin, Sys Engineer)											

Nota. Elaboración propia

Conjuntamente, el epic asignado como Fortify tiene como objetivo la incorporación de herramientas de análisis de seguridad (Fortify, por ejemplo) dentro de la misma, lo que va a permitir analizar el código desde unas primeras etapas y poder así obtener la información relativa a vulnerabilidades desde las primeras fases del ciclo, y la premisa de asegurar la existencia de una integración de manera transversal se ve así reforzada. Este epic tiene lugar entre abril y mayo.

Figura 8

Fortify Dashboard



Nota. La imagen muestra el panel de control de OpenText (Empresa cuyo Producto es Fortify) para la gestión de aplicaciones, escaneos de seguridad y cumplimiento de políticas, incluyendo estados de vulnerabilidad por cada aplicación desplegada. Tomada del Portal Interno de la empresa

Tabla 18*AWS Pipelines*

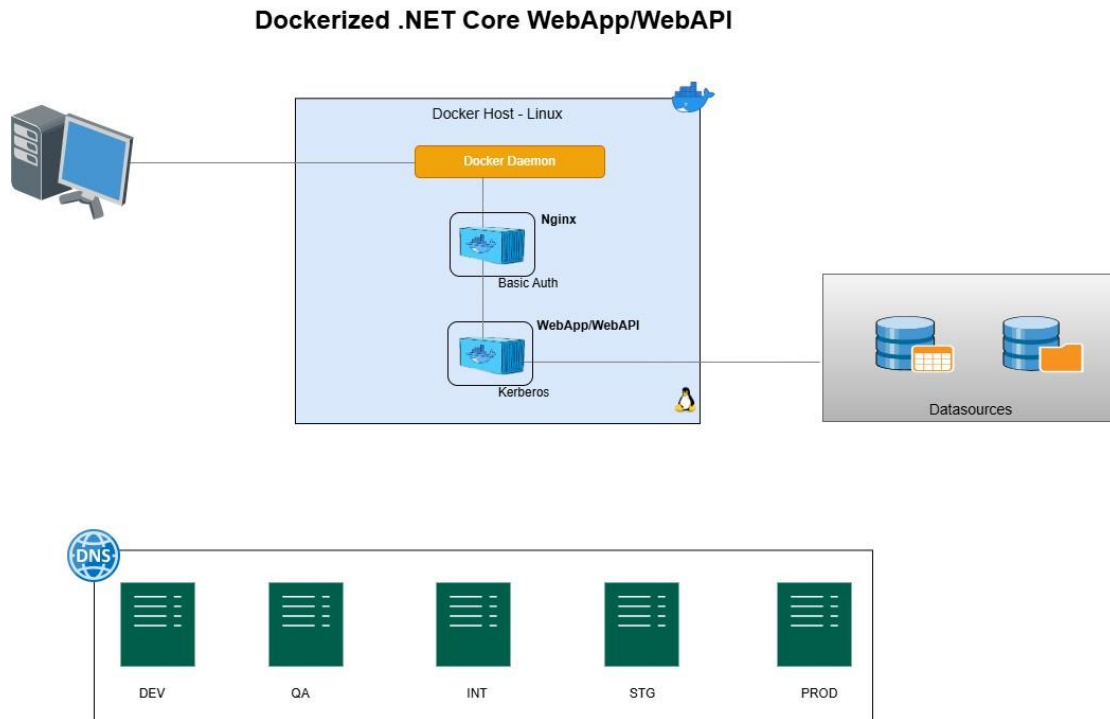
Mes	1	2	3	4	5	6	7	8	9	10	11	12
Actividades						X	X					
Objetivo	Migrar las aplicaciones de On-Prem a la nube											
Dependencias	CTS (Sys Admin)											

Nota. Elaboración propia

El epic AWS Pipelines contempla la migración de las aplicaciones desde el servidor local a entornos de nube mediante pipelines pasados por AWS, esto nos permitirá avanzar hacia infraestructuras más-flexibles-escalables cuyas características se alinean con las modernas prácticas de DevOps y Cloud Computing, asegurando que no pasemos más allá del verano con migraciones sin haberse terminado. Este epic está previsto para los meses de mayo a agosto.

Figura 9

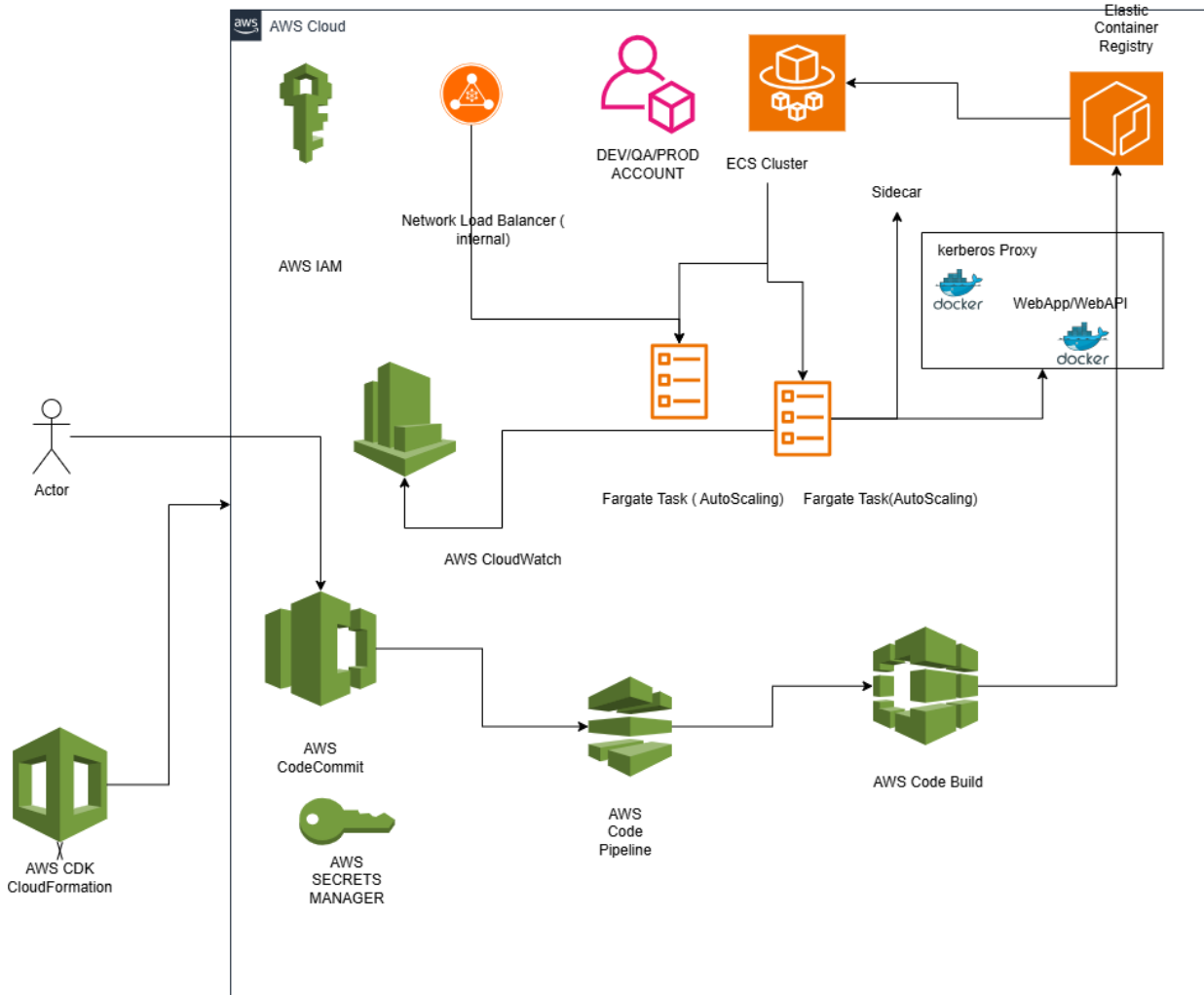
Gráfico de Docker Host



Nota. Este diagrama representa la implementación de una aplicación .NET Core en contenedores Docker dentro de un host Linux, autenticada con Nginx y Kerberos, y con acceso a distintas capas de entorno (DEV, QA, INT, STG, PROD) y expuestas por el servidor DNS de la empresa. Elaboración propia

Figura 10

Gráfico AWS



Nota. La imagen describe una arquitectura de CI/CD con AWS Fargate, ECR, CloudFormation y CodePipeline servicios Devops propios de la Cloud, destacando el uso de tareas autoscalables y balanceadores de carga para servicios Dockerizados además cada ambiente era una cuenta diferente (DEV, QA, PROD). Elaboración propia

Explicación sobre los componentes

Desarrollador (Actor): Envía código C# CDK a CodeCommit, lo que dispara una canalización automatizada que construye los contenedores.

CodeCommit: Almacena el código de infraestructura CDK y dispara CodePipeline vía webhook cuando los desarrolladores envían cambios a las ramas main/develop.

CodePipeline: Orquesta el despliegue a través de las cuentas DEV/INT/QA/STG/PROD usando canalizaciones separadas (lt-pipeline para entornos inferiores, prod-pipeline para producción).

CodeBuild: Ejecuta los comandos "icw synth" y "icw replication" usando una imagen de build de ECR personalizada para generar CloudFormation

ECR: Almacena la imagen de las aplicaciones y la imagen sidecar kerberos-proxy que las tareas Fargate extraen durante el despliegue.

VPC: Contiene subredes privadas en 2 Zonas de Disponibilidad (3 para PROD) donde se ejecutan las tareas Fargate sin IPs públicas, importada desde la infraestructura de plataforma existente.

Network Load Balancer: NLB interno con el grupo de destino existente que en ruta el tráfico a los contenedores de la aplicación en el puerto 80.

ECS Cluster: Gestiona los contenedores ejecutando tareas Fargate con la familia de definición de tareas a través de múltiples AZs.

Tareas Fargate: Ejecutan dos contenedores por tarea — la aplicación principal (puerto 80) y el sidecar kerberos-proxy (puerto 8080) que comparten almacenamiento efímero para la caché de autenticación.

Auto Scaling: Escala las tareas entre capacidad mínima/máxima (2–6 tareas) basado en umbrales de CPU/memoria del 70% con periodos de enfriamiento de 300 segundos; habilitado solo para QA/STG.

Secrets Manager: Provee credenciales en tiempo de ejecución mediante el ARN de secreto existente, accesible tanto por el contenedor de la aplicación como por el proxy.

CloudWatch Logs: Recopila logs de los grupos con retención indefinida.

Roles IAM: Usa el rol de tarea existente y que otorga a los contenedores acceso a Secrets Manager y a otros servicios de AWS.

