# C:\Users\gkrios\Downloads\isntitucional vertical azul.jpg

**UNIVERSIDAD TÉCNICA PARTICULAR DE LOJA**

*La Universidad Católica de Loja*

**FACULTAD DE ÁREA TÉCNICA**

**CARRERA DE COMPUTACIÓN Y SISTEMAS INFORMÁTICOS**

**Diseño e implementación de un enfoque de seguridad DevSecOps en el desarrollo de aplicaciones orientadas a microservicios**

Trabajo de titulación previo a la obtención del título de:

**INGENIERO EN SISTEMAS INFORMÁTICOS Y**

**COMPUTACIÓN**

**Autor:** Quichimbo Guamán Ronald Vicente

**Director:** Romero González, Karla Alexandra

LOJA

2022

# Aprobación del director del Trabajo de Titulación

Loja, día de mes de año

Título académico completo, (no colocar siglas)

Nombres y Apellidos completos del director de la carrera

**Director de la carrera de Xxxxxxxxxx**

Ciudad.-

De mi consideración:

Me permito comunicar que, en calidad de director del presente Trabajo de Titulación denominado: (nombre del trabajo) realizado por Nombres y Apellidos completos del autor o autores (as) ha sido orientado y revisado durante su ejecución, así mismo ha sido verificado a través de la herramienta de similitud académica institucional, y cuenta con un porcentaje de coincidencia aceptable. En virtud de ello, y por considerar que el mismo cumple con todos los parámetros establecidos por la Universidad, doy mi aprobación a fin de continuar con el proceso académico correspondiente.

Particular que comunico para los fines pertinentes.

Atentamente,

Director: Nombres y Apellidos completos del Director del Trabajo de Titulación y título académico.

C.I.:

Correo electrónico:

# Declaración de **autoría** y cesión de derechos

Yo, Nombres y Apellidos completos, declaro y acepto en forma expresa lo siguiente:

Ser autor (a) del Trabajo de Titulación denominado: …………………, de la carrera de…………….…., específicamente de los contenidos comprendidos en: (se debe colocar los nombres de los capítulos elaborados en el Trabajo de Titulación), siendo (nombres y apellidos completos), director (a) del presente trabajo; también declaro que la presente investigación no vulnera derechos de terceros ni utiliza fraudulentamente obras preexistentes. Además, ratifico que las ideas, criterios, opiniones, procedimientos y resultados vertidos en el presente trabajo investigativo, son de mi exclusiva responsabilidad. Eximo expresamente a la Universidad Técnica Particular de Loja y a sus representantes legales de posibles reclamos o acciones judiciales o administrativas, en relación a la propiedad intelectual de este trabajo.

Que la presente obra, producto de mis actividades académicas y de investigación, forma parte del patrimonio de la Universidad Técnica Particular de Loja, de conformidad con el artículo 20, literal j), de la Ley Orgánica de Educación Superior; y, artículo 91 del Estatuto Orgánico de la UTPL, que establece: “Forman parte del patrimonio de la Universidad la propiedad intelectual de investigaciones, trabajos científicos o técnicos y tesis de grado que se realicen a través, o con el apoyo financiero, académico o institucional (operativo) de la Universidad”, en tal virtud, cedo a favor de la Universidad Técnica Particular de Loja la titularidad de los derechos patrimoniales que me corresponden en calidad de autor/a, de forma incondicional, completa, exclusiva y por todo el tiempo de su vigencia.

La Universidad Técnica Particular de Loja queda facultada para ingresar el presente trabajo al Sistema Nacional de Información de la Educación Superior del Ecuador para su difusión pública, en cumplimiento del artículo 144 de la Ley Orgánica de Educación Superior.

.................................................................

Autor: Nombres y Apellidos completos del autor

C.I.:

Correo electrónico:

# Dedicatoria

Xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxxxxxxxx xxxxxxx xxxxxxx xxxxxx xxxxxx xxxxxxxx

# Agradecimiento

Xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxxxxxxxx xxxxxxx xxxxxxx xxxxxx xxxxxx xxxxxxxx xxxxxxxxxx xxxxxxxx x

# Índice de contenido

[1](#_Toc1)

[Aprobación del director del Trabajo de Titulación 2](#_Toc2)

[Declaración de autoría y cesión de derechos 3](#_Toc3)

[Dedicatoria 5](#_Toc4)

[Agradecimiento 6](#_Toc5)

[Índice de contenido 7](#_Toc6)

