UNIVERSIDAD PERUANA UNIÓN

FACULTAD DE INGENIERÍA Y ARQUITECTURA

Escuela Profesional de Ingeniería de Sistemas

Una Institución Adventista

Integración y entregas continuas con herramientas open source basado en DevOps: Una revisión sistemática de la literatura

Por

Brandux Didier Juárez Avila

Kevin Junior Mogollón Calle

Asesor:

M.Sc. Fredy Abel Huanca Torres

Lima, 2020

DECLARACIÓN JURADA
DE AUTORÍA DEL TRABAJO DE INVESTIGACIÓN

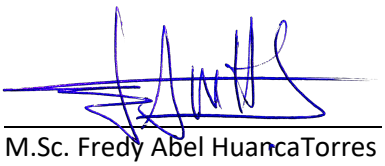
M.Sc. Fredy Abel Huanca Torres, de la Facultad de Ingeniería y Arquitectura, Escuela Profesional de Ingeniería de Sistemas, de la Universidad Peruana Unión.

DECLARO:

Que el presente informe de investigación titulado: ***"Integración y entregas continuas con herramientas open source basado en DevOps: Una revisión sistemática de la literatura"*** constituye la memoria que presentan los estudiantes Kevin Junior Mogollón Calle; Brandux Didier Juárez Avila para aspirar al grado de bachiller en Ingeniería de Sistemas, cuyo trabajo de investigación ha sido realizado en la Universidad Peruana Unión bajo mi dirección.

Las opiniones y declaraciones en este informe son de entera responsabilidad del autor, sin comprometer a la institución.

Y estando de acuerdo, firmo la presente constancia en *Lima,* al 06 de octubre del año 2020.



M.Sc. Fredy Abel HuancaTorres

ACTA DE SUSTENTACIÓN DE TRABAJO DE INVESTIGACIÓN

En Lima, Ñaña, Villa Unión, a……..los.....18.....día(s) del mes de…..septiembre.....del año 2020.... siendo las.....09:00.....horas, se reunieron los miembros del jurado en la Universidad Peruana Unión campus Lima, bajo la dirección del (de la) presidente(a): ..... Dra. Erika Inés Acuña Salinas....., el (la) secretario(a): ..... Ing. Diana Lidia Sánchez Torpoco..... y los demás miembros: ..… Ing. Jenson Daniel Chambi Aguilar …..y el (la) asesor(a): ……M.Sc. Fredy Abel Huanca Torres……con el propósito de administrar el acto académico de sustentación del trabajo de investigación titulado: "Integración y Entregas continuas con herramientas open source basados en DevOps: Una revisión sistemática de la literatura".

…….....….de los (las) egresados (as): a)……..... Kevin Junior Mogollon Calle …….....

……b) …… Brandux Didier Juarez Avila ……

…… conducente a la obtención del grado académico de Bachiller en

……Ingeniería de Sistemas…….....
(Denominación del Grado Académico de Bachiller)

El Presidente inició el acto académico de sustentación invitando …a los… candidato(a)/s hacer uso del tiempo determinado para su exposición. Concluida la exposición, el Presidente invitó a los demás miembros del jurado a efectuar las preguntas, y aclaraciones pertinentes, las cuales fueron absueltas por ..los… candidato(a)/s. Luego, se produjo un receso para las deliberaciones y la emisión del dictamen del jurado.

Posteriormente, el jurado procedió a dejar constancia escrita sobre la evaluación en la presente acta, con el dictamen siguiente:

Candidato/a (a): …… Kevin Junior Mogollon Calle …….

CALIFICACIÓN	ESCALAS			Mérito
	Vigesimal	Literal	Cualitativa	
Aprobado	**18**	**A-**	**Con nominación muy bueno**	Sobresaliente

Candidato/a (b): …… Brandux Didier Juarez Avila ……..

CALIFICACIÓN	ESCALAS			Mérito
	Vigesimal	Literal	Cualitativa	
Aprobado	**18**	**A-**	**Con nominación muy bueno**	Sobresaliente

(*) Ver parte posterior

Finalmente, el Presidente del jurado invitó …a los… candidato(a)/s a ponerse de pie, para recibir la evaluación final y concluir el acto académico de sustentación procediéndose a registrar las firmas respectivas.

Presidente Dra. Erika Inés Acuña Salinas		Secretario Ing. Diana Lidia Sánchez Torpoco
Asesor M.Sc. Fredy Abel Huanca Torres	Miembro	Miembro Ing. Jenson Daniel Chambi Aguilar
Candidato/a (a) Kevin Junior Mogollon Calle		Candidato/a (b) Brandux Didier Juarez Avila

ÍNDICE

Integración y entrega continuas con herramientas open source basado en DevOps: Una revisión sistemática de la literatura.

Continuous Integration and Delivery with open source tools based DevOps: A Systematic Review of the Literature.

Juárez Avila Brandux Didier[1] y Mogollón Calle Kevin Junior[2]

[1,2]Universidad Peruana Unión, Km 19 Carretera Central, Ñaña, Lurigancho, Lima, Perú

Resumen. La integración (CI) y entrega continua (CD) son las prácticas de desarrollo de software que han ayudado a impulsar el desarrollo ágil, debido a que fomentan la automatización en todo momento, permitiendo satisfacer las entregas de componentes de software rápidas y de calidad. Dichas prácticas requieren de herramientas, las mismas que no deben de ser impedimento alguno para poder llevar a cabo su implementación, por los costos excesivos y agobiantes, es así que el código abierto se convierte en la mejor opción. El presente estudio busca identificar, las herramientas open source, sus ventajas, limitaciones y por último cuáles son las más utilizadas. Para poder identificar los elementos mencionados se realizó una revisión sistemática de la literatura en bases de datos científicas reconocidas. De un total de ciento diez artículos revisados, se identificaron dieciocho artículos que ayudan al cumplimiento de nuestro objetivo. Al finalizar la revisión sistemática de la literatura se pudo evidenciar que existen algunas que destacan de otras por sus ventajas y desventajas, como es el caso de docker, gitlab y jenkins. Asimismo, se logró identificar sesenta y dos herramientas open source para CI/ CD y para finalizar las herramientas Jenkins y Github se presentan con una mayor frecuencia de uso.

Palabras claves: Integración continua, entrega continua, herramientas, open source, DevOps.

Abstract. Integration (CI) and Continuous Delivery (CD) are the software development practices that have helped drive agile development, because they encourage automation at all times, allowing to satisfy the deliveries of fast and quality software components. These practices require tools, which should not be an impediment to carry out their implementation, due to the excessive and overwhelming costs, thus open source becomes the best option. This study seeks to identify open source tools, their advantages, limitations and finally which are the most used. In order to identify the mentioned elements, a systematic review of the literature was carried out in recognized scientific databases. Out of a total of one hundred and ten articles reviewed, eighteen articles were identified that help fulfill our objective. At the end of the systematic review of the literature, it was evident that there are some that stand out from others due to their advantages and disadvantages, such as docker, gitlab and jenkins. Likewise, it was possible to identify sixty-two open source tools for CI / CD and to finalize the Jenkins and Github tools are presented with a greater frequency of use.