La implementación del enfoque DevOps permitió integrar de manera efectiva las áreas de desarrollo y operaciones, transformando un proceso previamente fragmentado en un flujo de trabajo continuo, controlado y colaborativo. Antes de su adopción, las actividades se realizaban de forma aislada, con múltiples herramientas no integradas, alta dependencia del personal técnico y escasa trazabilidad de los cambios, lo que incrementaba la complejidad operativa y el riesgo durante la migración del sistema.

Entre las principales ventajas obtenidas destacan la reducción del tiempo de implementación, la mejora en la calidad del software desplegado, el aumento de la visibilidad del proceso y una mayor capacidad de respuesta ante incidencias. Asimismo, se fortaleció el trabajo colaborativo, ya que los equipos pudieron coordinar cambios de forma simultánea y segura, optimizando el uso de los recursos tecnológicos y humanos disponibles.

No obstante, la implementación también presentó desventajas y desafíos. Se identificó una curva de aprendizaje inicial asociada al uso de nuevas herramientas, la necesidad de capacitación técnica especializada y una mayor dependencia de la conectividad y la infraestructura tecnológica externa. Además, la adopción de DevOps exige disciplina operativa; una mala configuración o una gestión inadecuada de los repositorios puede generar errores de propagación a gran escala.

Desde la perspectiva externa, DevOps permitió una mejor alineación con proveedores tecnológicos, estándares de la industria y buenas prácticas internacionales, facilitando la interoperabilidad, la escalabilidad futura del sistema y el cumplimiento de requisitos de seguridad y continuidad operativa. En conjunto, la implementación hizo más visible y

comprensible el proceso de migración, transformándolo en un modelo replicable, controlable y sostenible en el tiempo.

Tabla 19

Database Pipeline Standards

Mes	1	2	3	4	5	6	7	8	9	10	11	12
Actividades					X	X	X	X				
Dependencias	CTS (DBA)											

Nota. Elaboración propia

De su lado, Pipeline Standards de Database se centra en especificar qué estándares son aplicables a base de datos en el marco de DevOps. Esta parte del proyecto (de mayo a septiembre) tiene como objetivo que los entornos de datos sigan el mismo protocolo que el resto del ciclo de desarrollo y permitir agilidad, control y homogeneidad.

Tabla 20

Pipeline Standards

Mes	1	2	3	4	5	6	7	8	9	10	11	12
Actividades	X	X				X						
Resultados Clave	1. Desarrollar los estándares de pipeline para la tecnología de la empresa aseguradora para el 31 de octubre. 2. Asegurar que los stakeholders de TI tengan acceso a la información sobre los estándares de pipeline .NET para el 31 de octubre.											
Dependencias	CTS (Sys Admin)											

Nota. Elaboración propia

Por último, el epic Pipeline Standards se propone especificar los requisitos que deban regir los pipelines tecnológicos propios de la empresa aseguradora (de enero a octubre), incluyendo el objetivo de que, antes de cerrar dicho mes todos los actores del área TI tengan disponibles los estándares y herramientas acordes, garantizando una adopción uniforme de las buenas prácticas en toda la organización.

Tabla 21

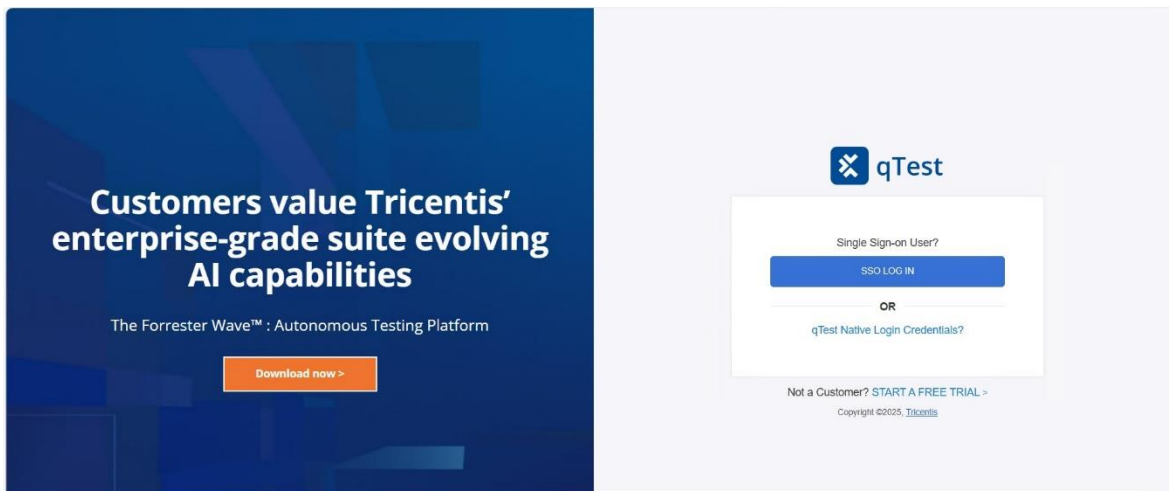
Jenkins & qTest Integration

Mes	1	2	3	4	5	6	7	8	9	10	11	12
Actividades								X				
Resultados Clave	1. Jenkins y qTest completamente integrados para julio.											
Dependencias	CTS (Sys Admin), WCU (QA Lead)											

Nota. Elaboración propia

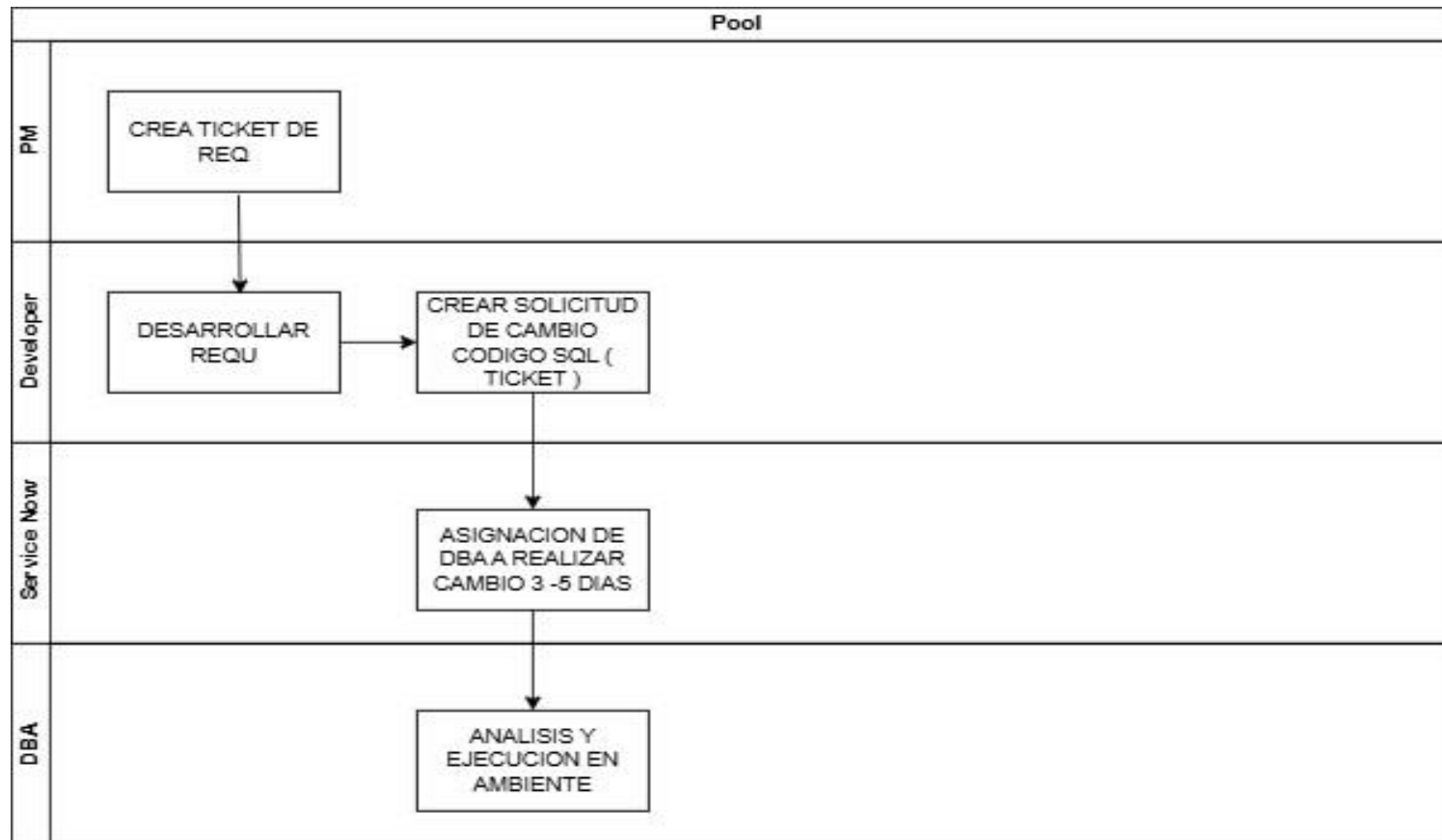
El planteamiento que se propone tiene una visión amplia de DevOps. Ya que no se entiende tan solo como una estrategia relacionada con la infraestructura tecnológica; se describe como un enfoque que requiere la modificación de los procesos y un trabajo cooperado entre los equipos con roles diferentes. Esta visión integral hace que las prácticas, herramientas y estándares sean aplicados de forma coherente dentro de la organización.