[Resumen 1](#_Toc7)

[Abstract 2](#_Toc8)

[Introducción 3](#_Toc9)

[Capítulo uno 4](#_Toc10)

[Nombre del capítulo 4](#_Toc11)

[1.1 Xxxxxxxxx xxxxx xxxxxxxxxx 4](#_Toc12)

[1.1.1 *hola xxxxxxxx xxxxxxxxxxx*  6](#_Toc13)

[1.1.1.1 Xxxxxxxxxxxxxxx. Xxxxxxxxxxxxxxxxxxxx xxxxxx xxxxxxx xxxxx xxxxxx xxxxxxxxxxxxx xxxxxxxxxxxxxxx. 6](#_Toc14)

[1.1.1.1.1 Xxxxxxxxxxx xxxxxxx xxxx. Xxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxx xxxxxxxxxx xxxxxxx xxxxxxxxxxxx. 6](#_Toc15)

[Capítulo dos 8](#_Toc16)

[Nombre del capítulo 8](#_Toc17)

[2.1 Xxxxxxxxxx xxxxxx xxxxxxx xxxx 8](#_Toc18)

[2.2 Xxxxxxx xxxxxx xxxxxxxxx xxxxxxx 8](#_Toc19)

[Capítulo tres 9](#_Toc20)

[Nombre del capítulo 9](#_Toc21)

[Conclusiones 10](#_Toc22)

[Recomendaciones 11](#_Toc23)

[Referencias 12](#_Toc24)

[Apéndice 13](#_Toc25)

[Apéndice A. Xxxxxxx xxxxxx xxxxxx 13](#_Toc26)

[Apéndice B. Xxxxxxx xxxxxx xxxxxx 14](#_Toc27)

[Apéndice C. Xxxxxxx xxxxxx xxxxxx 15](#_Toc28)

Aquí se debe hacer constar la paginación respectiva de los capítulos, temas y subtemas desarrollados, así como incluir índice de tablas y figuras.

**Índice de tablas**

[**Tabla 1** **Xxxxxxx xxxxxx xxxx......................................................................................………4**](#_Toc40382068)

**Índice de figuras**

[**Figura 1 Xxxxxxxx xxxxxx xxx**.**................................. …………………………………………..5**](#_Toc40381998)

# Resumen

El resumen se presentará en un único párrafo con un máximo de **180 palabras**, sintetiza el aporte que brinda el trabajo realizado. **Obligatoriamente** debe contener las palabras clave **(máximo tres)**.

Ejemplo:

Xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxxxxxxxx xxxxxxx xxxxxxx xxxxxx xxxxxx xxxxxxxx xxxxxxxx xxxxxxxxx xxxxxx xxxxx xxxxx xxxxxxxxxx xxxxxxx xxxxxxxxxxxxxxxx xxxxxxxxxxxxxx.(2009)(2009; et al., 2011)

*Palabras clave****:*** xxxxxxxxx, xxxxxxxx, xxxxxx

# Abstract

Abstract es el resumen traducido al idioma inglés en donde se incluyen las palabras claves. **Obligatoriamente** debe contener las palabras claves **(máximo tres)**.

Ejemplo:

Xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxxxxxxxx xxxxxxx xxxxxxx xxxxxx xxxxxx xxxxxxxx xxxxxxxxxx

# Introducción

Se sugiere presentar en máximo **dos páginas** y considerar los siguientes puntos:

Cómo dio respuesta al problema planteado,

El alcance de los objetivos y su cumplimiento,

Las facilidades u oportunidades, los inconvenientes o limitantes con los que se enfrentó en el desarrollo del trabajo,

La metodología utilizada, ( et al., 2019)(1990; et al., 2010, 2015, 2019)

Una breve explicación de los capítulos,

La importancia que tiene la investigación para la institución, empresa o usuarios y la sociedad en general.

