Universidad de Buenos Aires

Facultades de Ciencias Económicas, Cs. Exactas y Naturales e Ingeniería

Carrera de Especialización en Seguridad Informática

Trabajo Final de Especialización

Tema:

Automatización de Procesos

Título:

Automatización de Pruebas de Seguridad en el Ciclo de Vida del Desarrollo de Software

Autor: Lic. Rubén Darío Aybar

Tutor: Ing. Hugo Pagola

Año 2020
Cohorte 2017

Declaración Jurada de origen de los contenidos

Por medio de la presente, el autor manifiesta conocer y aceptar el Reglamento de Trabajos Finales vigente y se hace responsable que la totalidad de los contenidos del presente documento son originales y de su creación exclusiva, o bien pertenecen a terceros u otras fuentes, que han sido adecuadamente referenciados y cuya inclusión no infringe la legislación Nacional e Internacional de Propiedad Intelectual.

FIRMADO
Rubén Darío Aybar
D.N.I. 28.061.798

Resumen

En el siguiente Trabajo Final de Especialización, se desarrollará el estado del arte de la práctica denominada DevOps (siendo un acrónimo inglés de las palabas development -desarrollo- y operations -operaciones-), se buscará introducir al lector en sus orígenes hasta llegar a su actualidad, realizando una revisión de textos varios en los que se describe esta práctica de Ingeniería de Software.

Se abordarán diferentes capítulos en los cuales se explicara el Ciclo de vida del desarrollo de software, pasando desde el modelo de Cascada hasta los modelos modernos que utilizan tecnologías agiles, se repasara la historia de DevOps, su concepto y sus principales componentes, se desarrollara el concepto más moderno DevSecOps, con lo que se buscara, acercar al lector a un contexto general del mismo y por último se describirá un ejemplo práctico, mediante la implementación de un laboratorio desarrollado con diferentes herramientas de código abierto. [1]

Palabras claves

Transformación Digital – SDLC – DevOps – DevSecOps – Cultura – Personas – Procesos – Tecnología – Automatización – OWASP Top 10

Agradecimientos

A mí familia (Monserrat, Luca y Alicia) que son mi pilar fundamental, ellos son los que día a día están apoyando mis proyectos, mis locuras y los que ante toda adversidad me sostienen y no me dejan caer. Los Amo!!!

A mis padres Alicia y Eduardo, mi mamá Alicia siempre guiándome desde el cielo, ella está en mi mente y mi corazón, y sé que poniéndose orgullosa por los logros que fui consiguiendo en mi vida.

Mi papa siempre brindándome su apoyo en cada proyecto y poniéndose orgulloso con cada hito que voy consiguiente en este camino llamado vida.

A mi tutor el Ing. Hugo Pagola quien tomo la responsabilidad de brindarme su tiempo, que es lo más valioso que tiene el ser humano, y su conocimiento aportándome sus recomendaciones sobre cada versión de la obra.

Un agradecimiento especial a mí Amigo, Colega y Compañero de Especialización el Lic. Darío Osvaldo Rizzo quien siempre me brindó su apoyo para que escriba esta obra y me ayudo con sus correcciones, leyendo cada una de las nuevas versiones.

A mis compañeros de Especialización quienes hicieron el viaje al cual todos nos embarcamos más placentero, ya que fue una Cohorte 2017 de excelencia, siempre predispuestos a ayudar a colaborar y que todos lleguemos al mismo puerto.

Por ultimo a mis amigos y colegas a quienes agradezco de todo corazón el siempre confiar en mí y estar brindando sus apoyo.

Contenido

Declaración Jurada de origen de los contenidos I
Resumen II
Palabras claves III
Agradecimientos IV
Tabla de Imagenes VII
Capítulo 1 1
Introducción 1
1 SDLC 4
1.1 Concepto 4
1.2 Historia 5
2 Costo de Reparar un Error según la Etapa del SDLC 7
Capítulo 2 9
3 Historia de DevOps 9
3.1 Acuñar concepto DevOps (Andrew Clay Shafer y Patrick Debois) 9
3.2 Nacen los DevOpsDays 10
3.3 Otros hechos relevantes 12
4 DevOps 14
4.1 Etimología de DevOps 14
4.2 ¿Que no es DevOps? 15
4.3 Cultura DevOps 16
5 CI/CD y la otra CD 17
5.1 CI: Integración Continua 19
5.2 CD: Distribución Continua 20
5.3 La otra CD: Implementación Continua 20
6 Pipeline 22
6.1 Definición de Pipeline 22
6.2 Estructura de un Pipeline 22
Capítulo 3 24
7 DevSecOps 24
7.1 ¿Qué es DevSecOps? 24
7.2 Seguridad en DevOps = DevSecOps 25
7.3 Modelado de Amenazas 26
7.4 SAST 28
7.5 DAST 29
7.6 Test de Intrusión 29

7.7 OWASP Top 10 30
Capítulo 4 32
8 Herramientas de Código Abierto para Laboratorio de DevSecOps 32
8.1 SonarLint 32
8.2 Github 33
8.3 Jenkins 34
8.4 SonarQube 36
8.5 OWASP ZAP 37
8.6 Diferencia entre SonarLint y SonarQube 37
9 Laboratorio de DevSecOps 39
9.1 Arquitectura del Laboratorio 39
9.2 Desarrollo del Proceso de Automatización 40
10 Conclusiones 45
Bibliografía 46
ANEXO 1 1

Tabla de Imagenes

Imagen 1 Fases del Ciclo de Vida del Desarrollo de Software 5
Imagen 2 Evolución de las Metodologías de Desarrollo 6
Imagen 3 Costo de Reparar un Error 8
Imagen 4 Logo de la Agile 2008 Conference 9
Imagen 5 Afiche de la conferencia Velocity 09 11
Imagen 6 Logo del primer DevOpsDays 2009 11
Imagen 7 Línea de Tiempo DevOps 13
Imagen 8 development -desarrollo- y operations -operaciones- 14
Imagen 9 DevOps Comunicación en tiempo real 14
Imagen 10 Modelo Tradicional vs Modelo DevOps 18
Imagen 11 Integración Continua - Despliegues Continuos - Implementaciones Continuos 21
Imagen 12 Pipeline 22
Imagen 13 Seguridad en todo el SDLC 25
Imagen 14 Pruebas de Seguridad en SDLC 26
Imagen 15 Modelado de Amenazas en el SDLC 27
Imagen 16 OWASP Top 10 2017 31
Imagen 17 SonarLint en el Marketplace IDE Pycharm 33
Imagen 18 Cuenta de Github para el Proyecto de Automatización 34
Imagen 19 Jenkins con Jobs generados 35
Imagen 20 SonarQube con Análisis estático de la Aplicación DVWA 37
Imagen 21 Arquitectura Laboratorio DevSecOps 40
Imagen 22 Jenkins con Jobs del Pipeline Especialización 41
Imagen 23 Pipeline Especialización en ejecución 41
Imagen 24 Finishid: SUCCESS Job DVWA_Job 42
Imagen 25 Finishid: SUCCESS Job SonarQube_Job 42
Imagen 26 Finishid: SUCCESS Job OWASPZAP_Job 42
Imagen 27 Pipeline Especialización finalizado correctamente 43
Imagen 28 GUI de SonarQube con el resultado del Análisis Estático de Código 43
Imagen 29 Resultado del Job OWASPZAP_Job con el reporte generado 44
Imagen 30 reporte del Análisis Dinámico de Código con OWASP ZAP 44

"Hay una fuerza motriz más poderosa que el vapor, la electricidad y la energía atómica: la voluntad."

Albert Einstein

Capítulo 1

Introducción

A día de hoy, año 2020, las organizaciones de diferentes sectores están sumergidas en las nuevas tecnologías para poder desarrollar sus actividades diarias.

Vamos a mencionar unos ejemplos, los bancos cambiaron la forma de presentar sus productos acercándoles a sus clientes nuevas ventajas como la de poder operar desde la comodidad de su propio hogar a través del Homebanking, o desde cualquier lugar, solamente con tener un punto de acceso a internet y un dispositivo móvil para poder realizar sus operaciones desde la banca móvil estratégicamente desarrollada.

Como otros ejemplos también podemos tomar las aplicaciones móviles para solicitar autos como lo son Uber o Cabify o las aplicaciones de las grandes cadenas de supermercados que se pueden acceder desde los diferentes navegadores de una computadora o desde los dispositivos móviles, solamente por citar algunos casos de una forma no exhaustiva.

Pero entonces, ¿cómo definimos este concepto que se dio en llamar Transformación Digital?, según Salesforce "La transformación digital es el uso de la tecnología para mejorar radicalmente el desempeño y el alcance de las empresas.

Por un lado, la transformación digital la ha impuesto la creciente demanda de los consumidores por obtener experiencias digitales simples e integradas a través de los diversos canales de servicio.

