DOI: 10.15649/2346030X.3011
Aibi revista de investigación, administración e ingeniería. Volumen 11, Número 1, Pág 20-38, Enero - Abril de 2023, ISSN 2346-030X

Entorno experimental de procesamiento de datos distribuidos integrando devops en el ciclo de entrega de software.
Experimental environment of distributed data processing integrating devops in the software delivery cycle.

Charlie Alberto Angulo-Angulo[1]
[1]*Escuela Colombiana de Ingeniería Julio Garavito, Bogotá - Colombia*
ORCID: [1]*0009-0009-1707-5811*

Recibido: 15 de septiembre de 2022.
Aceptado: 13 de diciembre de 2022.
Publicado: 01 de enero de 2023.

Resumen- En este artículo de investigación se muestra la implementación y puesta en marcha de una arquitectura orientada a la experimentación y práctica en procesamiento de datos distribuido con la opción de integrar un flujo de entrega continua de software con devops; mediante un despliegue automatizado de infraestructura como código en la nube. Permitiendo obtener un entorno experimental para estudiantes interesados en asumir cargos laborales Cloud Engineering y Data Engineering, logrando minimizar el tiempo y el esfuerzo que se consume en la configuración del clúster, promoviendo el aprendizaje práctico y experimental de tecnologías demandadas en el mercado.
Palabras clave: data processing, big data devOps, hadoop, spark, data, cloud. aws, cluster, infrastructure as code**.**

Abstract— This research article shows the implementation and commissioning of an architecture oriented to experimentation and practice in distributed data processing with the option of integrating a continuous software delivery flow with devops; through an automated deployment of infrastructure as code in the cloud. Allowing to obtain an experimental environment for students interested in assuming Cloud Engineering and Data Engineering job positions, minimizing the time and effort consumed in the configuration of the cluster, promoting practical and experimental learning of technologies demanded in the market.
Keywords: data processing, big data devOps, hadoop, spark, data, cloud. aws, cluster, infrastructure as code.

I. INTRODUCCIÓN

*Autor para correspondencia.
Correo electrónico: Charlie.angulo@mail.escuelaing.edu.co (Charlie Alberto Angulo Angulo).

Como citar este artículo: C. A. Angulo-Angulo, "Entorno experimental de procesamiento de datos distribuidos integrando devops en el ciclo de entrega de software", *Aibi revista de investigación, administración e ingeniería*, vol. 11, no. 1, pp. 20-38 2023, doi: 10.15649/2346030X.3011

En la actualidad se generan miles de millones de datos, Big data es el termino describe en la generación grandes volúmenes de datos permitiendo una gran cantidad, variedad, a una gran velocidad, en donde se logra identificar la veracidad y aportar valor a través de los mismos, por esta razón requieren nuevas formas de procesamiento para permitir una mejor toma de decisiones a las organizaciones, facilita el descubrimiento de información valiosa, logrando incluso a optimizar muchos procesos.

Gracias a la analítica de datos altamente avanzada, a menudo impulsada por técnicas de IA, podemos dar sentido y trabajar con flujos de datos enormemente complejos y variados. [1] A medida que los datos se trasladan al ecosistema de la nube, los datos y la analítica componibles son cada vez más relevantes y el enfoque más preferido a la hora de crear aplicaciones analíticas.

Cada año seguimos viendo cómo la industria evoluciona y acelera la capacidad de ofrecer software con más velocidad y mejor estabilidad. Las organizaciones que se someten a una transformación DevOps mediante la adopción de la entrega continua son más propensas a tener procesos de alta calidad, bajo riesgo y rentables. [2].

Ahora bien, la introducción de la computación en la nube trajo consigo una serie de nuevos retos de ingeniería. Los desarrolladores de software que se enfrentan a con estos retos tienden a converger en soluciones similares basadas en buenas prácticas de diseño. Uno de los patrones y prácticas más se utilizada en la nube actualmente es la infraestructura como código. [3].

Por ello, las empresas necesitan especialistas en estas tecnologías y un tema importante es el cómo los estudiantes que quieren enfocar su carrera al desarrollo en cargos como ingenieros en la nube e ingenieros de datos pueden adquirir estas habilidades de manera práctica y experimental.

En esta tesis se propone un ambiente que permita evitar las configuraciones extensas, sacando provecho e integrando los mejores beneficios que ofrece en una plataforma de procesamiento de datos distribuidos en un entorno cloud mediante un despliegue de infraestructura como código, manipulando herramientas dominantes en el mercado y utilizadas en los ambientes laborales. De esta forma los desarrolladores cloud tendrán una experiencia práctica a la hora de interactuar en el mundo de los datos, la nube, infraestructura como código y algunas herramientas utilizadas en devops, con un conocimiento mucho más sólido, enfocados en desarrollo y no en las configuraciones complejas, obteniendo así habilidades demandados laboralmente en el mercado y viviendo un símil de lo se encontrarán en un entorno laboral.

II. METODOLOGÍA

Se estructuro el artículo de la siguiente manera:

- Fase 1: Análisis:
  En la fase de análisis el objetivo es revisar la literatura mediante una búsqueda sistémica.

- Fase 2: Planeación y Diseño:
  En esta fase se realizará el diseño de la arquitectura de referencia para el procesamiento de datos distribuidos integrada con herramientas Devops mediante IaC.

- Fase 3: Configuración:
  En la fase de ejecución se implementará la arquitectura propuesta.

- Fase 4: Ejecución y Diagnóstico:
  Se colocará a prueba el despliegue de la infraestructura en la nube con todas las configuraciones de las herramientas a utilizar.

- Fase 5: Pruebas y Análisis de resultados:
  Se probará el entorno completo implementando un caso de uso, en búsqueda de probar el ambiente, recolectando un set de datos, con las métricas del resultado de las ejecuciones.

- Fase 6: Cierre:
  Conclusiones, trabajos futuros.

III. MARCO TEÓRICO

A continuación, se presentan varios términos y principales características utilizados procesamiento de datos distribuidos, Devops y la nube, con el fin de tener claridad en el momento de abordar los temas propuestos.

a. Cloud Engenieering

Un ingeniero de la nube es un profesional de TI responsable de cualquier tarea tecnológica asociada a la computación en nube, incluidos el diseño, la planificación, la gestión, el mantenimiento y el soporte. [4].

b. Data Engeniering

Los ingenieros de datos son una parte fundamental en cualquier proceso de ciencia de datos. Son perfiles muy demandados en cualquier entorno donde se manejen datos. Una data engineer es aquel profesional enfocado en el diseño, desarrollo y mantenimiento de los sistemas de procesamiento de datos dentro de un proyecto de Big data. [5].

c. Teorema CAP

El teorema CAP o teorema Brewer, dice que en sistemas distribuidos es imposible garantizar a la vez: (consistencia): al realizar una consulta o inserción siempre se tiene que recibir la misma información, (disponibilidad) todos los clientes puedan leer y escribir, aunque se haya caído uno de los nodos y (tolerancia a particiones) el sistema tiene que seguir funcionando, aunque existan fallos (ConsistencyAvailability-Partition Tolerance). [6].

d. Big data

Es la agrupación de múltiples tendencias tecnológicas, el termino describe grandes volúmenes de datos generados de diferentes fuentes de información o un grupo de registros de datos muy grandes y complejos que resultan difíciles de procesar utilizando las aplicaciones tradicionales de procesamiento de datos o las herramientas de los sistemas de bases de datos disponibles. [7].

Hace referencia a un sistema informático capaz de buscar, capturar, procesar, visualizar y acumular grandes cantidades de datos digitales. El objetivo central del *Big data* es analizar los datos capturados con el fin de identificar patrones o tendencias al interior de los fenómenos que se están analizando. [8].

e. Visualización

Es la forma de como mostrar y comunicar los datos de forma eficaz en diferentes entornos se ha convertido cada vez más en un importante contenido de investigación el uso de métodos de visualización como gráficos, imágenes y animaciones.[9]

f. Arquitectura de software

La arquitectura de las aplicaciones de *software* es el proceso de definición de una solución estructurada que satisface todos los requisitos técnicos y operativos, al tiempo que optimiza atributos de calidad comunes como el rendimiento, la seguridad y la capacidad de gestión. [10].

La arquitectura de aplicaciones trata de tender un puente entre los requisitos empresariales y los técnicos mediante la comprensión de los casos de uso y la búsqueda de formas de implementar esos casos de uso en el software. [11].

g. Computación en la nube

Es un término usado para referirse a un modelo de infraestructura en el cual un usuario puede acceder a un servicio o aplicación bajo demanda y de forma remota. [12][13].

h. Modelos de despliegue [12][13]

1. Nube pública:

La infraestructura de la nube se pone a disposición del público en general o de un gran grupo industrial y es propiedad de una organización que vende servicios en la nube.

2. Nube privada:

La infraestructura de la nube se opera únicamente para una organización. Puede ser administrado por la organización o un tercero y puede existir dentro o fuera de las instalaciones.