Letra Arial N° 11, con sangría (1.27 cm), interlineado doble y justificado

# Capítulo uno

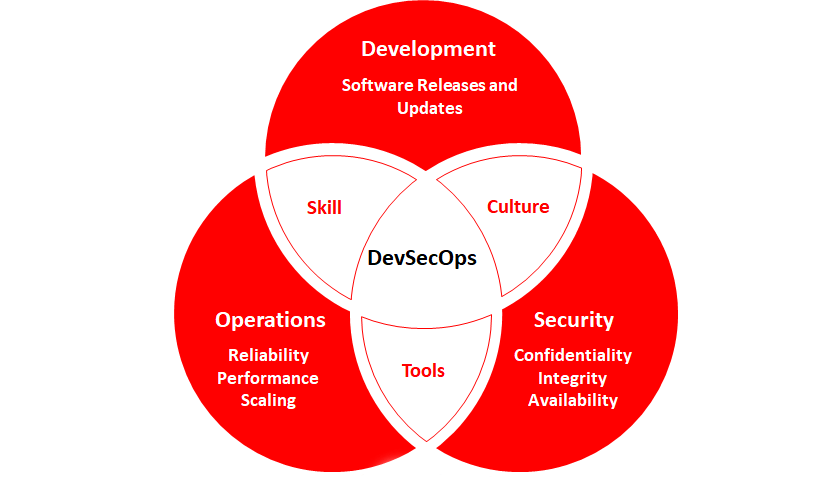
# Marco Teórico

## **¿Qué es DevSecOps?**

Es un enfoque, que se encarga de implementar la seguridad desde el inicio de la creación de una aplicación para evitar ataques de hacking. Combinando la parte de desarrollo + seguridad + operaciones, en un ciclo de entrega continuo y automatizado siendo responsables todos de la seguridad. Devsecops integra la seguridad a través de herramientas automatizas con un conjunto de estándares de comportamientos, conocimientos y hábitos(Shrivastava & Services, 2019) que con llevan a una cultura de seguridad como código figura 1. El propósito es ayudar a mejorar el tiempo de entrega y enviar automáticamente el código a producción cumpliendo los objetivos de seguridad.

**Figura 1**

*Que es DevSecOps*



Nota. Adapto de (Shrivastava & Services, 2019)

El enfoque Devsecops surge de la principal interrogante que presentaba devops, de cómo controlar la presencia de vulnerabilidades en sus equipos evitando afectar la calidad de la seguridad del software. Se debe conocer que Devsecops no es una idea o conceptos diferentes de Devops (Santana et al., 2021). Si no que amplía la filosofía de Devops integrando la seguridad en todo su proceso desde el inicio como un componente integral de todo el ciclo de vida de desarrollo de software (SDLC[[1]](#footnote-1)).

**Figura 2**

*Implementación de DevSecOps*



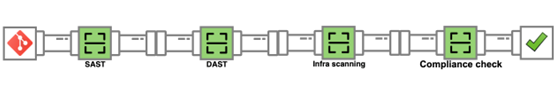
Nota. Adaptado de (Sentrio, 2021)

El objetivo principal para imprentar DevSecOps es automatizar, monitorear y aplicar la seguridad bien sea políticas, pruebas enfocadas a la seguridad, trazas de log, auditorías, monitorización o alarmado (Sentrio, 2021), en todas las fases del SDLC devops, es decir: planificar, desarrollar, construir, probar, lanzar, entregar, implementar, operar y monitorear Figura 2. Utilizando la arquitectura de microservicios, la integración continua y despliegue continuo (CI/CD)(Chandramouli, 2022). Basándose en la forma de un pipeline[[2]](#footnote-2) para realizar una variedad de pruebas y validaciones de seguridad automatizadas, sin requerir el trabajo manual de un operador humano. Estas pueden variar de una aplicación a otra según las necesidades de la empresa, el proceso de entrega de software, la madurez y el nivel de automatización. Uno de los problemas más comunes que se puede encontrar para poder implementar DevSecOps, es la velocidad porque es el elementó diferencial de Devops. Donde la integración de la seguridad no debe afectar al proceso sino acoplarse al ciclo del SDLC.

Para añadir la seguridad figura 2, hay diferentes formas como marcos o guías que recomiendan algunos pasos para realizar como “la Guía OWASP DevSecOps”. Que se centra en implementar un pipeline seguro utilizando las mejores prácticas e introducir herramientas desde la fase de desarrollo (Owasp, 2021). Además, trata de ayudar a promover la cultura de la seguridad en el proceso de desarrollo.

**Figura 3**

*Guía del proceso de implementación de DevSecOps*



      Nota. Adaptado de (Owasp, 2021).