Por otro lado, esta transformación ahora es posible gracias a las diversas revoluciones tecnológicas que permiten a las empresas digitalizar todos los procesos de sus negocios.” [2]

Así también Salesforce nos brinda unas cifras en dólares del año 2017 que nos da un panorama de cómo va creciendo la transformación digital “De hecho, según la consultora IDC, en 2017, las empresas alrededor del mundo invertirán $1.2 billones de dólares estadounidenses en transformación digital. Las empresas de manufactura discreta y de procesos serán responsables del 30% de esta inversión. Otros sectores como los servicios financieros (bancos y aseguradoras), minoristas y de salud también impulsarán las inversiones en el tema.” [2]

Pero siempre se deben tener en cuenta los problemas de seguridad y por ello diversos estudios como el del NIST[1] en su Publicación especial 800-64 Revisión 2 dice que “El 92% de las vulnerabilidades están en las aplicaciones, no en las redes” [3], con esta afirmación se está comunicando que las vulnerabilidades en su mayoría se encuentran en las piezas de software desarrollado y no en los dispositivos o servicios que soportan a las aplicaciones.

El dato del NIST no es con el único que nos podemos quedar, encontramos en una entrada del blog de la página de MTP, que es una compañía líder en Servicios de Aseguramiento Digital que cuenta con veinte años de experiencia y con capacidad para operar a nivel global, nos dice:

“Según datos del National Institute for Standard and Technology (NIST), el 92% de las vulnerabilidades están en el software. Según SAP, el 84% de los ciberataques se centran en la capa de aplicación. Según BlackDuck, el 67% de las aplicaciones contienen vulnerabilidades de código abierto… Estos son solo algunos datos que podemos encontrar haciendo una sencilla búsqueda en Internet. La búsqueda puede ser todo lo exhaustiva que queramos, pero siempre va a apuntar en la misma dirección: nuestros negocios dependen cada vez más de las aplicaciones; las

[1] El Instituto Nacional de Estándares y Tecnología (NIST por sus siglas en inglés, National Institute of Standards and Technology)

aplicaciones se han convertido en la puerta de entrada preferida de los hackers; como consecuencia, deberíamos centrarnos en desarrollar aplicaciones seguras. Ese puede ser un buen punto de partida.” [4]

Este preámbulo nos presenta un nuevo desafío, ya que al crecer la demanda de sitios y aplicaciones que serán accedidos por los clientes de las organizaciones que se vayan introduciendo en la trasformación digital se deberá brindar mayor seguridad a cada una de las fases del ciclo de vida del desarrollo de software desde su nacimiento hasta su extinción.

Por todo lo expuesto se debe comenzar a pensar en instaurar el concepto de DevOps y DevSecOps en las organizaciones, ello es muy importante, ya que estos no son conceptos nuevos, pero que se pusieron muy de moda en los últimos años, lo que representa una transformación y un cambio cultural en las mismas, este cambio se debe comenzar en la importancia de poner el foco en las personas, los procesos y la tecnología como se podrá apreciar en las secciones posteriores de esta obra.

1 SDLC

1.1 Concepto

El Ciclo de Vida del Desarrollo de Software (SDLC de sus siglas en inglés Software Development Life Cicle) es un proceso compuesto por fases, de las cuales se lleva adelante el desarrollo de una pieza de software en la Ingeniería de Software.

Si citamos una definición más formal se puede decir que el "Ciclo de vida del software. Es el proceso que se sigue para construir, entregar y hacer evolucionar el software, desde la concepción de una idea hasta la entrega y retiro del sistema. Se definen las distintas fases intermedias que se requieren para validar el desarrollo de un software, es decir, para garantizar que el software cumpla los requisitos para la aplicación y verificación de los procedimientos de desarrollo, se asegura de que los métodos utilizados son apropiados." [5]

Este proceso se compone generalmente por cinco fases estándar, aunque según la metodología que se utilice pueden ser o más o menos, las cuales podemos enumerar como:

- Requerimientos/Análisis
- Diseño
- Implementación
- Testeo
- Evolución

La norma ISO[2] 12207 define al Ciclo de Vida del Desarrollo de Software como "un marco de referencia que contiene las actividades y las tareas involucradas en el desarrollo, la explotación y mantenimiento de un producto de software desde la definición hasta la finalización de su uso." [6]

En el siguiente grafico se pueden apreciar las etapas genéricas que componen el Ciclo de Vida del Desarrollo de Software.

[2] Organización Internacional de Normalización (ISO por sus siglas en inglés, International Organization for Standardization)

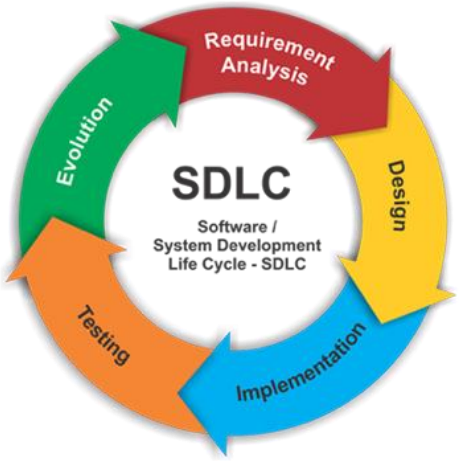


Imagen 1 Fases del Ciclo de Vida del Desarrollo de Software
Fuente: *[7]*

1.2 Historia

El Ciclo de Vida del Desarrollo de Software tiene sus orígenes en la década de los 60’ ya que se contaba con la necesidad de apalear a una metodología que delinee los pasos a seguir y en base a ella desarrollar software de una forma ordenada, eficiente y que pueda ser repetida a lo largo del tiempo.

En estas primeras metodologías se seguían las fases de forma secuencial, ya que se esperaba la finalización de cada una de la fase anterior para poder ser utilizada como entrada de datos de la siguiente, finalizado el ciclo se volvía a comenzar con el proceso.

En la década de los 90’ desde “el Departamento de Defensa de Estados Unidos (el mayor contratista mundial de proyectos de desarrollo de software) cambió oficialmente sus estándares de desarrollo de software y descartó el modelo en cascada para introducir el estándar 498, que utiliza un modelo iterativo de desarrollo de software.

Los modelos iterativos consisten en descomponer un proyecto de desarrollo de software en una serie de subproyectos de menor envergadura.

Estos subproyectos deben diseñarse de tal forma que cada uno de ellos aporte funcionalidad nueva para el sistema desde el punto de vista del usuario final del mismo.” [8]

En el año 2000 surgen nuevas metodologías a las cuales denominaron “agiles”, como las metodologías Scrum, kanban o el Modelo de Programación Extrema entre otras que son anteriores al 2000 pero también consideradas metodologías agiles, y se caracterizan por realizar entregas por etapas de los sistemas desarrollados, buscando de esta manera ser más flexibles a cambios, fomentar el trabajo en equipo y la comunicación entre sí, minimizar riesgos y dejando al cliente ser parte del equipo.

Aunque también hay que destacar que las metodologías agiles son más adecuadas para proyectos cortos y no de largo alcance.

A continuación se nombran las principales metodología conforme fueron surgiendo en la historia del Ciclo de Vida del Desarrollo de Software:

- Modelo Corregir y Codificar
- Modelo de Ciclo de Vida Clásico o Modelo de Cascada (Waterfall).
- Modelos Evolutivos como el Modelo Evo de Tom Gilb
- Modelo en espiral de Boehm
- Modelos Agiles como el Modelo de Programación Extrema.

En el grafico siguiente se puede apreciar una línea de tiempo con la evolución de las metodologías de desarrollo según fueron surgiendo en el tiempo:

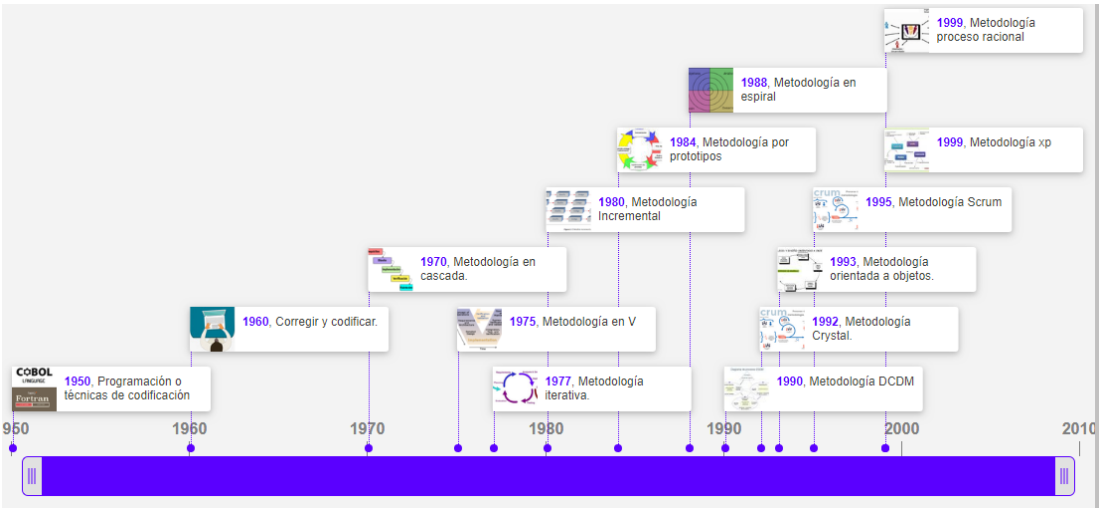


Imagen 2 Evolución de las Metodologías de Desarrollo
Fuente: *[9]*

2 Costo de Reparar un Error según la Etapa del SDLC

Uno de los principales problemas que se producen al comenzar un nuevo proyecto de desarrollo es la inclusión de errores en el mismo, según la metodología y la etapa en la que se encuentre el proyecto y sea detectado el error, generara un costo de reparación exponencialmente mayor.

