# Migración y adopción de tecnologías azure devops en la compañía Suramericana

John Fredy Mejía Serna

Informe de práctica presentado para optar al título de Ingeniero de Sistemas

Asesora

Sandra Patricia Zabala Orrego, Especialista en gerencia de proyectos

Universidad de Antioquia

Facultad de ingeniería

Ingeniería de sistemas

Medellín

2023

**TABLA DE CONTENIDO**


RESUMEN 7

ABSTRACT 8

INTRODUCCIÓN 9

1 OBJETIVOS 10

Objetivo general 10

Objetivos específicos 10

2 MARCO TEÓRICO 11

3 METODOLOGÍA 14

4 RESULTADOS 16

5 CONCLUSIONES 26

REFERENCIAS 28

## LISTA DE TABLAS

TABLA 1 Pipelines ci/cd a migrar 16

## LISTA DE FIGURAS

Fig. 1. Pipeline ms Sarlaft Web hook 18

Fig. 2. Plantilla gradle-openshift 19

Fig. 3. Pipeline de ms Firmavirtual-integrador 21

Fig. 4. Vista de sitio sharepoint Sura 23

Fig. 5. Editor html de Sharepoint 24

Fig. 6. Editor logic apps Azure 25

## SIGLAS, ACRÓNIMOS Y ABREVIATURAS

| | |
|---|---|
| **AKS** | Azure kubernetes services |
| **CI/CD** | Continuous Integration/Continuous Delivery |
| **DevOps** | Development and Operations |
| **GIT** | Global Information Tracker |
| **HTML** | HyperText Markup Language |
| **IaC** | Infrastructure as Code |
| **PaaS** | Platform as a service |
| **PO** | Product Owner |
| **SVN** | Subversion |
| **TI** | Tecnología de la Información |
| **YAML** | Yet Another Markup Language |

## RESUMEN

En este documento se da una descripción del proceso que se realizó en el periodo de semestre de industria en la compañía Suramericana, el cual estuvo focalizado en la migración de los diferentes pipelines de sus aplicativos y adopción de nuevas plantillas ci/cd para lograr estandarizar dichos flujos.

Entre las actividades realizadas está la migración de los pipelines ci/cd de distintos microservicios desarrollados en java implementados con el framework spring boot,que al momento se encontraban ya en azure devops pero aún no estaban adoptando los lineamientos de plantillas pipelines que la compañía lleva realizando desde 2022, también se migra microservicios que sus pipelines estaban operando en la plataforma Jenkins y de esta forma se unifica en una sola suite tecnológica lo concerniente a despliegues en azure devops.

Gracias a la adopción de Azure DevOps, se logró una consolidación efectiva de todas las herramientas tecnológicas. Esto permite optimizar y simplificar los procesos de despliegue de aplicaciones, mejorar la eficiencia y fomentar una mayor colaboración entre los equipos de trabajo.

Adicionalmente se efectuaron tareas de apoyo en otras áreas, como le fue la gestión de incidentes que se reportaban en la intranet de la compañía,la cual está soportada por el ecosistema de microsoft sharepoint,donde se atendieron distintos casos, según se fueran notificando por usuarios a nivel administrativo.

***Palabras clave*** **— Pipeline, ci/cd, azure devops, microservicio, despliegue, gestión de incidentes, microsoft sharepoint.**

## ABSTRACT

This document provides a description of the process carried out during the industry semester at Suramericana company, which was focused on the migration of various application pipelines and the adoption of new CI/CD templates to standardize these workflows.

Among the activities carried out was the migration of CI/CD pipelines for different microservices developed in Java and implemented with the Spring Boot framework. These pipelines were already in Azure DevOps but had not yet adopted the pipeline template guidelines that the company has been implementing since 2022. Additionally, microservices with pipelines operating on the Jenkins platform were migrated, thus unifying all aspects related to deployments in Azure DevOps into a single technological suite.

Thanks to the adoption of Azure DevOps, an effective consolidation of all technological tools was achieved. This allows for the optimization and simplification of application deployment processes, improved efficiency, and the promotion of greater collaboration among teams.

Additionally, support tasks were carried out in other areas, such as incident management reported on the company's intranet, which is supported by the Microsoft SharePoint ecosystem. Various cases were addressed as they were reported by users at the administrative level.

***Keywords*** **— Pipeline, ci/cd, azure devops, microservice, deployment, incident management, microsoft sharepoint.**

# INTRODUCCIÓN

La compañía Suramericana, es una filial del grupo Sura especializada en la industria de seguros, que como gestora de tendencias y riesgos entrega capacidades a las personas y las empresas en nueve países.
En el área de T.I la compañía cuenta con varios dominios encargados de gestionar y dar soporte a toda la demanda tecnológica que se presenta.

La compañía se encuentra en un proceso de migrar a nuevas tecnologías en todo lo relacionado con los procesos de despliegues de aplicativos, que se viene manejando en la plataforma Jenkins, también de repositorios de sus códigos fuente actualmente alojados en Apache Subversion (SVN) una tecnología que surgió antes del conocido git y el manejo de los boards para la gestión de los distintos equipos de trabajo en la plataforma Jira; por ende el propósito a llevar a cabo es que todas estas herramientas queden unificadas en una suite tecnológica y es allí donde entra azure devops quien suple todas las necesidades que las anteriores soluciones provee y agregando algunas más , entre ellas tenemos azure boards,azure repos,azure pipelines, azure test plans, azure artifact.

Entre las actividades que se realizará se encuentra la migración de los pipelines de los distintos aplicativos , sea frontend, backend o librerías, desde Jenkins hacia azure devops, también la creación de templates de algunas tecnologías como lo es openshift,aks,weblogic entre otras.