3. Nube híbrida:

Es la combinación de uno o más entornos de nube pública y privada, de tal forma que las organizaciones empresariales se benefician de las ventajas que proporcionan los dos tipos de infraestructura *cloud*. Así, se dispone de un conjunto de recursos virtuales gestionados por software de administración y automatización que permite a los usuarios acceder a lo que necesiten.

i. Modelos de servicios [12][14]

4. Plataforma como servicio (PaaS)

Ofrece una solución completa para la construcción y puesta en marcha de aplicaciones y servicios *Web* que estarán completamente disponibles a través de Internet. Algunos ejemplos: *Google App Engine*.

5. Software como servicio (SaaS)

Consiste en la distribución de software donde una empresa proporciona el mantenimiento, soporte y operación que usará el cliente durante el tiempo que haya contratado el servicio. Ejemplos: Gmail.

6. Infraestructura como servicio (IaaS)

Proporciona al cliente una infraestructura de computación como un servicio, usando principalmente la virtualización. El cliente compra recursos a un proveedor externo, para hosting, capacidad de cómputo, mantenimiento y gestión de redes, etc. Ejemplos: *Amazon EC2*.

j. Infraestructura como código(IasC)

La infraestructura como código (*IaC* del inglés *Infrastructure as Code*) permite gestionar y preparar la infraestructura a través del código, en lugar de hacerlo mediante procesos manuales.

Con este tipo de infraestructura, se crean archivos de configuración que contienen las especificaciones que esta necesita, lo cual facilita la edición y la distribución de las configuraciones. Asimismo, garantiza que usted siempre prepare el mismo entorno. La infraestructura como código codifica y documenta sus especificaciones para facilitar la gestión de la configuración, y le ayuda a evitar los cambios *ad hoc* y no documentados. [16].

Los desarrolladores de software que se enfrentan a estos retos tienden a converger en soluciones similares basadas en buenas prácticas de diseño. Este patrón es de los más utilizados en la nube como podemos observar en la Figura 1, donde se representa el lenguaje de patrones para el software de ingeniería para la nube, mostrando las relaciones entre los patrones (flechas) y las categorías en las que se encuadran, azul Descubrimiento y comunicación, amarillo orquestación y supervisión, verde supervisión y café gestión automatizada de la infraestructura. [18].

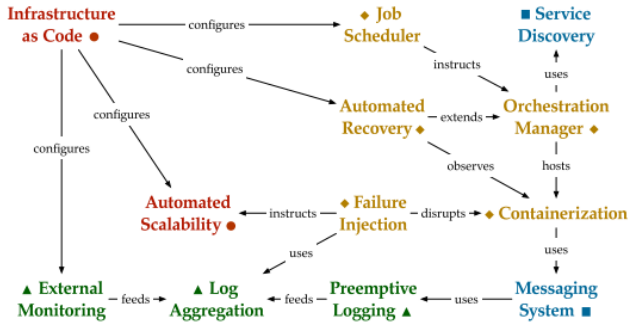

Figura 1: Lenguaje de patrones para el software de ingeniería para la nube. [18].
Fuente: Elaboración propia.

k. *Devops*

Significa un cambio de cultura hacia la colaboración entre el desarrollo, el control de calidad y las operaciones. El objetivo es integrar mejor los procesos de desarrollo, producción y los procesos de negocio de las operaciones con la tecnología adecuada. [15].

l. Entrega continua (Continuous Delivery):

Trata de optimizar la gestión de la infraestructura y la necesidad crítica de equilibrar el tiempo y los recursos. Permitiendo en los equipos de desarrollo realizar entregas de software en ciclos de vida cortos, para así generar valor en entregas de mínimo productos viables (MVP), liberando software en cualquier momento de forma confiable.

IV. ESTADO DEL ARTE

a. *Entornos* de procesamiento de datos distribuidos para experimentación

Existen algunos ambientes con implementaciones de arquitecturas lideradas por big data similares pero no a plenitud como se quiere proponer, hay muchos acercamientos con respecto a componentes que se quieren trabajar en este trabajo, por ejemplo con spark tenemos un esquema de utilización de recursos adaptable para cargas de trabajo de big data en entornos de contenedores aprovechando la elasticidad vertical de Docker, realizando experimentos con apache spark y hadoop en la nube.(Choi, Cho, and Kim 2021), llevando a cabo la implementación de un clúster en máquinas virtuales como en docker con un único nodo, cada máquina virtual tiene su propio sistema operativo, así como su propio espacio de trabajo de procesamiento de datos Spark para gestionar los archivos de ejecución. Pero está enfocada a investigar la diferencia de rendimiento de ambas implementaciones.

b. Ambientes experimentales

Se encuentra s una plataforma big data, basada en contenedores docker, utilizando herramientas como redis y nginx.[19] Proponiendo establecer una arquitectura avanzada en la nube, que puede construirse utilizando servidores recién adquiridos o servidores propios de la universidad. La plataforma gestiona el clúster de servidores y divide los recursos mediante tecnología de virtualización, carga las imágenes experimentales de big data en la máquina virtual dividida y sirve como entorno del sistema experimental para los estudiantes.

También existe una plataforma de aplicación de ciencia de datos y análisis de big data basada en la arquitectura de microservicios para la educación en el campo de la investigación no profesional, utilizando Docker, JupyterHub, spark, hdfs, y spring boot para la parte web.[20]Ayuda a compartir datos, potencia de cálculo y recursos de infraestructura. Además, permite a los usuarios utilizar un entorno más amigable para registrar y compartir el proceso experimental y el modelo visualizado. Utiliza JupyterHub y el modelado visualizado como sus dos aplicaciones principales. La plataforma utiliza micro-servicios como dependencia técnica; en particular, utiliza Docker. El sistema de archivos distribuidos del entorno utiliza HDFS (Hadoop Distributed File System. La arquitectura de computación paralela utiliza Apache Spark, se utiliza Jupyter

Notebook como entorno de ingeniería de código del usuario. Y JupyterHub gestiona un entorno de cuaderno multi usuario de forma unificada. Esta solución se acerca en gran medida a lo que se quiere lograr, pero se realizando la construcción y el despliegue mediante infraestructura como código.

c. Infraestructura como código

Existe el desarrollo de un entorno que despliega un clúster utilizando hadoop ansible y terraform, reduciendo tiempo y esfuerzo de implementación para el almacenamiento y procesamiento de datos distribuidos. Utilizando terraform como aprovisionamiento y ansible, para la gestión de configuraciones. Todo automatizado sin intervención humana, configurando un clúster con hadoop, sobre un servidor en la nube, permitiendo conectarse a maquinas Linux y Windows, mediante conexión ssh.[21].

d. Devops

Se ha construido un framework con continius delivery para data science, utilizando algoritmos de reconocimiento fácil en un marco de prácticas devops.[22] Desarrollando un canal de entrega para un sistema de aprendizaje automático mediante la extracción de información y reconocimiento a partir de datos, aplicando cuatro prácticas de devops en un sistema de aprendizaje automático, estas prácticas son el control de versiones, el servidor de modelos, la contenerización y CI/CD.

En la misma línea se ha utilizado herramientas para una solución de gestión de información educativa, utilizando herramientas como Jenkins Kubernetes y git.[23]. Para mejorar la calidad de la gestión de los estudiantes, desarrollado un nuevo sistema de información educativa basado en el concepto de devops en in sistema de integración continúa basado en Git, despliegue continúo basado en Jenkins y Kubernetes, gestión de logs basado en ELK y Sistema de análisis de calidad de código basado en SonarQube.

V. OBJETIVOS PROPUESTO

a. Propósito

El propósito y el aporte de este artículo es lograr integrar un clúster para análisis, procesamiento y visualización de datos distribuidos mediante un despliegue de infraestructura como código enfocado a la práctica y experimentación, integrando la opción de devops en el ciclo de entrega continua, con una configuración end to end desde ceros y en lenguaje utilizando un lenguaje de configuración de alto nivel llamado HCL (HashiCorp Configuration Language).

Si bien algunos desarrollos de los componentes ya existen, trataremos de colocar un grado mayor de dificultad en la arquitectura propuesta, cuanto la configuración por ejemplo en la cantidad de nodos, notebook apuntando al clúster como interfaz de usuario, sonarCloud y git actions como herramientas para el flujo de trabajo devops, realizando toda la configuración con bash, y HCL como ilustra la figura 1. Este desarrollo se disponibilizara en el repositorio de código git, para que los usuarios lo descarguen en sus equipos físicos y puedan ejecutar el entorno con comandos terraform.