Keyworks: Continuous integration, Continuous delivery, tools, open source, DevOps.

1 Introducción

Los enfoques modernos en Ingeniería de Software requieren ciclos de desarrollo cada vez más ágiles y que a su vez garanticen la calidad , todo ello involucra una mejor armonización o colaboración efectiva dentro del equipo de desarrollo, son muchas las tendencias centradas en el desarrollo ágil que se han esforzado por resolver todo ello, dentro de ellas últimamente se destaca a DevOps con sus prácticas que respaldan la integración continua (CI) y entrega continua (CD), las mismas que son consideradas las mejores en el desarrollo de software. [5], [6], [8], [9], [23].

CI y CD a su vez impulsan la automatización, lo cual resulta de suma utilidad para más de una empresa, a fin de construir un software menos propenso a errores de incompatibilidad, asimismo el ahorro de tiempo al no realizar tareas repetitivas que demandan esfuerzo innecesario [11] .
Hacer realidad la implementación de estas prácticas involucra el uso de herramientas, las cuales están disponibles en una amplia variedad, las mismas que no deben convertirse en un barrera que trunque el objetivo debido a los costos agobiantes y excesivos que se generan, es así que el código abierto se convierte en una opción apropiada para estos casos [10].

El objetivo principal de esta revisión sistemática es identificar, las herramientas open source, sus ventajas, limitaciones y por último cuáles son las más utilizadas.

Este documento está organizado de la siguiente manera: sección dos presenta el marco conceptual; la sección tres ilustra el método de revisión sistemática de literatura (RSL); para la sección cuatro los resultados obtenidos. Las conclusiones e ideas para el trabajo futuro se encuentran en la sección cinco.

2 Marco Conceptual.

En la siguiente *sección* se presentan definiciones que se abarcara el presente objeto de estudio.

2.1 Desarrollo de software ágil.

El desarrollo de software tradicional sigue un ciclo de vida lineal, el cual sigue las etapas comunes: análisis, diseño, codificación y las pruebas las cuales se ejecutan justo antes de la implementación, asimismo este último y el mantenimiento posterior está a cargo de los profesionales de Operaciones de TI (Ops) [6].

Por otra parte, el desarrollo de software ágil busca en todo momento satisfacer al cliente a través de entregas tempranas y continuas del software. En efecto, se requiere de ciclos de desarrollo cortos, lo más importante aún, una mejor comunicación y cooperación de los equipos de desarrollo. Con el afán de responder a todos estos desafíos e ir al ritmo que el cliente demanda se ha recurrido a prácticas ágiles, las mismas que incluyen frameworks, metodologías e inclusive culturas, entre los cuales se destacan Kanban, Scrum, Extreme Programing y el recientemente emergido DevOps [5], [12], [9].

2.2 DevOps.

DevOps: un acrónimo de los equipos de desarrolladores (Dev) y Operaciones (Ops), estos equipos de tecnología de la información trabajan de manera colaborativa con el fin de eliminar los llamados “silos de información” [12], asimismo es una cultura, movimiento o práctica enfatizada en la colaboración y comunicación [14], cuyo objetivo es mejorar la velocidad del ciclo de lanzamiento (por ejemplo, de dos semanas a un día) de aplicaciones en producción y la automatización en la creación de nuevos componentes de software, manteniendo una alta calidad [15].

En las últimas dos décadas, el proceso de desarrollo de software ha presentado cambios pasando de un enfoque en cascada y con documentación extensa a modelos de desarrollo ágiles, esto ha generado un nuevo desafío en los equipos

multifuncionales en aumentar la velocidad de entregas a los usuarios finales, esto se logra mediante el empleo de un conjunto de prácticas y herramientas de ingeniería en toda la linea del ciclo de vida de software: implementación, lanzamiento, operación y monitoreo [8], [12].

Los estudios recientes que informan la adopción exitosa de DevOps, demuestran que dicha implementación requiere de cambios en las estructuras y roles organizacionales, arquitectura del sistema, procesos y herramientas, entre otros [15].

Sin embargo su correcta implementación trae una buena adopción de sus principios resumidos en "CALMS": Cultura, Automatización, Lean, Medición, Compartir. Dichos principios están incluidos dentro del proceso de mejora llamado Integración y Entrega continua con la intención de aumentar la estabilidad y el rendimiento de los activos de desarrollo y operaciones de la organización [6].

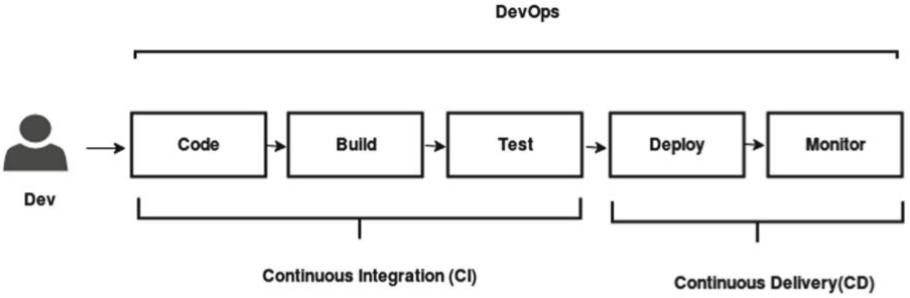


Fig. 1 CI/CD para el desarrollo de software [6].

2.3 Integración continua (CI).

El término de integración continua (CI) fue acuñado por primera vez por Grady Booch durante 1991, con la idea de ofrecer mejores versiones incrementales de software, asimismo posteriormente fue incluida como una práctica central de Extreme Programing XP, para el año 2000 comenzó a tener mayor acogida luego de una publicación de Martin Fowler en su blog [13], [25].

CI es una práctica de desarrollo de software que permite a los desarrolladores integrar a menudo las copias de codificación de manera rápida y segura, todo ello no sería posible sin la automatización de los procesos de compilación, vinculación y ejecución de pruebas, los mismos que necesitan apoyarse en el sistema de control de versiones y servidor de integración. Por otro lado, la finalización de estos procesos debe incluir indicadores que informen el éxito de los mismos para la toma de decisiones del equipo de desarrollo [13], [16], [17] .

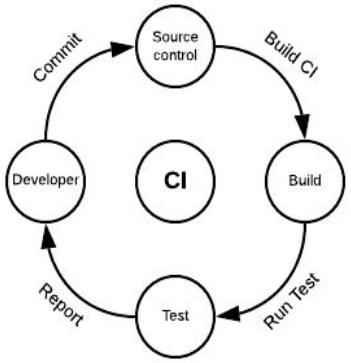


Fig. 2 Flujo de trabajo de integración continua [16], [21]

2.4 Entrega continua (CD)

Entrega continua nos permite preparar de manera automatizada una versión de software probada y lista para un lanzamiento bajo demanda en cualquier entorno. Es decir, nos permite disponer de una aplicación configurable y desplegable en cualquier momento sin tener que efectuar modificaciones en la codificación, garantizando el ahorro de tiempo y reducción de costos [18], [19], [12], [4], [10].