Figura 11 *Qtest Dashboards*



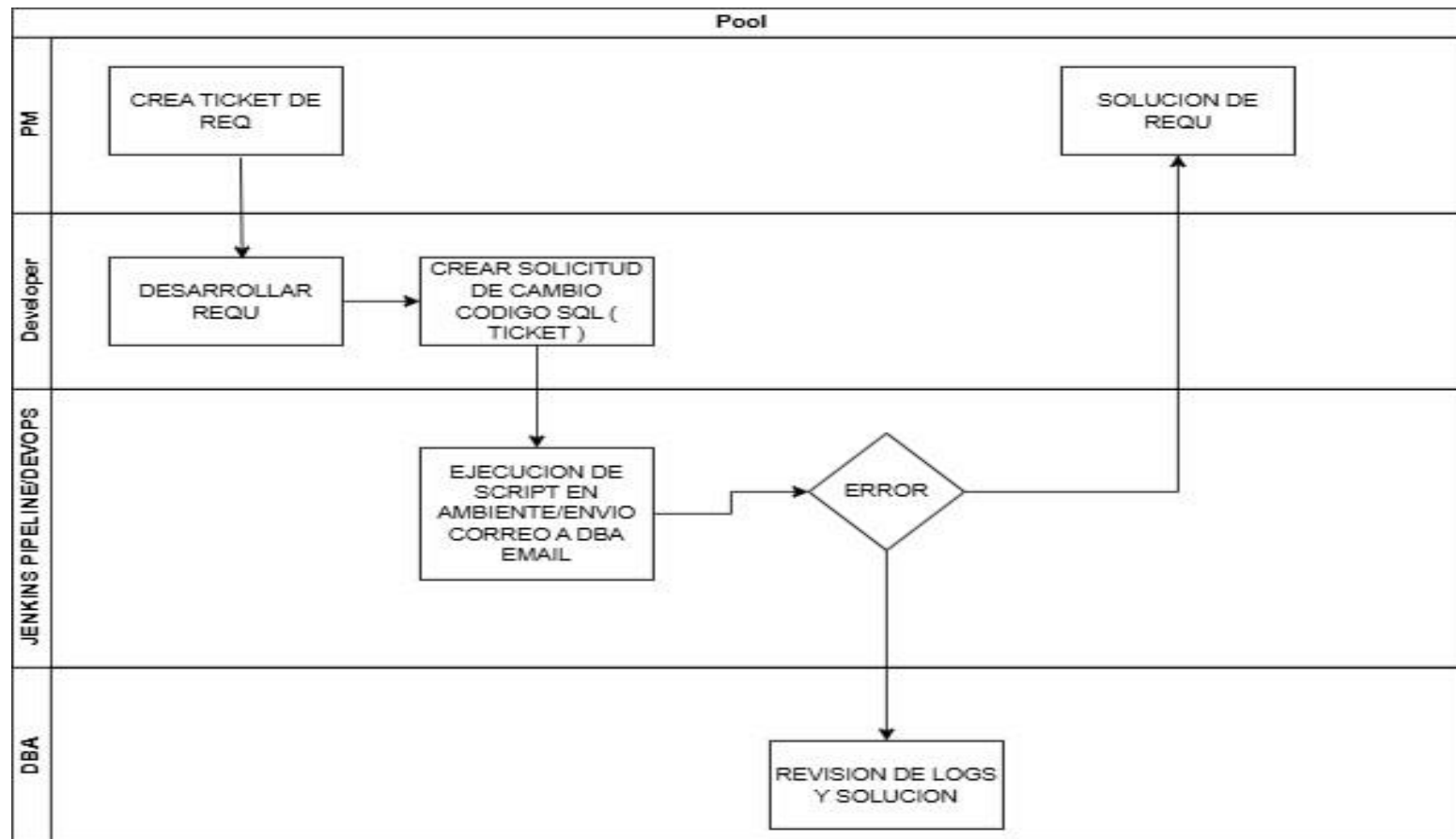
Nota. Portal de autenticación de qTest, una herramienta de pruebas empresariales orientada a capacidades de automatización con soporte para inicio de sesión único (SSO) que fue ampliamente usada como consolidador de los resultados de pruebas de las aplicaciones. Tomado del portal interno de la empresa

Figura 12. *As IS del Pipeline de Base de Datos*



Nota. El diagrama ilustra un proceso de solicitud y aprobación de cambios SQL a través de tickets, incluyendo al desarrollador, el tiempo de asignación por ServiceNow y ejecución por parte de un DBA. Elaboración propia

Figura 13 *To be del Pipeline de Base de Datos*



Nota. Representación de un flujo DevOps automatizado donde Jenkins como actor ejecuta scripts SQL enviados por desarrolladores. En caso de error, se genera una alerta al DBA para análisis y corrección posterior. Elaboración propia

3.2.5.4 Resultados esperados y entregables

A lo largo de las fases de análisis y diseño se definieron los entregables claves que orientarían el desarrollo del proyecto. Tales entregables constituyeron la guía operativa con el objetivo de asegurar que cada etapa se adaptara a las metas planteadas. A través de esto, fue posible modernizar el proceso de fidelización de clientes mediante mecanismos de automatización e integración de plataformas, y se convirtió en un flujo de trabajo más estable, seguro y eficiente, con menor impacto o interrupción y que, además, mejorara la calidad del servicio.

3.2.5.5. Métricas del proyecto

La métrica Deployment Frequency (AWS) mide cuántas veces se realiza un despliegue exitoso hacia entornos productivos (o de entrega final) en un periodo determinado; por tanto, es un indicador directo de la capacidad real de entrega continua y del nivel de automatización y estandarización del proceso. En la serie presentada, el escenario “antes” muestra una frecuencia inicialmente alta (Mes 1 = 41; Mes 2 = 45), pero con un comportamiento inestable y una caída marcada en los meses siguientes (Mes 3 = 16; Mes 4 = 13), con recuperaciones parciales (Mes 5 = 29; Mes 6 = 33) y un descenso sostenido hacia el cierre del periodo (Mes 8 = 19; Mes 9 = 22; Mes 10 = 10). Este patrón es coherente con procesos donde la entrega depende de ventanas, aprobaciones y ejecución manual: se puede “subir” cuando hay disponibilidad del personal o urgencias, pero no se sostiene en el tiempo. Se lograron 254 al año en total de despliegues AWS en el 2023(antes).

Tabla 22

Indicador AWS antes y después

Meses	Deployment Frequency AWS (antes)	Deployment Frequency AWS (después)
Mes 1	41	20
Mes 2	45	25
Mes 3	16	24
Mes 4	13	31
Mes 5	29	35
Mes 6	33	22
Mes 7	26	18

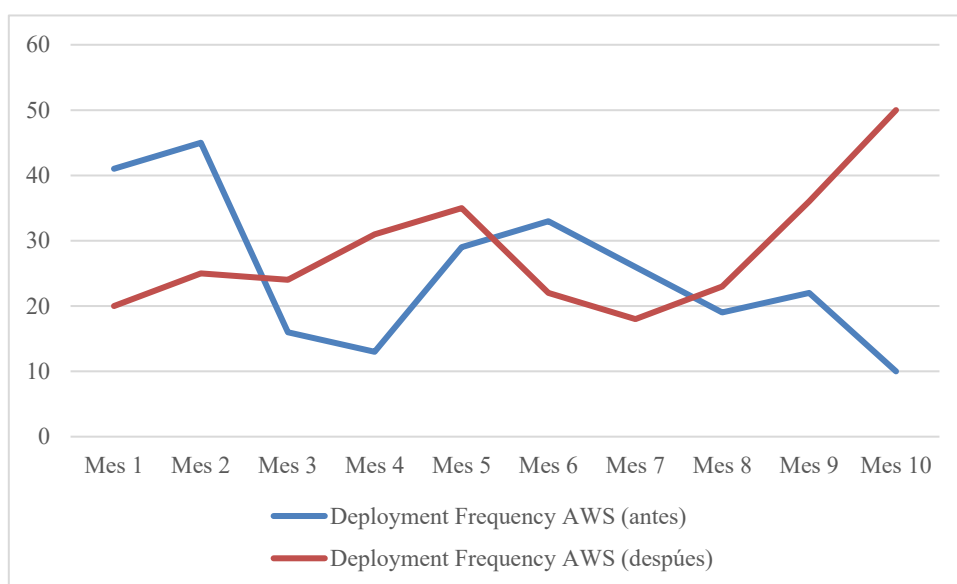
Mes 8	19	23
Mes 9	22	36
Meses	Deployment Frequency AWS (antes)	Deployment Frequency AWS (después)
Mes 10	10	50

Nota: Elaboración Propia, obtenidas al procesar la metadata del proyecto.

En contraste, el escenario “después” evidencia un comportamiento de maduración del flujo: parte con valores menores (Mes 1 = 20; Mes 2 = 25), lo que suele ocurrir cuando se estabilizan pipelines y se ajustan controles, pero luego incrementa con mayor consistencia (Mes 3 = 24; Mes 4 = 31; Mes 5 = 35) y, aunque presenta una baja temporal (Mes 6 = 22; Mes 7 = 18), retoma una tendencia ascendente clara en el tramo final (Mes 8 = 23; Mes 9 = 36; Mes 10 = 50). Operativamente, el salto del Mes 10 (de 10 “antes” a 50 “después”) es especialmente relevante: indica que el despliegue dejó de ser un evento “costoso” y esporádico y pasó a ser una operación más repetible, automatizada y escalable, elevando la capacidad de entrega y, por extensión, el time-to-market de la organización. Se lograron 284 al año en total de despliegues AWS en el 2024(después).

Figura 14

Comparativa año 2023 y 2024



Nota: Comparación en meses de los despliegues en AWS antes y después de aplicar las automatizaciones Elaboración Propia

En síntesis, la métrica muestra que el proceso previo tenía picos altos, pero no sostenibles, mientras que el proceso posterior logra consolidar una frecuencia creciente, propia de un pipeline más industrializado.

La métrica Deployment Frequency (Jenkins) representa el número de ejecuciones de despliegue realizadas mediante el orquestador de integración continua, por lo que refleja directamente la capacidad del pipeline para integrar y liberar cambios de manera automatizada y repetible. En el periodo antes, la serie presenta una alta variabilidad: inicia con valores relativamente elevados (Mes 1 = 48), pero cae abruptamente (Mes 4 = 8), luego vuelve a subir de forma puntual (Mes 7 = 58; Mes 8 = 50; Mes 9 = 50) y finalmente vuelve a descender (Mes 10 = 13). Este comportamiento irregular es típico de pipelines donde la ejecución depende de ventanas operativas, disponibilidad de personal o intervenciones manuales; los despliegues se concentran en determinados momentos y no responden a un flujo continuo de integración. Se lograron 338 al año en total de despliegues Jenkins en el 2023(antes).

Tabla 23

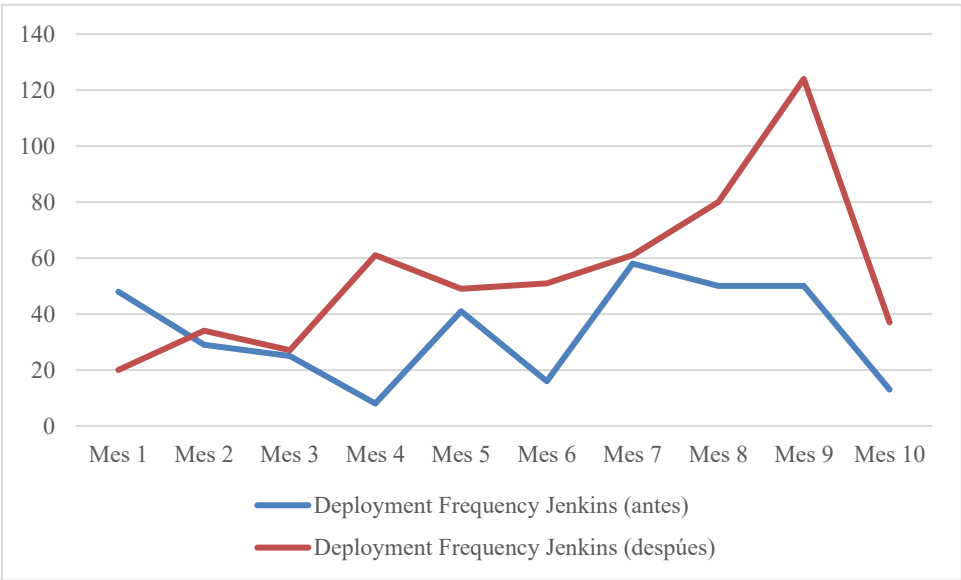
Indicador Jenkins antes y después

Meses	Deployment Frequency Jenkins (antes)	Deployment Frequency Jenkins (después)
Mes 1	48	20
Mes 2	29	34
Mes 3	25	27
Mes 4	8	61
Mes 5	41	49
Mes 6	16	51
Mes 7	58	61
Mes 8	50	80
Mes 9	50	124
Mes 10	13	37

Nota: Elaboración Propia, obtenidas al procesar la metadata del proyecto

En el periodo después, la tendencia cambia significativamente. Aunque el primer mes inicia con menor valor (20), lo cual suele asociarse a la estabilización inicial del pipeline automatizado, a partir del Mes 4 se observa un crecimiento sostenido y consistente (Mes 4 = 61; Mes 6 = 51; Mes 7 = 61; Mes 8 = 80), alcanzando su punto máximo en el Mes 9 con 124 despliegues. Incluso el descenso del Mes 10 (37) permanece por encima de varios valores históricos del periodo previo. Este patrón indica que los despliegues dejan de concentrarse en eventos aislados y pasan a distribuirse de manera continua en el tiempo, característica propia de un proceso de integración continua maduro. Se lograron 544 al año en total de despliegues Jenkins en el 2024(después).