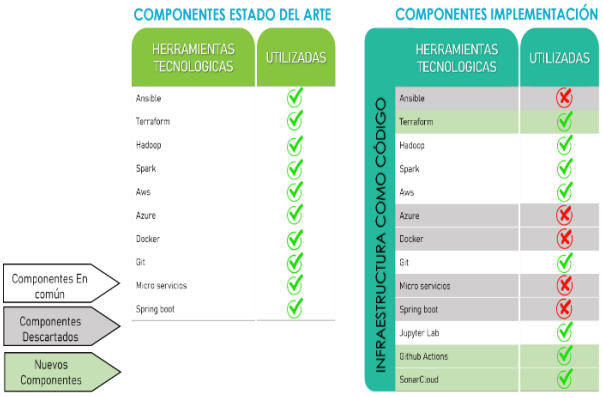

Figura 2: Integración de componentes en arquitectura bajo despliegue de IaC.
Fuente: Elaboración propia.

b. Alcance

La solución planteada está enfocada a los estudiantes interesados en asumir cargos laborales Cloud Engineering y Data Engineering, minimizando el tiempo y el esfuerzo que se consume en la configuración del clúster para el almacenamiento y procesamiento de datos distribuidos. Promoviendo el aprendizaje práctico y experimental de tecnologías demandadas en el mercado. Se propone realizar la configuración e implementación y el aprovisionamiento de un clúster de procesamiento, almacenamiento, análisis, y visualización de información distribuida mediante un despliegue de infraestructura como código en la nube (IaC), integrando herramientas líderes en el mercado para el procesamiento de datos y la opción de incluir herramientas devops, para que los estudiantes interactúen con estas herramientas en un entorno que desencadene un conjunto de acciones de validaciones de código, cobertura, luego de que los estudiantes elijan la opción de versionar el código, en la búsqueda de ofrecerle experimentar un símil de lo que se encontraran, cuando realicen desarrollos en espacios colaborativos laboralmente.

FASE 1: ANÁLISIS DE ARQUITECTURAS DE REFERENCIAS BIG DATA

Se seleccionan 6 de las arquitecturas de referencia más interesantes que tomamos para elegir u diseñar la arquitectura a plantear mediante una búsqueda sistémica en repositorios académicos confiables (*Scopus*, *ieeexplore*, *google scholar*).

Como se observó en el estado del arte, las arquitecturas big data son influyentes en temas de procesamientos de datos distribuidos, por lo que se toman como base para poder analizarlas y diseñar nuestra arquitectura.

Las particularidades de los datos involucran un desafío para las organizaciones, debido que se requieren gestionar datos con sus variantes características, de este modo se requiere arquitecturas específicas que posean componentes que almacenen, procesen, analicen y visualicen un gran volumen de variedad de datos.[24].

1. Arquitectura Lambda

La arquitectura *Lambda* tiene tres capas: *batch*, *speed* y *serving*. La capa de lotes gestiona los datos históricos y vuelve a calcular los resultados (aprendizaje automático). itera a través de todos los datos y tiene una alta latencia. La capa de velocidad procesa los datos entrantes y proporciona resultados en tiempo casi real. Los resultados no son tan Los resultados no son tan precisos como los de la capa de procesamiento por lotes. La capa de servicio obtiene los resultados de las otras dos capas y permite las consultas.

Aunque la arquitectura *Lambda* ofrece una gran flexibilidad, tiene un precio. La complejidad y la necesidad de mantener dos bases de código diferentes para el procesamiento de datos hace que sea difícil de implementar y soportar. [10].

2. Arquitectura Kappa

La arquitectura Kappa, mostrada en la Fig.3, trata de simplificar la arquitectura *Lambda*. Combina las capas *batch* y capas de velocidad en una sola y tiene sólo dos capas: procesamiento de flujos y servicio. La capa de procesamiento de flujos ejecuta trabajos de procesamiento. Dependiendo de los datos que necesitemos procesar se ejecutan diferentes trabajos (datos en tiempo real, datos históricos). La capa de servicio se utiliza para consultar los resultados como en la arquitectura *Lambda*. Kappa cambia la flexibilidad por la simplicidad. [10].

3. Arquitectura big data centro hospitalario de Portugal

La arquitectura de Big Data, en tiempo real, establecida e implementada por *Gonçalves* y otros en la unidad de cuidados intensito del Centro Hospitalario de Porto en Portugal, establece una solución utilizando herramientas de código abierto como lo es el clúster de apache hadoop,en un entorno online para la extracción, el gobierno, análisis, almacenamiento y procesamiento de la información, La arquitectura de *Big Data* tiene para los agentes Adquisición de Datos, Inferencia, Interfaz y Gestión del Conocimiento. El agente hace que el sistema funcione mediante acciones automáticas, que realizan algunas tareas esenciales, como la recogida automática de datos y la actualización de los modelos predictivos, en tiempo real, sin necesidad de intervención humana. [25].

4. Procesamiento de Big Data basado en contenedores Docker en múltiples nubes

El sistema explora otra dimensión potencial para el análisis basado en contenedores *Docker* en un marco económico y fácil de usar, obteniendo conocimientos y habilidades básicas de TI. Además, se puede desarrollar fácilmente en una sola máquina, en varias máquinas o en varias nubes.[26].

5. *Aplicación de Big Data mediante un marco virtualización ligera en la Nube*

Nos plantea una arquitectura implementada constantemente en los entornos empresariales de *big data* en la nube, como *MapReduce* y sus implementaciones como *Hadoop y Spark* proporcionando una interfaz de programación de alto nivel productiva para el procesamiento de datos a gran escala y la analítica. *MapReduce* es un marco de software que permite a un clúster de ordenadores procesar un gran conjunto de datos en paralelo. Esto en un marco de trabajo de maestro en virtualizaciones basadas en hipervisores virtuales y *Docker* como contenidización. [27].

6. Despliegue de una arquitectura utilizando hadoop. ansible y terraform

La configuración del *clúster* consume mayor tiempo y esfuerzo de implementación para el almacenamiento y procesamiento de datos distribuidos. Una alternativa para automatizar este proceso se puede realizar mediante *terraform* como aprovisionamiento y *Ansible*, para la gestión de configuraciones. Todo automatizado sin intervención humana, configurando un clúster con *hadoop*, sobre un servidor en la nube, permitiendo conectarse a maquinas Linux y Windows, mediante conexión ssh.[21].

FASE 2: PLANEACIÓN Y DISEÑO

En la revisión de las arquitecturas anteriores nos permitió identificar e integrar diferentes capas en el ciclo de vida de los datos, algunas herramientas de código libre disruptivas y que prevalecen en el tiempo y con ambientes de trabajo *on premise* como en la nube, y sin duda nos da idea de cómo diseñar la arquitectura para el entorno experimental que buscamos, además se observó en las mayorías de arquitecturas de referencia definen 4 capas las cuales aplicaremos para la arquitectura agregando una 1 capa más adicional para la parte de visualización de datos.:

1. Fuentes de datos

Esta capa permite acceder a los orígenes o fuentes de los datos recolectarlos. Las fuentes pueden ser: bases de datos internas o externas, documentos, imágenes u otros.[25].

2. Carga de datos

En este punto se cargan los datos y se debe aplicar el concepto de metadatos, es decir, se deben describir las características de los datos que se cargan, como el nombre, la importancia y la relación con otros datos u objetos de la empresa. Uno de los objetivos de esta etapa es obtener una estructura homogénea que ayude a que los datos sean trazables y fáciles de acceder.[28].

3. Transformación y almacenamiento

Durante esta capa los datos se transforman mediante la aplicación de reglas de negocio y el procesamiento de los datos y luego se almacenan.

4. Analítica

Esta capa es responsable de procesar todo tipo de datos y realizar los análisis requeridos por la organización, el cual puede ser descriptivo, predictivo o prescriptivo.

5. Visualización

En este punto se generan los resultados con informes de visualización y monitoreo de información. La generación de informes y tableros de control es una función crítica en las arquitecturas de *Big Data*, ya que esta capa debe permitir visualizar información útil.

6. Diseño Arquitectura de referencia definida

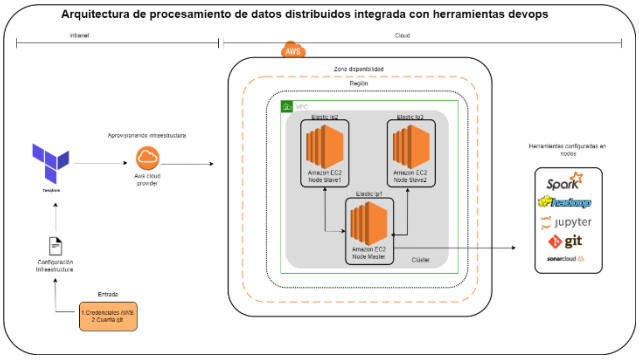

Figura 3: Arquitectura de procesamiento de datos distribuidos definida integrada con herramientas *Devops*.
Fuente: Elaboración propia.

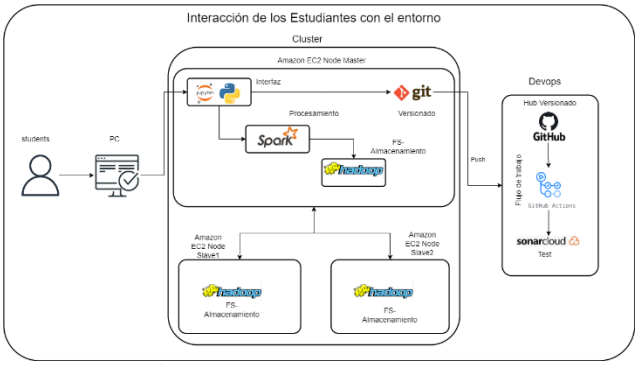

Figura 4: Interacción de los estudiantes con la arquitectura.
Fuente: Elaboración propia.

7. Descripción de Tecnologías o herramientas a utilizar

En la implementación del entorno de procesamiento de datos distribuido se utilizarán diferentes tecnologías que trabajan entre sí, para lograr el objetivo de almacenar, procesar, analizar y visualizar datos.