Estos errores pueden ser de funcionalidad o de seguridad, según el tipo de error puede ser descubierto por el personal de QA en el caso de ser funcionales o por el personal de seguridad en caso de ser una vulnerabilidad, aunque también pueden ser detectados por personas ajenas al proyecto, como ser ciber delincuentes y aprovecharse de ellos para su propio beneficio.

Por lo tanto siempre se espera que surjan errores durante las etapas del proyecto. Una desalineación con los requisitos o incluso si los requisitos son incorrectos, generalmente conducen a un defecto que podría impedir implementar la aplicación en producción y retrasar todo el proceso.

Si se toma como ejemplo la metodología en cascada, corregir un error en la fase de requisitos es muy sencillo: simplemente se debe actualizar los diagramas/documentación.

Si un ingeniero de control de calidad captura el mismo error en la fase de evaluación, se deberán realizar las siguientes acciones para su corrección:

- Actualizar los diagramas/documentos
- Crear una nueva versión de la aplicación
- Implementar la nueva versión en el entorno de control de calidad

Como se puede apreciar en el siguiente gráfico, cuanto más a la izquierda sea detectado un error menor será el costo monetario y de esfuerzo humano para su corrección, esto quiere decir en etapas más tempranas, más fácil será su solución y menos costo generara para la empresa.

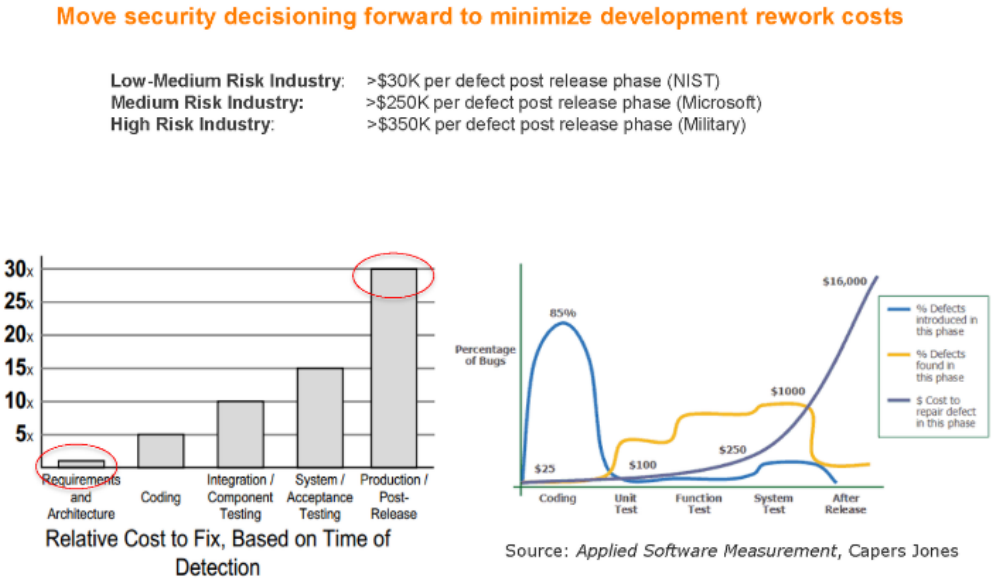
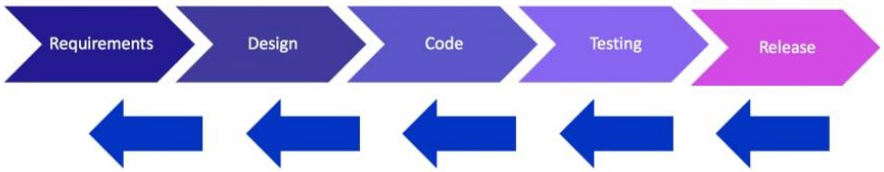


Imagen 3 Costo de Reparar un Error
Fuente: *[10]*

Esto último es comentado en su blog por Tanya Janca y lleva el nombre de “desplazarse a la izquierda” (del inglés “push left”[3]) “En todas las charlas y artículos que he escrito y en todos los consejos que he dado, siempre les digo a las personas que deben "desplazarse a la izquierda". Cuando las personas de seguridad dicen que quieren "desplazarse a la izquierda", se refieren al lado izquierdo del Ciclo de vida del desarrollo del sistema (SDLC), que es la forma en que los ingenieros de software describen la metodología o el proceso para crear software. Digo "empujar" porque a veces no estoy invitado a "cambiar".

Si mira la imagen a continuación, cuanto más "izquierda" mire, más temprano estará en el proceso. Cuando decimos que queremos "empujar a la izquierda", queremos decir que queremos iniciar la seguridad desde el principio y realizar la seguridad en cada paso del SDLC.” [11]



[3] “Pushing Left with Tanya Janca” https://www.youtube.com/watch?v=Ff-ZZYYDpI8

"La primera regla de cualquier tecnología que se utilice en una empresa, es que la automatización aplicada a una operación eficiente aumentara la eficiencia.
La segunda es que la automatización aplicada a una operación ineficiente aumentara la ineficiencia."

Bill Gates

Capítulo 2

3 Historia de DevOps

3.1 Acuñar concepto DevOps (Andrew Clay Shafer y Patrick Debois)

Para poder conocer la historia del movimiento o cultura DevOps, se debe presentar la historia de dos personas quienes acuñaron este término por el año 2009, el primero de ellos es Andrew Clay Shafer y el segundo es Patrick Debois. [12]

La historia comienza en el año 2008, cuando Andrew Clay Shafer iba a brindar una charla en un evento llamado "Agile 2008 Conference" que se realizó en Toronto, Canadá.

Imagen 4 Logo de la Agile 2008 Conference
Fuente: *[13]*

Al momento de brindar su charla surgió un inconveniente, solo había un participante en el público para escucharlo, por lo tanto Andrew decide cancelar su charla.

Lo que no sabía Andrew es que la persona que estaba esperando sentado para escuchar su charla era Patrick Debois que había concurrido específicamente para ello desde Bélgica.

Patrick Debois, oriundo de Bélgica, era un Administrador de Infraestructura que por los problemas que había detectado en la empresa en la que trabajaba al tener que realizar demasiados despliegues solicitados por el área de desarrollo, comenzaba a pensar en la automatización de estos procesos.

Aunque Andrew había cancelado su exposición en el evento, se pudieron encontrar y charlar sobre como agilizar las tareas de despliegue para el área de infraestructura y el área de desarrollo. De esta interacción surgió la idea de crear un grupo en Google para que la comunidad pueda opinar y aportar ideas, que dieron como nombre “Agile System Adminstrators Group”. [14]

3.2 Nacen los DevOpsDays

Un año después del encuentro entre Andrew Clay Shafer y Patrick Debois en Toronto, se celebra una conferencia llamada “Velocity web Performance and Operations Conference ” en San José de (California), en la cual “John Allspaw y Paul Hammond presentaron "10 despliegues por día: Cooperación entre desarrollo y operaciones en Flickr", donde mostraron cómo colaboraron (desarrollo y operaciones) y utilizaron prácticas de integración continua para hacer del despliegue una parte del trabajo diario de todos.” [15]

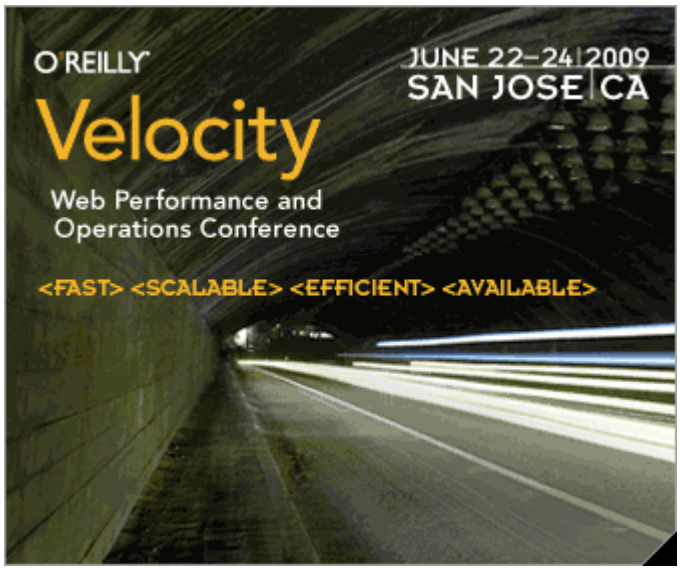


Imagen 5 Afiche de la conferencia Velocity 09
Fuente: *[16]*

A esta conferencia no pudo asistir Patrick Debois por su lejanía, pero de ella le surgió la posibilidad de crear su propia conferencia que llamo "DevopsDays", este evento se llevó a cabo en la ciudad Ghent, Belgica para fines de ese mismo año y en la cual se utilizó el hashtag #DevOps para ser referenciada en las redes sociales, teniendo una amplia repercusión.