# 1 OBJETIVOS

## *1.1 Objetivo general*

Implementar migración de pipelines para su despliegue en plataforma azure devops en la compañia Suramerica.

## *1.2 Objetivos específicos*

- Analizar cómo están construidos los template de los distintos pipelines según el lenguaje de programación usando diagramas y la documentación oficial.
- Adquirir el conocimiento sobre las actuales prácticas de devops que se realizan en la compañía, valiéndome de la documentación oficial de la compañía centralizada en su plataforma Confluence.
- Implementar las migraciones de los pipelines en ambiente de desarrollo
- Implementar las migraciones de los pipelines en ambiente laboratorio(pruebas).
- Implementar las migraciones de los pipelines en ambiente de producción.
- Crear y habilitar templates que faciliten el despliegue de los distintos aplicativos ,siguiendo lineamientos de la empresa.

# 2 MARCO TEÓRICO

DevOps es un enfoque de colaboración y comunicación que busca unir el desarrollo de software (Development) y la operación de sistemas (Operations) para mejorar la velocidad y la calidad de la entrega de software. Se basa en prácticas ágiles, automatización y monitorización continua, con el objetivo de mejorar la eficiencia y la capacidad de respuesta a los cambios del mercado. Algunas prácticas para adoptar la cultura DevOps son las siguientes:

Colaboración y comunicación: Fomentar una cultura de colaboración comunicación constante entre los equipos de desarrollo y operaciones y promueva la integración continua.

Automatización: Automatizar tareas repetitivas y procesos mediante el uso de herramientas y tecnologías, como scripts de programación y herramientas de integración y entrega continua (CI/CD).

Pruebas continuas: Integrar pruebas automatizadas en el proceso de entrega de software para mejorar la calidad y la confiabilidad.

Infraestructura como código: Adoptar un enfoque de infraestructura como código (IaC) para gestionar la infraestructura de manera eficiente y repetible.

Mejora continua: Promover un enfoque de mejora continua en todos los aspectos del proceso de entrega de software, desde el desarrollo hasta la operación.

Dentro de esta estrategia de desarrollo de software existen procesos que toman vital importancia como lo son la integración continua, entrega continua y el despliegue continuo (CI/CD por sus siglas en Inglés).

La integración continua es una de las prácticas más difundidas en el ecosistema DevOps. Es un proceso que se origina en un servidor de versionado como lo puede ser Git y finaliza con la ejecución de un conjunto de pruebas que se ejecutan sobre el código compilado o interpretado.

Las etapas de integración continua son las siguientes:

Planificación: El ciclo de integración continua y del proceso completo, comienza con una etapa de planeamiento. Ya sea que estemos desarrollando una funcionalidad nueva, o resolviendo un problema, lo que sea que se vaya a desarrollar, se planifica. Si bien la etapa es agnóstica a la metodología, lo normal es que se planifique usando metodologías ágiles o afines.

Codificación: El equipo de desarrollo empieza a escribir el código que da vida a la funcionalidad o resuelve el problema encontrado. Acá se hacen uso herramientas de gestión de código como Git o similares.

Construcción: El proceso de compilación del código, donde se ejecutan las herramientas y/o scripts que generan el *artefacto* del componente o aplicación.

Pruebas de funcionamiento: Esta etapa introduce los primeros chequeos a ejecutar sobre el código desarrollado y sobre la aplicación desplegada. La idea es hacer pruebas sencillas que descarten problemas evidentes. Este tipo de pruebas se le conoce como *Smoke tests* o *Pruebas de humo*.

El despliegue continuo comienza luego de que la etapa de integración finaliza, dando lugar a las tareas esenciales para asegurar la calidad del código o del artefacto generado y a las tareas que involucran el despliegue y entrega final. En esta línea de razonamiento, no hay un inicio, sino una continuación del flujo originado por el proceso de integración. Por el contrario, esta etapa sí tiene un final, siendo este, el servidor o contenedor donde se quiere impactar el código o artefacto de la aplicación.

Las etapas del despliegue continuo son las siguientes:

Testing: En esta etapa se agregan pruebas de calidad de código, cobertura de código, performance, seguridad, deuda técnica, etc. Es crucial que en esta etapa se definan de manera precisa los criterios de aceptación, ya que de ellos depende la continuación o detención del ciclo.

Es importante destacar que esta etapa no es completamente nueva, sino una extensión de la anterior.

Release: En esta etapa se realizan todas las tareas relacionadas con la liberación, desde la confección de los documentos como notas, changelogs, etc. Además, se configuran las bases necesarias para poder desplegar el código o artefacto, como, por ejemplo, el etiquetado del código (version tagging), congelamiento del código, etc.

Despliegue: Se escriben e impactan los automatismos para desplegar la aplicación o componente, en cada uno de los ambientes que formen parte del ciclo. Este despliegue suele ser concatenado, es decir, primero se despliega en un ambiente réplica de producción llamado “staging”, el cual sirve para hacer las validaciones finales y funcionales del despliegue. La cantidad de ambientes previos a producción va a estar determinada por la necesidad de la organización y no tanto por definición, por lo que podría haber, entre los ambientes de desarrollo y de staging, un ambiente de integración, donde se pone a prueba la aplicación junto con el ecosistema con el cual va a interactuar, como lo pueden ser servicios de LDAP, Single Sign On, APIs, etc.