En la figura 3 se muestra los pasos de la guía de Owasp para poder implementar en un pipeline los cuales son los siguientes:

* Detección de secretos
* SAST (prueba de seguridad de aplicaciones estáticas)
* DAST (prueba de seguridad de aplicaciones dinámicas)
* Escaneo de infraestructura
* Comprobación del cumplimento

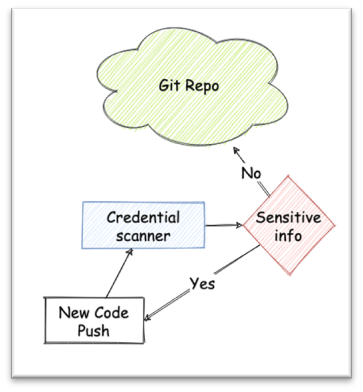
Estos procesos se pueden personalizar de acuerdo con el ciclo de vida de desarrollo de software (SDLC) o arquitectura de software. CI / CD es una ventaja para DevSecOps porque es un punto de entrada privilegiado para medidas y controles de seguridad. Sin embargo, cuando se utiliza herramientas de CI / CD para proporcionar automatización. Se debe colocar controles de seguridad en el software de construcción, implementación y automatización.

1. **Detección de secretos**

Al momento de desarrollar se debe asegurarse de que la información confidencial no se envíe a la ubicación donde se almacena todo el proyecto (GitHub, GitLab, etc). Porque puede ocasionar varios problemas como ser visibles en el historial. En el caso de las plataformas de alojamiento de código, los secretos aún pueden permanecer en la web y se pueden buscar después de ser eliminado el repositorio. Siguiendo el proceso de la figura 4 se debe escanear los commits y el repositorio para detectar cualquier información sensible como contraseña, clave secreta, confidencial, etc.

**Figura 4**

Proceso de detección de secretos



Nota. Adaptado de (Owasp, 2021)

Para esto se debe escanear el repositorio en busca de información confidencial y luego eliminarla; tenga en cuenta que cuando se filtra una credencial, ya está comprometida y debe invalidarse. Para detectar los secretos se debe verificar en varios lugares como:

* Detectar secretos existentes mediante la búsqueda en un repositorio de secretos existentes.
* Usar pre-commit[[3]](#footnote-3) para reconocer problemas simples antes de enviarlos a revisión de código.
* Detectando secretos en el pipeline.

La mejor ubicación es la de pre-commit, Esto asegura que antes de que un secreto entre en la base de código, es interceptado y el desarrollador o el committer recibe un mensaje. A continuación, se incluyen algunas herramientas útiles para escanear automáticamente los repositorios en busca de información confidencial. Los escaneos se pueden implementar directamente en nuestra tubería y ser repetibles y eficientes.

* Gittyleaks[[4]](#footnote-4) .- Busca información sensible en un repositorio de git.
* Git-secrets[[5]](#footnote-5).- Impide que se envié secretos y credenciales  a los repositorios de git.
* TruffleHog[[6]](#footnote-6). - Busca secretos en los repositorios de git, profundizando en el historial de confirmaciones y las ramas. Esto es eficaz para encontrar secretos cometidos accidentalmente.
* Repo-supervisor[[7]](#footnote-7).- Es una herramienta que ayuda a detectar secretos y contraseñas en el código.
* Git Hound[[8]](#footnote-8).- Es un complemento de Git donde ayuda a evitar que se confirmen datos confidenciales.

1. **Sast**

Las pruebas estáticas de seguridad de las aplicaciones (SAST) comprueban los puntos débiles de seguridad en los pipelines de CI. Existe desde hace tiempo y se basa en el concepto de white-box testing (Quintero & Joaquín, 2019). Una herramienta SAST funciona analizando códigos dados en un entorno sin tiempo de ejecución en busca de errores y rutas que conduzcan a debilidades o riesgos de seguridad. Permite a los desarrolladores encontrar vulnerabilidades de seguridad en el código fuente de la aplicación en una fase más temprana del ciclo de vida de desarrollo del software. Esto también garantiza la conformidad con las normas de codificación antes de que se ejecuten los códigos (Akujobi, 2021). Gracias a los hallazgos de vulnerabilidades en una fase temprana, los desarrolladores pueden resolver los problemas de seguridad antes de que causen un daño real. Por lo tanto, los desarrolladores utilizan las herramientas SAST en los IDE o en los pipelines. El escaneo estático es una buena manera de encontrar problemas de codificación como:

* Violaciones de sintaxis
* Vulnerabilidades de seguridad
* Errores de programación
* Codificación de violaciones estándar
* Valores indefinidos

La mayoría de las herramientas de análisis estático tienen el alcance de prueba limitado a un componente limitando la posibilidad de poder realizar pruebas en diferentes componentes. Por ejemplo, para una arquitectura de microservicio, las herramientas SAST probarán cada microservicio de forma independiente. A continuación, algunas herramientas que nos permiten realizar este proceso.