A propósito, en [10], [20], [24] se menciona que la integración continua es un subconjunto de la entrega continua, además se destaca que esta última es usada bajo el mismo contexto de despliegue continuo, pero no deben confundirse. La diferencia radica en que la entrega continua no necesariamente implementa el software para los usuarios finales pero sí incluye un proceso de despliegue durante el tiempo de desarrollo de nuevos entregables, por otra parte el despliegue continuo necesita de CD debido a que es necesario disponer de una confirmación probada del producto de software antes de ser entregado automáticamente a producción.

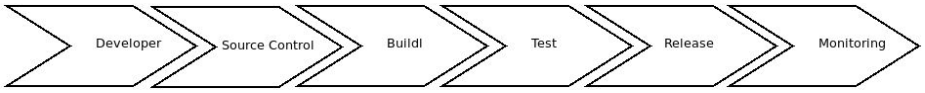


Fig. 3 Flujo de trabajo de entrega continua [22]

3 Revisión sistemática de la literatura

Los pasos del método de revisión sistemática de la literatura (RSL) se documentan a continuación [2].

3.1 Necesidad de la revisión sistemática.

La revisión sistemática de la literatura que se desarrolla en el presente proyecto de investigación, surge a partir de la necesidad de identificar las herramientas open source para poder realizar la integración y entrega continuas basada en DevOps. Esta necesidad se manifiesta con la evolución desmesurada sobre esta cultura durante los últimos años, en [1] se menciona que es debido al interés por nuevos enfoques ágiles en las organizaciones enfocadas no necesariamente al mundo de las tecnologías de información.

A fin de establecer el objetivo de investigación se empleo la plantilla Goal, Question, Metric (GQM por sus siglas en inglés) [3], en la siguiente tabla se detalla:

Tabla 1: Elaboración del objetivo de la investigación

Campo	Valor
Objeto de estudio	Equipo de desarrollo
Proposito	Herramientas de código abierto
Foco	Mejor herramienta realizada
Involucrados	DevOps, Integración continua, Entrega continuas
Factores de contexto	Uso de herramientas bajo devops

3.2 Preguntas para la revisión sistemática.

Tomando como referencia la tabla anterior se realizó la definición de las preguntas de investigación para este artículo, que se muestran en la siguiente *Tabla 2,* dichas preguntas están acompañadas por su motivación respectiva.

Además en la *Tabla 3*, se detallan las preguntas bibliométricas a fin de identificar la tendencia de los estudios presentados en el tiempo.

Tabla 2: Preguntas de investigación

ID	Pregunta	Motivación
PI-1	¿Cuáles son las ventajas y limitaciones de las herramientas open source encontradas?	Identificar la herramienta open source más significativas.
PI-2	¿Qué herramientas se han empleado para la integración y entrega continua?	Identificar las herramientas en las diferentes publicaciones sobre CI y CE.
PI-3	¿Qué herramientas para integración y entrega continua son las más utilizadas?	Determinar las herramientas más utilizadas para la integración y entrega continua.

Tabla 3: Preguntas bibliométricas

ID	Pregunta	Motivación
PB-1	¿Cuál es la cantidad de publicaciones por tipo de artículo?	Identificar la cantidad de estudios en base al tipo de artículos.
PB-2	¿Cómo ha evolucionado en el tiempo la frecuencia de las publicaciones sobre este tema?	Identificar la continuidad de publicaciones referentes al análisis de estudio, en 4 años.
PB-3	¿Cuales son las publicaciones en las que se han encontrado estudios relacionados al tema?	Determinar en qué dominio de publicación se encuentra la mayor concentración de artículos relacionados con el objeto de estudio.
PB-4	¿Cuál es la cantidad de publicaciones del tema por base de datos?	Identificar la cantidad de publicaciones por cada fuente bibliográfica.

3.3 Protocolo de la investigación.

3.3.1 Definición de las cadenas de búsqueda y fuentes bibliográficas

La elaboración de la cadena de búsqueda está en base a la estrategia PICO (Population, Intervention, Comparison, Outcome) [4], siendo este un proceso iterativo donde se realizaron ajustes convenientes para la posterior selección de resultados.

- **Población**:
  - Entidad: *Development Team*
  - Término principal 1: *devops*
  - Justificante: *Es seleccionada por ser objeto y giro de estudio de la RSL.*
- **Intervención**:
  - Entidad: *Tools open source*
  - Término principal 1: *Continuous delivery*
  - Término principal 2: *Continuous integratión*
  - Término principal 3: *Tools*
  - Términos alternativos 1: *CD*
  - Términos alternativos 2: *CI*
  - Justificante: *se selecciona los términos por ser elementos sobre los cuales se realizará el análisis y se obtienen dichos términos alternos por ser aquellos los tipos de objetos.*
- **Comparación**: *no aplica ya que en este estudio no se hace diferencia entre las herramientas.*
- **Resultado**:
  - Entidad: *Open sources tools most used*
  - Término principal 1: *Open Sources.*
  - Termino alternos: *most used, tools*
  - Justificante: *Los términos propuestos es lo que se busca obtener como resultado de investigación.*

Tabla 4: Términos y conectores lógicos usados para la búsqueda

Concepto	Terminos
Población	("devops")
Intervención	("CI" OR ("integratio*" and "continuous")) AND ("CD" OR ("deliver*"and "continuous"))
Comparación	No aplica (NA)
Resultado	("open source" and "tools")

Se eligió cuatro librerías digitales indexadas para este RSL, ya que tienen la cobertura suficiente sobre literatura de ingeniería de software y contienen búsqueda de metadatos, "búsqueda de comando" con "palabras clave" con una cadena de búsqueda, por ejemplo, una cadena de búsqueda: ("NFR" O "atributo de calidad") Y "ingeniería de software", las fuentes bibliográficas son las siguientes:

- ACM Digital Library (https://dl.acm.org/)
- IEEE Xplore (https://ieeexplore.ieee.org/Xplore/home.jsp)
- Science Direct (https://www.sciencedirect.com/)
- Springer Link (https://link.springer.com/)

3.3.2 **Criterios de inclusión y exclusión.**

Criterios de Inclusión.

- **CI-1**. Se considerarán todos los artículos dentro del 2016 hasta la actualidad.
- **CI-2**. Los artículos deben estar en Inglés
- **CI-3**. Se aceptarán artículos provenientes de Journal Article y Conference Proceeding, en las siguientes base de datos indexadas IEEE Xplore, Science Direct, ACM Digital Library y Springer Link.
- **CI-4**. Se aceptarán artículos relacionados con integración y entrega continuas.
- **CI-5**.Se aceptarán artículos que contengan herramientas Open Source para la implementación de integración y entrega continuas.
- **CI-6**. Se aceptarán artículos que contengan un proceso detallado de implementación de las herramientas de integración y entrega continuas deseadas para el estudio.