Operación: En esta etapa se monitorea y se mide el estado de salud y performance del despliegue. Un buen sistema de monitoreo nos permite anticiparnos a los problemas y aprender de ellos. Recuperarnos más rápido de lo que fallamos es parte esencial de la ecuación, y todo un desafío sino se cuenta con las herramientas adecuadas. Acá suelen integrarse automatismos que nos permitan reaccionar espontáneamente a determinados eventos.

Se exponen los conceptos teóricos que sustentan el desarrollo del trabajo, debidamente referenciados.

# 3 METODOLOGÍA

El desarrollo de todas las actividades se realiza con el marco de trabajo ágil SCRUM, el cual es un proceso en el que se aplican de manera regular un conjunto de buenas prácticas para trabajar colaborativamente, en equipo, y obtener el mejor resultado posible de un proyecto. Estas prácticas se apoyan unas a otras y su selección tiene origen en un estudio de la manera de trabajar de equipos altamente productivos.

En Scrum se realizan entregas parciales y regulares del producto final, priorizadas por el beneficio que aportan al receptor del proyecto. Por ello, Scrum está especialmente indicado para proyectos en entornos complejos, donde se necesita obtener resultados pronto, donde los requisitos son cambiantes o poco definidos, donde la innovación, la competitividad, la flexibilidad y la productividad son fundamentales.

Está conformado por los siguientes roles:

Un Product Owner (PO): Es la persona responsable del valor del producto y del trabajo en equipo.

Un Scrum Master o facilitador: Se encarga del diálogo y la organización de las daily meeting, los Sprints y dar solución a cualquier eventualidad que se pueda presentar.

Desarrolladores: Son los encargados de dar solución a los diferentes problemas presentados por los PO o a las prioridades que se presenten a través de la célula de operaciones.

Diseñador técnico: Es la persona encargada de todo el proceso relacionado a la creación y diseño de nuevos sistemas.

Las actividades que se realizan dentro del equipo scrum del cual se realiza la presente propuesta son:

Sprint: Ejecución de tareas en iteraciones de dos semanas , dentro de este equipo cada sprint tiene una duración de 11 días.

Planning: Ceremonia en la que se planea el siguiente Sprint y se escoge el Sprint Backlog.

Daily Meeting: Reunión que se realiza tres veces por semana a las 8:00 am y que dura 30 minutos, en la que cada miembro del Scrum Team esboza su avance con sus tareas, o reporta si tiene dificultades.

Sprint Review: Al finalizar un sprint se muestra a los stakeholders y al Product Owner los incrementos en el producto.

Sprint Retrospective: Se analiza el sprint que terminó evaluando que se hizo bien, que se hizo mal y que cosas se pueden mejorar.

Refinement Session: Se refina el backlog de cara al próximo Sprint.

La herramientas de la cual el equipo está haciendo uso para el desarrollo de las tareas es Azure Boards, con esta se logra total gestión de las actividades del sprint.

# 4 RESULTADOS

En esta sección, se presentaran los resultados de trabajo durante el semestre de prácticas en el dominio soluciones administrativas de TI de la compañía Suramericana.

En el equipo de DevOps, se han asignado historias de usuario relacionadas con la migración de los pipelines de cada microservicio desplegado. Esta acción fue tomada debido a que la compañía ha adoptado nuevos lineamientos para las diversas plataformas y/o nubes en las cuales se despliegan sus aplicativos. A continuación, se presenta la tabla 1 que enlista las migraciones realizadas durante los primeros sprints y la plataforma a la cual se desplegaron:

| APM | AKS Migrables | WebLogic Migrables | Repo |
|---|---|---|---|
| 892 | Sarlaft WebHook | | https://dev.azure.com/SuraColombia/Gerencia_Tecnologia/_git/adm_y_fin-sarlaft-function-webhook-mi |
| 892 | Sarlaft Validador Cliente Identidad | | https://dev.azure.com/SuraColombia/Gerencia_Tecnologia/_git/adm_y_fin-validadorcliente-identidad-ms |
| 892 | Sarlaft Function CCM | | https://dev.azure.com/SuraColombia/Gerencia_Tecnologia/_git/adm_y_fin-sarlaft-function-ccm-mi |
| 892 | Sarlaft Function Clientes | | https://dev.azure.com/SuraColombia/Gerencia_Tecnologia/_git/adm_y_fin-sarlaft-function-clientes-mi |
| 892 | Sarlaft Function ClientPJ | | https://dev.azure.com/SuraColombia/Gerencia_Tecnologia/_git/adm_y_fin-sarlaft-function-clientPJ-mi |
| 892 | Sarlaft Function Identity | | https://dev.azure.com/SuraColombia/Gerencia_Tecnologia/_git/adm_y_fin-sarlaft-function-identity-mi |
| 892 | Sarlaft Function Peps | | https://dev.azure.com/SuraColombia/Gerencia_Tecnologia/_git/adm_y_fin-sarlaft-function-peps-mi |
| 892 | Sarlaft Function Rrcc | | https://dev.azure.com/SuraColombia/Gerencia_Tecnologia/_git/adm_y_fin-sarlaft-function-rrcc-mi |
| 892 | Sarlaft Function Listas | | https://dev.azure.com/SuraColombia/Gerencia_Tecnologia/_git/adm_y_fin-sarlaft-function_listas-mi |
| 892 | Sarlaft Function P8 | | https://dev.azure.com/SuraColombia/Gerencia_Tecnologia/_git/adm_y_fin-sarlaft-function_p8-mi |
| 892 | Sarlaft Function Requisitos | | https://dev.azure.com/SuraColombia/Gerencia_Tecnologia/_git/adm_y_fin-sarlaft-function_requisitos-mi |
| 892 | Sarlaft Function ClientPN | | https://dev.azure.com/SuraColombia/Gerencia_Tecnologia/_git/adm_y_fin-sarlaft-function-clientPN-mi |
| 892 | Sarlaft Function Saveclientes | | https://dev.azure.com/SuraColombia/Gerencia_Tecnologia/_git/adm_y_fin-sarlaft-function_saveclientes-mi |
| 892 | Sarlaft Webapp Asesores | | https://dev.azure.com/SuraColombia/Gerencia_Tecnologia/_git/adm_y_fin-sarlaft-webapp-asesores-mi |
| 892 | Sarlaft Function Batch | | https://dev.azure.com/SuraColombia/Gerencia_Tecnologia/_git/adm_y_fin-sarlaft-function-batch-mi |
| 892 | Sarlaft Clientes | | https://dev.azure.com/SuraColombia/Gerencia_Tecnologia/_git/892-sarlaft4-sarlaftclientes-ms |
| 907 | Talento Humano Nomina | | https://dev.azure.com/SuraColombia/Gerencia_Tecnologia/_git/adm_y_fin-talento_humano-nomina-ms |