Imagen 6 Logo del primer DevOpsDays 2009
Fuente: *[17]*

La conferencia DevOpsDays sigue creciendo año tras año, durante todo el año 2019 fue realizada en 80 sedes alrededor del mundo.

3.3 Otros hechos relevantes

En los años 2009 y 2010 también se pudo apreciar la proliferación de herramientas Open Source para la automatización de procesos, como fueron los casos de Jenkins, Chef y Puppet.

En el año 2013 se escribe una novela, la cual es una referencia que no puede faltar al momento de contar la historia de DevOps, “The Phoenix Project: A Novel about IT, DevOps, and Helping Your Business Win” es una novela escrita de 2013 por Gene Kim, Kevin Behr y George Spafford, que, además de haber sido un “best seller” en el área de tecnología, debe ser en estos últimos años el libro más citado cuando se habla de DevOps. Así, el libro se ha merecido un post.

“La novela cuenta la historia de un manager de TI, llamado Bill, que trabaja para una empresa llamada Parts Unlimited, que se dedica al área de la automoción, etc., y al que le dan noventa días para rescatar a un proyecto que va muy retrasado y por encima de presupuesto, cuyo nombre es el proyecto Phoenix. Vamos, la vida misma.” [18]

A continuación se muestra una línea de tiempo en la cual se puede apreciar cómo fue la evolución desde el año 2007 con Patrick Debois de DevOps

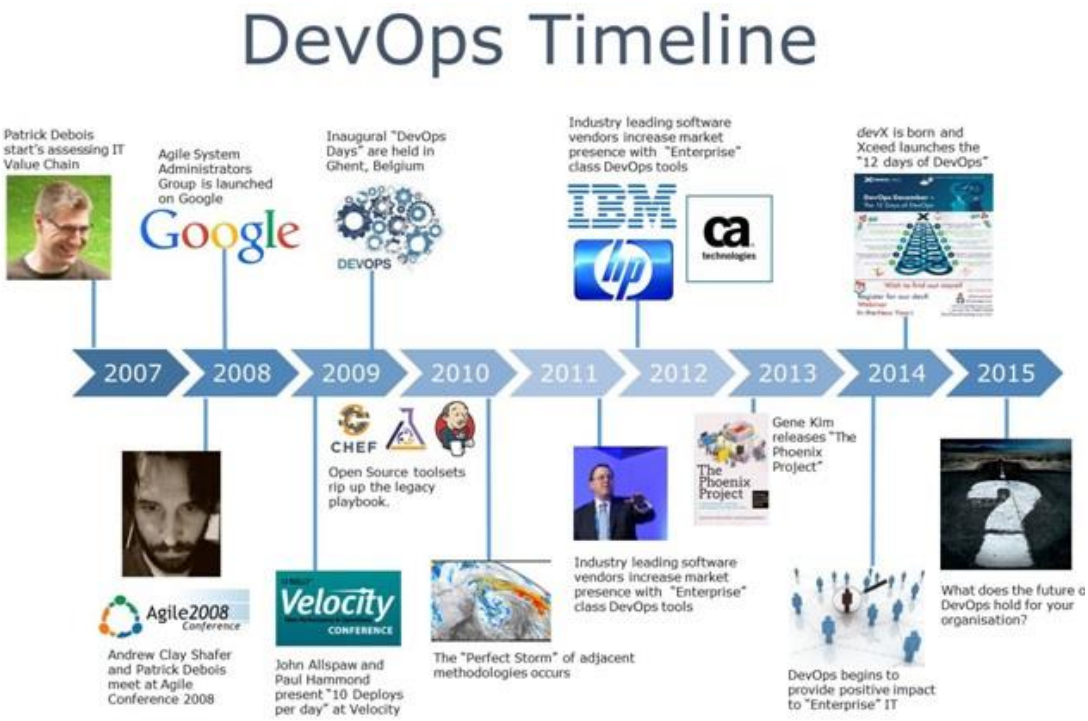


Imagen 7 Línea de Tiempo DevOps
Fuente: *[19]*

4 DevOps

4.1 Etimología de DevOps

El termino DevOps en su definición etimológica deriva del acrónimo inglés de las palabas development -desarrollo- y operations -operaciones- y como se desarrolló en el apartado 3.1 Acuñar concepto DevOps (Andrew Clay Shafer y Patrick Debois) no es un término nuevo, sino que el uso del mismo data del año 2009 y desde entonces fue creciendo cada vez más, hasta nuestros días convirtiéndose en una palabra de moda, un Buzzword[4].

Imagen 8 development -desarrollo- y operations -operaciones-
Fuente: *[20]*

DevOps se refiere a un movimiento cultural y profesional que se centra en la comunicación, colaboración e integración de los equipos dentro de las organizaciones.

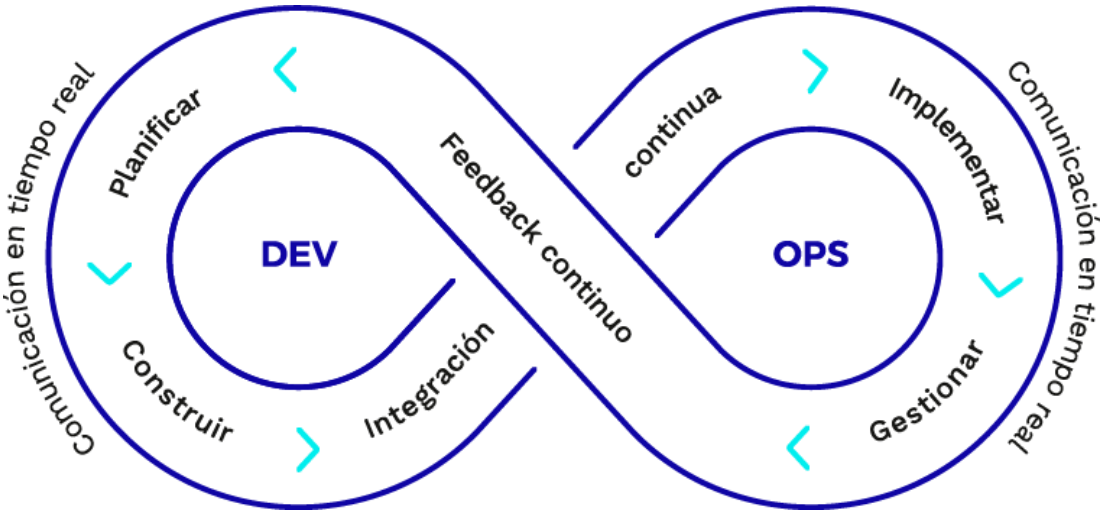


Imagen 9 DevOps Comunicación en tiempo real
Fuente: *[21]*

[4] Buzzword: palabra de moda.

Por lo tanto para entender mejor el termino DevOps se puede definir como: “DevOps intenta llenar estos vacíos desarrollando una asociación entre los equipos de desarrollo y operaciones. El movimiento DevOps enfatiza la comunicación, la colaboración y la integración entre los desarrolladores de software y las operaciones de TI.

DevOps promueve la colaboración, y la colaboración se ve facilitada por la automatización y la orquestación para mejorar los procesos. En otras palabras, DevOps esencialmente extiende los objetivos de desarrollo continuo del movimiento ágil a la integración y liberación continuas.

DevOps es una combinación de prácticas y procesos ágiles, que aprovecha los beneficios de las soluciones en la nube. Las metodologías ágiles de desarrollo y prueba nos ayudan a cumplir los objetivos de integrar, desarrollar, construir, implementar, probar y lanzar aplicaciones continuamente.”

4.2 ¿Que no es DevOps?

Ahora que se definió lo que es DevOps se puede decir lo que no es DevOps, ya que muchas veces el término es mal utilizado.

El termino DevOps no es un puesto laboral, ya que muchas organizaciones realizan sus búsquedas solicitando personas con perfiles “DevOps”, esto no es correcto, ya que lo que las organizaciones están buscando es una persona con un perfil diferente.

Un empleado que sea desarrollador, ósea que sepa desarrollar en varios lenguajes y sysadmin[5], ósea que conozca y sepa implementar diferentes soluciones de infraestructura como lo son los servidores web, balanceadores de carga, WAF[6], etc. y de automatización de servicios.

Entonces como se puede apreciar DevOps no es solo aplicar herramientas dentro de los procesos de la organización para generar

[5] Sysadmin: Administrador de Sistemas (Acronimo del inglés System Administration)
[6] WAF: Firewall de Aplicaciones Web (WAF del inglés Web Application Firewall)

servicios automatizados, ya que es mucho más que ello, como se apreció en su definición.