Criterios de exclusión:

- **CE-1.** Serán excluidos los artículos de años menores al 2016.
- **CE-2.** No se considerarán artículos en español.
- **CE-3.** Serán excluidos los artículos duplicados..
- **CE-4.** Serán excluidos artículos no relacionados con integración y entrega continuas.
- **CE-5.** Serán excluidos artículos que no contengan al menos una herramienta open source para integración y entrega continuas.
- **CE-6.** Serán excluidos artículos que no contengan un proceso de implementación de las herramientas de integración y entrega continuas deseadas para el estudio.

Procedimiento de selección. En base a lo mencionado en la sección 3.3.1, los documentos resultantes de las fuentes bibliográficas seleccionadas se guardaron en una carpeta de la herramienta Mendeley (https://www.mendeley.com/) y con Parsifal (https://parsif.al/), se aplico los criterios de inclusión y exclusión en base a 4 pasos:

Tabla 5: Procedimiento y criterios de selección

Paso	Procedimiento	Criterios de selección	
		CI	**CE**
1	Se realizó la extracción de artículos en las librerías indexadas mediante la cadena de búsqueda PICO, considerando los criterios de inclusiòn y exclusiòn mostrados en la columna derecha, los resultados encontrados se alojaron en el parsifal.	CI-1 CI-2 CI-3	CE-1 CE-2 CE-3
2	Se revisó el título y el resumen de los resultantes del Paso 1, aplicando los criterio de inclusión y exclusión CI-4 y CE-4 quedando aquellos que tienen contenido relacionado con integración y entrega continuas.	CI-4	CE-4
3	Se realizó con los resultados del Paso 2, se hizo una revisión de la introducción, conclusiones y por último una revisión rápida de todo el documento,a fin de aplicar los criterios de inclusión y exclusión, motivados a identificar aquellos artículos que contengan herramientas Open Source para la implementación de integración y entrega continuas.	CI-3	CE-3
4	En base a los artículos resultantes del Paso 3, se hizo una revisión del contenido completo para identificar aquellos artículos en el que se han detallado los procesos de implementación de dichas herramientas openSource para integración y entrega continuas.	CI-4	CE-4

3.3.3 Criterios de calidad.

Una vez se hayan seleccionado varios artículos en base a los criterios de inclusión y exclusión se procede a evaluar su calidad, según el puntaje empleado en [7] consiste en asignarle: Si cumple (Y) = 1, Cumple parcialmente (P) = 0.5 y No cumple (N) = 0. Las preguntas de evaluación de calidad definidas en esta revisión sistemática de la literatura están definidas como:

- **¿El contenido es relevante frente al objeto de estudio?**
  - Y*: El contenido es relevante para el objeto de estudio.*
  - P*: El contenido es parcialmente relevante para el objeto de estudio.*
  - N*: El contenido no es relevante para el objeto de estudio.*
- **¿Existe una discusión sobre los resultados del estudio?**
  - Y: *Existe una discusión sobre el resultado del estudio.*
  - P: *Existe parcialmente una discusión de los resultados del estudio.*
  - N*: No contiene una discusión de los resultados del estudio.*
- ❖ **¿El tema de integración y entrega continuas se adaptan al objeto de estudio?**

Y: *El tema de investigación se adapta al objeto de estudio.*
P: *El tema se adapta parcialmente al objeto de estudio.*
N: *El tema no se adapta al objeto de estudio.*

- **¿Se responden adecuadamente todas las preguntas de investigación?**
  Y: *Las preguntas son respondidas adecuadamente.*
  P: *Las preguntas de investigación son respondidas parcialmente.*
  N: *No se responden todas las preguntas de investigación.*
- **¿Están bien definidos los objetivos de investigación?**
  Y: *Si están bien definidos los objetivos de investigación.*
  P: *Los objetivos están parcialmente definidos.*
  N: *No están bien definidos los objetivos de investigación.*
- **¿Se describen detalladamente y se justifica su selección de las herramientas open source para integración y entrega continuas?**
  Y: *Si se describe y se justifica la selección de las herramientas.*
  P: *Se describe y se justifica la selección de herramientas parcialmente.*
  N: *No se describe, ni se justifica la selección de herramientas.*

Formulario de extracción de datos. La extracción de datos permite recopilar la información necesaria de los diferentes estudios seleccionados para poder responder las preguntas de investigación [7]. La Tabla 5 muestra el formulario de extracción de datos que se emplea en esta revisión sistemática.

Tabla 6: Formulario de extracción de datos.

Criterio	Descripción	Relevancia
Identificador		
Fuente		
Titulo		
Autores		
Publicacion		
Año de publicación		
Tipo de publicación		
Objetivo de la investigación		
DOI		
Herramientas		
Ventajas y limitaciones		

Validar protocolo de investigación. La validación se llevó de acuerdo a los estándares presentados por Kitchenham: Un investigador extrajo los datos y otro verificó la extracción. Se coordinó las tareas de extracción y verificación de datos, que involucraron a todos los autores de este artículo. La asignación no fue aleatoria, se basó en la disponibilidad de tiempo de los investigadores individuales. Cuando hubo un desacuerdo, discutimos los problemas hasta llegar a un acuerdo [2].

4 Resultados.

Dada la conformidad del protocolo mencionada en el apartado anterior. En esta sección se presentan los resultados del estudio.

4.1 Resultados de la búsqueda.

A fin de obtener los mejores resultados de las fuentes bibliográficas se realizaron ciertos cambios en la cadena de búsqueda, presentado en la *tabla 7*, con la intención que se adapte a las diferentes características que cuenta cada una de las librerías indexadas, para tener artículos que mejor se adapten al objeto de estudio de esta revisión sistemática.

Tabla 7: Resultados de búsqueda

Base de datos	Fecha	Total
Cadena de busqueda		
ACM Digital Library	mayo 2020	41
("CI" OR ("integratio*" AND "continuous")) AND ("CD" OR("deliver*" AND "continuous")) AND ("devops") AND ("open source ") AND ("tool*")		
IEEE Xplore	mayo 2020	5
("CI" OR ("integratio*" and "continuous")) AND ("CD" OR ("deliver*"and "continuous")) and "open*" and "source*" and "tool*"		
Science Direct	mayo 2020	19
"integration" and "delivery" and ("continuous" or "fast") and "open" and "source" and "tool" and "DevOps"		
Springer Link	mayo 2020	45
("CI" OR ("integratio*" and "continuous")) AND ("CD" OR ("deliver*"and "continuous")) and "open source" and "tool*" and "devops"		

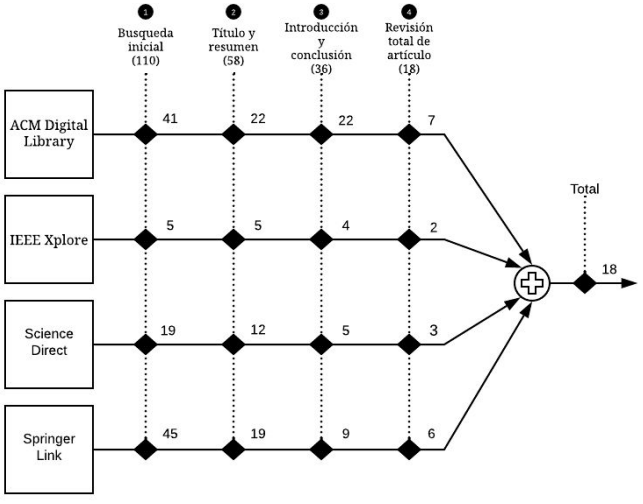
4.2 Seleccionar los estudios.