Tabla 1. Pipelines ci/cd a migrar

Para la ejecución de estas migraciones fue necesario una capacitación por parte de analistas de la compañía y también un exhaustivo análisis de la documentación que la empresa tiene a disposición en sus distintos canales.

El apoyo que se realizó para cumplir con las tareas asignadas fue migrar cada microservicio, puesto que sus pipelines ci/cd estaban alineados con prácticas que la compañía está dejando a un lado para adoptar el uso de plantillas predeterminadas por cada plataforma de despliegue.

Las plantillas de pipeline CI/CD ofrecen una forma eficiente de definir y configurar estos pasos en un formato reutilizable. En lugar de tener que crear y configurar manualmente cada paso en un pipeline, las plantillas proporcionan una estructura predefinida que se puede adaptar a los requisitos específicos de un proyecto.

Las plantillas de pipeline CI/CD generalmente están escritas en un lenguaje de definición de flujo de trabajo, como YAML, y se pueden almacenar en repositorios de código fuente o en herramientas de integración y entrega continuas, como en este caso Azure DevOps.
Al utilizar plantillas de pipeline CI/CD, los equipos de desarrollo pueden ahorrar tiempo y esfuerzo al no tener que configurar repetidamente los mismos pasos en diferentes proyectos. Además, las plantillas promueven la consistencia y la estandarización en el proceso de CI/CD, lo que facilita la colaboración y la detección de errores.

En la figura 1 se puede observar la migración del pipeline del microservicio Sarlaft webhook a una plantilla preestablecida para todo despliegue gradle-aks , en este caso un microservicio desarrollado con spring boot y como gestor de dependencias y compilación gradle , y a su vez el despliegue a producción se realiza en AKS, la cual es un servicio administrado de Azure que permite implementar, administrar y escalar clústeres de contenedores de Kubernetes de manera sencilla. Kubernetes es una plataforma de código abierto para automatizar la implementación, el escalado y la administración de aplicaciones en contenedores.