4.3 Cultura DevOps

DevOps es un cambio cultural y con enfoque en el conocimiento compartido dentro de la organización, que debe emanar de arriba hacia abajo, en forma de Top-Down, desde la alta gerencia que debe estar convencida del cambio, hasta llegar a las diferentes áreas y equipos.

Por lo tanto esta cultura organizacional no solo debe involucrar a los desarrolladores de software, sino también a los administradores de infraestructuras para que puedan colaborar entre ambos desde el inicio del proyecto, con equipos interdisciplinarios y cambiando el pensamiento de “si siempre fue hecho así y funciona para que cambiarlo”.

5 CI/CD y la otra CD

CI/CD, que consta de dos prácticas relacionadas: integración continúa y entrega continúa, es el proceso de creación, empaquetado, implementación de aplicaciones y, por lo tanto, generar valor para su negocio. En otras palabras, es la práctica la que le permite entregar de manera rápida y segura nuevos códigos y actualizaciones a sus usuarios.

Para poder dar sentido a lo que se detalló como DevOps y posteriormente lo que se detallara en el apartado sobre DevSecOps, se tiene que definir lo que se conoce como Integración Continua (CI[7]), Despliegues Continuos (CD[8]) y la Implementación Continua (CD[9]) que son el núcleo en el que se basa DevOps y DevSecOps.

“La CI/CD es un método para distribuir aplicaciones a los clientes con frecuencia mediante el uso de la automatización en las etapas del desarrollo de aplicaciones. Los principales conceptos que se atribuyen a la CI/CD son la integración continua, la distribución continua y la implementación continua. La CI/CD es una solución para los problemas que puede generar la integración del código nuevo a los equipos de desarrollo y de operaciones (también conocida como "el infierno de la integración").

En concreto, la CI/CD incorpora la automatización continua y el control permanente en todo el ciclo de vida de las aplicaciones, desde las etapas de integración y prueba hasta las de distribución e implementación. Este conjunto de prácticas se conoce como "canales de CI/CD", y cuenta con el soporte de los equipos de desarrollo y de operaciones que trabajan de forma conjunta y con agilidad.” [22]

A continuación se puede apreciar en la imagen como era el modelo tradicional en el cual se desarrollaba, se testeaba el código y por último se realizaba de una forma manual el despliegue a los diferentes ambientes por el equipo de operaciones.

[7] CI: Integración Continua (del inglés Continuous Integration)
[8] CD: Despliegues Continuos (del inglés Continuous Delivery)
[9] CD: Implementación Continua (del inglés Continuous Deployment)

Los diferentes equipos de Desarrollo, QA e Infraestructura trabajaban de una forma independiente, por lo tanto si se detectaba un problema y al no haber una colaboración, se retrasaba todo el proceso.

En cambio en un modelo DevOps se realiza un trabajo colaborativo entre los diferentes equipos buscando un bien común el cual es la consagración de llevar el proyecto hasta su despliegue en producción de una forma ordenada y automatizada.

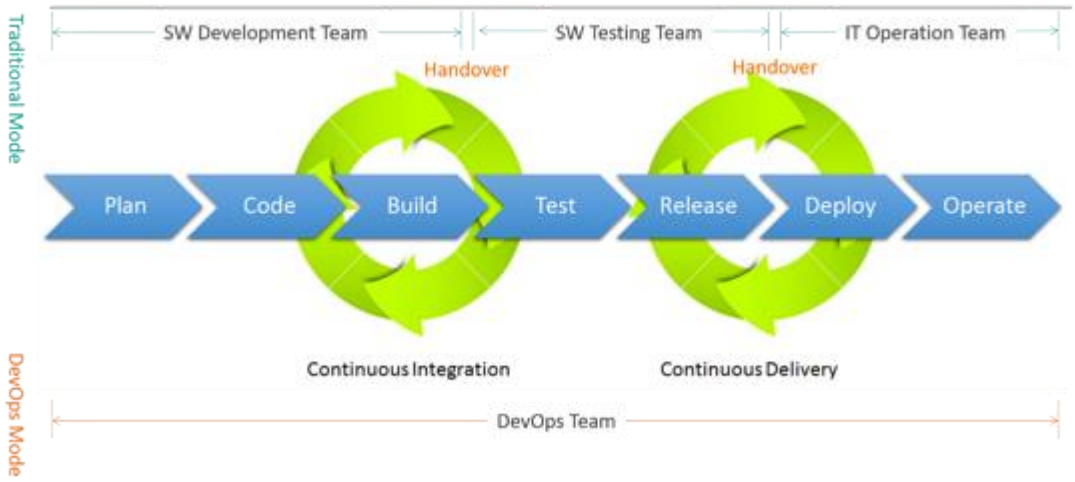


Imagen 10 Modelo Tradicional vs Modelo DevOps
Fuente: *[23]*

“En palabras simples, la integración continúa o CI es una práctica de ingeniería de software, donde cada check-in realizado por un desarrollador se verifica por uno de los siguientes:

- Mecanismo de extracción: ejecución de una compilación automatizada a una hora programada
- Mecanismo de inserción: ejecutar una compilación automatizada cuando los cambios se guardan en el repositorio.

A este paso le sigue la ejecución de una prueba unitaria con los últimos cambios disponibles en el repositorio de código fuente. La integración continua es una práctica popular de DevOps que requiere que los desarrolladores integren el código en repositorios de código como Git y SVN varias veces al día para verificar la integridad del código.

Luego, cada check-in es verificado por una compilación automatizada, lo que permite a los equipos detectar problemas temprano.

CI, e incluso CD, son las bases para que las empresas incluso archiven DevOps. No podemos hacer DevOps sin buenas implementaciones de CI y CD en su organización.” [24]

5.1 CI: Integración Continua

“El objetivo del desarrollo de las aplicaciones modernas es contar con múltiples desarrolladores que trabajen de forma simultánea en distintas funciones de la misma aplicación. Sin embargo, si una empresa fusiona todo el código fuente diversificado en un solo día (conocido como el "día de la fusión"), las tareas resultantes pueden ser tediosas y manuales, y pueden tomar mucho tiempo. Esto sucede porque, cuando un desarrollador que trabaja de forma aislada implementa un cambio en una aplicación, existe la posibilidad de que este cambio entre en conflicto con aquellos cambios implementados simultáneamente por otros desarrolladores. Este problema puede agravarse aún más si cada desarrollador personaliza su propio entorno de desarrollo integrado (IDE) local, en lugar de que el equipo acuerde un IDE basado en la nube.

La integración continua (CI) ayuda a que los desarrolladores fusionen los cambios que introducen en el código para incorporarlos a una división compartida (o "rama") con más frecuencia, incluso diariamente. Una vez que se fusionan los cambios implementados por un desarrollador en una aplicación, se validan con el desarrollo automático de la aplicación y la ejecución de distintos niveles de pruebas automatizadas (generalmente, pruebas de unidad e integración) para verificar que los cambios no hayan dañado la aplicación. Esto significa probar todo, desde las clases y el funcionamiento hasta los distintos módulos que conforman toda la aplicación. Si una prueba automática detecta un conflicto entre el código nuevo y el actual, la CI facilita la resolución de esos errores con frecuencia y rapidez.” [25]

5.2 CD: Distribución Continua

"Después de la automatización de los diseños y las pruebas de unidad e integración de la CI, la distribución continua automatiza la liberación de ese código validado hacia un repositorio. Por eso, para que el proceso de distribución continua sea eficaz, es importante que la CI ya esté incorporada a su canal de desarrollo. El objetivo de la distribución continua es tener una base de código que pueda implementarse en un entorno de producción en cualquier momento.

En la distribución continua, cada etapa (desde la fusión de los cambios en el código hasta la distribución de los diseños listos para la producción) implica la automatización de las pruebas y de la liberación de código. Al final de este proceso, el equipo de operaciones puede implementar una aplicación para que llegue a la etapa de producción de forma rápida y sencilla." [25]

5.3 La otra CD: Implementación Continua

"La última etapa de un canal de CI/CD listo es la implementación continua, que es una extensión de la distribución continua. Recordemos que esta última automatiza la liberación de un diseño listo para la producción a un repositorio del código. A su vez, la implementación continua automatiza la liberación de una aplicación para que llegue a la etapa de producción. Debido a que no hay una entrada manual en la etapa del canal anterior a la producción, la implementación continua depende, en gran medida, del correcto diseño de la automatización de pruebas.