Los artículos encontrados en las diferentes bases de datos fueron exportados en formato BibTex para ser almacenados en una carpeta en Mendeley y con el sistema web Parsifal se llevó a cabo las iteraciones para los criterios de inclusión y exclusión, como se menciona en la sección 3.3.2, A continuación se detalla los pasos ejecutados para selección de estudios:

- **Paso 1:** Se ejecutó la cadena de búsqueda en las fuentes bibliográficas, la lista obtenida se filtró y se descarto los artículos duplicados con la herramienta mendeley. Asimismo, en dicha lista se aplicó los criterios de inclusión y exclusión mencionados en el **Paso 1** en **Tabla 5** con ayuda del sistema Parsifal.
- **Paso 2:** En base a la lista proveniente del *Paso 1*, se revisó el título y resumen de cada uno de los artículos a fin que queden seleccionados aquellos donde su contenido está relacionado con la integración y entrega continuas, siguiendo con las métricas mostradas en la **Tabla 5**.
- **Paso 3:** Los artículos que fueron seleccionados en el *Paso 2* fueron revisados en base a la introducción, conclusiones y una revisión rápida del contenido completo para identificar dichos artículos con herramientas Open Source para implementación de integración y entrega continuas, de acuerdo con los criterios en la **Tabla 5**.
- **Paso 4.** En el último paso a seguir se llevó a cabo la revisión completa del contenido en base a los criterios respectivos de la **Tabla 5** resaltando contenido relevante, las herramientas, procesos y pasos ejecutados, con la finalidad de solo obtener aquellos artículos que han detallado procesos de implementación con las herramientas OpenSource.

En la *Figura 4* se visualiza los resultados por cada paso realizado, se muestra también todos los artículos resultantes por cada fuente bibliográfica.

Fig. 4 Filtración de artículos.



4.3 Evaluar la calidad de los estudios.

En base al total de diciocho artículos seleccionados en el *Paso 4* se aplicó los criterios de calidad mencionados en la *sección 3.3.3,* Observamos que el 67% de los artículos sobrepasa la puntuación tres, dicha puntuación está definida al ser ma media de las preguntas de calidad realizadas, lo que se puede considerar como un buen indicador de la calidad de los estudios. En la *figura 5* se muestra la asignación de puntos usando el Parsifal, asimismo en la *tabla 8* se visualiza los resultados de la evaluación.

Fig. 5 Puntuación Y;P;N, con Parsifal.

Fuente:https://parsif.al/kevinmogollon/buenas-practicas-para-el-desarrollo-de-software-basada-en-scrum-y-devops/conducting/quality/

Tabla 8: Evaluación de la calidad del contenido

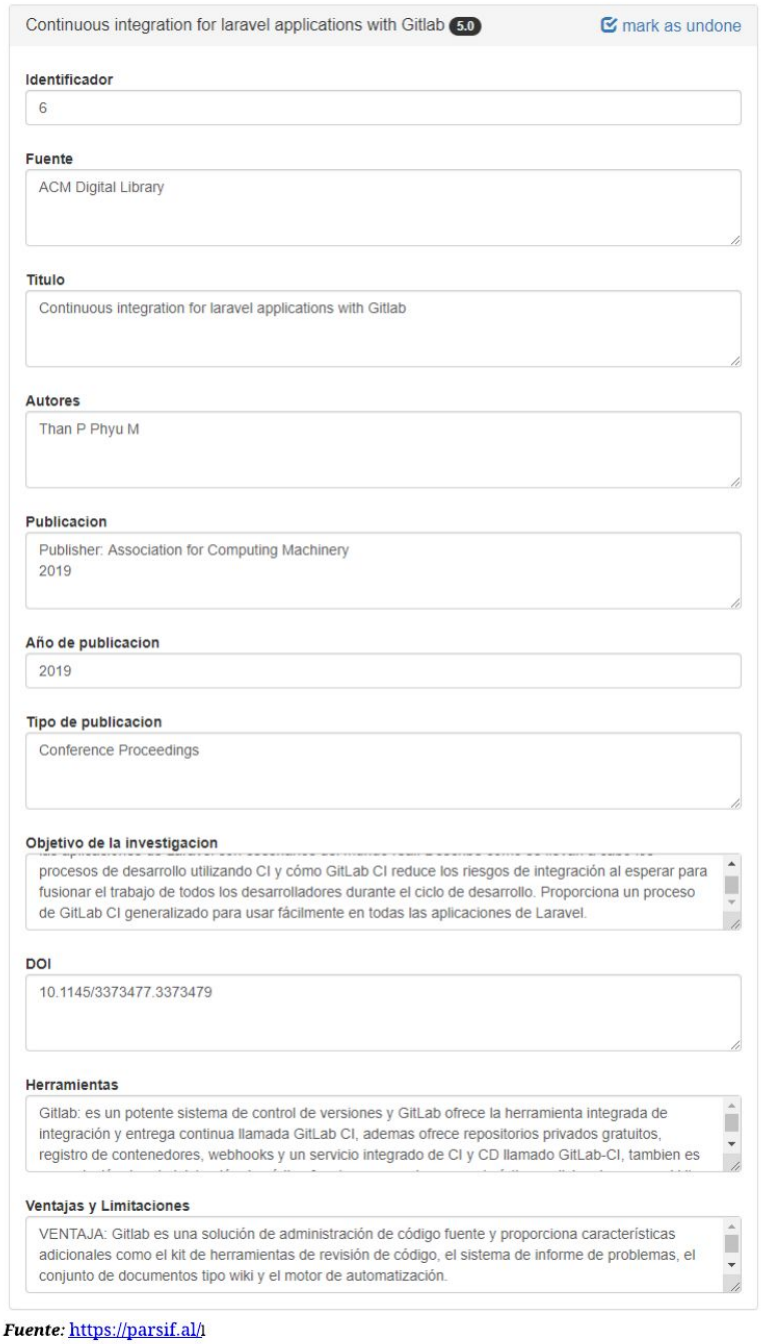
ID	Titulo	Nivel de calidad
1	Enabling continuous integration in a formal methods setting	3
2	Building science gateway infrastructure in the middle of the pacific and beyond: Experiences using the agave deployer and agave platform to build science gateways	4
3	Building lean continuous integration and delivery pipelines by applying devops principles: A case study at varidesk	2,5
4	Streamlining APIfication by Generating APIs for Diverse Executables Using Any2API	3