7.1. Terraform (Infraestructura como código)

Es una herramienta de software de infraestructura como código abierto, que proporciona un flujo de trabajo CLI coherente para gestionar cientos de servicios en la nube. *Terraform* codifica las API de la nube en archivos de configuración declarativos. Permite expresar la infraestructura como código en un lenguaje sencillo y legible para el ser humano llamado HCL (*HashiCorp Configuration Language*). Lee los archivos de configuración y proporciona un plan de ejecución de los cambios, que puede ser revisado para la seguridad y, a continuación, aplicado y aprovisionado.[29] Esta herramienta permitirá realizar el despliegue de la infraestructura en Aws(*Amazon Web Services*).

7.2. Amazon Web Services (AWS):

Es la plataforma en la nube ofrece funciones sólidas, como entrega de contenidos, potencia computacional, funciones de redes y administración de bases de datos con más 200 servicios. Esta plataforma nos facilitara desplegar mediante instancias Amazon EC2 linux, el clúster que utilizaremos para nuestra arquitectura en la nube. [30].

7.3. Apache Hadoop:

Es una plataforma que permite el procesamiento de grandes volúmenes de datos a través de clúster, usando un modelo simple de programación. Proporciona un *framework*, escrito en Java, sobre el cual desarrollar aplicaciones distribuidas que requieren un uso intensivo de datos y de alta escalabilidad. Este proyecto es administrado por Apache Software *Foundation*. (Jasim Hadi et al., 2015). En el caso de la arquitectura propuesta se utilizará el *FileSystem* de *hadoop* como almacenamiento de datos. El sistema de archivos distribuidos *hadoop* (HDFS) es un sistema de archivos distribuidos diseñado para funcionar en hardware básico. Tiene muchas similitudes con los sistemas de archivos distribuidos existentes. Sin embargo, las diferencias con otros sistemas de archivos distribuidos son significativas. Es altamente tolerante a fallos y está diseñado para ser desplegado en hardware de bajo coste. HDFS proporciona un acceso de alto rendimiento a los datos de la aplicación y es adecuado para aplicaciones que tienen grandes conjuntos de datos. [31].

7.4. Apache Spark (Plataforma Big data)

Spark es un *framework* de código abierto para computación en clúster. Desarrollado para el rendimiento, puede resultar hasta 100x veces más rápido que *Hadoop* para el proceso de grandes cantidades de información. Proporciona *API´s* de alto nivel con más de 100 operadores para Java, *Python*, *Scala* y R, además de un motor optimizado. [32] *Spark* tiene una mayor velocidad de procesamiento gracias a la computación en memoria. Adicional se puede resolver muchas tareas y problemas importantes en el campo de los grandes datos, incluyendo el procesamiento por lotes fuera de línea, la consulta interactiva, la computación de flujo en tiempo real y el aprendizaje automático. [9].

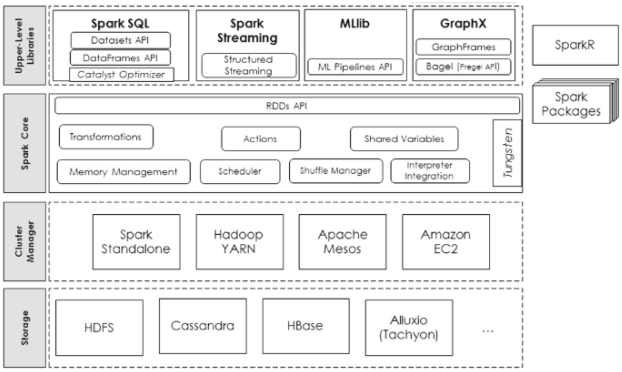

Figura 5: Arquitectura de alto nivel *Apache Spark*.[33].
Fuente: Elaboración propia.

La forma en la que utilizaremos *apache Spark* con *Hadoop* será mediante un gestor de clústeres se utilizará para adquirir recursos de clúster para ejecutar trabajos. El núcleo de *Spark* se ejecuta en el gestor de clústeres *Hadoop* YARN ubicado en instancias Amazon EC2 y el gestor de clústeres integrado (es decir, independiente). El gestor de clústeres se encarga de compartir los recursos entre las aplicaciones de Spark y puede acceder a datos en HDFS mediante *Yarn*.

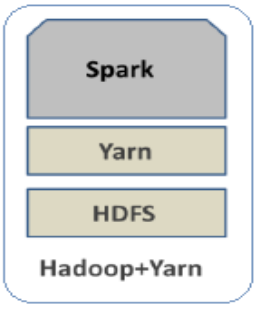

Figura 6: Despliegue de *Spark* en la arquitectura *hadoop*.
Fuente: Elaboración propia.

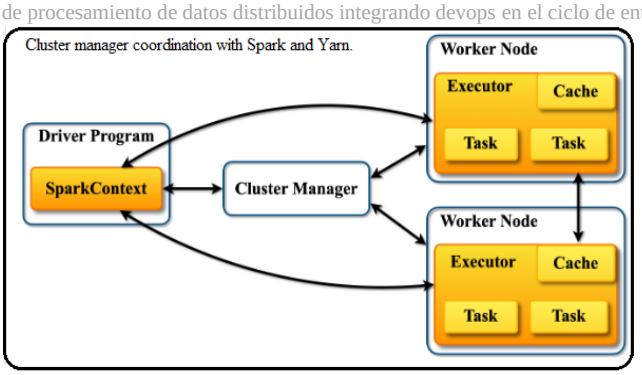

Figura 7: Funcionamiento Clúster Manager Spark y Yarn.[33].
Fuente: Elaboración propia.

7.5. JupyterHub

Una representación de todo el contenido visible en la aplicación web, incluidas las entradas y salidas de los cálculos, el texto explicativo, las matemáticas, las imágenes y las representaciones multimedia de los objetos.

Jupyter Notebook, que surge de *JupyterHub*, pro-vierte una interfaz programática interactiva de fácil uso, accesible desde el navegador, que alivia la carga de los estudiantes de implementar su trabajo en Scala o Python mediante engorrosos métodos de línea de comandos. [20] Utilizaremos *Jupyter Notebook* como interfaz de usuario para desarrollo y prácticas de código en el entorno y estará conectado con el clúster de *spark* y el *yarn*.

7.6. SonarQube(cloud)

Es una plataforma de código abierto para la gestión de la calidad del código, que se utiliza para gestionar la calidad del código fuente. A través del mecanismo de *plug-in*.[23].

7.7. Git

Sistema de control de versiones Git, para el desarrollo de código. Cuando se utiliza para el desarrollo local, los archivos se convierten entre tres estados: modificado, en fase y confirmado. Estos estados están relacionados con el área de trabajo, el área de preparación y el repositorio local que mantiene localmente. En primer lugar, después del usuario extrae el código del repositorio local, todas las modificaciones del código fuente se registran en el área de trabajo. En un periodo de tiempo, el código fuente actual modificado puede añadirse al área de almacenamiento temporal a través del comando *git add*. El usuario puede enviar los cambios en el área de almacenamiento temporal al repositorio local a través del comando *git commit*. Cada *commit* registrará la estructura de árbol actual correspondiente al directorio de código fuente, el archivo actual actualizado y metadatos como los comentarios y finalmente un *git push* para enviarlos a repositorio en la nube.[23].

Al realizar el *push Git* llamará ahora a la URL que desencadena la tarea de construcción y ejecución de *sonarQube Cloud* mediante la obtención el último código en la rama de distribución y utilizar las *GitHub Action* o flujos de trabajo de *git* para construir, compilar y finalmente probar los componentes de software.

FASE 3. EJECUCIÓN

1. Implantación del clúster realizado en terraform

Todas la paramétricas y configuraciones configuradas en terraform para que el ambiente se ejecute de manera automática con la acción anterior.

1.1. Configuración de recursos en Aws

Inicia con la creación y parametrización de los servicios a utilizar en Aws, para el caso del ambiente experimental, se han definido la parametrización de *Amazon Elastic Compute Cloud* (EC2) bajo una instancia *Linux ubuntu*, *Amazon Virtual Private Cloud* (VPC), una Subred, un *Gateway* de internet y Grupo de seguridad. Esto sin dejar atrás la generación previa de manera manual del *Personal Acces Token*, que permite crear todos estos recursos en Amazon, y el par de llaves para concedernos acceso a cada una de las instancias y realizar las respectivas configuraciones del *clúster*.

1.2. Configuraciones del clúster

Esta configuración empieza en la instancia del nodo master con la actualización del Linux, instalación del jdk 8, la descarga y parametrización de *Apache hadoop* y *Apache spark*, al igual que *Git* y la creación de algunas variables de entorno requeridas y luego se replica los mismos ajustes de *hadoop* para los 2 nodos esclavos.

1.3. Parametrización de flujos de trabajo en GitHub

Utilizando las variables de estrada requeridas para *git*, se da comienzo a la configuración de unas variables de entorno, se inicia el repositorio en una carpeta llamada(Versionado), creando la carpeta *workflows* y en ella un archivo de configuración. *yml*, el cual contiene la parametría de *sonarcloud* y la rama en la cual se realizarán las respectivas validaciones de código para *python*.