En la práctica, la implementación continua implica que el cambio implementado por un desarrollador en una aplicación pueda ponerse en marcha unos cuantos minutos después de escribirlo (en el supuesto de que haya pasado las pruebas automatizadas). Esto facilita mucho más el proceso de recibir e incorporar continuamente los comentarios de los usuarios. En conjunto, todas estas prácticas de CI/CD relacionadas hacen

que la implementación de una aplicación se lleve a cabo con menos riesgos, ya que es más fácil liberar cambios en las aplicaciones en fragmentos pequeños, en vez de hacerlo todo de una sola vez. Sin embargo, también deben realizarse muchas inversiones iniciales, ya que las pruebas automatizadas deben diseñarse para que se adapten a las distintas etapas de prueba y liberación en el canal de la CI/CD.” [25]

Imagen 11 Integración Continua - Despliegues Continuos - Implementaciones Continuos
Fuente: *[25]*

6 Pipeline

6.1 Definición de Pipeline

En la Ingeniería de software se conoce con el término “pipeline” comúnmente como un flujo de trabajo que permite a los desarrolladores y profesionales de DevOps compilar, construir e implementar sus códigos de manera confiable y eficiente en la plataforma de cómputo de producción.

Un pipeline de CI/CD es el proceso por el cual se despliega una pieza de software hasta que esta lista para ser desplegada en producción. A partir de esto se cierra la brecha entre el equipo de desarrollo y los equipos de operación al automatizar las fases de construcción, prueba e implementación del ciclo de vida del desarrollo del software.

El objetivo de un pipeline de CI/CD es permitir un flujo constante de actualizaciones de software, lanzarlas a producción, acortar los ciclos de lanzamiento, reducir costos y reducir los riesgos asociados con el desarrollo manual.

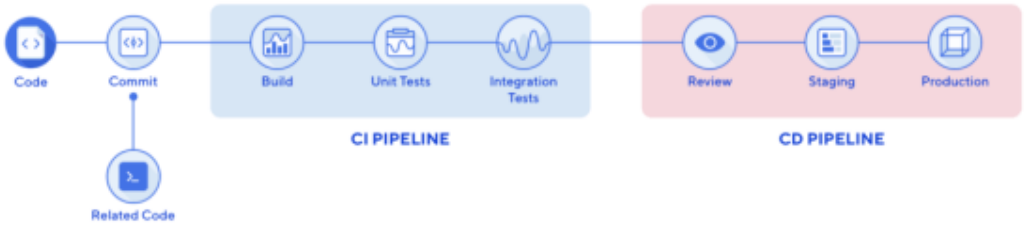


Imagen 12 Pipeline
Fuente: *[26]*

6.2 Estructura de un Pipeline

La estructura genérica de los componentes de un pipeline está compuesta por los siguientes elementos:

- **La fase del Commit**: Esta es la primera fase del pipeline y es en la cual los desarrolladores envían su código a un

repositorio, luego de esta acción manual, el servidor de integración continua detecta el cambio de forma automática y comienza con la compilación.

- **La fase de Compilación**: En esta fase se comienzan a ejecutar las primeras pruebas sobre el código de forma estática (SAST[10]), estas pruebas pueden ser funcionales, como también de seguridad. De esta fase se obtiene como resultado un componente ejecutable del producto.
- **Fase de Pruebas**: En esta tercer fase se comienzan a realizar las pruebas dinámicas (DAST[11]), de estas pruebas se ve la calidad del software en ejecución de forma funcional y también si se pueden detectar vulnerabilidades en el código desplegado.
- **Fase de implementación**: Si las pruebas son exitosas, se activa la fase de implementación. Básicamente, esta es la fase en la que la versión integrada se entrega a producción. La implementación se puede implementar en el servidor de producción o en el servidor de pre-producción.

[10] Pruebas de seguridad de aplicaciones estáticas (SAST por sus siglas en inglés, Static application security testing)
[11] Pruebas de seguridad de análisis dinámico (DAST por sus siglas en inglés, Dynamic application security testing)

> "Si piensas que la tecnología puede solucionar tus problemas de seguridad, está claro que ni entiendes los problemas ni entiendes la tecnología"
>
> Bruce Schneier

Capítulo 3

7 DevSecOps

7.1 ¿Qué es DevSecOps?

El termino DevSecOps tiene menos tiempo que DevOps, ya que se comenzó a adoptar la cultura y filosofía organizacional en el año 2015 si lo tenemos que situar en un año determinado.

Ahora ¿A que nos referimos cuando decimos "el termino DevSecOps"?

Deloitte Insights en su artículo de la Edición del 10° aniversario sobre tendencias tecnológicas 2019 dice:

"PARA MEJORAR SUS ENFOQUES ANTE EL RIESGO CIBERNÉTICO Y OTROS RIESGOS, las organizaciones de pensamiento prospectivo están incrustando seguridad, privacidad, política, y controles, en su cultura, procesos, y herramientas de DevOps. En la medida en que DevSecOps gane impulso, más compañías probablemente harán que la automatización de la modelación de amenaza, la valoración del riesgo, y la seguridad de la tarea sean componentes fundamentales de las iniciativas de desarrollo de producto, desde generación de ideas hasta la iteración para lanzar y hasta operaciones. DevSecOps de manera fundamental transforma la administración cibernética y del riesgo desde ser actividades basadas-en-cumplimiento – típicamente realizadas más tarde en el ciclo de vida del desarrollo. Por otra parte, DevSecOps codifica las políticas y las mejores prácticas en las herramientas y plataformas subyacentes, permitiendo que la seguridad se convierta en una responsabilidad compartida de toda la organización de TI." [27]

Con ello se vuelve a ratificar que DevSecOps sigue los mismos principios que DevOps, buscando un cambio cultural de trabajo organizacional entre equipos desde el principio del proyecto, transformando las personas, procesos y las herramientas de la practicas DevOps pero con la capa de seguridad como premisa desde el principio del Ciclo de Vida del Desarrollo de Software.

Como fue comentado en la Introducción de esta obra, uno de los ejemplos fue el de Uber, pero también se debe comentar que en el año 2017 Uber fue víctima de una brecha de seguridad en la cual 57 millones de clientes sufrieron la exposición de sus datos personales, debido a que los desarrolladores utilizaron una solución alternativa para administrar las credenciales en un repositorio de software. [28]

7.2 Seguridad en DevOps = DevSecOps

"Los objetivos de DevSecOps son mover las actividades de seguridad de derecha a izquierda durante el ciclo de vida del desarrollo y tener prácticas de seguridad integradas en el pipeline de integración continua." [29]

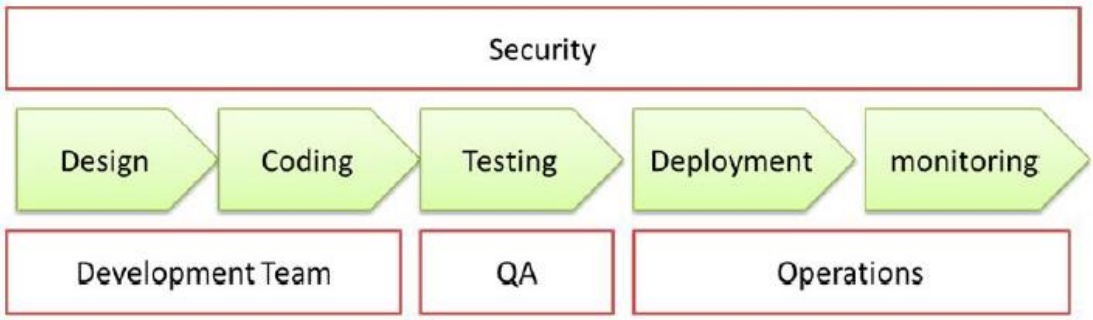


Imagen 13 Seguridad en todo el SDLC
Fuente: *[29]*

Pero para llegar a la consagración de este objetivo se debe también tener en cuenta que no solo todos los actores deben estar alineados, sino se deben cumplir con las siguientes consideraciones de seguridad:

- Cumplimiento de las Normas de Seguridad (ISO 2700x, FIPS, CSA-CCM)

- Complimiento Legal/Regulatorio
- Nuevas tecnologías (de terceros, en la nube, contenedores y virtualización)
- Hackeos/Abusos de servicios en la nube

Esto es importante porque si solo se piensa en llevar adelante la práctica de DevOps en la organización, se estaría cayendo en olvidar uno de los principios más importantes de todos que es la seguridad en el Ciclo de Vida del Desarrollo de Software.

A continuación se muestran las diferentes fases del Ciclo de Vida del Desarrollo de Software y cada una de las prácticas que se pueden desarrollar para generar la capa de seguridad en todo el proceso.