5	Towards combined process {\&} tool variability management in software testing	2,5
6	Devops meets dynamic orchestration	4
7	Migrating to Cloud-Native Architectures Using Microservices: An Experience Report	4,5
8	One size does not fit all: An empirical study of containerized continuous deployment workflows	2,5
9	Usage, costs, and benefits of continuous integration in open-source projects	3,5
10	Agile Development and Operation of Complex Systems in Multi-technology and Multi-company Environments: Following a DevOps Approach	4
11	Continuous integration and delivery for HPC: Using Singularity and Jenkins	3,5
12	Empowering Continuous Delivery in Software Development: The DevOps Strategy	4,5
13	Continuous integration for laravel applications with Gitlab	5
14	Towards cloud native continuous delivery: An industrial experience report	3
15	Continuous Integration and Continuous Deployment Pipeline Automation Using Jenkins Ansible	4
16	Utilising CI environment for efficient and effective testing of NFRs	3,5
17	Improving the delivery cycle: A multiple-case study of the toolchains in Finnish software intensive enterprises	5,5
18	DevOps in practice: A multiple case study of five companies	4

4.4 Extraer los datos relevantes

En la extracción de datos de acuerdo al formulario de extracción mencionado en la *tabla 5,* se realizó mediante la lectura de los artículos seleccionados y en paralelo se llenó el formulario correspondiente, en la *figura 6* se visualiza un ejemplo de la información relevante de cada artículo para ello se utilizó el apartado de Data Extraction del Parsifal.

Fig. 6 Ejemplo de extracción de datos.



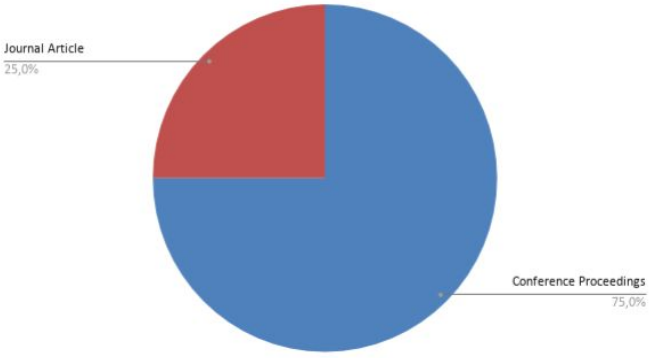
Fuente: https://parsif.al/

4.5 Análisis bibliométrico

4.5.1 PB-1 ¿Cuál es la cantidad de publicaciones por tipo de artículo?

En la *figura 7* se visualiza el porcentaje que corresponde a cada tipo de artículo encontrado en la RSL. Se observa que el 75% de ellos corresponden a conferencias (Conference Proceedings); seguidamente los Journal Article provenientes de las fuentes bibliográficas anteriormente mencionadas representan un 25% del total. En base a los resultados encontrados se puede concluir que las conferencias son la mayor fuente de los estudios sobre herramientas openSource para integración y entrega continua.

Fig. 7 Tipos de artículos.



4.5.2 PB-2 ¿Cómo ha evolucionado en el tiempo la frecuencia de las publicaciones sobre este tema?

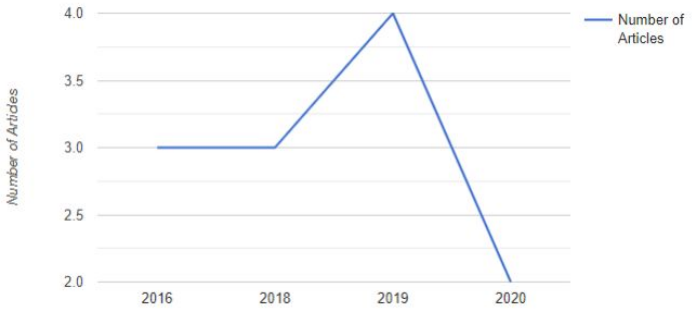


Fig. 8 Frecuencia de publicaciones en el tiempo.

Al analizar los resultados en la *Figura 8* se puede observar un incremento en publicaciones a partir del 2018 al 2019, asimismo podemos identificar que existe un bajón considerable del 2019 hasta la actualidad, esto indica que la coyuntura respecto al nuevo coronavirus (Cov-19) ha afectado significativamente la continuidad de publicaciones respecto al tema, cabe recalcar que la extracción de datos de la cadena de búsqueda corresponde entre el periodo del 2016 hasta mayo del 2020 tal y como se muestra en la *tabla 6.* De un total de doce artículos resultantes de la evaluación de calidad (*sección 4.3*), cinco (41.7%) fueron publicados en el año 2019, en los años de 2016 al 2018 se hicieron tres (25%) publicaciones cada uno y, por último, uno (8.3%) en el 2020.

4.5.3 PB-3 ¿Cuáles son las publicaciones en las que se han encontrado estudios relacionados?

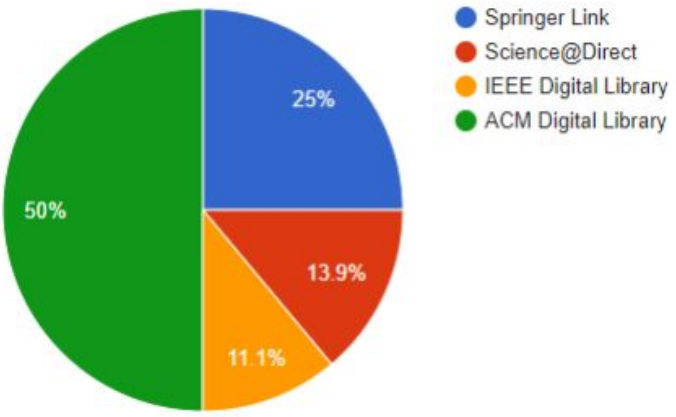
Las publicaciones donde fueron extraídos los artículos seleccionados están mostrados en la *tabla 9*; A partir de este análisis se muestra que existe concurrencia de publicaciones referentes al tema de investigación en dominios de ciencias de computación, ingeniería de software y sistemas de información.

Tabla 9: Publicaciones correspondiente a los artículos.

Publicacion	Cantidad
Springer International Publishing Switzerland	1
Association for Computing Machinery	4
Springer Verlag	3
Information and Software Technology	3
Institute of Electrical and Electronics Engineers.	1

4.5.4 PB-4 ¿Cuál es la cantidad de publicaciones del tema por base de datos?

Fig. 9 Fuentes bibliográficas.



Al llevar a cabo la ejecución de la cadena de búsqueda en cuatro fuentes bibliográficas previamente definidas, se obtuvo como resultado: el 50% de los artículos provienen de ACM Digital Library, seguidamente de Springer Link con un 25% artículos encontrados, 13.9% fueron obtenidos de Science Direct y, por último, el 11.1% restante proviene de IEEE Digital Library.

4.6 Preguntas de investigación.

4.6.1 PI-1 ¿Cuáles son las ventajas y limitaciones de las herramientas Open Source encontradas?

Entre las ventajas y limitaciones existen algunas que destacan de otras; asimismo algunas similitudes. Entre las más destacadas por sus ventajas se encuentran: Jenkins, Docker y Gitlab. En la tabla 10 se puede identificar detalladamente cada una de las ventajas y limitaciones de las herramientas ya mencionadas.