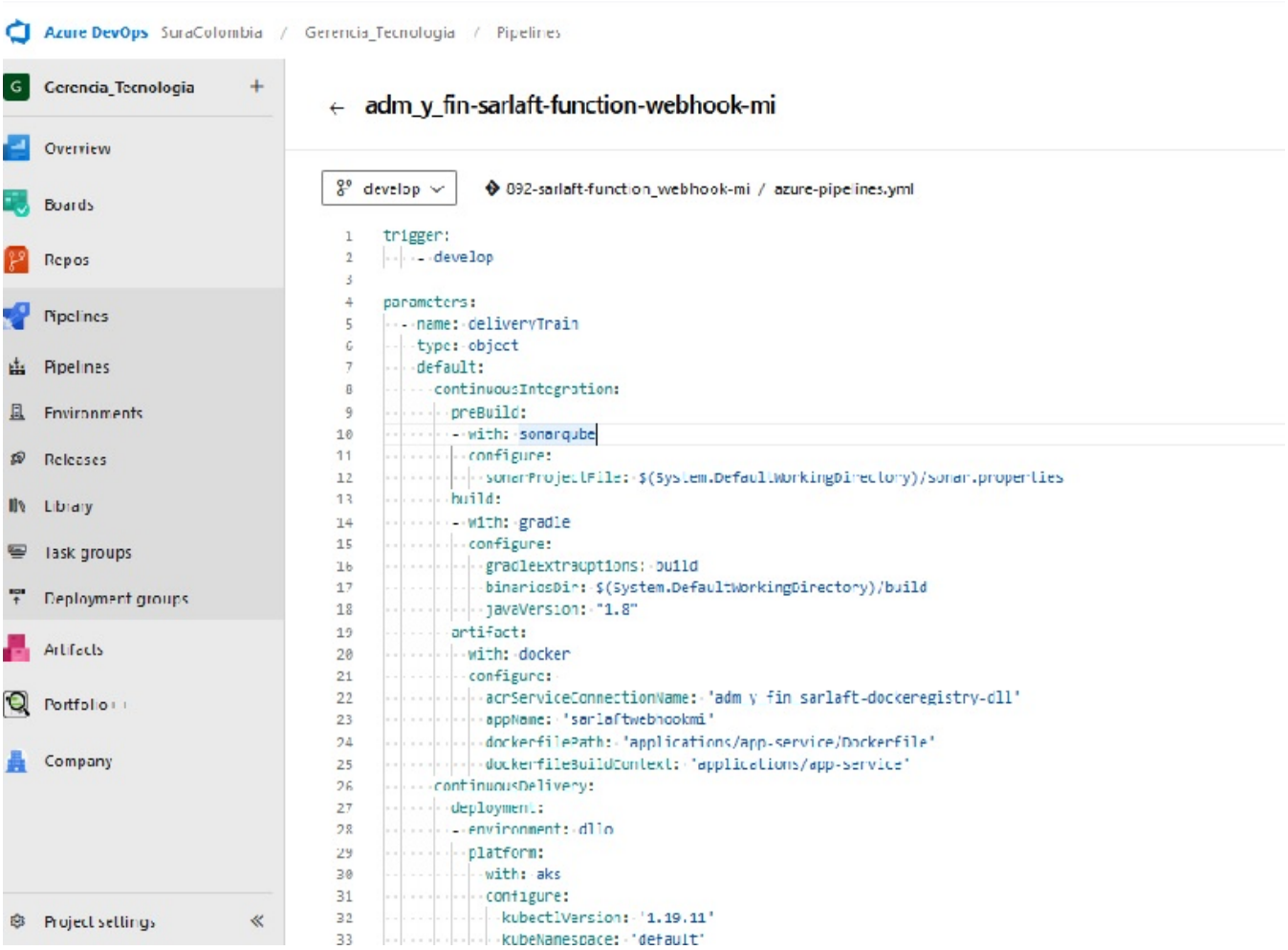


Fig. 1. Pipeline ms Sarlaft Web hook

En cuanto a migraciones para otras plataformas, fue necesario desarrollar la plantilla para despliegues gradle-openshift y este en ambiente on-premise. OpenShift On-Premise es una solución de plataforma como servicio (PaaS) basada en contenedores que permite a las organizaciones implementar y administrar clústeres de OpenShift en su propio entorno de infraestructura local o privada. A diferencia de OpenShift en la nube pública, OpenShift On-Premise brinda la flexibilidad de tener el control total sobre la infraestructura subyacente y los recursos utilizados.

Al elegir OpenShift On-Premise, las organizaciones pueden construir y administrar su propia plataforma de aplicaciones basada en contenedores sin depender de proveedores externos de

servicios en la nube. Esto les permite mantener sus aplicaciones y datos dentro de su infraestructura privada o local, lo que puede ser beneficioso en términos de seguridad, cumplimiento normativo y políticas internas, se basa en tecnologías de código abierto, como Kubernetes y Docker, y proporciona una amplia gama de características y herramientas para el desarrollo, implementación y administración de aplicaciones en contenedores.

Para el desarrollo de la plantilla fue de suma importancia analizar como se tenía implementado otras plantillas ya implementadas como fue el caso de  gradle-aks , en base a esta se dio inicio y realizaron los ajustes necesarios para la solución que se necesitaba en openshift , en la figura 2 se muestra el template finalizado:

templates > gradle-openshift.yaml

```yaml
parameters:
  - name: deliveryTrain
    type: object

stages:
- template: ../transversal/stages/continuous-integration.yaml
  parameters:
    continuousIntegration: ${{ parameters.deliveryTrain.continuousIntegration }}

- ${{ if ne(variables['Build.Reason'],'PullRequest') }}:
  - ${{ if parameters.deliveryTrain.continuousDelivery }}:
    - template: ../transversal/stages/approvals.yaml
      parameters:
        deployment: ${{ parameters.deliveryTrain.continuousDelivery.deployment }}
    - template: ../transversal/stages/continuous-delivery.yaml
      parameters:
        continuousDelivery: ${{ parameters.deliveryTrain.continuousDelivery }}
        artifact: ${{ parameters.deliveryTrain.continuousIntegration.artifact }}

  - ${{ if parameters.deliveryTrain.continuousTesting }}:
    - template: ../transversal/stages/continuous-testing.yaml
      parameters:
        continuousTesting: ${{ parameters.deliveryTrain.continuousTesting }}
```

Fig. 2. Plantilla gradle-openshift

En la figura 2 se puede observar la construcción final de la plantilla gradle-openshift.yaml, quedando en tan solo 23 líneas de código, esto gracias a la modularización que se aplicó valiéndonos del parámetro template el cual renderiza un nuevo yaml y estos a su vez podrian renderizar otros yaml, se explica con más detalle:

Parameters: En esta sección, se definen los parámetros que se utilizarán en el pipeline. En este caso, se define un parámetro llamado "deliveryTrain" de tipo objeto.

Stages: En esta sección, se definen las etapas (stages) del pipeline. Cada etapa puede contener uno o varios trabajos (jobs) que se ejecutan secuencialmente.

Template: Se utiliza la directiva "template" para hacer referencia a una plantilla YAML externa que contiene la configuración de una etapa específica del pipeline. En este caso, se hace referencia a una plantilla llamada "continuous-integration.yaml" ubicada en la ruta "../transversal/stages/continuous-integration.yaml". Esta plantilla se utilizará para la etapa de integración continua.

Parameters: Se definen los parámetros específicos de la plantilla referenciada. En este caso, se pasa el valor del parámetro "continuousIntegration" del parámetro "deliveryTrain" al parámetro "continuousIntegration" de la plantilla.