1.4 Formateo desde el nodo master

Se le aplica el comando *hdfs namenode –format*, para darle formato a cada uno de los nodos del clúster, con el fin de que tome las configuraciones realizadas en *hadoop*.

1.5. Iniciar sistema de archivos

En una terminal del nodo maestro se ejecutó la instrucción *start-dfs.sh* para que el nodo maestro inicie los nodos esclavos y ponga en marcha el clúster, de esta manera iniciará *NameNode* y *SecondaryNameNode* en *node-master*, y *DataNode* en nodo1 y nodo2, de acuerdo con la configuración del archivo workers.

1.6. Iniciar yarn

Luego de que los procesos del *dfs* estén corriendo se lanza el comando start-yarn.sh, iniciando ResourceManager en node-master y un NodeManager en node1 y node2.

1.7. Configuración Python y jupyter notebook con clúster

Instalar *Jupyterlab* implica una larga lista de dependencias de Python. Para evitar conflictos de dependencia, se instala en un entorno virtual llamado *venv*. Este directorio contiene un intérprete de Python, bibliotecas y scripts que están aislados del resto del sistema.

Se descarga e instala python3, luego Jupyterlab y Jupyter notebook, la instalación genera un archivo de configuración con muchas configuraciones importantes, como la contraseña de usuario codificada, los enlaces de IP y el control de acceso remoto, se modifica para preferencias del ambiente.

Finalmente se realiza la configuración e instalación del kernel de *py Pyspark 3*, recordar que los *kernels* son procesos específicos del lenguaje de programación que se ejecutan de forma independiente e interactúan con las aplicaciones de Jupyter y sus interfaces de usuario. Tendremos que agregar un *kernel* en el directorio dl-venv. En nuestro caso, crearemos uno personalizado, para direccionado al *clúster Spark* implementado.

2. Despliegue de infraestructura mediante terraform

Terraform aprovisionará infraestructura como código de las instancias EC2, Grupos de Seguridad, VPC en AWS. Cuenta con 4 comandos esenciales que nos permiten abordar un flujo de trabajo.

2.1. Terraform init

Permite inicializar *terraform* y descargar versiones tanto versiones como archivos del proveedor necesarios en este caso *Aws* como se ilustra en la imagen.

2.2. Terraform plan

Este comando permite crear el plan de ejecución de la infraestructura programada en cada uno de los archivos de configuración definidos.

2.3. Terraform apply

Se utiliza para aplicar los cambios necesarios para alcanzar el estado deseado de la configuración, o el conjunto predefinido de acciones generado por un plan ejecución. Se utilizó el comando apply -auto-approve, que no pide confirmación de la acción.

2.4. Terraform Destroy

Facilita el borrado de todos los recursos creados en AWS y de la infraestructura en si, por lo que se aplicara únicamente cuando no se desee utilizar el ambiente de experimentación.

3. Ejecución entorno jupyterLab

Para ingresar al *notebook*, hay que ingresar a la ip publica y puerto definido 4242(ippublica:4242) del nodo master para lograr interactuar con el notebook y el clúster, mediante la contraseña “1234” y este nos permitirá realizar almacenamiento, procesamiento, análisis y visualización de datos mediante *Python* y disponer del uso de la terminal para manipular el entorno.

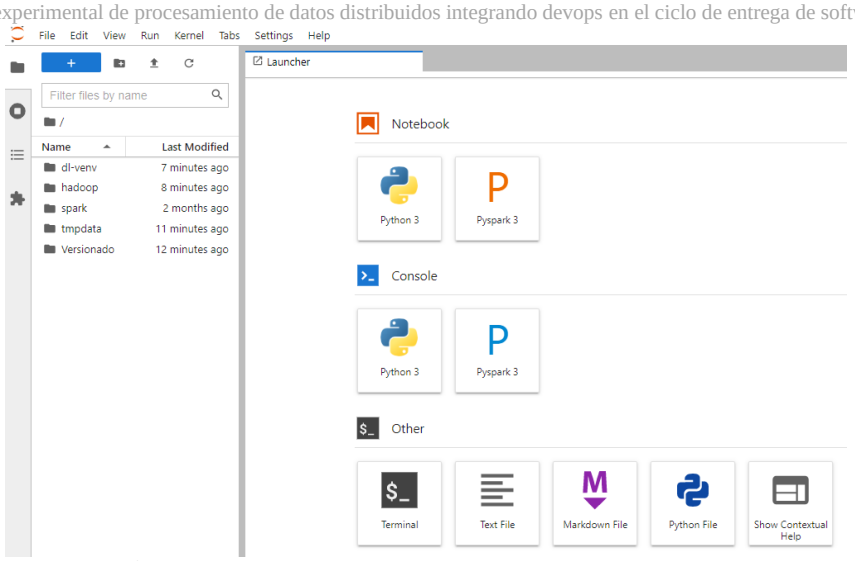

Figura 8: Interfaz Jupyter lab listo para experimentar.
Fuente: Elaboración propia.

VI. ANÁLISIS Y RESULTADOS

a. Desarrollo caso de uso

Para probar la infraestructura fue necesario buscar un caso de uso con la suficiente complejidad para realizar una prueba de todos los elementos de la misma, por lo que se eligió el covid 19. Se realizará un análisis de los casos confirmados en la ciudad de Bogotá, identificaremos información más detallada por localidad de la que nos ofrece la página de reporte de coronavirus del gobierno colombiano, limitando la muestra a 30.000 registros. Se toma este caso de uso por ser un caso real que impacto en la sociedad y cuyos datos yacen en un repositorio público, para ser analizados y verificar de cómo esta pandemia afectó a la ciudad de Bogotá DC.

b. Identificación de los orígenes de los datos

Para esta fase se usarán datos abiertos del gobierno nacional (Bogotá D.C), utilizando una muestra de la información de casos confirmados de *covid* 19.

1. Obtención de los datos

Luego de identificar la fuente de datos a analizar en el punto anterior se procede con la descarga de la información, la cual permite descargar 2 archivos, el primero para la descarga de la muestra correspondiente a todos los sets de datos a analizar de casos confirmados de *covid* 19 y el segundo la información de cada uno de los metadatos correspondiente a la descripción de las columnas e información que contendrá la muestra.

2. Almacenamiento de los datos:

Este proceso se efectúa a través del cargue de información que nos permite realizar *jupyter lab*, de manera gráfica permitiendo ubicando en nuestro pc local el archivo descargado; en este caso cargando el archivo “osb_enftransm-covid-19_18082022.csv”.

c. Implementación caso de uso

En esta sección se ejecuta un análisis de datos obtenidos para validar el funcionamiento entorno.

1. Importación librerías spark

Realizaremos la importación de librerías de *spark* y crearemos nuestra aplicación mediante los siguientes parámetros:

```
from pyspark.sql import SparkSession
from pyspark.sql import functions as F
import pandas as pd
spark = SparkSession. \
builder. \
appName('implementacion-caso-de-uso'). \
master('yarn'). \
getOrCreate()
```

2. Lectura archivo mediante spark

Ya situados en el notebook listo para implementar el caso de uso y con el archivo subido al *jupyter* realizaremos la lectura mediante el siguiente comando y realizaremos un conteo de registros, asignando a la variable data todo el *dataframe* que se crear al momento de la lectura del insumo.

```
data =spark.read.format("csv").option("header",
"true").load("file:///home/ubuntu/Versionado/osb_enftransm-
covid-19_18082023.csv")
```

3. Caso 1. Fallecidos y recuperados por género

```
    estadoGenero =
data.groupby("ESTADO","SEXO").count().orderBy("SEXO")
    estadoGenero.show(truncate=False)
    estadoGeneroPd = estadoGenero.toPandas()
    estadoGeneroPd.groupby(['ESTADO',
'SEXO'])['count'].mean().unstack(level=0).plot(figsize=(10,
10),title='Estado de contagio por sexo', kind = 'barh')
```

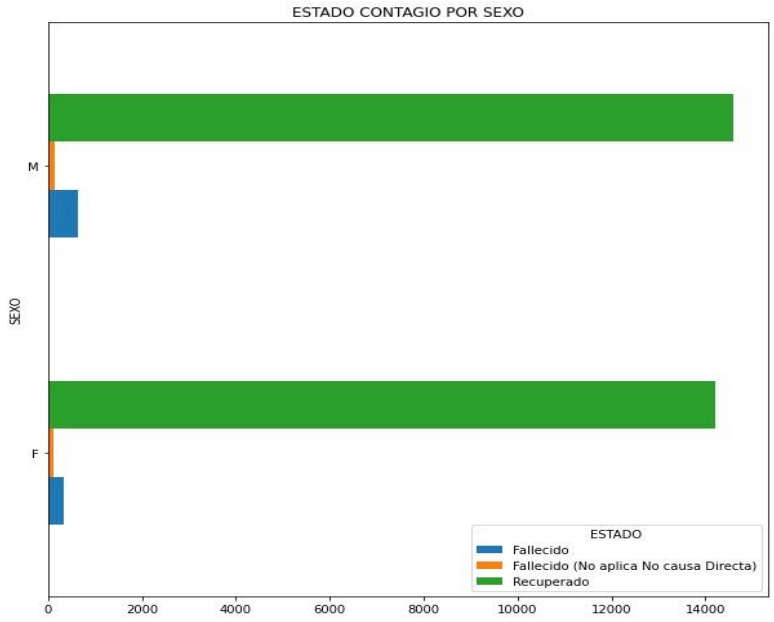

Figura 9: Representación gráfica estado contagios por sexo.
Fuente: Elaboración propia.