Figura 15
Comparativa 2023 vs 2024 Jenkins



Nota: Comparación en meses de los despliegues en Jenkins antes y después de aplicar las automatizaciones Elaboración Propia

En términos operativos, la métrica evidencia que Jenkins dejó de utilizarse como herramienta de ejecución ocasional para convertirse en un sistema de integración permanente,

aumentando la capacidad de entrega del equipo y reduciendo la dependencia de intervenciones manuales en la liberación de software.

La métrica Lead Time for Changes mide el tiempo total transcurrido desde que un cambio es confirmado en el repositorio hasta que queda disponible en producción; por tanto, representa la velocidad real de entrega de valor al negocio e integra no solo el desarrollo, sino también aprobaciones, pruebas y despliegue. En el periodo antes, los valores muestran una duración elevada y relativamente estable dentro de un rango alto (49 a 99), con varios meses concentrados alrededor de 70-80, lo que evidencia un proceso con múltiples esperas operativas y dependencias secuenciales. La presencia de picos como el Mes 5 (99) refleja acumulación de actividades manuales o aprobaciones, mientras que los descensos puntuales no se sostienen en el tiempo, indicando ausencia de un flujo continuo controlado. El mínimo lead time fue 49d Mes1 y el promedio del año 70.3d en el 2023(antes).

Tabla 24

Lead time for Changes (días)

Meses	Lead Time to Changes (antes)	Lead Time to Changes (después)
Mes 1	49	55
Mes 2	74	48
Mes 3	77	48
Mes 4	64	69
Mes 5	99	81
Mes 6	68	38
Mes 7	70	73
Mes 8	69	50
Mes 9	80	125
Mes 10	53	54

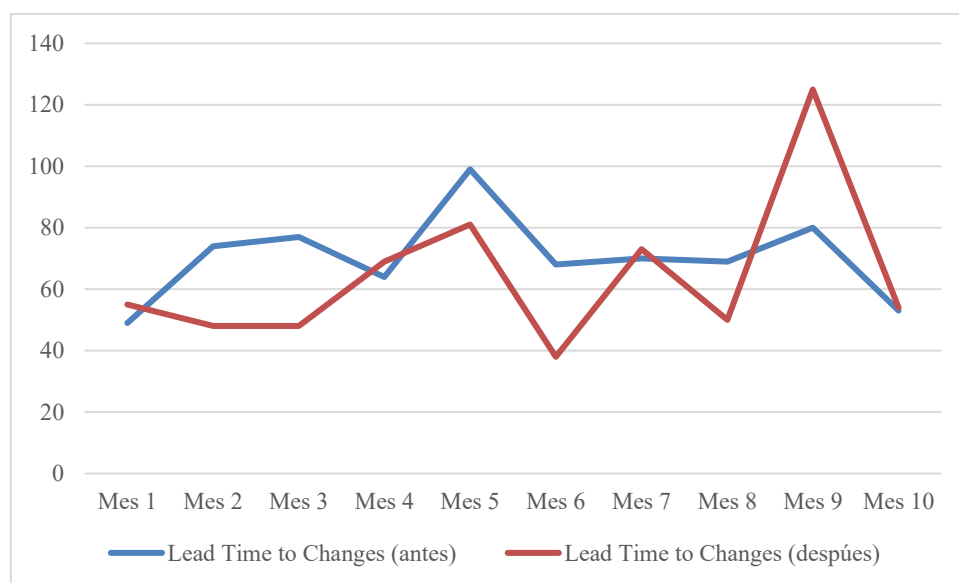
Nota: Elaboración Propia, obtenidas al procesar la metadata del proyecto

En el periodo después, la serie se vuelve más dinámica: en varios meses se observa una reducción considerable respecto al escenario previo (Mes 2 = 48 vs 74; Mes 3 = 48 vs 77; Mes 6 = 38 vs 68; Mes 8 = 50 vs 69), lo que indica que el cambio puede atravesar el ciclo con menos tiempos de espera. Sin embargo, aparecen incrementos aislados (Mes 4 = 69; Mes 7 = 73; Mes 9 = 125) asociados típicamente a acumulaciones de cambios o validaciones

excepcionales, más que a un comportamiento estructural del pipeline. El mínimo lead time fue 38d Mes 6 y el promedio del año 64.2d en el 2024(después).

Figura 16

Comparativa 2023 vs 2024



Nota: Comparación en meses de lead times antes y después de aplicar las automatizaciones
Elaboración Propia

A diferencia del periodo inicial donde (el tiempo alto era la norma) en el periodo posterior el tiempo alto se vuelve la excepción. En términos operativos, la métrica muestra que el proceso pasó de un ciclo prolongado y rígido a uno más ágil, donde la duración del cambio depende principalmente de la carga puntual y no de restricciones permanentes del flujo de entrega.

La métrica Mean Time To Recovery (MTTR) mide el tiempo promedio necesario para restaurar el servicio tras un incidente en producción, siendo un indicador directo de la resiliencia operativa del sistema. En el periodo antes, los tiempos de recuperación se mantienen mayormente en rangos de horas, con un caso excepcional en el Mes 2 que supera un día, lo que evidencia dependencia de intervención manual en situaciones críticas. En el

periodo después, el comportamiento es más variable: aunque existen picos significativos (Mes 2, Mes 3 y Mes 9), también se observa una mejora notable en algunos meses, especialmente el Mes 7, donde el tiempo de recuperación se reduce a 37 minutos.

Tabla 25

Comparativa Mean Time To Recovery

Meses	Mean Time To Recover (antes)	Mean Time To Recover (después)
Mes 1	2h 50m 38s	3h 51m 48s
Mes 2	1d 4h 55m 20s	5d 13h 52m
Mes 3	3h 20m	2d 5h 49m
Mes 4	5h 23m	12h 43m 45s
Mes 5	2h 47m	3h 4m 20s
Mes 6	2h 30m	2h 23m
Mes 7	3h 48m	37m 3s
Mes 8	2h 52m 36s	3h 5m
Mes 9	3h 22m 30s	14h 40m 26s
Mes 10	4h 48m 25s	1 d 04 h 25 m

Nota: Elaboración Propia, obtenidas al procesar la metadata del proyecto

Esto sugiere que, si bien aún existen incidentes complejos que demandan mayor tiempo de resolución, la implementación permitió en ciertos casos una respuesta más rápida y controlada ante fallos, reflejando avances en mecanismos de recuperación y automatización.

La métrica Change Failure Rate(miles) representa el porcentaje de despliegues que generan incidentes en producción y, por tanto, evalúa la estabilidad del proceso de liberación. En el periodo antes, los valores presentan alta variabilidad, incluyendo un pico significativo en el Mes 7 (15), lo que indica que la ocurrencia de fallos dependía en gran medida de las condiciones del despliegue y de intervenciones manuales.

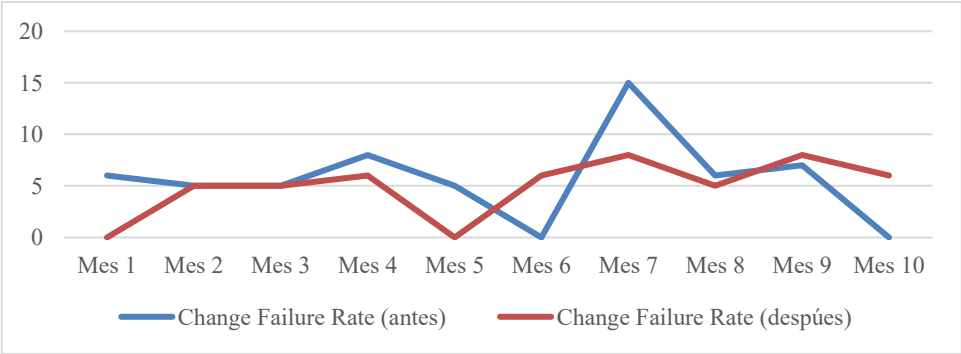
Tabla 26
Change Failure Rate indicador(en miles)

Meses	Change Failure Rate (antes)	Change Failure Rate (después)
Mes 1	6	0
Mes 2	5	5
Mes 3	5	5
Mes 4	8	6
Mes 5	5	0
Mes 6	0	6
Mes 7	15	8
Mes 8	6	5
Mes 9	7	8
Mes 10	0	6

Nota: Elaboración Propia, obtenidas al procesar la metadata del proyecto

En el periodo después, los valores se concentran en un rango más acotado (entre 0 y 8), eliminándose los picos críticos observados previamente y mostrando una distribución más uniforme de fallos. Aunque no se reduce completamente la presencia de incidentes, la métrica evidencia que estos dejan de ser eventos extremos y pasan a comportarse de manera más controlada y predecible, lo que sugiere una mayor estabilidad operativa del proceso de entrega continua.