En la figura 3 se muestra la plantilla ya  aplicada en el pipeline del ms Firmavirtual-integrador:

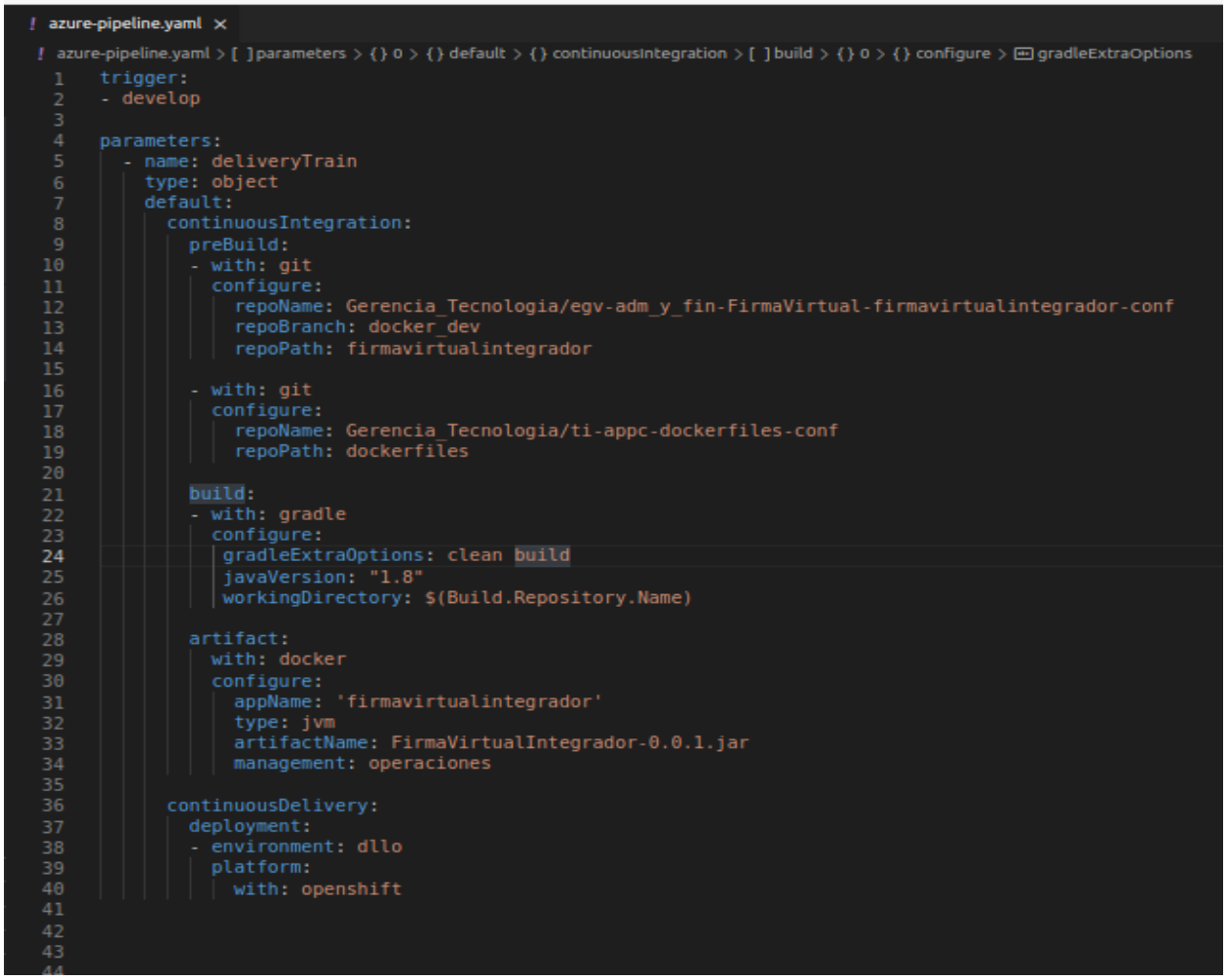


Fig. 3. Pipeline de ms Firmavirtual-integrador

A continuación, se explica su estructura y función:

Trigger: En esta sección, se especifica el disparador (trigger) del pipeline. En este caso, el pipeline se activará cuando se realicen cambios en la rama "develop" del repositorio.

Parameters: En esta sección, se definen los parámetros que se utilizarán en el pipeline. En este caso, se define un parámetro llamado "deliveryTrain" de tipo objeto con un valor predeterminado.

Default: Dentro de la sección "parameters", se define el valor predeterminado del parámetro "deliveryTrain". Este valor predeterminado incluye la configuración de integración continua (continuousIntegration) y entrega continua (continuousDelivery) para el pipeline.

Resources: En esta sección, se definen los recursos utilizados en el pipeline. En este caso, se define un repositorio de tipo git que se utilizará para acceder a plantillas adicionales.

Extends: En esta sección, se utiliza la directiva "extends" para extender una plantilla de pipeline existente. Se hace referencia a la plantilla "gradle-openshift.yaml" ubicada en el repositorio de plantillas,la cual se explicó anteriormente y se pasan los parámetros al template utilizando la variable "parameters.deliveryTrain".

Dentro de las actividades que también se desarrollaron durante el semestre, se atendió el requerimiento de algunos incidentes reportados por usuarios finales en la plataforma microsoft sharepoint que a rasgos generales es una plataforma de gestión y colaboración de contenido empresarial y permite a las organizaciones crear sitios webs, portales y aplicaciones para compartir información y documentos de manera eficiente.

Los usuarios de SharePoint pueden crear sitios web y subsitios para diferentes equipos, departamentos o proyectos dentro de una organización. Estos sitios pueden incluir bibliotecas de documentos, listas de tareas, calendarios, foros de discusión, wikis y otros componentes que facilitan la colaboración y el intercambio de información, en la figura 4 se observa una pantalla de un sitio sharepoint el cual se actualizo siguiendo los requerimientos de determinados usuarios.


Fig. 4. Vista de sitio sharepoint Sura

Algunas de esas actividades de soporte en sharepoint fueron la actualización de sitios según los requerimientos de usuario final , desde el editor html de sharepoint figura 5.