4. Caso 2. Fallecidos y recuperados por localidad por covid 19

```
    fallecidos =
data.groupby("LOCALIDAD_ASIS","ESTADO").count().orderBy("LOCALIDAD_ASIS")
    fallecidos.show(truncate=False)
    fallecidosPd = fallecidos.toPandas()
    fallecidosPd.groupby(['ESTADO',
'LOCALIDAD_ASIS'])['count'].mean().unstack(level=0).plot(figsize=(10,  10),title='Estado
contagio por localidad', kind = 'bar')
```

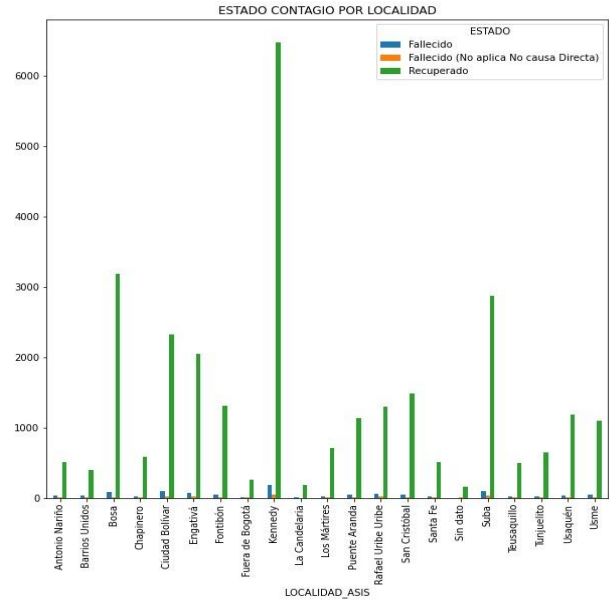

Figura 10: Representación gráfica estado contagios por localidad.
Fuente: Elaboración propia.

5. Caso 3. Fuente tipo de contagio por edad

```
tipoContagio = data.groupby("FUENTE_O_TIPO_DE_CONTAGIO",F.col("EDAD").cast("int").alias("EDAD")).count().orderBy("EDAD")
tipoContagio.show(truncate=False)
tipoContagioPd = tipoContagio.toPandas()
tipoContagioPd.groupby(['FUENTE_O_TIPO_DE_CONTAGIO', 'EDAD'])['count'].mean().unstack(level=0).plot(figsize=(10, 10),title='Fuete de contagio', kind = 'line')
```

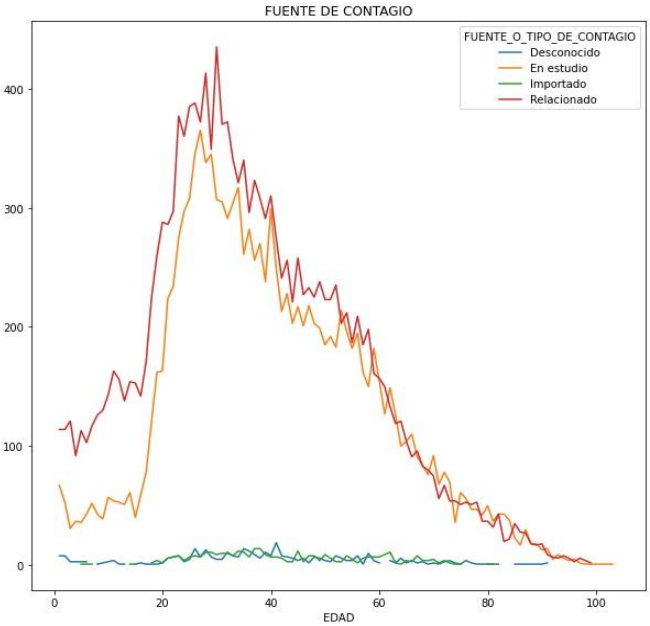

Figura 11: Representación gráfica fuente de contagio por edad.
Fuente: Elaboración propia.

6. Caso 4. Fallecidos y recuperados por edad

```
estado = data.groupby("ESTADO",F.col("EDAD").cast("int").alias("EDAD")).count().orderBy("EDAD")
estado.show(truncate=False)
estadoPd = estado.toPandas()
estadoPd.groupby(['ESTADO', 'EDAD'])['count'].mean().unstack(level=0).plot(figsize=(10, 10),title='Estado de contagio por edad', kind = 'area')
```

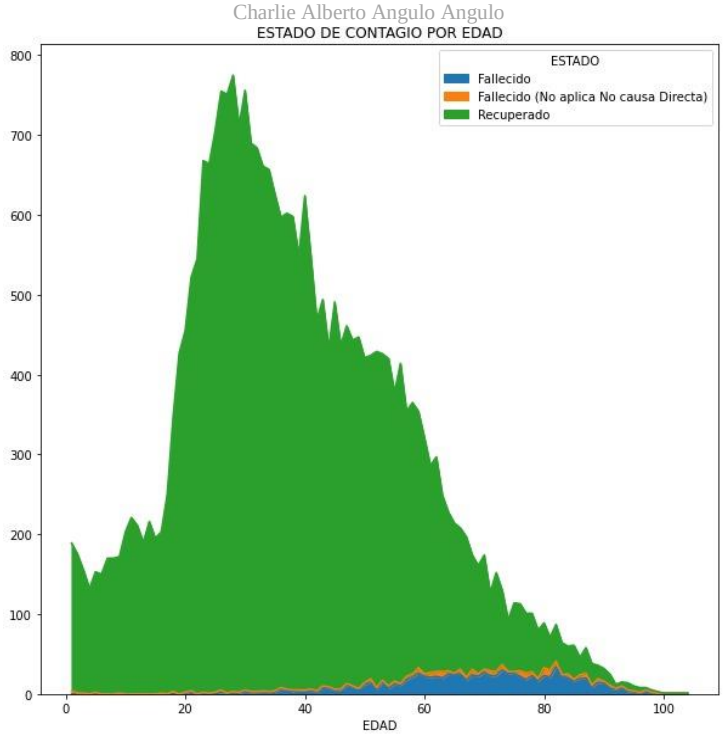

Figura 12: Representación gráfica estado contagios por edad.
Fuente: Elaboración propia.

7. Ejecución devops en el ciclo de entrega de software

Para finalizar el proceso es necesario realizar el ciclo de versionado de código y tan pronto se ejecuten los comandos *git* status, *git add*, *git commit* y *git push origin main* al repositorio, Tener en cuenta que el flujo de integración corre únicamente en la rama ***main*.** Con las configuraciones del entorno, *gitflow* iniciará el llamado a *sonarcloud*, en el cual se realizará las validaciones de cobertura de código y duplicidad del mismo.

Importante agregar los archivos *sonar-project.properties*, y *.github/* el cual contiene una configuración necesaria para la validación de código en sonar, el notebook que se desarrolló.

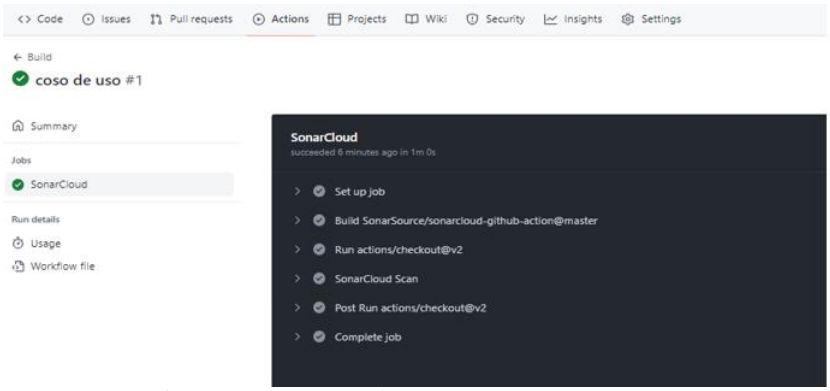

Figura 13: Ejecución SonarCloud mediante flujos de trabajo en gitHub.
Fuente: Elaboración propia.

Para verificar el análisis de código de manera gráfica, abrimos *sonarcloud* y en el proyecto que se crea por rama del repositorio que deseamos aplicarle validaciones, se obtendrá una visual del análisis realizado.

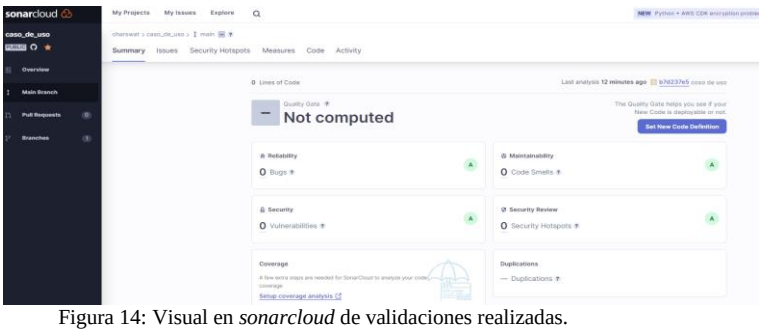

Figura 14: Visual en *sonarcloud* de validaciones realizadas.
Fuente: Elaboración propia.