* SonarQube[[9]](#footnote-9).- Es una herramienta de revisión automática de código para detectar errores, vulnerabilidades y problemas en el código. Puede integrarse con su flujo de trabajo actual para permitir la inspección continua del código.
* Clair[[10]](#footnote-10).- Es un software de código abierto para el análisis estático de vulnerabilidades en contenedores de aplicaciones (actualmente incluye OCI y Docker ).
* Veracode[[11]](#footnote-11).- Es una herramienta de análisis estático que ofrece una evaluación automatizada bajo demanda del código base de su aplicación. Simplemente envíe su código a su plataforma en línea y recibirá un plan de reparación con resultados detallados de las vulnerabilidades y fallas dentro de la aplicación o dentro del código de terceros que contiene.
* CodeSweep[[12]](#footnote-12).- Es una extensión que permite a los desarrolladores verificar su código en busca de vulnerabilidades en cada extracción. La extensión está configurada para ejecutarse como una acción de GitHub y devuelve las vulnerabilidades identificadas en el código modificado.

1. **Dast**

Las pruebas de seguridad de aplicaciones dinámicas (DAST) se centran en determinar cómo responde una aplicación en funcionamiento a las solicitudes maliciosas. Puede encontrar vulnerabilidades de seguridad y debilidades en una aplicación en ejecución inyectando cargas útiles maliciosas para identificar fallas potenciales que permitan ataques como inyecciones de SQL o scripts entre sitios (XSS), etc. Las herramientas DAST son especialmente útil para detectar:

* Validación de entrada o salida
* Problemas de autenticación
* Errores de configuración del servidor

Las herramientas DAST permiten realizar sofisticados escaneos del lado del cliente y del lado del servidor sin necesidad del código fuente o la forma como está construida la aplicación, suelen requerir mínimas interacciones del usuario una vez configuradas. A continuación, algunas herramientas DAST:

* Zed attack proxy (Owasp zap)[[13]](#footnote-13).- es una herramienta de código abierto que ofrece OWASP para realizar pruebas de seguridad.
* StackHawk[[14]](#footnote-14).- Facilita a los desarrolladores la búsqueda, clasificación y corrección de errores de seguridad de las aplicaciones. Escanea el código de la aplicación en busca de errores, clasificando y corrigiendo, permitiendo acoplarse para automatice su canalización para evitar que errores futuros afecten a producción.

1. **Escaneo de Infraestructura.**

La infraestructura es un punto clave al momento de desplegar una aplicación en producción. Porque contiene todo lo necesario para ejecutar la aplicación. Si no está configurada correctamente esta puede afectar a la seguridad de la aplicación así sea que el código este seguro. El software moderno se basa en el uso de la automatización tanto a nivel de desarrollo como de operaciones gracias a la adopción de DevOps. Para impulsa dicha automatización se utiliza la práctica de Infraestructura como Código (IaC) donde se encarga de configurar y automatizar la administración de la infraestructura (contenedores, redes, balanceo de cargar, etc), permitiendo gestionar y preparar la infraestructura con código, en vez de hacerlo mediante procesos manuales.

Por lo tanto, para preparar la automatizar de la infraestructura con la IaC involucra a no tener que preparar ni gestionar manualmente los servidores, los sistemas operativos, el almacenamiento ni ningún otro elemento de la infraestructura los desarrolladores cada vez que desarrollan o implementan una aplicación. De este modo hay herramientas que permiten realizar este proceso como:

* Chef[[15]](#footnote-15) .- Es una herramienta de automatización que convierte la infraestructura en código administrando y configurar múltiples sistemas con facilidad.
* Red Hat Ansible Automation Platform[[16]](#footnote-16).-  Es una plataforma para diseñar y gestionar la automatización de la TI según sus necesidades.
* Terraform[[17]](#footnote-17).-Es una herramienta que proporciona un flujo de trabajo CLI coherente para gestionar cientos de servicios en la nube.
* AWS CloudFormation[[18]](#footnote-18).- Es un servicio que brinda a desarrolladores una manera sencilla de crear una colección de recursos de AWS y de terceros.