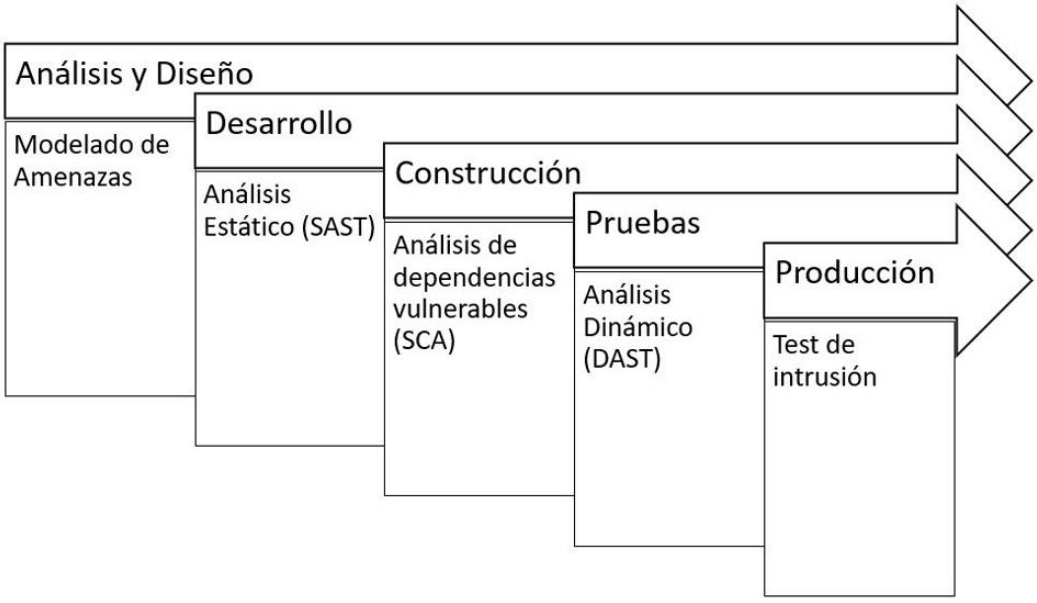


Imagen 14 Pruebas de Seguridad en SDLC
Fuente: *[33]*

7.3 Modelado de Amenazas

El modelado de amenazas debe ser parte fundamental de la práctica de DevSecOps, este se debe desarrollar en las etapas de análisis y diseño del Ciclo de Vida del Desarrollo de Software.

Como se puede apreciar en la siguiente imagen en las etapas de análisis y diseño debe ser llevado adelanté el modelado de amenazas.

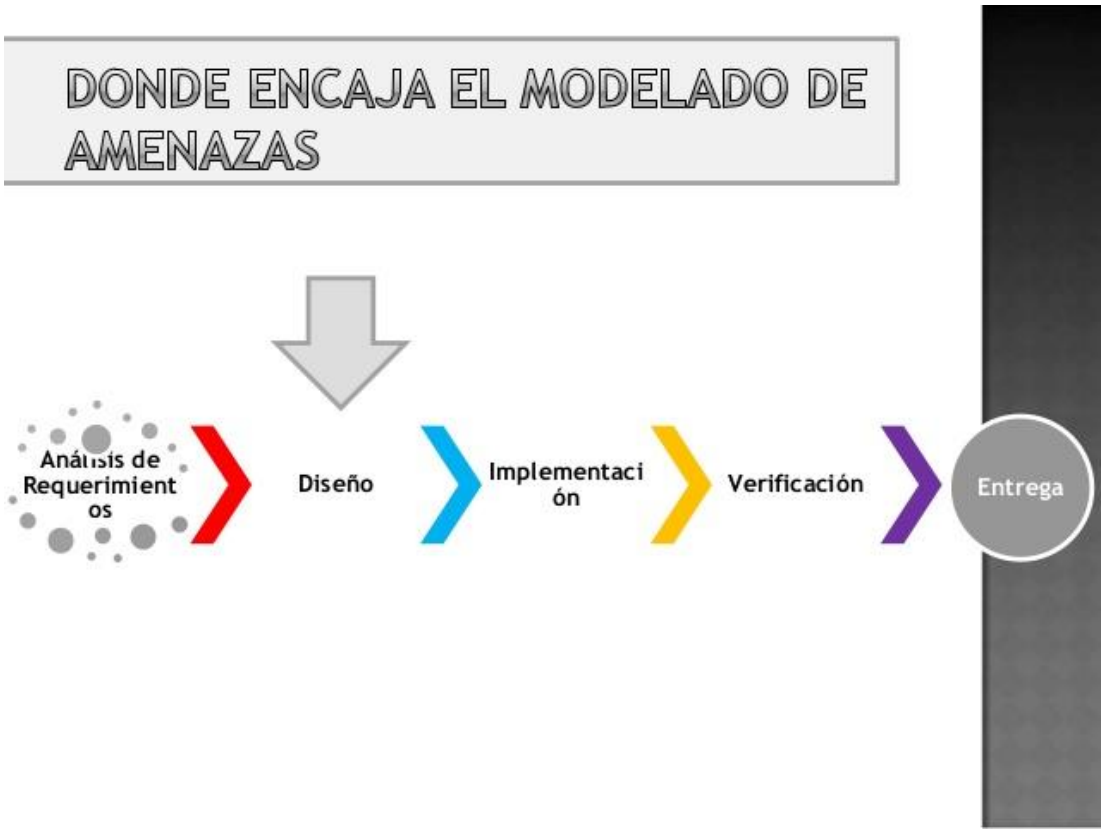


Imagen 15 Modelado de Amenazas en el SDLC
Fuente: *[30]*

Se puede definir el modelado de amenazas como “una técnica de comprobación cuyo objetivo es ayudar a identificar y planificar de forma correcta, la mejor manera de mitigar las amenazas de una aplicación mediante un enfoque moderno de análisis de gestión de riesgos, y la implementación de medidas o controles que contribuyan a mejorar la seguridad. Está técnica tiene como principales ventajas la aplicación durante una etapa temprana del SDLC, y más allá de que su implementación es relativamente nueva está despertando un gran interés.” [30]

Existen varios modelos que se pueden llevar a cabo a la hora de realizar un modelado de amenazas, a continuación se listan algunos de ellos que pueden ser tomados como referencia.

- Microsoft SDL
- BSIMM (Building Security In Maturity Model)
- SSE
- CMM (System Security Engineering Capability Matury Model)

- OWASP (Open Web Application Security Project)

Dentro de los beneficios que nos puede aportar el modelado de amenazas podemos enumerar los siguientes:

- “Entender los requisitos de seguridad: al ir realizando el modelado de amenazas, irán surgiendo requisitos de seguridad, y podremos hacernos a la idea de que controles van a aplicar, teniendo en cuenta que algunos no se podrán llevar a cabo, ya porque no se alinean con negocio, ser muy costosos, etc. (no hay que olvidar llevar un registro y poner los motivos de porque no se aplican).
- Detectar defectos en fases tempranas: ayuda a identificar los posibles problemas antes del desarrollo, por lo tanto, hará a los desarrolladores “escapar” de los errores, y por lo tanto evitar la necesidad de cambios que puedan afectar otro código y hacernos perder tiempo. Lo que nos lleva al siguiente punto y al final lo que a todos interesa…
- Entregar mejores productos” [31]

7.4 SAST

Las pruebas de seguridad de aplicaciones estáticas (SAST por sus siglas en inglés, Static Application Security Testing) es un conjunto de tecnologías diseñadas para analizar código fuente, byte code, y binarios con el fin de obtener un indicativo de nivel de seguridad del aplicativo. Las soluciones SAST analizan una aplicación desde "adentro hacia afuera" en un estado que no se ejecuta. [33]

Las pruebas de seguridad de aplicaciones estáticas pueden ser desarrolladas en la etapa de desarrollo a través de plugins que se instalan en los IDEs de los desarrolladores o una vez en la integración continua del pipeline de seguridad y se realizan los commits en el repositorio.

Para realizar estas pruebas se pueden utilizar las siguientes herramientas:

- Checkmarx Static Code Analysis
- Veracode
- SonarQube
- PMD
- FindBugs

7.5 DAST

Las pruebas de seguridad de aplicaciones dinámicas (DAST por sus siglas en inglés, Dynamic Application Security Testing) están diseñadas para detectar condiciones indicativas de una vulnerabilidad de seguridad en una aplicación en su estado de ejecución. La mayoría de las soluciones DAST prueban solo las interfaces HTTP y HTML expuestas de las aplicaciones habilitadas para la Web; sin embargo, algunas soluciones están diseñadas específicamente para el protocolo no web y la malformación de datos (por ejemplo, llamada a procedimiento remoto, protocolo de inicio de sesión [SIP], etc.). [34]

Las pruebas de seguridad de aplicaciones dinámicas pueden ser desarrolladas en la etapa de testing, ya que se realizan con la aplicación corriendo en memoria y ya desplegada por el personal de infraestructura.

Para realizar estas pruebas se pueden utilizar las siguientes herramientas:

- Acunetix
- Burp Suite
- OWASP ZAP
- Arachni
- W3af

7.6 Test de Intrusión

Un test de intrusión o Pentest es la simulación de un ataque a un sistema real de forma controlada en búsqueda de vulnerabilidades que le permitan al atacante obtener información y llegar a conseguir el control del sistema víctima.

Se pueden encontrar dos tipos de test de intrusión que se detallan a continuación:

- Tipo externo: Este tipo de test de intrusión es el más real ya que se va a realizar desde fuera de la organización, teniendo que sobrepasar todas las medidas de seguridad al igual que si lo estaría llevando a cabo un cibercriminal.
- Tipo Interno: Los test de intrusión de tipo interno simulan a un atacante intentando conseguir accesos desde la parte interna de la organización, ya sea haciéndose pasar por un empleado y consiguiendo acceso a una boca de red o a un access point wifi.