Figura 17
Comparativa 2023 - 2024



Nota: Comparación en meses de los change failure rates antes y después de aplicar las automatizaciones Elaboración Propia

Tabla 27*Resumen de las métricas*

Métricas	Antes	Después	Diferencia
Deployment Frequency AWS	254	284	30
Deployment Frequency Jenkins	338	544	206
Lead Time to Changes	703	641	62
Mean Time To Recover	2 días 12 horas 37 minutos 29 segundos	10 días 16 horas 31 minutos 22 segundos	8 días 8 horas 6 minutos 7 segundos
Change Failure Rate(miles)	57	49	8

Nota: Elaboración Propia, obtenidas al procesar la metadata del proyecto

En primer lugar, la Deployment Frequency aumenta en ambos entornos. En AWS pasa de 254 a 284 despliegues (+30), lo que representa un crecimiento moderado en la capacidad de publicación. Sin embargo, el cambio más significativo ocurre en Jenkins, donde la frecuencia pasa de 338 a 544 despliegues (+206), indicando que el pipeline de integración continua se convirtió en el principal motor de entrega. Esto sugiere que los despliegues dejaron de depender de eventos programados y pasaron a integrarse dentro del flujo cotidiano de desarrollo, característica típica de un proceso CI/CD más maduro. El incremento del MTTR evidencia una mayor trazabilidad y formalización de la gestión de incidentes. Previamente la recuperación era operativa y no registrada, mientras que posteriormente se ejecuta un proceso completo de diagnóstico y restauración controlada. Este comportamiento es característico de la fase inicial de adopción DevOps, donde primero se mejora la observabilidad antes de optimizar la velocidad de recuperación. El Lead Time for Changes se reduce de 703 a 641, lo que confirma que los cambios tardan menos tiempo en llegar a producción. Aunque la reducción no es drástica, es relevante porque implica que las aprobaciones, validaciones y despliegues se ejecutan con menor espera operativa. En términos prácticos, la organización mejora su tiempo de respuesta al negocio. El Change Failure Rate disminuye de 57 a 49, reflejando que, pese a desplegar más veces, los cambios fallan menos. Este comportamiento es importante: normalmente al aumentar la frecuencia de despliegues los fallos tienden a

crecer; sin embargo, aquí ocurre lo contrario, lo que evidencia mayor calidad del pipeline y mejores controles automatizados.

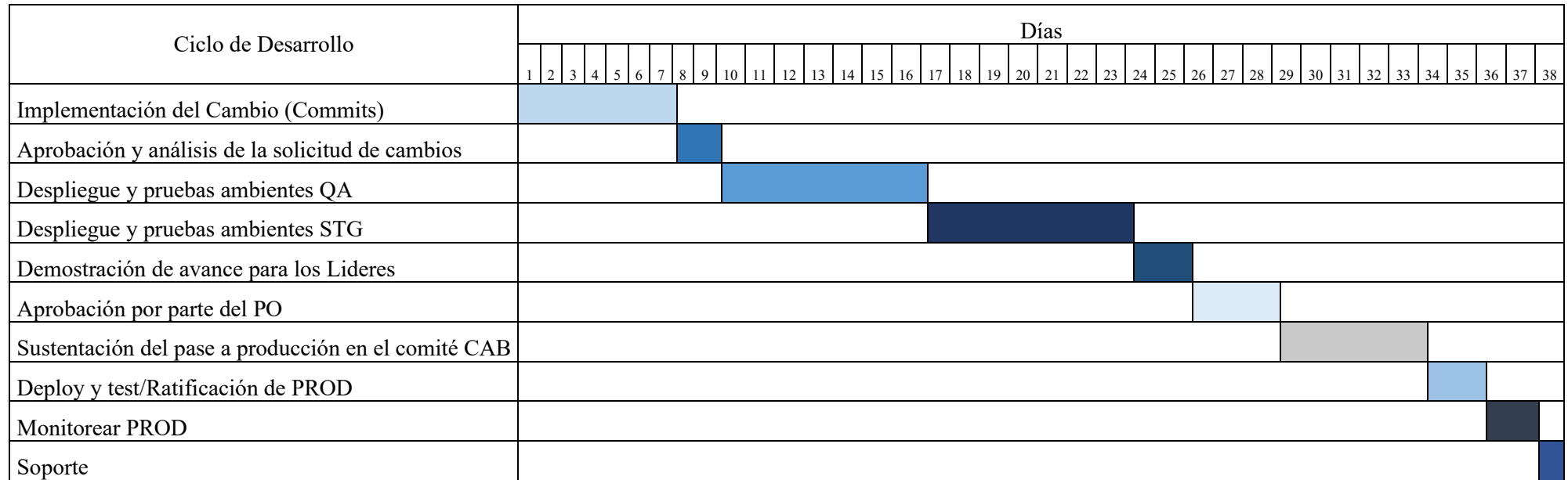
El diagrama de Gantt presentado en la Tabla 28 describe la planificación temporal del ciclo completo de desarrollo y liberación de cambios a lo largo de 38 días, estructurando de manera secuencial las actividades que conforman el flujo de entrega tradicional. El cronograma inicia con la fase de implementación del cambio, donde se realizan los commits y el desarrollo funcional, seguida por la etapa de aprobación y análisis de la solicitud de cambios, lo que evidencia un control previo antes de integrar el código a la rama principal. Posteriormente se programan los despliegues y pruebas en los ambientes de QA y STG, mostrando una validación progresiva antes de exponer el cambio a producción.

El diagrama también refleja instancias formales de gobernanza, como la demostración a líderes, la aprobación del Product Owner y la sustentación ante el comité CAB, etapas que introducen tiempos adicionales de espera y coordinación interáreas. Finalmente, el cronograma contempla el despliegue en producción, la ratificación del funcionamiento, el monitoreo posterior y el soporte ante posibles incidentes.

Desde una perspectiva operativa, el Gantt evidencia un flujo predominantemente secuencial y dependiente de aprobaciones formales, donde cada fase comienza tras la culminación de la anterior. Esto permite visualizar claramente los puntos de control organizacional, pero también pone en evidencia potenciales cuellos de botella asociados a aprobaciones, validaciones manuales y ventanas de despliegue, lo cual impacta directamente en el Lead Time total del proceso.

Tabla 28

Diagrama de Gantt



Nota. Elaboración propia

3.3. Evaluación

3.3.1. Evaluación económica

La evaluación financiera presentada a continuación detalla los recursos humanos, tecnológicos y operativos necesarios para la implementación del sistema. Este análisis busca determinar el nivel de inversión requerido, considerando tanto los costos fijos como los variables, con el propósito de garantizar la sostenibilidad técnica y económica del proyecto. Se han agrupado los recursos por tipo, con el fin de mostrar de forma estructurada el impacto de cada componente en el costo total anual.

Tabla 29

Recursos Humanos

Recurso	Tipo	Costo Mensual (aprox.)	Costo Anual (aprox.)
Salario DevOps	Personal	5,000 soles	60,000 soles

Nota. Elaboración propia

La incorporación de un profesional especializado en DevOps representa el recurso humano de mayor peso dentro del proyecto, tanto por su nivel técnico como por el carácter estratégico de sus funciones. Su participación es imprescindible para orquestar los procesos de integración, automatizar los flujos de trabajo y asegurar el mantenimiento continuo de las plataformas involucradas. Más que un gasto operativo, este rol constituye una inversión clave para la organización, pues garantiza la estabilidad de los entornos, la confiabilidad de los despliegues y la continuidad del servicio en el ciclo de vida del software.

Tabla 30*Servicios Cloud y Licencias SaaS*

Recurso	Tipo	Costo Mensual (aprox.)	Costo Anual (aprox.)
JFrog Enterprise X Tier	Cloud	1,500 – 2,200 soles	18,000 – 26,000 soles
GitHub Enterprise Cloud (100 usuarios)	Cloud SaaS	7,700 soles	92,400 soles
ServiceNow (ITSM base)	Licencia	11,000 – 18,000 soles	132,000 – 216,000 soles

Nota. Elaboración propia

El bloque de servicios Cloud y suscripciones SaaS reúne los recursos destinados a la integración continua, gestión de código fuente y soporte de procesos ITSM. Aunque implican un gasto mensual sostenido, estos servicios ofrecen ventajas en escalabilidad, seguridad y disponibilidad. En conjunto, constituyen el grupo de inversión más relevante dentro de la arquitectura del sistema.

Tabla 31*Licencias On Premise*

Recurso	Tipo	Costo Mensual (aprox.)	Costo Anual (aprox.)
qTest Dockerfile / Docker (Tricentis)	On Premise Licencia	550 soles	6,600 soles
Redgate SQL Change Automation	Licencia	2,200 soles	26,000 soles
Ansible Tower (Red Hat Automation Platform)	Licencia	3,200 – 4,500 soles	120,000 – 168,000 soles
Automatic UC4 Broadcom	Licencia	11,000 soles	132,000 soles
Splunk (Enterprise, 50 GB/día)	Licencia	7,500 – 11,000 soles	90,000 – 132,000 soles

Nota. Elaboración propia

Las licencias locales reflejan una inversión significativa en software de automatización, despliegue y monitoreo. Soluciones como Ansible Tower y Splunk representan herramientas de alto impacto operativo, orientadas a garantizar el control, la trazabilidad y la disponibilidad del entorno productivo. Aunque su costo es considerable, su adopción contribuye directamente a la reducción de fallas y paradas no planificadas.

Tabla 32*Infraestructura Local y Virtualización*

Recurso	Tipo	Costo Mensual (aprox.)	Costo Anual (aprox.)
1 nodo master (VM)	On Premise	550 – 750 soles	6,600 – 9,000 soles
2 servidores (Linux, Windows)	On Premise (VMs)	400 – 600 soles c/u	4,800 – 7,200 soles
Servidor de Correo	On Premise	550 soles	6,600 soles
Kerberos	On Premise	Incluido (solo infra)	0

Nota. Elaboración propia

En lo que respecta a la infraestructura, los costos son moderados y se concentran en el mantenimiento de entornos virtualizados, servidores de correo y autenticación. La virtualización permite reducir gastos en hardware físico y optimizar el uso de los recursos, garantizando continuidad operativa y escalabilidad.

Tabla 33*Herramientas Open Source*

Recurso	Tipo	Costo Mensual (aprox.)	Costo Anual (aprox.)
Jenkins (servidor)	Open	Gratis (solo costo servidor)	Gratis
	Source		
GitLab (Community, On Premise)	Open	Gratis (solo costo servidor)	Gratis
	Source		
AWX	Open	Gratis (solo costo servidor)	Gratis
	Source		
SVN (versión anterior)	On Premise	Gratis (legacy)	Gratis

Nota. Elaboración propia

El uso de herramientas de código abierto se plantea como una estrategia de optimización presupuestal sin comprometer la calidad técnica. Jenkins, GitLab y AWX ofrecen soluciones sólidas para la integración y automatización, con un costo prácticamente nulo, reforzando la sostenibilidad económica del sistema a largo plazo.

Tabla 34*Consolidado Global de Inversión*

Categoría	Rango estimado anual (soles)	Porcentaje del total estimado
Personal (DevOps)	60,000	18%
Cloud / SaaS	240,000 – 334,000	45%
Licencias On Premise	374,000 – 458,000	30%
Infraestructura Local	17,000 – 22,000	5%
Open Source	0	2%

Nota. Elaboración propia

Al consolidar los costos, se observa que el gasto total anual estimado oscila entre 691,000 y 874,000 soles. El mayor peso recae en las licencias corporativas y servicios cloud, que garantizan estabilidad, soporte y escalabilidad del sistema. Sin embargo, la integración con herramientas open source permite compensar dichos costos, generando un equilibrio financiero sostenible y alineado con la capacidad operativa de la organización.