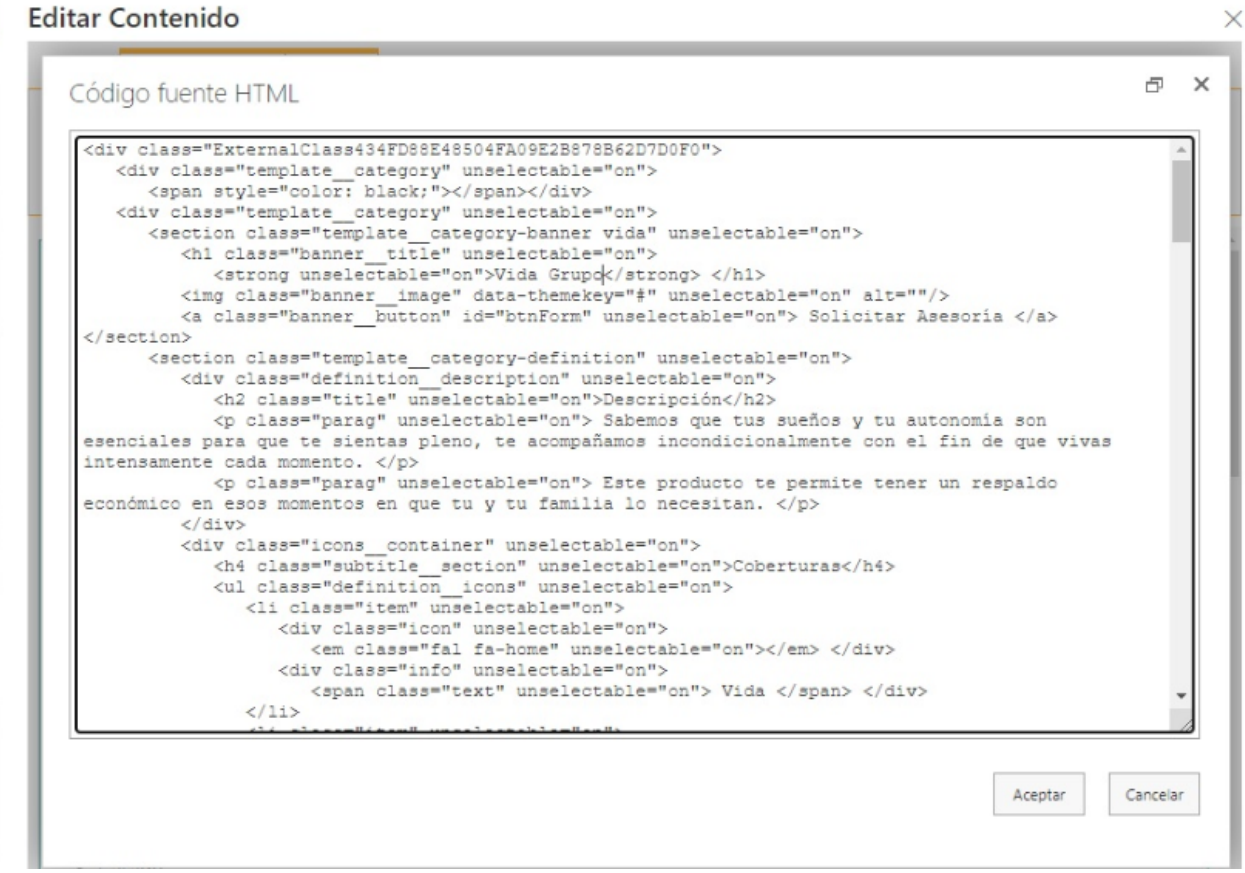


Fig. 5. Editor html de Sharepoint

Usando una herramienta de Azure llamada logic app , la cual es un servicio de integración en la nube proporcionado por Microsoft Azure, permite a los usuarios crear flujos de trabajo y procesos automatizados para integrar y coordinar la interacción entre diferentes aplicaciones, servicios y sistemas en la nube y en las instalaciones., se efectuaron algunas tareas relacionadas con dicha herramienta, por ejemplo crear flujos de automatización que se conectaran con sitios de sharepoint para actualizar datos, o notificaciones push via correo electrónico. Algunas características de las logic apps son:

Conectividad y compatibilidad: Admiten una amplia gama de conectores y protocolos para facilitar la integración con servicios locales o en la nube. Además, ofrecen soporte para estándares como REST, SOAP, FTP, entre otros.

Orquestación de servicios y aplicaciones: Permiten coordinar y orquestar la interacción entre diferentes servicios y aplicaciones mediante la definición de pasos y acciones secuenciales.

Monitoreo y administración: Proporcionan capacidades de monitoreo y registro de actividad para rastrear el flujo de trabajo y detectar posibles problemas o errores. También ofrecen opciones de administración, como escalado y administración de versiones.