La opción de incluir herramientas devops en el entorno, favorece a los *cloud* y data *Engineering*, permitiéndoles interactuar con estas herramientas ofrecidas en la nube como lo son *sonarCloud* y *git* y su *marketplace* de *git action* para automatizaciones de diversos proceso en flujos de trabajo en *git*, desarrollando habilidades de versionado de código, familiarizándose con términos como calidad de código, cobertura del mismo, *commits*, *pull request*, entre otros en la búsqueda de ofrecerles experimentar un símil de lo que se encontraran laboralmente.

Cuando no requiera utilizar más el entorno de esto para terminar el *clúster* mediante los comandos de *terraform Destroy*, eliminara todas las instancias y recursos utilizados en *Aws*.

d. Resultados de la solución

1. Métricas del desarrollo

Se realizaron diferentes acciones en el entorno experimental, acciones como, *count*, *gruopBy*, *joins*, *show*, funciones de ventana con 30.000 y 1.000.000 de registros como se muestra en las figuras 15 y 17, realmente el entorno se comporta muy bien y es apto para que sea utilizado como ambiente de experimentación, se obtienen tiempo de respuestas óptimos, y al ser una plataforma experimental no requiere compararla con otros proyectos dado el alcance que se le está brindando.

Figura 15: Ejecuciones con treinta mil y un millón de registros.
Fuente: Elaboración propia.

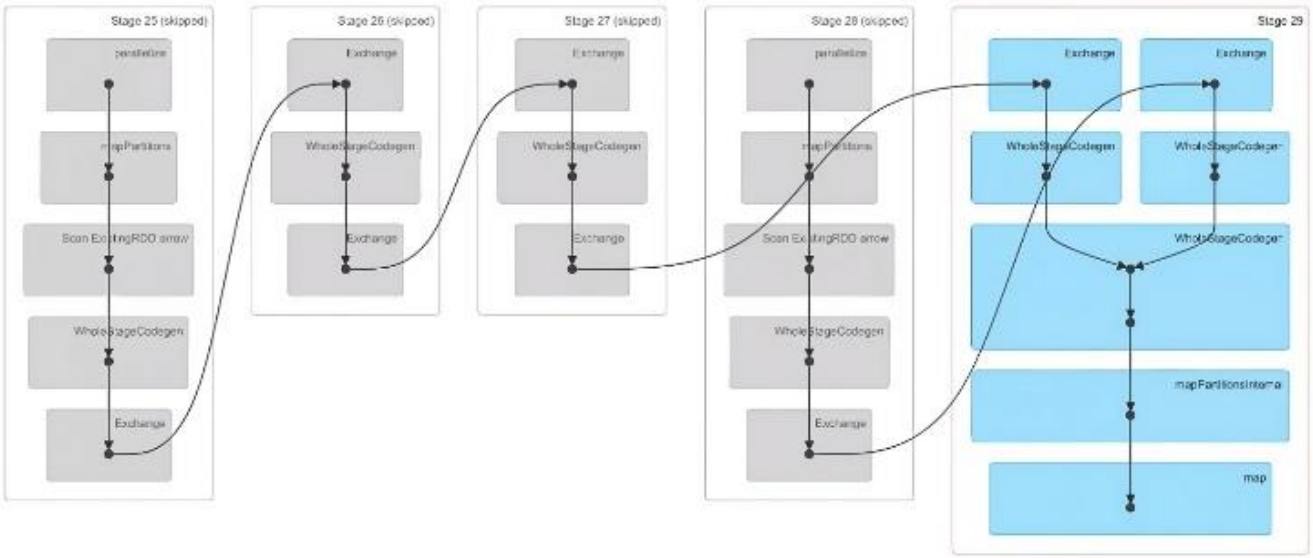

Figura 16: Visualización ejecución en DAG.
Fuente: Elaboración propia.

Figura 17: Tiempo finalización ejecuciones.
Fuente: Elaboración propia.

2. Retroalimentación público objetivo

Al final de implementada la solución se procedió a obtener una retroalimentación o *feedback* de 9 estudiantes que quieren enfocar sus estudios al desarrollo y de 4 profesionales desarrolladores y expertos en procesamiento de datos distribuidos, para recibir de estos últimos con concepto profesional de la solución. Al final se reciben comentarios muy positivos y una muy buena aceptación del ambiente experimental.

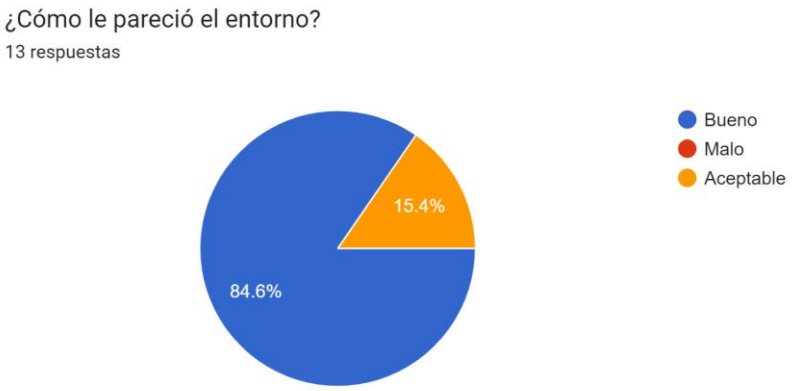

Figura 18: Retroalimentación Entorno.
Fuente: Elaboración propia.

Figura 19: Retroalimentación del valor de evitar configuraciones en el ambiente.
Fuente: Elaboración propia.

Figura 20: Retroalimentación tiempo despliegue entorno experimental.
Fuente: Elaboración propia.

3. Objetivos cumplidos

Al final de la implementación se cumple con el objetivo de disponer un entrono experimental de procesamiento de datos distribuidos integrando *devops* en el ciclo de entrega de software, cumpliendo con cada uno de sus objetivos específicos de analizar arquitecturas de referencia en temas de procesamiento de datos y *devops*, diseñando y construyendo una arquitectura de procesamiento de datos distribuidos e implementando el caso de uso, para corroborar su funcionamiento. Con el fin de que los futuros ingenieros de datos y los ingenieros en la nube logren enfocar sus esfuerzos de aprendizaje en la práctica de manipulación de datos y el uso de tecnologías en la nube.

4. Ventajas

Sin duda alguna es un ambiente completo para los estudiantes, está construido con herramientas, lenguaje de desarrollo y tecnologías demandadas y utilizadas en el mercado, logra desplegar ágilmente aproximadamente 10 min, y desmontar en menos de 2 minutos mediante infraestructura como código(*IaC*) en la nube, minimiza el esfuerzo y errores en configuraciones manuales, siempre se tendrá disponible el mismo entorno con un clúster de procesamiento ideal para la práctica en manipulación, análisis y visualización de datos. Este último punto importante para lograr tener una experiencia visual de lo desarrollado mediante una representación de los datos y manipulación del entorno mismo. Funcional desde cualquier sistema operativo que sea compatible con *terraform*, dado a que su infraestructura de despliega en la *Aws*.

5. Limitaciones:

El entorno cuenta con un lenguaje definido de análisis y manipulación de los datos como lo es Python, el script solo se probó en 2 ambientes de especificaciones similares, no se realizaron pruebas del resultado de modificar el clúster, se garantizó el funcionamiento correcto del entorno hasta con 1.000.000 de registros, aunque se realizaron pruebas con 3.000.000 de registros y funciono correctamente, solo que no se plasmaron en el documento. Y finalmente no es una infraestructura elástica u escalable

VII. CONCLUSIONES

En este documento se muestra la implementación y puesta en marcha de una arquitectura y plataforma orientada la experimentación y practica en manipulación y procesamiento de datos distribuidos integrado herramientas devops, empleando tecnologías demandadas en el mercado, en un entorno que permite a los usuarios ser competitivos ante la interacción , manipulación y práctica con un lenguaje de programación como python, clúster de procesamiento de datos (Spark y hadoop), herramientas Devops el ciclo de entrega de software, como es SonarCloud y Github permitiendo experimentar a los estudiante un símil de lo que se encontraran, cuando realicen desarrollos en espacios colaborativos laboralmente, utilizando tecnologías como infraestructura como código(terraform) y la nube mediante Aws, para interactuar y consumir una variedad de servicios que la cloud ofrece; y que en el sector tecnológico son habilidades que requieren las empresas en los perfiles de alta demanda. Todos los aspectos mencionados, convierten a esta plataforma en una alternativa para el aprendizaje e interacción con tecnologías líderes en el sector tecnológico.

Adicional el poder desplegar la infraestructura como código del entorno de manera ágil, realmente genera valor, y evita el hecho de configurar manualmente todos los recursos necesarios que se crean en la nube, se realiza la configuración del clúster automáticamente en cuestión de minutos y así mismo se destruye para no generar cobros adicionales en los servicios requeridos en Aws.