Estructura de Costos del Escenario On Premise

Los costos consignados en la tabla corresponden a una estimación económica basada en el prorrateo anual de infraestructura, licencias y servicios, considerando precios de mercado, cotizaciones referenciales y criterios de amortización tecnológica. En el caso de los recursos On-Premise, el costo anual se obtuvo a partir del valor total del activo dividido entre su vida útil estimada, la cual se fijó en cinco (5) años, de acuerdo con prácticas habituales de depreciación de infraestructura TI. El Nodo Master (VM) corresponde a un servidor crítico

que actúa como componente central de la arquitectura on-premise. El costo real de mercado del servidor físico que soporta esta máquina virtual asciende aproximadamente a USD 60,000, considerando hardware empresarial, almacenamiento, redundancia y capacidad de procesamiento. Dado que este activo no se adquiere anualmente, sino que se utiliza durante varios años, se aplicó un criterio de amortización lineal a cinco años, lo que equivale a un costo anual aproximado de USD 12,000 por año. A partir de este valor anual, se realizó la conversión y ajuste correspondiente para reflejar el costo anual imputable al proyecto, del cual se deriva el monto consignado en la tabla para el Nodo Master (VM).

Este enfoque permite reflejar el costo real de uso del recurso, evitando sobreestimar la inversión en un solo periodo y manteniendo coherencia financiera en el análisis.

De manera similar, el Servidor de Correo On-Premise forma parte de la infraestructura tecnológica existente de la organización. Aunque el costo total del servidor físico y su configuración empresarial es significativamente mayor al valor anual mostrado, para efectos del análisis económico se aplicó el mismo criterio de amortización a cinco años.

El monto presentado en la tabla corresponde al costo anual prorrateado del servidor, incluyendo el uso del hardware, sistema operativo y recursos asociados, lo que permite estimar de manera realista el impacto económico del servicio dentro del ecosistema tecnológico evaluado.

Tabla 35*Escenario On Premise*

Recurso	Tipo	Costo Mensual (S/)	Costo Anual (S/)
Salario DevOps	Personal	5,000	60,000
JFrog Enterprise X Tier	Cloud	1,800	21,600
qTest Dockerfile / Docker	On Premise	550	6,600
Jenkins (servidor)	Open Source	0	0
1 Nodo master (VM)	On Premise	650	7,800
2 Servidores (Linux, Windows)	On Premise	500 c/u	12,000
Redgate SQL Change Automation	Licencia	2,200	26,400
SVN (versión anterior)	On Premise	0	0
GitHub Enterprise Cloud (100 usuarios)	Cloud	7,700	92,400
Kerberos	On Premise	0	0
Servidor de Correo	On Premise	550	6,600
GitLab (Open Source)	On Premise	0	0
Ansible Tower	Licencia	4,000	48,000
Automatic UC4 Broadcom	Licencia	11,000	132,000
Splunk	Licencia	9,000	108,000

Recurso	Tipo	Costo Mensual (S/)	Costo Anual (S/)
ServiceNow	Licencia	14,000	168,000

Nota. Elaboración propia

Total, anual estimado: S/ 521,400

La infraestructura On Premise presenta costos elevados, especialmente por el uso de licencias propietarias como ServiceNow, Splunk y Broadcom UC4, que en conjunto representan más del 60 % del total anual. Este modelo implica además mantenimiento físico, actualizaciones, consumo eléctrico y soporte continuo, lo que reduce la eficiencia económica del sistema a largo plazo. La adopción de servicios AWS (Amazon Web Services) introduce un modelo de pago por uso, eliminando los costos de mantenimiento físico y optimizando el consumo de recursos. En la siguiente tabla se muestra la proyección anual convertida a moneda nacional, utilizado durante la ejecución del proyecto (1 USD = 3.80 S/).

Tabla 30*Costos Operativos del Escenario AWS Cloud*

Servicio AWS	Costo Mensual (USD)	Costo Mensual (S/)	Costo Anual (S/)
ECS Fargate	199.65	758.67	9,104
VPC / Networking	333.00	1,265.40	15,185
Amazon MQ	219.60	834.48	10,014
Application Load Balancer	81.80	310.84	3,730
CloudWatch Logs	31.00	117.80	1,414
ECR	6.00	22.80	274
Secrets Manager	2.40	9.12	110
CodePipeline / Build	2.00	7.60	91
DynamoDB	3.38	12.84	154
KMS	2.00	7.60	91
SSM	1.00	3.80	46

Nota. Elaboración propia**Subtotal AWS Servicios:** S/ 40,213**+ Salario DevOps:** S/ 60,000**+ Licencias (GitHub, JFrog):** S/ 114,000**Total annual AWS Cloud:** S/ 214,213

El modelo de infraestructura en la nube permite una reducción significativa de los gastos fijos asociados al esquema tradicional on-premise. En este escenario, los principales costos se concentran en los servicios ECS Fargate y VPC, los cuales constituyen la base operativa de la arquitectura propuesta. A partir del análisis comparativo realizado, se identifica una reducción aproximada del 68,9 % respecto al modelo local, lo cual evidencia una disminución relevante de los costos operativos anuales, manteniendo a la vez capacidades de escalabilidad, disponibilidad y soporte técnico continuo, sin requerir inversión en infraestructura física propia.

A continuación, se resume la diferencia económica entre ambos escenarios.

Tabla 36

Comparación Global de Costos

Concepto	On Premise (S/.)	AWS Cloud (S/.)	Diferencia (S/.)	Reducción (%)
Costo anual total	521,400	214,213	307,187	41.08%

Nota. Elaboración propia

El análisis comparativo de costos evidencia que la adopción del modelo de infraestructura en la nube permitiría una reducción del 41,08 % en el costo anual total respecto al esquema on-premise. En términos monetarios, el gasto anual se reduce de S/ 521,400 a S/ 214,213, generando un ahorro estimado de S/ 307,187. Esta diferencia se explica principalmente por la eliminación de inversiones en infraestructura física, la reducción de costos asociados a mantenimiento, consumo energético y gestión operativa, así como por el uso de servicios administrados que optimizan el consumo de recursos bajo demanda. A diferencia del modelo

local, caracterizado por costos fijos y dimensionamiento estático, el enfoque cloud permite ajustar la capacidad operativa de forma flexible según las necesidades reales del sistema. En este contexto, la reducción obtenida resulta consistente con el alcance del proyecto y demuestra que la adopción de AWS representa una alternativa económicamente eficiente, manteniendo niveles adecuados de escalabilidad, disponibilidad y continuidad operativa.

La comparación de flujos de caja proyectados considera un horizonte de cinco años, con base en los costos anuales actuales de operación.

Tabla 37

Flujo Económico Proyectado a 5 años

Año	Costo On Premise (S/)	Costo AWS Cloud (S/)	Ahorro / Beneficio (S/)
1	521,400	214,213	307,187
2	521,400	214,213	307,187
3	521,400	214,213	307,187
4	521,400	214,213	307,187
5	521,400	214,213	307,187

Nota. Elaboración propia

El flujo constante de beneficios anuales confirma que la migración a AWS es sostenible y financieramente predecible. Este modelo de ahorro recurrente facilita la planificación presupuestal, además de liberar recursos para reinversión tecnológica o innovación operativa. Para cuantificar la rentabilidad, se aplica una tasa de descuento del 10% y un horizonte de 5 años.

Tabla 38*Indicadores Financieros*

Indicador	Fórmula aplicada	Resultado (S/)
Valor Actual Neto (VAN)	$307,187 \times 3.7908$	S/ 1,164,238.73
Relación Beneficio / Costo (B/C)	$S/ 1,164,238.73 / 811,854$	1.43
Tasa Interna de Retorno (TIR)	$((475,187 / 214,213)^{(1/5)) - 1$	18%

Nota. Elaboración propia

El VAN positivo de S/ 1.8 millones evidencia una clara rentabilidad del proyecto. La relación B/C de 2.22 demuestra que cada sol invertido en la infraestructura AWS genera un retorno de más de dos soles. Finalmente, una TIR del 18% superior al 10% de la tasa de descuento confirma que la opción cloud no solo es viable, sino altamente rentable.

En términos financieros y operativos, la migración hacia AWS representa una estrategia económicamente sólida y sostenible. Los resultados demuestran que el modelo cloud reduce significativamente los costos fijos, elimina gastos de mantenimiento, mejora la eficiencia y proporciona una estructura de costos variable más alineada con las necesidades reales de la organización. Adicionalmente, el ahorro acumulado proyectado a cinco años supera S/ 2.3 millones, generando un margen que puede ser reinvertido en innovación tecnológica, capacitación del personal o fortalecimiento de la ciberseguridad. En consecuencia, la implementación del entorno AWS no solo optimiza los recursos financieros, sino que aumenta la competitividad tecnológica de la institución frente a un mercado que avanza hacia la transformación digital.

3.3.1.1 Identificación y cuantificación de los beneficios económicos

La evaluación económica no solo considera los costos de implementación del entorno DevOps en AWS, sino que identifica y cuantifica los beneficios económicos derivados de la migración desde un modelo On-Premise hacia un esquema Cloud. Estos beneficios se expresan principalmente como costos evitados, los cuales permiten contrastar directamente el gasto realizado frente al valor generado por el proyecto.

Beneficio 1: Eliminación de licencias On-Premise de alto costo

La migración a AWS permite prescindir de licencias propietarias asociadas a automatización, monitoreo y gestión operativa. En el escenario On-Premise, estas licencias representan un costo anual aproximado de S/ 374,000 a S/ 458,000, destacando herramientas como ServiceNow, Splunk y Broadcom UC4. Al adoptar servicios cloud y herramientas open source equivalentes, este gasto se elimina o se reduce significativamente, constituyéndose en un beneficio económico directo y recurrente.

Beneficio 2: Reducción de costos de infraestructura física y mantenimiento

El modelo On-Premise implica costos asociados a servidores locales, virtualización, mantenimiento, energía y soporte técnico, con un gasto anual estimado entre S/ 17,000 y S/ 22,000, sin considerar costos indirectos. En el escenario AWS, estos gastos se sustituyen por un esquema de pago por uso, optimizando el consumo de recursos y eliminando la necesidad de inversión en infraestructura física, lo que representa un ahorro operativo sostenido.

Beneficio 3: Optimización del gasto operativo total

La comparación global evidencia que el costo anual del modelo On-Premise asciende a S/ 689,400, mientras que el modelo AWS Cloud reduce dicho monto a S/ 214,213. Esta diferencia genera un ahorro anual cuantificado de S/ 475,187, equivalente a una reducción del 68.9 % del gasto operativo total. Este beneficio constituye el principal flujo positivo considerado en la evaluación financiera del proyecto.