Tabla 10: Ventajas y Limitaciones de las herramientas open source encontradas CI/CD

Herramienta	Ventajas	Limitaciones

Jenkins	Varios estilos de configuración de trabajos	En ciertos casos si se requiere hacer automatizaciones mucho más complejas habrá que dedicar tiempo y esfuerzo, ya que conlleva integrar y personalizar cada paso de una tarea, para ello se necesita gente cualificada y con experiencia.
	Administración de secretos	
	Facilmente configurable	
	Compatibilidad con diferentes sistemas operativos	
	Control de acceso basado en roles	
Github	Fácil integración con Jenkins por medio de los Webhooks	Permite tener repositorios privados pero sólo permite a 3 colaboradores puedan trabajar en un mismo repositorio.
Gitlab	Servicio integrado de CI llamado Gitlab-CI	Para poder realizar integración continua es necesario contar con Gitlab Runner
	Contiene herramientas de revisión de código, informe de problemas, wiki y motor de automatización.	
Docker	Excelente para componentes y delgados.	Demasiada basura al crear y destruir continua de imágenes y contenedores
	Respaldo por los vendedores más importantes en el ecosistema de TI	
	Mismo comportamiento en diferentes entornos.	
	Independencia del sistema operativo	
	Ahorro de recursos.	
	Mismo comportamiento en diferentes entornos.	
Ansible	Formato de configuración basado en yaml, el cual es de fácil lectura	Al especificarlo todo en YAML a veces necesitamos crear nuestro propio módulo es un salto a lo desconocido y requiere de mayor tiempo para aprender cómo hacerlo.
	Ahorro de tiempo	En ciertas ocasiones todo los módulos disponibles con los que cuenta, no siempre están actualizados.

4.6.2 PI-2 ¿Qué herramientas Open Source se han empleado para la integración y entrega continua?

Las herramientas open source para CI /CD encontradas fueron un total de sesenta y dos, existe una amplia variedad de estas, debido a que cada una de ellas están adaptadas para un fin específico. En la figura 10, se agrupó dichas herramientas en base a las dos prácticas consideradas (CI /CD).

Por otro lado, como se puede apreciar en la figura 11, las herramientas para CI son las más sobresalientes en cantidad con 81%, seguidamente de las herramientas de CD con un 17,5 %, por último y no menos importante una herramienta "Jenkins" que involucra a ambas prácticas representada con un 1.6 % del total.

Fig. 10 Herramientas Open Source para CI/CD encontradas.

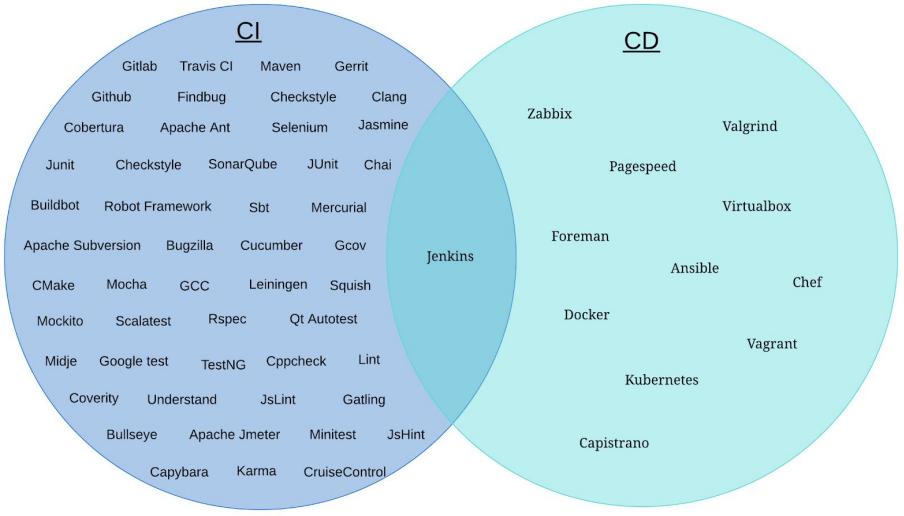
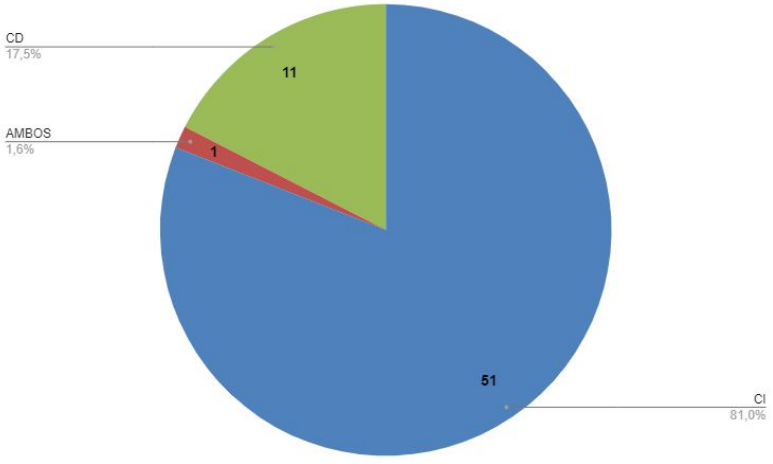


Fig. 11 Cantidades y porcentaje de herramientas CI/CD encontradas.



4.6.3 PI-3 ¿Qué herramientas para integración y entrega continuas son las más utilizadas?

De las herramientas open source para CI /CD encontradas existe un porcentaje que es usado comúnmente en comparación a otras. En la figura 10, se puede observar que son ocho las herramientas sobresalientes, Jenkins con una mayor relevancia representada por una frecuencia de ocho , asimismo Github con cinco y para finalizar Junit con una menor relevancia en comparación a todas representada por una frecuencia de dos.

Tabla 11: Herramientas más usadas CI/CD

Herramienta	Cantidad mencionada
Jenkins	8
Github	5
Ansible	4
Docker	3
Maven	3
Selenium	3
Vagrant	2
Junit	2

5 Conclusiones y trabajo futuro.

En este estudio están reflejados los resultados de una revisión sistemática aplicada a dieciocho trabajos de investigación del tipo conferencia y artículo, los mismos que fueron seleccionados de las bases de datos científicas indexadas de gran relevancia en el ámbito científico y académico. Los estudios anteriormente mencionados fueron escogidos gracias al análisis bibliométrico en el cual se busca encontrar aquellos estudios que alcancen el objetivo definido y hayan sido publicados dentro de los últimos cuatro años.

Como se puede apreciar en los resultados observamos que existen algunas que destacan de otras por sus ventajas y desventajas, como es el caso de docker, gitlab y jenkins. Asimismo, se logró identificar sesenta y dos herramientas open source para CI/ CD, donde las herramientas para CI son las más sobresaliente en frecuencia, por otro lado las herramientas de CD con una menor frecuencia, asimismo se pudo identificar que existe una herramienta "Jenkins" que involucra a ambas prácticas . Para finalizar las herramientas como Jenkins y Github se destacan como las más usadas

Como trabajo futuro, se propone desarrollar un framework para integración y entregas continuas haciendo uso de herramientas de código abierto. Asimismo para definir el uso de dichas herramientas se debe emplear métricas que fundamenten su elección.