1. **Comprobación del cumplimiento**

Devsecops promueve la adopción de la comprobación del cumplimiento o compliance check ayudando a definir los requisitos de cumplimiento de una manera que los humanos y las máquinas puedan leer fácilmente. Esto permite al personal de SecOps desarrollar políticas de cumplimiento como código, sin necesidad de utilizar lenguajes de programación técnicos. Estas políticas de cumplimiento se pueden almacenar en un sistema de control de versiones de código fuente como Git para un monitoreo continuo del cumplimiento durante la fase de desarrollo para poder obtener comentarios en tiempo real sobre sus compromisos de código utilizando herramientas de monitoreo de cumplimiento como es Chef InSpec[[19]](#footnote-19), donde permite convertir los marcos de cumplimiento y las políticas de TI en pruebas automatizadas incorporando en los pipelines. Esto permite a los desarrolladores evaluar de manera proactiva la revisión del código para verificar el funcionamiento de la aplicación o servicio si se está acoplando al cumplimiento de la empresa o políticas de TI (Spiros Psarris, 2020).

Todos estos pasos que indica la guía de owasp se debe añadir en un ciclo de vida de DevOps implicando fases repetitivas que representan las capacidades, procesos y técnicas cruciales para el desarrollo figura 2. Cada ciclo encuentra una colaboración separada pero constante de equipos y comunicación para garantizar que se mantengan la velocidad, la alineación y la calidad del proceso. A continuación, se indica cada fase de devops incorporando los procesos de seguridad antes mencionados

Toda vida no está finalizada

* **Plan**

El proceso implica un planteamiento del problema y una definición del alcance para identificar los recursos necesarios que se utilizarán durante todo el periodo. Después, los desarrolladores revisan los últimos sistemas y establecen sus objetivos. Posteriormente, establecen la practicabilidad y la viabilidad del estudio para determinar el periodo que se tardaría en realizar. Por último, el equipo considera las posibles amenazas, riesgos, limitaciones, seguridad e integraciones del sistema. Elaborando un documento de especificaciones de requisitos de software (SRS) viable para todo el proyecto.

* **Build**

Una vez que la etapa inicial de la planificación del proyecto se logra y se aprueba, el desarrollador hace que el sistema diseñe y cambie el documento SRS en un componente lógico con especificaciones detalladas para la implementación a través de la codificación. A esto le sigue la creación de contingencias, capacitación de equipos y mantenimiento a través de un plan de operación. Una vez que se completa el diseño, los documentos pasan al nivel de implementación donde implica codificar, utilizando uno o varios lenguajes de programación específicos para el código fuente. Los módulos del sistema se combinan en un entorno de test para la detección de errores y defectos. Luego, se prepara un informe de prueba mediante la asignación de recursos una vez que se ha integrado el sistema.

* **Deploy**

Las aplicaciones se despliegan en los entornos de puesta en escena y producción, entre las herramientas utilizadas en estas etapas figuran los instrumentos de gestión y aprovisionamiento para la infraestructura como código. También desempeñan un papel fundamental en estas etapas, debido a la gran escala que existe en estas etapas. Con el advenimiento de las tecnologías de virtualización y de la nube, los entornos de puesta en escena y producción pueden estar compuestos hoy en día por cientos o incluso miles de servidores y los instrumentos de vigilancia permiten a las organizaciones supervisar las aplicaciones desplegadas en producción.

* **Monitoring**

Se encarga de supervisar el código desplegado en producción para verificar si esta correctamente funcionando o identificar los diversos errores y fallas para generar alertas. Estas se generan y resuelven automáticamente o recomiendan las acciones necesarias que los desarrolladores y el equipo de operaciones deben emprender para solucionarlas. El proceso de mantenimiento y soporte se ejecuta, por tanto, de forma automática sin necesidad de intervención física. Eso ayuda a los desarrolladores a reenfocar su energía en otras acciones productivas.

# Capítulo dos

# Nombre del capítulo

## **Xxxxxxxxxx xxxxxx xxxxxxx xxxx**

Xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx.

Xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx.

## **2.2 Xxxxxxx xxxxxx xxxxxxxxx xxxxxxx**

Xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx.

Xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx.

Xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

# Capítulo tres

# Nombre del capítulo

* 1. **Xxxxxxxxx**

Xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx.

Xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx.

* 1. **Xxxxxxxxx xxxxxxx xxxxxx xxxxxx xxxxxxx**

Xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx.

Xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx.

# Conclusiones

Se redactan los puntos más sobresalientes, debilidades o fortalezas del proyecto o investigación, observados o descubiertos durante la ejecución del Trabajo de Titulación, se recomienda redactar por cada conclusión, una recomendación.

Xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx.

Xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx.

Letra Arial N° 11, las conclusiones empiezan en una nueva página, cada párrafo inicia con sangría y no colocar viñetas o numeración

# Recomendaciones

En esta parte debes sugerir temas para futuras investigaciones y puedan aportar a la academia.

Xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx.

Xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx.

Letra Arial N° 11, las recomendaciones empiezan en una nueva página, cada párrafo inicia con sangría y no colocar viñetas o numeración

# Referencias

Akujobi, A. C. (2021). *A Model For Measuring Improvement Of Security In Continuous Integration pipelines*.

Chandramouli, R. (2022). Implementation of DevSecOps for a Microservices-based Application with Service Mesh Implementation of DevSecOps for a Microservices-based Application with Service Mesh. *NIST Special Publication*.

Owasp. (2021, May 1). *DevSecOpsGuideline Secrets Management*. Github. https://github.com/OWASP/DevSecOpsGuideline/blob/master/document/01a-Secrets-Management.md

Quintero, C., & Joaquín, J. (2019). *DevSecOps : integración de herramientas SAST , DAST y de análisis de Dockers en un sistema de integración continua .*

Santana, G., Neto, M., Sapata, F., Muñoz, M., Moraes, A. M. S. P., Morais, T., & Goldfarb, D. L. (2021). DevSecOps in AWS. *AWS Certified Security Study Guide*, 405–441. https://doi.org/10.1002/9781119658856.app3

Sentrio. (2021). *¿Qué es DevSecOps? Integra la seguridad en DevOps*. https://sentrio.io/blog/que-es-devsecops-vs-devops/

Shrivastava, A., & Services, N. G. (2019). DevSecOps What, Why and How Anant Shrivastava NotSoSecure Global Services. *Black Hat*. https://anantshri.info

Spiros Psarris. (2020, September 1). *Using DevSecOps to Facilitate Compliance - Reblaze Blog*. https://www.reblaze.com/blog/using-devsecops-to-facilitate-compliance/

# Apéndice

Se incluye de acuerdo al orden citado en el cuerpo del Trabajo de Titulación (TT).

# Apéndice A. Xxxxxxx xxxxxx xxxxxx

Xxxxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxx xxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxx.

Arial N° 10, negrita y sin punto final

Arial N° 10, sin negrita, cursiva y sin punto final

**Tabla A1**

*Xxxxxxxxxxx*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Internet** | **Urbano** | | **Rural** | |
| 06 a 09 | 10 a 11 | 06 a 09 | 10 a 11 |
| Grado a | 10 | 15 | 2 | 4 |
| Grado b | 15 | 18 | 3 | 3 |
| Grado c | 15 | 10 | 3 | 1 |
| Grado d | 10 | 20 | 2 | 1 |
| **Total** | 50 | 63 | 10 | 9 |

*Nota*. En esta tabla se observa que los niños del sector urbano tienen mayor

acceso al Internet.

Arial N° 10, sin negrita, la palabra *Nota* en cursiva e interlineado doble

# Apéndice B. Xxxxxxx xxxxxx xxxxxx

Xxxxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxx xxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxx.