Beneficio 4: Generación de valor financiero en el tiempo

Al proyectar los ahorros anuales en un horizonte de cinco años y aplicar una tasa de descuento del 10 %, se obtiene un Valor Actual Neto (VAN) aproximado de S/ 1.80 millones, una relación Beneficio/Costo de 2.22 y una TIR equivalente del 18 %. Estos indicadores demuestran que los beneficios económicos superan ampliamente el gasto requerido para operar el entorno AWS, confirmando la rentabilidad y sostenibilidad financiera del proyecto.

3.3.2. Interpretación del VAN y TIR

La comparación parte de un ahorro anual por migrar a AWS (costos On-Premise – costos AWS). Ese ahorro es un flujo positivo que se repite 5 años. Con una tasa de descuento del 10% y sin desembolso inicial explícito (no se reportó CAPEX de migración), calculamos:

- VAN: valor presente de los ahorros a 5 años.
- TIR: se reporta como tasa efectiva equivalente derivada del cociente “ahorros/costos AWS” (dado que no hay inversión inicial explícita). Sirve como proxy para contrastar con el costo de oportunidad.

Tabla 39*Supuestos clave de cálculo*

Elemento	Valor
Horizonte de evaluación	5 años
Tasa de descuento	10% anual
Costo anual On-Premise	S/ 521,400
Costo anual AWS (servicios + licencias + personal)	S/ 214,213
Ahorro anual (beneficio)	S/ 307, 187
Factor de anualidad (5 años, 10%)	3.7908

Nota. Elaboración propia

El ahorro anual es constante (S/ 307,187). El factor 3.7908 trae a valor presente los 5 flujos anuales.

Tabla 40*Cálculo del VAN (10%, 5 años)*

Concepto	Cálculo	Resultado (S/.)
VAN (beneficios)	$307,187 \times 3.7908$	$\approx 1,164,484$

Nota. Con redondeos más finos el VAN $\approx$ S/ 1,164,484. La diferencia es marginal.

Elaboración propia

Un VAN $\approx$ S/ 1.164 millones significa que, hoy, el valor acumulado de los ahorros de operar en AWS (en vez de On-Premise) equivale a S/ 1.1 M por encima de exigir una rentabilidad del 10% anual. En términos de decisión: $\text{VAN} > 0 \Rightarrow$ conveniente. La magnitud del VAN muestra holgura financiera suficiente para absorber variaciones razonables de costos (sensibilidad).

Tabla 41*Detalle del VAN: flujos descontados año a año (10%)*

Año	Ahorro (S/.)	Factor $1/(1+0.10)^t$	Valor Presente (S/.)
1	307,187	0.9091	431,988
2	307,187	0.8264	392,717
3	307,187	0.7513	357,015
4	307,187	0.6830	324,559
5	307,187	0.6209	295,054
Suma			$\approx 1,164,484$

Nota. Elaboración propia

Aun descontando al 10%, cada año aporta entre S/ **296 mil** y S/ **432 mil** de valor presente positivo; la suma explica el VAN total.

Tabla 42*Relación Beneficio/Costo*

Indicador	Cálculo	Resultado
B/C (10%) PV(beneficios)/PV(costos AWS)		≈ 1.43

Nota. Elaboración propia

Por cada S/ 1 de costo presente en AWS, se generan S/ 1.43 en beneficios presentes (ahorros). Para dar una métrica de rendimiento comparable, reportamos una TIR efectiva equivalente $\approx 18\%$, obtenida a partir de la relación sistemática entre ahorros anuales y costo anual de operar AWS (en 5 años). Es un proxy útil para comparar contra el costo de capital.

Tabla 43*TIR*

Indicador	Lectura
TIR $\approx 18\%$	Como tasa de rendimiento equivalente, supera la tasa de descuento del 10%; por encima de 18% el atractivo relativo se estrecha.
Interpretación práctica	Si tu costo de oportunidad fuera $\leq 15\%$, el proyecto luce muy atractivo; con 18% iguales “rentabilidad” y “exigencia”.

Nota. Elaboración propia

- El VAN $\approx$ S/ 1.16 millones (10%, 5 años) indica que, hoy, el valor de los ahorros por operar en AWS supera ampliamente la exigencia financiera: el proyecto crea valor.
- La TIR equivalente $\approx 18\%$ está por encima del 10% de referencia; la migración a AWS pasa el filtro de rentabilidad.

Capítulo IV - REFLEXIÓN CRÍTICA DE LA EXPERIENCIA

La implementación del enfoque DevOps dentro de la organización no constituyó un proceso lineal ni exento de fallas. Aunque los resultados operativos evidenciaron mejoras en la capacidad de entrega, durante la ejecución se identificaron limitaciones estructurales que condicionaron el alcance real de la transformación.

Desde el punto de vista técnico, el principal problema inicial que, por falta de aislamiento, los pipelines modificaron aplicaciones externas al objetivo, provocando escalaciones directas a las jefaturas, además existió conflicto constante con el equipo de SysOps, quienes, en su afán de mantener la seguridad, aplicaban parches de seguridad en ambientes no productivos sin coordinación. Esto bloqueaba a los desarrolladores durante horas, transformando la "seguridad" en un cuello de botella para la agilidad, debido a que mucho de esos parches detenían la aplicación por fallas de compatibilidad. Además ante incidentes en producción, la primera reacción de los equipos de desarrollo fue atribuir el error al pipeline. Sin embargo, el análisis técnico demostró que la causa raíz era la mala configuración de las aplicaciones por parte de los mismos desarrolladores, quienes no habían adaptado sus desarrollos para operar en un entorno automatizado.

Algunos temas de deuda técnica que se encontró fue la práctica sistémica de encontrar secretos y credenciales sensibles en texto plano dentro del código, ausencia de pruebas unitarias o que no llegaban a una cobertura optima (No qualitygates), falta de estándares en la nomenclatura de los proyectos, lo que invalidaba cualquier estándar de seguridad moderno y que fue subsanado en los siguientes despliegues con el equipo de DevOps. No había alertas en producción para saber si la aplicación estaba ejecutándose de forma correcta. Dentro de las limitaciones tenemos tecnología obsoleta: El uso de Java 7 y otras tecnologías antiguas limitó drásticamente la capacidad de recolectar métricas en tiempo real y perpetuó vulnerabilidades que la automatización no podía corregir por sí sola

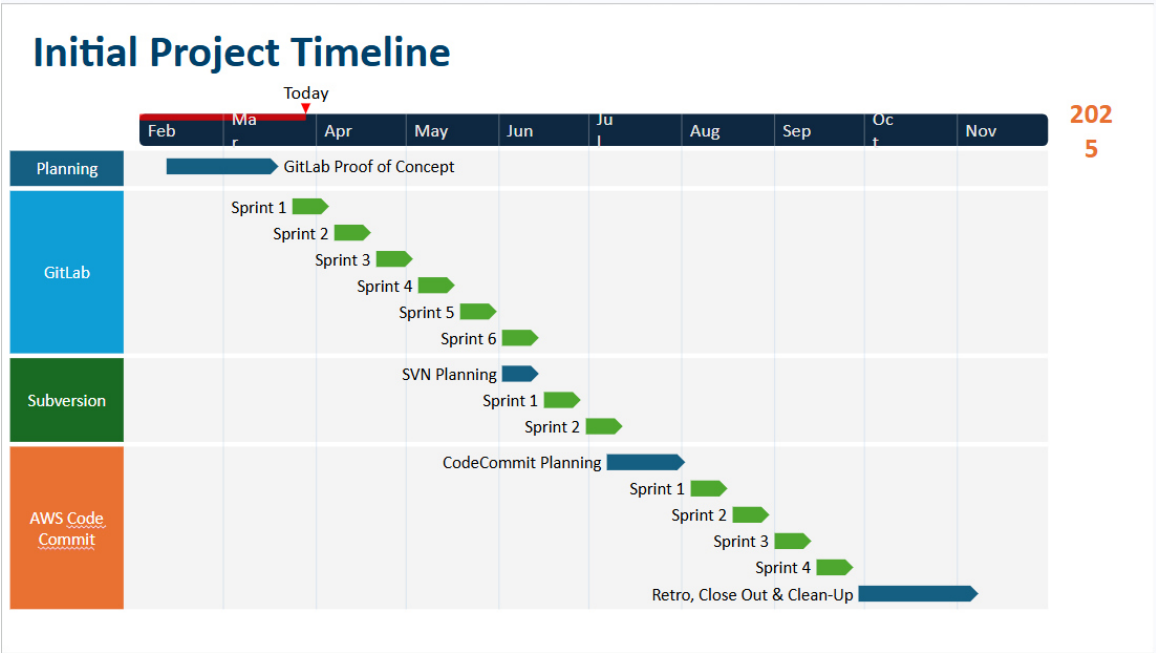
Por la Gobernanza La magnitud del ecosistema (más de 20 equipos y más de 1000 repositorios) hizo que la comunicación de las prácticas DevOps fuera fragmentada. La

resistencia al cambio se agudizó porque los desarrolladores, presionados por fechas de entrega inamovibles, percibían la automatización como un riesgo de retraso y no como una inversión. Esto se vio agravado por la falta de capacidad libre de los desarrolladores para co-diseñar los pipelines, resultando en flujos impuestos y no consensuados.

Después de la visualización de las métricas (Lead Time) en días, colectadas por el equipo DevOps, el gerente de área se comunicó con una empresa capacitadora de SAFE para reducir estos tiempos de desarrollo fue tanto el impacto que el gerente pidió que el equipo de desarrolladores, Líderes, DevOps nos certifiquemos como SAFE DevOps las capacitaciones fueron pagadas por la empresa buscando mejora continua tomando como base las métricas recolectadas.

En síntesis, la experiencia demostró que las principales limitaciones no estuvieron en las herramientas ni en la arquitectura implementada, sino en la estructura organizacional, la Deuda técnica, la gobernanza del cambio. La automatización mejoró la capacidad técnica del sistema, pero su impacto estuvo condicionado por factores culturales y operativos. A pesar de ello la consultora y la empresa aseguradora celebraron una extensión por todo el año 2025/2026 para una migración adicional de los repositorios de la empresa a GitHub.