6 Referencias

[1] Guillermo Jiménez Marco, "DevOps, la nueva tendencia en el desarrollo de sistemas TI, un caso práctico en el análisis de incidencias de software," Universidad Politécnica de Catalunya, 2016.

[2] B. Kitchenham, O. Pearl Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, "Systematic literature reviews in software engineering - A systematic literature review," Inf. Softw. Technol., vol. 51, no. 1, pp. 7–15, 2009.

[3] Caldiera, V. R. B. G., & Rombach, H. D. (1994). The goal question metric approach. Encyclopedia of software engineering, 528-532.Caldiera, V. R. B. G., & Rombach, H. D. (1994). The goal question metric approach. Encyclopedia of software engineering, 528-532.

[4] C. Da Costa Santos, C. de Mattos Pimenta y M. Nobre, «The PICO strategy for the research question construction and evidence search» Rev. Lat. Am. Enfermagem, 2007.

[5] K. Meixner, D. Winkler, and S. Biffl, "Towards combined process & tool variability management in software testing," in ACM International Conference Proceeding Series, Feb. 2019, doi: 10.1145/3302333.3302339.

[6] K. Bahadori and T. Vardanega, "Devops meets dynamic orchestration," in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2019, vol. 11350 LNCS, pp. 142–154, doi: 10.1007/978-3-030-06019-0_11.

[7] B. D. Rouhani, M. N. Z. R. Mahrin, F. Nikpay, R. B. Ahmad, and P. Nikfard, “A systematic literature review on Enterprise Architecture Implementation Methodologies,” Information and Software Technology, vol. 62, no. 1. Elsevier B.V., pp. 1–20, 2015, doi: 10.1016/j.infsof.2015.01.012.

[8] G. Benguria, J. Alonso, I. Etxaniz, L. Orue-Echevarria, and M. Escalante, “Agile Development and Operation of Complex Systems in Multi-technology and Multi-company Environments: Following a DevOps Approach,” in Communications in Computer and Information Science, 2018, vol. 896, pp. 15–27, doi: 10.1007/978-3-319-97925-0_2.

[9] L. D. Couto, P. W. V. Tran-Jørgensen, R. S. Nilsson, and P. G. Larsen, “Enabling continuous integration in a formal methods setting,” Int. J. Softw. Tools Technol. Transf., 2019, doi: 10.1007/s10009-019-00546-y.

[10] A. Hakli, D. Taibi, and K. Systa, “Towards cloud native continuous delivery: An industrial experience report,” in Proceedings - 11th IEEE/ACM International Conference on Utility and Cloud Computing Companion, UCC Companion 2018, Jan. 2019, pp. 314–320, doi: 10.1109/UCC-Companion.2018.00074

[11] S. Mysari and V. Bejgam, “Continuous Integration and Continuous Deployment Pipeline Automation Using Jenkins Ansible,” in International Conference on Emerging Trends in Information Technology and Engineering, ic-ETITE 2020, Feb. 2020, doi: 10.1109/ic-ETITE47903.2020.239.

[12] Mäkinen, S., Leppänen, M., Kilamo, T., Mattila, A. L., Laukkanen, E., Pagels, M., & Männistö, T. (2016). Improving the delivery cycle: A multiple-case study of the toolchains in Finnish software intensive enterprises. Information and Software Technology, 80, 175–194. https://doi.org/10.1016/j.infsof.2016.09.001

[13] M. Hilton, T. Tunnell, K. Huang, D. Marinov, and D. Dig, “Usage, costs, and benefits of continuous integration in open-source projects,” in ASE 2016 - Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering, Aug. 2016, pp. 426–437, doi: 10.1145/2970276.2970358.

[14] Siebra, C., Lacerda, R., Cerqueira, I., Quintino, J. P., Florentin, F., da Silva, F. B. Q., & Santos, A. L. M. (2019). Empowering Continuous Delivery in Software Development: The DevOps Strategy. In Communications in Computer and Information Science (Vol. 1077, pp. 247–265). Springer Verlag. https://doi.org/10.1007/978-3-030-29157-0_11

[15] Lwakatare, L. E., Kilamo, T., Karvonen, T., Sauvola, T., Heikkilä, V., Itkonen, J., … Lassenius, C. (2019). DevOps in practice: A multiple case study of five companies. Information and Software Technology, 114, 217–230. https://doi.org/10.1016/j.infsof.2019.06.010

[16] P. P. Than and M. P. Phyu, “Continuous integration for laravel applications with Gitlab,” in ACM International Conference Proceeding Series, Nov. 2019, doi: 10.1145/3373477.3373479.

[17] Z. Sampedro, A. Holt, and T. Hauser, “Continuous integration and delivery for HPC: Using Singularity and Jenkins,” in ACM International Conference Proceeding Series, Jul. 2018, doi: 10.1145/3219104.3219147.

[18] A. Balalaie, A. Heydarnoori, and P. Jamshidi, “Migrating to Cloud-Native Architectures Using Microservices: An Experience Report,” in Springer International Publishing Switzerland, 2016, vol. 567, doi: 10.1007/978-3-319-33313-7.

[19] S. B. Cleveland, R. Dooley, D. Perry, J. Stubbs, J. M. Fonner, and G. A. Jacobs, “Building science gateway infrastructure in the middle of the pacific and beyond: Experiences using the agave deployer and agave platform to build science gateways,” in ACM International Conference Proceeding Series, Jul. 2018, doi: 10.1145/3219104.3219151.

[20] Y. Zhang, B. Vasilescu, H. Wang, and V. Filkov, “One size does not fit all: An empirical study of containerized continuous deployment workflows,” in ESEC/FSE 2018 - Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Oct. 2018, pp. 295–306, doi: 10.1145/3236024.3236033.

[21] L. Yu, E. Alégroth, P. Chatzipetrou, and T. Gorschek, “Utilising CI environment for efficient and effective testing of NFRs,” Inf. Softw. Technol., vol. 117, Jan. 2020, doi: 10.1016/j.infsof.2019.106199.

[22] J. Humble and D. Farley, Continuous delivery, Pearson Education. United States of America, 2011.

[23] V. Debroy, S. Miller, and L. Brimble, “Building lean continuous integration and delivery pipelines by applying devops principles: A case study at varidesk,” in ESEC/FSE 2018 - Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Oct. 2018, pp. 851–856, doi: 10.1145/3236024.3275528.

[24] J. Wettinger, U. Breitenbücher, and F. Leymann, “Streamlining APIfication by Generating APIs for Diverse Executables Using Any2API,” doi: 10.1007/978-3-319-29582-4.

[25] M. Fowler, “Continuous Integration,” 2006. https://martinfowler.com/articles/continuousIntegration.html.