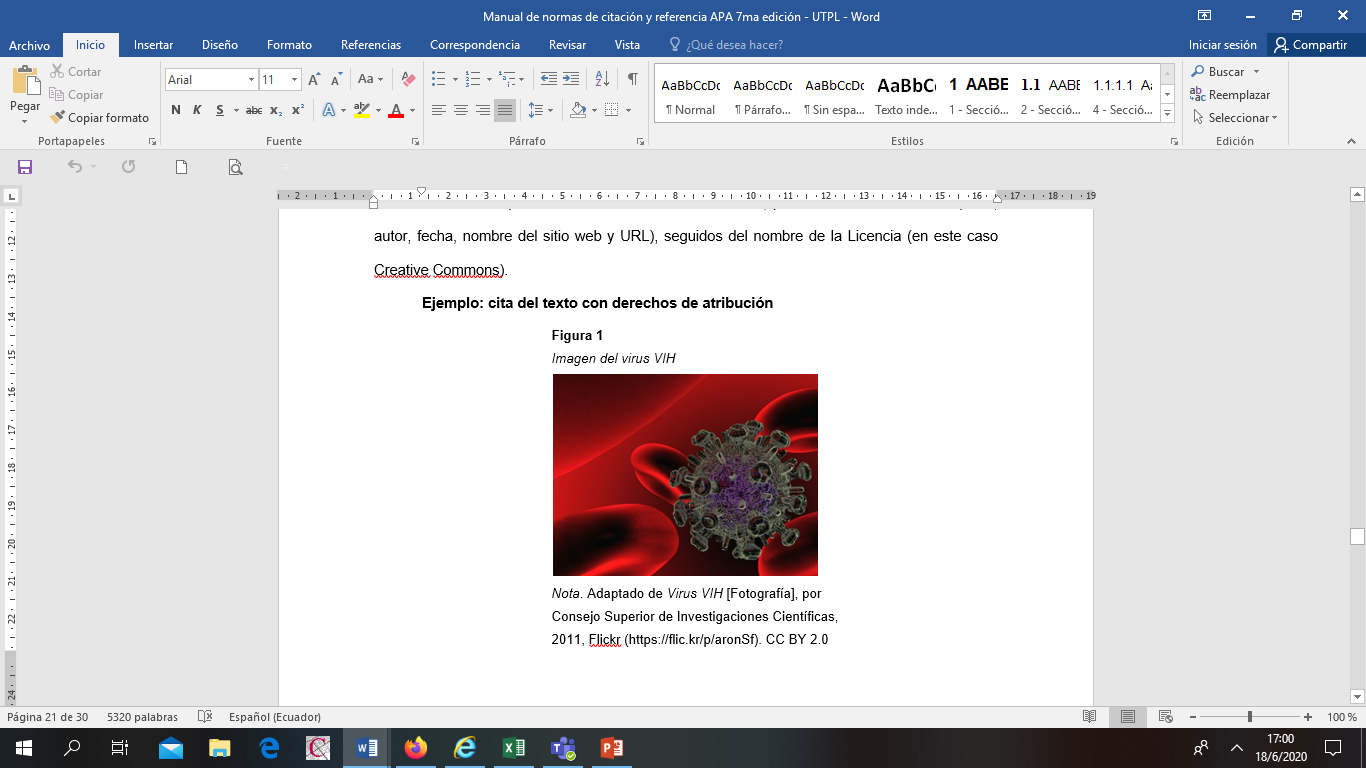
Interlineado doble

Arial N° 10, en negrita y sin punto final

Arial N° 10, sin negrita, cursiva y sin punto final

**Figura B1**

Imagen del virus VIH



*Nota.* Adaptado de *Virus VIH* [Fotografia], por Consejo Superior de

Investigaciones Científcas, 2011, Flickr ([flic.kr/p/aronSf](https://flic.kr/p/aronSf)). CC BY 2.0.

Arial N° 10, sin negrita, la palabra *Nota* en cursiva e interlineado doble

# Apéndice C. Xxxxxxx xxxxxx xxxxxx

Xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxx.

Xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxx.

Xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxx.

Letra Arial N° 11, interlineado doble y sin cursiva

1. https://bit.ly/3NTzju2 [↑](#footnote-ref-1)
2. https://bit.ly/37uiuoW [↑](#footnote-ref-2)
3. https://bit.ly/3rbILiu [↑](#footnote-ref-3)
4. https://bit.ly/3D74o7F  [↑](#footnote-ref-4)
5. https://bit.ly/3n163WW [↑](#footnote-ref-5)
6. https://bit.ly/30efZU5 [↑](#footnote-ref-6)
7. https://bit.ly/3ogwasi [↑](#footnote-ref-7)
8. https://bit.ly/3wxuq1s  [↑](#footnote-ref-8)
9. https://bit.ly/3qCM4jD [↑](#footnote-ref-9)
10. https://bit.ly/3kwC2MQ [↑](#footnote-ref-10)
11. https://bit.ly/3oojsHV [↑](#footnote-ref-11)
12. https://bit.ly/3qznJem [↑](#footnote-ref-12)
13. https://bit.ly/3HhmWEK [↑](#footnote-ref-13)
14. https://bit.ly/3niUHh0 [↑](#footnote-ref-14)
15. https://bit.ly/3FnXJXC [↑](#footnote-ref-15)
16. https://red.ht/3ckdnGY [↑](#footnote-ref-16)
17. https://bit.ly/3Fe83Bo [↑](#footnote-ref-17)
18. https://go.aws/3DjzAAK  [↑](#footnote-ref-18)
19. https://bit.ly/3qMKiwk  [↑](#footnote-ref-19)