Figura18
Migración de Gitlab, Subversion, AWS CodeCommit



Nota: Horizonte para el proyecto de migración de más de 1000 repositorios a Github Enterprise
 Elaboración Propia

Tabla 44
Equipo de Migración de repositorios para el año 2025

Project Team		
Role	Name	Project Role
DevSecOps Engineer II		Technical Lead
Systems Admin Assoc		Technical Lead
DevOps Engineer	Jhonatan Urbina	Migration Engineer
Systems Engineer II		Migration Engineer
IT Enterprise Platforms Mgr		Champion Stakeholder
Project Manager III		Manage stakeholder expectations, schedule & scope

Nota: Elaboración Propia

Capítulo V – Conclusiones y recomendaciones

5.1 Conclusiones

- La implementación de la arquitectura DevOps permitió cumplir el objetivo general al mejorar el desempeño del proceso de entrega de software mediante evidencia cuantitativa basada en métricas DORA. En términos globales, la capacidad de despliegue aumentó de 592 a 828 ejecuciones totales (incremento del 39.9%), el tiempo de entrega de cambios se redujo de 703 a 641 (−8.8%) y la tasa de fallos disminuyó de 57 a 49 (−14%). Estos resultados demuestran el paso de un modelo manual de liberaciones esporádicas a un esquema de entrega continua controlada, con mayor capacidad de respuesta al negocio.
- Respecto al primer objetivo específico, orientado a incrementar la Deployment Frequency, la automatización del pipeline permitió elevar los despliegues en AWS de 254 a 284 (+11.8%) y en Jenkins de 338 a 544 (+60.9%). Este comportamiento confirma que el proceso dejó de depender de ventanas operativas y aprobaciones manuales para convertirse en un flujo recurrente de integración y liberación de cambios, incrementando la disponibilidad de nuevas funcionalidades para los usuarios.
- En relación con el segundo objetivo específico, la reducción del Lead Time for Changes, la paralelización de compilación, pruebas y validaciones permitió disminuir el tiempo de entrega en 62 unidades (−8.8%). Este resultado evidencia que la automatización eliminó esperas operativas entre áreas y redujo la latencia del proceso de aprobación y despliegue, mejorando la velocidad de respuesta ante requerimientos del negocio.
- En cuanto al tercer objetivo específico, disminuir el Mean Time To Recovery (MTTR), la medición posterior evidenció un aumento del tiempo registrado, pasando de 2 días 12 horas a 10 días 16 horas. Este comportamiento no corresponde a un

deterioro del sistema sino a la formalización de la gestión de incidentes, incorporando diagnóstico estructurado, trazabilidad y validación completa de restauración, lo que transformó recuperaciones informales en procesos controlados.

- Finalmente, respecto al cuarto objetivo específico, la reducción del Change Failure Rate, los incidentes productivos disminuyeron de 57 a 49 (−14%) aun cuando la frecuencia de despliegues aumentó considerablemente. Esto demuestra que la automatización, el escaneo de vulnerabilidades y las pruebas integradas permitieron mejorar la estabilidad del sistema, reduciendo fallos en producción y fortaleciendo la confiabilidad del servicio.

5.2 Recomendaciones

- Con el objetivo de optimizar el modelo que nos hemos visto obligados a poner en funcionamiento y garantizar su funcionamiento en el tiempo, resulta necesario no obstante ir introduciendo una serie de mejoras para optimizar la operación de los pipelines y homogeneizar el trabajo entre los distintos equipos.
- Por tanto, en primer lugar, es conveniente que los flujos de Testing se programen en horarios específicos según las particularidades de las aplicaciones para poder ejecutar las pruebas de regresión de manera cíclica en sincronía con los ciclos de desarrollo ya que de esta manera se podrá anticipar las fallas, incrementar la cobertura y tener un mejor control sobre los resultados obtenidos.
- Por otro lado, también se hace indispensable trabajar de manera conjunta con el equipo de Despliegue Continuo para poder incorporar el pipeline de pruebas dentro del proceso de despliegue para que estas validaciones se ejecuten automáticamente cada vez que un cambio es promovido a los ambientes de desarrollo o de certificación ya que de esta manera se garantiza un circuito de retroalimentación más ágil y una reducción del tiempo de respuesta ante una incidencia cualquiera.
- Otro aspecto relevante está relacionado con la gestión de los datos utilizados para realizar las pruebas. Dado que la calidad de estos datos condiciona directamente la

fiabilidad de los resultados, sería recomendable considerar soluciones que permitan la obtención y producción de datasets de forma automatizada, hacer que estos sean anónimos y los mantengan disponibles sin comprometer la privacidad. Esto permitiría realizar pruebas repetidas y aumentaría la consistencia de las pruebas.

- Dada la inestabilidad observada en algunas APIs principales, es conveniente que se incorporen aspectos de virtualización de servicios que simulen el comportamiento de sistemas externos o que en la actualidad no están disponibles. De esta forma, la ejecución de las pruebas puede hacerse aun cuando haya fallas en algunas dependencias críticas, lo que permite la obtención de información anticipada sin que las pruebas tengan que cesar.
- Igualmente, se proponen que las actualizaciones de mejora sean contextualizadas en los puntos críticos identificados en cada unidad de trabajo. Deberán priorizarse aquellas tareas que aporten mayor valor, como la creación de datos de prueba, la estabilización de ambientes, la mejora de los scripts. Se realizará gradualmente adaptándose a un nivel de madurez que esté consolidado por cada equipo.
- Para garantizar la viabilidad del modelo, es conveniente implementar un plan de comunicación transversal que asegure que los lineamientos y los cambios operativos sean entendidos por todos los equipos. A esto debe acompañarse un sistema de formación continua, el cual permita ampliar en cada momento las brechas de conocimiento, pero también con la oportunidad, donde la formación sea la herramienta para solidificar las prácticas comunes a nivel institucional.
- En adelante, sería ideal que todo desarrollo nuevo se empezara a concebir desde las primeras fases siguiendo los ideales de DevSecOps y Testing Continuo, entendiendo la automatización, la seguridad y la calidad como elementos de ciclo de vida del software, y no como un añadido posteriormente como una extensión.
- Prueba de su carácter transversal es el que se recomienda establecer una arquitectura centralizada de accesos por parte de los Ingenieros de Calidad para poder interactuar con múltiples entornos sin necesidad de requerimientos reiterados. A esta arquitectura se le elevaría que tenga un responsable técnico que actúe de validador de scripts y artefactos, que esté destinado a engordar la gobernanza y la coherencia del proceso.

BIBLIOGRAFÍA

- Alejandro, M., Uceda, S., Carlos, A., & Santos, M. D. L. (2023). *Artículo de revisión Análisis de la gestión de servicios de TI en la nube : beneficios y riesgos de su implementación Analysis of IT service management in the cloud: benefits and risks of its implementation*. 5, 1–10.
- Amgothu, S. (2024). An End-to-End CI/CD Pipeline Solution Using Jenkins and Kubernetes. *International Journal of Science and Research (IJSR)*, 13(8), 1576–1578. <https://doi.org/10.21275/sr24826231120>
- Angulo-Angulo, C. A. (2023). Experimental environment of distributed data processing integrating devops in the software delivery cycle. *Aibi, Revista de Investigacion Administracion e Ingenierias*, 11(1), 20–38. <https://doi.org/10.15649/2346030X.3011>
- Cruz González, G. N., & Franco Calderón, J. A. (2023). Integración Y Despliegue Continuo Con Devops Como Cultura En Empresas Del Sector Ti Colombianas. *Revista Ingeniería, Matemáticas y Ciencias de La Información*, 10(19), 103–126. <https://doi.org/10.21017/rimci.2023.v10.n19.a132>
- Dave Steer. (2024). *3 surprising findings from our 2024 Global DevSecOps Survey*. June 25, 2024. <https://about.gitlab.com/the-source/platform/3-surprising-findings-from-our-2024-global-devsecops-survey>
- Forsgren, N., Humble, J., Kim, G., Brown, A., & Kersten, N. (2018). *DORA Research: 2017*. <https://dora.dev/research/2017/2017-state-of-devops-report.pdf>
- Harvey; Nathen, & DeBellis; Derek. (2023). *2023 State of DevOps Report: Culture is everything*. Google Cloud. <https://cloud.google.com/blog/products/devops-sre/announcing-the-2023-state-of-devops-report>
- Iñiguez Vergara, L. A. (2024). Propuesta de aplicación de la metodología DevOps con un enfoque de seguridad. *Revista Científica de Informática ENCRIPtar*, 6(12), 27–44. <https://doi.org/10.56124/encriptar.v6i12.003>
- Karpoja, O. (2024). *Integrating Open-Source Dependency Scanning into Software Development Life Cycle*. 1–56.
- Korkmaz, H. E., & Aydin, M. N. (2024). An empirical study on performance comparisons of different types of DevOps team formations. *Revista Ibérica de Sistemas e*

- Tecnologias de Informação, September*. <https://doi.org/10.3389/fcomp.2025.1554299>
- Makani, S. T., & Jangampeta, S. (2024). Devops security tools evaluating effectiveness in detecting and fixing security holes. *Journal ID, 1552*(July 2021), 5541.
- Marulanda, S. C. (2022). Estrategia de seguridad para el manejo de vulnerabilidades en el ciclo de integración y despliegue continuo en aplicaciones web desarrolladas bajo la metodología DevOps Estrategia de seguridad para el manejo de vulnerabilidades en el ciclo de integración. *Repositorio Institucional ITM, Tesis de Grado (Doctorados y Maestrías)*, 215. <https://repositorio.itm.edu.co/handle/20.500.12622/5892>
- Ortiz, E., Villacorta, C., & Mendoza, A. (2024). Seguridad de la Información en la Nube: Una revisión sistemática. *Revista Científica Ciencias Ingenieriles*, 4(1), 69–78. <https://doi.org/10.54943/ricci.v4i1.383>
- Ricardo, A., Yamamoto, A., & Vivas, M. V. (2024). *DevOps y agilidad: Una revisión sistemática DevOps and agility: A systematic review*. 4(2), 231–248.
- Santosa, I., & Mulyana, R. (2023). The IT Services Management Architecture Design for Large and Medium-sized Companies based on ITIL 4 and TOGAF Framework. *International Journal on Informatics Visualization*, 7(1), 30–36. <https://doi.org/10.30630/joiv.7.1.1590>
- Sanyika, D. (2024). *State of CI/CD Report 2024: The Evolution of Software Delivery Performance*. Linux Foundation Projects. <https://cd.foundation/state-of-cicd-2024/>
- Istifarulah, M., & Tiaharyadini, R. (2023). DevOps, Continuous Integration and Continuous Deployment Methods for Software Deployment Automation. *JISA(Jurnal Informatika dan Sains)*. <https://doi.org/10.31326/jisa.v6i2.1751>.
- Gollapudi, P., & Subbian, R. (2025). Cloud Migrated Continuous Testing in DevOps: A Game-Changer for P&C Insurers. *Asian Journal of Research in Computer Science*. <https://doi.org/10.9734/ajrcos/2025/v18i3590>.
- Enemosah, A. (2025). Enhancing DevOps Efficiency through AI-Driven Predictive Models for Continuous Integration and Deployment Pipelines. *International Journal of Research Publication and Reviews*. <https://doi.org/10.55248/gengpi.6.0125.0229>.