En la figura 6 se observa el editor de flujos logics app desde el portal Azure:

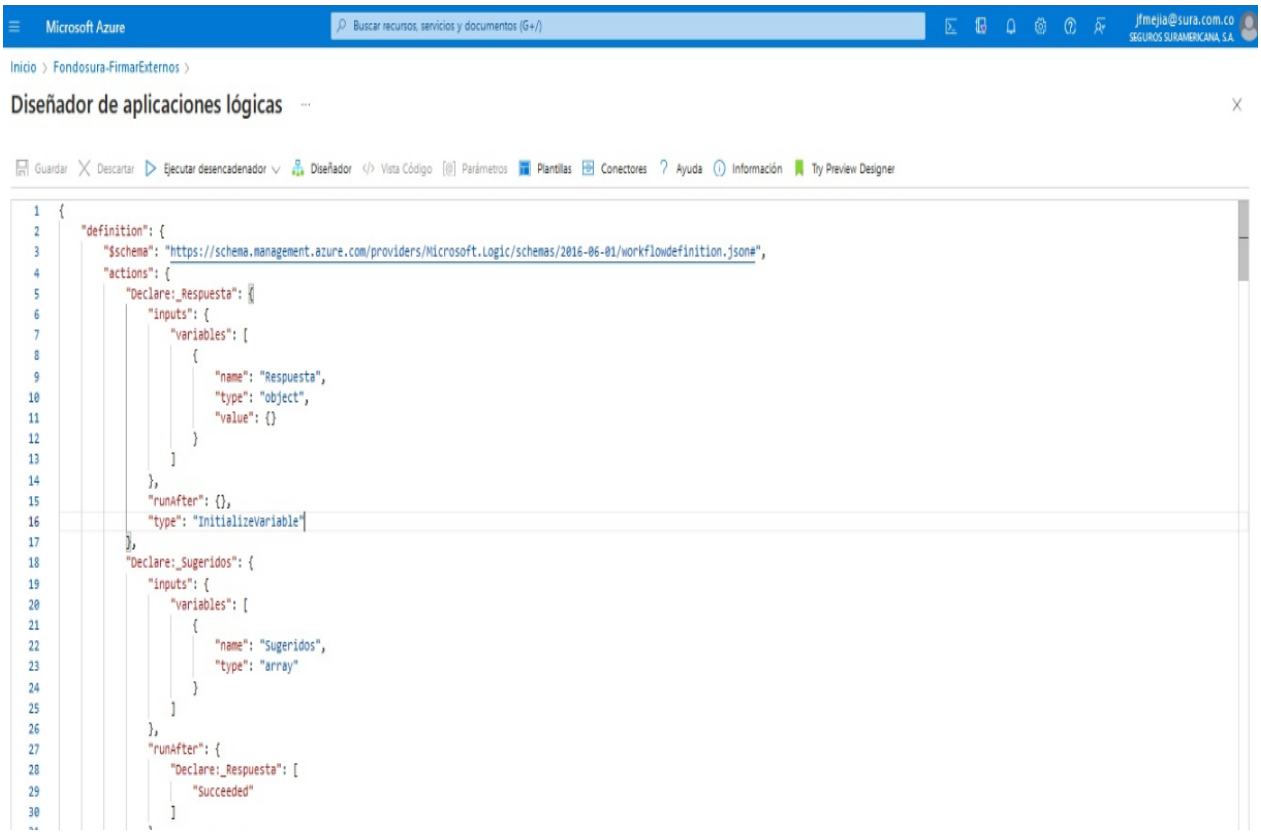


Fig. 6. Editor logic apps Azure

# 5 CONCLUSIONES

- Se puede concluir la importancia del marco de trabajo DevOps en una compañía, porque mejora la velocidad de entrega de software,esto con el uso de herramientas de automatización y pruebas, se puede acelerar el proceso de desarrollo y despliegue, lo que puede permitir que la empresa entregue productos y servicios más rápido, también se disminuye los errores el uso de herramientas DevOps puede ayudar a identificar y solucionar errores de forma más rápida y eficiente, lo que puede disminuir el tiempo de inactividad y los costos asociados con errores.

- En vista del desarrollo de templates para la ejecución de los distintos pipelines, se concluye que el uso de templates predefinidos puede ayudar a los equipos de desarrollo a ahorrar tiempo al no tener que crear pipelines desde cero. Esto también puede reducir el riesgo de errores al estandarizar el proceso de construcción y despliegue, también estos templates pueden ayudar a mantener la consistencia entre los diferentes proyectos y equipos de desarrollo, lo que puede mejorar la calidad del código y la experiencia del usuario final.

  En general, el uso de templates de pipelines en Azure DevOps puede ser beneficioso para los equipos de desarrollo al permitirles ahorrar tiempo, mejorar la consistencia, fomentar la colaboración, personalizar el proceso de construcción y despliegue y aumentar la eficiencia.

- También se concluye en la eficacia de trabajar con la herramienta sharepoint, esto es muy beneficioso para la compañía , porque todo el ecosistema de Microsoft hace parte de su stack tecnológico.

  En cuanto al desarrollo de sitios web de la compañía, usando sharepoint se reducen los tiempos de entrega y la mantenibilidad de dichos sistemas

se hace de una manera más transparente que usando otras tecnologías. Una de las ventajas más importantes de SharePoint es la capacidad de almacenar y compartir información de manera centralizada y accesible desde cualquier lugar y dispositivo. Esto ha permitido a los empleados acceder rápidamente a documentos y recursos importantes, reduciendo el tiempo dedicado a buscar información.

## REFERENCIAS

[1] Proyectos Ágiles. (s.f.), de https://proyectosagiles.org/beneficios-de-scrum/

[2] Ámbito. (2022, 28 de noviembre). ¿Qué es la cultura DevOps y para qué sirve a las empresas? https://www.ambito.com/opiniones/tecnologia/que-es-laculturadevops-y-que-le-sirve-las-empresas-n5138878

[3] CLAIRE, D. (16 de abril de 2011). ¿Qué es scrum? Obtenido de Atlassian: https://www.atlassian.com/es/agile/scrum

[4] GRUPO EMPRESARIAL SURAMERICANA DE SEGUROS SAS . (18 de septiembre de 2016). GRUPO SURA. Obtenido de GRUPO SURA: https://www.gruposura.com/nuestra-compania/sobre-grupo-sura/