Sin duda la maestría nos brindó la base del conocimiento para poder profundizar y lograr crear esta solución, la cual ayudo a profundizar aún más en herramientas, arquitecturas de procesamiento de datos, patrones de diseño y desarrollo de software mediante infraestructura como código y otras tecnologías que continúan creciendo en el mercado, en donde cada problema presentado en la configuración del clúster, la creación de recursos en la nube y así su optimización, la parametrización del notebook y el flujo devops, él envió de archivos de configuración a la nube, y otros muchos inconvenientes presentados mediante lenguaje de configuración HashiCorp Configuration Language, nos permitió obtener un sin número de habilidades a la hora de enfrentarse con soluciones data processing, cloud. Devops y modelos de despliegue como código.

VIII. TRABAJOS FUTUROS

Para lograr que el entorno experimental sea funcional en otros ambientes diferente al desarrollado sería necesario realizar más pruebas para garantizar la generalidad del mismo.

Por otro lado, lograría ser interesante realizar la infraestructura bajo una arquitectura de contenedores y virtualización. En la parte de visualización mediante el notebook, resultaría útil agregar el kernel de scala para que se tenga también la facilidad de desarrollar bajo este lenguaje de programación, teniendo en cuanta que junto a Python son los lenguajes más demandados en lo que a manejo de datos se trata; e incluso escala esta embebido en algunas tecnologías lideres para el procesamiento de datos.

IX. REFERENCIAS

[1] C. Howard, "Top Priorities for IT: LeadershipVision for 2021, Data and Analytics Leaders," 2020, [Online]. Available: gartner.com.

[2] D. Smith, D. Villaba, M. Irvine, D. Stanke, and N. Harvey, "Accelerate State of DevOps 2021," p. 45, 2021, [Online]. Available: https://cloud.google.com/blog/products/devops-sre/announcing-dora-2021-accelerate-state-of-devops-report.

[3] T. Sousa, H. S. Ferreira, and F. F. Correia, "A Survey on the Adoption of Patterns for Engineering Software for the Cloud," IEEE Trans. Softw. Eng., vol. 5589, no. c, pp. 1–13, 2021, doi: 10.1109/TSE.2021.3052177.

[4] "What is a Cloud Engineer and How Do You Become One?" https://www.techtarget.com/searchcloudcomputing/definition/cloud-engineer (accessed Mar. 14, 2023).

[5] E. Bello, "¿Qué es Data Engineering? Funciones, requisitos y salario," Think. Innov., Oct. 2022, Accessed: Mar. 14, 2023. [Online]. Available: https://www.iebschool.com/blog/data-engineering-big-data/.

[6] S. Ananthi and S. Hariganesh, "A comprehensive study on cloud computing," ICIIECS 2015 - 2015 IEEE Int. Conf. Innov. Information, Embed. Commun. Syst., 2015, doi: 10.1109/ICIIECS.2015.7193151.

[7] Q. Rida, "A Roadmap Towards Big Data Opportunities, Emerging Issues and Hadoop as a Solution," Int. J. Educ. Manag. Eng., vol. 10, no. 4, pp. 8–17, 2020, doi: 10.5815/ijeme.2020.04.02.

[8] B. Leonel Goldman Cita and B. Leonel Goldman, "El Big Data y la Analítica de Negocios en el capitalismo informacional," p. 8, 2017, [Online]. Available: https://www.aacademica.org.

[9] J. Cao, M. Lin, and X. Ma, "A survey of big data for IoT in cloud computing," IAENG Int. J. Comput. Sci., vol. 47, no. 3, pp. 585–592, 2020.

[10] S. Zhelev and A. Rozeva, "Big data processing in the cloud - Challenges and platforms," AIP Conf. Proc., vol. 1910, no. December 2017, 2017, doi: 10.1063/1.5014007.

[11] "Chapter 1: What is Software Architecture? | Microsoft Docs." https://docs.microsoft.com/en-us/previous-versions/msp-n-p/ee658098(v=pandp.10)?redirectedfrom=MSDN (accessed Apr. 11, 2022).

[12] P. Mell and T. Grance, "The NIST-National Institute of Standars and Technology- Definition of Cloud Computing," NIST Spec. Publ. 800-145, p. 7, 2011.

[13] M. I. Malik, "Cloud Computing-Technologies," Int. J. Adv. Res. Comput. Sci., vol. 9, no. 2, pp. 379–384, 2018, doi: 10.26483/ijarcs.v9i2.5760.

[14] I. Ashraf, "An Overview of Service Models of Cloud Computing," Int. J. Multidiscip. Curr. Res., vol. 2, no. August 2014, pp. 779–783, 2014, [Online]. Available: http://ijmcr.com/wp-content/uploads/2014/08/Paper18779-783.pdf.

[15] C. Ebert, G. Gallardo, J. Hernantes, and N. Serrano, "DevOps," 2016.

[16] M. Artac, T. Borovssak, E. Di Nitto, M. Guerriero, and D. A. Tamburri, "DevOps: Introducing infrastructure-as-code," Proc. - 2017 IEEE/ACM 39th Int. Conf. Softw. Eng. Companion, ICSE-C 2017, no. May, pp. 497–498, 2017, doi: 10.1109/ICSE-C.2017.162.

[17] S. E. Bibri and J. Krogstie, "Towards a novel model for smart sustainable city planning and development: A scholarly backcasting approach," J. Futur. Stud., vol. 24, no. 1, pp. 45–62, 2019, doi: 10.6531/JFS.201909_24(1).0004.

[18] 宗成庆, "State of Software development," p. 48, 2021.

[19] G. Ruijun, "A Lightweight Experimental Platform for Big Data Based on Docker Containers," J. Phys. Conf. Ser., vol. 1437, no. 1, 2020, doi: 10.1088/1742-6596/1437/1/012104.

[20] K. Miao, J. Li, W. Hong, and M. Chen, "A Microservice-Based Big Data Analysis Platform for Online Educational Applications," Sci. Program., vol. 2020, 2020, doi: 10.1155/2020/6929750.

[21] M. Gupta, M. N. Chowdary, S. Bussa, and C. K. Chowdary, "Deploying Hadoop Architecture Using Ansible and Terraform," 2021 5th Int. Conf. Inf. Syst. Comput. Networks, ISCON 2021, pp. 1–6, 2021, doi: 10.1109/ISCON52037.2021.9702299.

[22] S. Saxena, S. K. Gupta, S. Poongodi, and P. Singh, "Turkish Journal of Computer and Mathematics Education Vol . 12 No . 11 (2021), 2507- 2521 Research Article A modern approach to building a data science framework delivery pipeline using DevOps practices," vol. 12, no. 11, pp. 2507–2521, 2021.

[23] D. Yang et al., "DevOps in practice for education management information system at ECNU," Procedia Comput. Sci., vol. 176, pp. 1382–1391, 2020, doi: 10.1016/j.procs.2020.09.148.

[24] D. Blazquez and J. Domenech, "Big Data sources and methods for social and economic analyses," Technol. Forecast. Soc. Change, vol. 130, no. March 2017, pp. 99–113, 2018, doi: 10.1016/j.techfore.2017.07.027.

[25] A. Gonçalves, F. Portela, M. F. Santos, and F. Rua, "Towards of a Real-time Big Data Architecture to Intensive Care," Procedia Comput. Sci., vol. 113, pp. 585–590, 2017, doi: 10.1016/j.procs.2017.08.294.

[26] N. Naik, "Docker container-based big data processing system in multiple clouds for everyone," 2017 IEEE Int. Symp. Syst. Eng. ISSE 2017 - Proc., 2017, doi: 10.1109/SysEng.2017.8088294.

[27] J. Bhimani, Z. Yang, M. Leeser, and N. Mi, "Accelerating big data applications using lightweight virtualization framework on enterprise cloud," 2017 IEEE High Perform. Extrem. Comput. Conf. HPEC 2017, 2017, doi: 10.1109/HPEC.2017.8091086.

[28] V. L., Camargo, J. J. Camargo-Ortega, and J. F. . Joyanes-Aguilar;, "Vista de Arquitectura vertida," vol. 1, pp. 7–18, 2015, doi: https://doi.org/10.14483/udistrital.jour.RC.2015.21.a1.

[29] "Terraform by HashiCorp." https://www.terraform.io/ (accessed Apr. 14, 2022).

[30] "¿Qué es AWS?" https://aws.amazon.com/es/what-is-aws/ (accessed Nov. 13, 2020).

[31] "Apache Hadoop 3.3.2 – HDFS Architecture." https://hadoop.apache.org/docs/r3.3.2/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html#Introduction (accessed Apr. 14, 2022).
[32] "Overview - Spark 3.2.1 Documentation." https://spark.apache.org/docs/latest/ (accessed Apr. 15, 2022).
[33] S. Salloum, R. Dautov, · Xiaojun Chen, · Patrick, X. Peng, and J. Z. Huang, "Big data analytics on Apache Spark," Int. J. Data Sci. Anal., vol. 1, pp. 145–164, 2016, doi: 10.1007/s41060-016-0027-9.