"El resultado de un Test de Intrusión debe ser analizado para estudiar y abordar soluciones que eliminen, o minimicen en su mayor medida, todos los problemas de seguridad detectados, reforzando el nivel de seguridad de los Sistemas de Información evaluados." [36]

7.7 OWASP Top 10

El proyecto OWASP (OWASP es un acrónimo de la abreviatura de Open Web Application Security Project) que es una organización sin fines de lucro reconocida internacionalmente, enfocada en colaborar para fortalecer la seguridad del software en todo el mundo. OWASP es conocido por mantener una lista de las 10 principales vulnerabilidades de OWASP.

"El grupo que apoya el proyecto está compuesto por una gama de expertos en seguridad web de todo el mundo. Comparten su conocimiento y experiencia sobre vulnerabilidades, amenazas, ataques y contramedidas existentes.

La idea es reunir la información más importante que le permita acceder a los riesgos de seguridad y formas de combatirlos de manera efectiva." [36]

Si se quiere obtener más información detallada de cada una de las vulnerabilidades del OWASP Top 10[12] se puede acceder a su sitio web donde se encuentran de forma online detalladas una por una.

Imagen 16 OWASP Top 10 2017
Fuente:

[12] OWASP TOP 10 Sitio Oficial: https://owasp.org/www-project-top-ten/

> "Cuando quería hackear algún sistema operativo, básicamente comprometía el código fuente y luego lo copiaba para llevarlo a la universidad, ya que no tenía espacio suficiente en mi disco duro de 200 MB."
>
> Kevin Mitnick

Capítulo 4

8 Herramientas de Código Abierto para Laboratorio de DevSecOps

Se presentaran a continuación una cierta cantidad de herramientas de código abierto con las cuales en la sección siguiente se implementara un laboratorio de Automatización de Pruebas de Seguridad para el Ciclo de Vida del Desarrollo de Software en modo DevSecOps.

Cabe aclarar y resaltar como fue explicado con anterioridad en las secciones en las cuales se explicaron los conceptos de CI/CD y la otra CD, Pipeline y DevSecOps, que la automatización a través de la utilización de herramientas en el proceso DevSecOps es agnóstico a la metodología o las herramientas utilizadas.

8.1 SonarLint

SonarLint es una extensión para diferentes IDE[13], la cual ayuda a detectar y solucionar problemas de calidad o seguridad a medida que se escriben las líneas de código, esto es muy importante ya que si se detectan errores serán en la fase de desarrollo y su corrección tendrá un costo mucho menor que si fuera descubierta en una fase posterior de la metodología utilizada cumpliendo con el concepto de "Push Left".

Principales Características de SonarLint:

- Entre los lenguajes soportados se destacan: Java, JavaScript, PHP, Python y HTML entre otros.

[13] IDE: Entorno de Desarrollo Integrado (IDE por sus siglas en inglés, Integrated Development Environment)

- SonarLint se puede obtener directamente desde el Eclipse Marketplace.
- SonarLint tiene la posibilidad de ser integrado de forma nativa con SonarQube y SonarCloud.

Más información al respecto, puede encontrarse en:

- Sitio Web de SonarLint https://www.sonarlint.org/

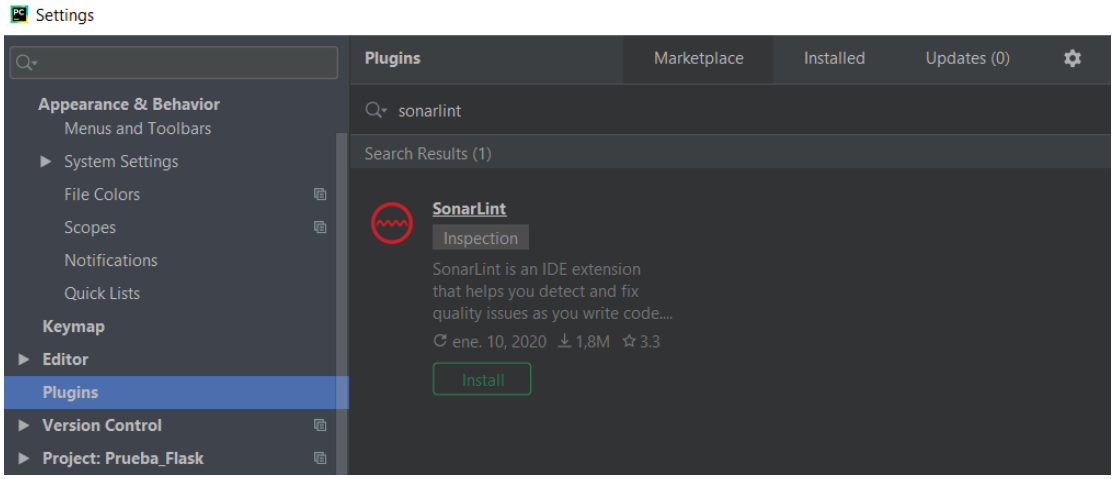


Imagen 17 SonarLint en el Marketplace IDE Pycharm
Fuente: Imagen Propia

8.2 Github

Github es un sistema para alojar proyectos utilizando un control de versiones (Versionado), utilizando y basándose en el sistema de control de versiones de GIT para poder realizar el control de versiones.

Sirve como plataforma para desarrollo de forma colaborativa, con lo cual cada desarrollador puede trabajar de forma independiente e ir subiendo diferentes piezas de código que se irán agregando al versionado del proyecto.

Permite la generación de pipelines para CI/CD (Integración Continua y Deploy Continuos), y diferentes herramientas, entre las cuales se destacan:

- Una Wiki para el mantenimiento de las distintas versiones de las páginas.

- Un sistema de seguimientos de problemas que permiten a los miembros de un equipo detallar un problema con su software o una sugerencia que deseen hacer.
- Una herramienta de revisión de código donde se pueden añadir anotaciones en cualquier punto de un fichero y debatir sobre determinados cambios realizados en un commit especifico.
- Un visor de ramas donde se pueden comparar los progresos realizados en las distintas ramas de los repositorios.

Más información al respecto, puede encontrarse en:

- Sitio Web de Github https://github.com/

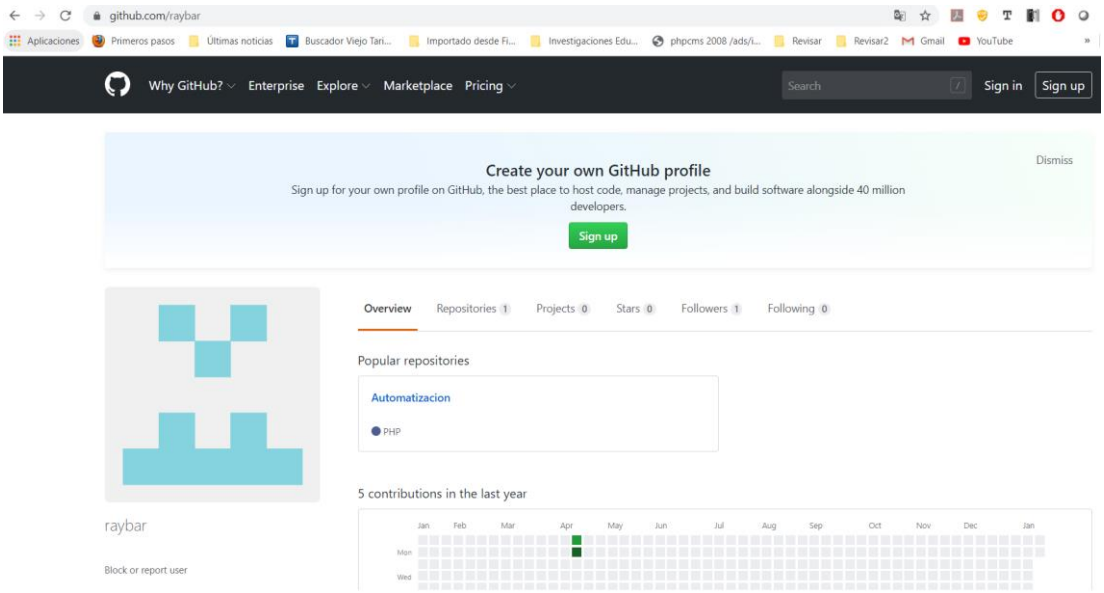


Imagen 18 Cuenta de Github para el Proyecto de Automatización
Fuente: Imagen Propia

8.3 Jenkins

Jenkins es una herramienta de código abierto, la cual nace como un fork[14] del proyecto Hudson, la primera versión del proyecto Hudson nace en el año 2005 hasta que en el año 2011 se dividen los proyectos y da nacimiento de Jenkins.

[14] Fork: Bifurcación, cuando se toma como base un proyecto ya existente y se sigue una rama diferente.

Jenkins es un servidor de automatización el cual se integra en el ciclo de vida del desarrollo del software mediante la integración continua, trabaja con Jobs, pipelines, plugins, etc.

Se puede ampliar la definición de Jenkins citando la siguiente descripción “Un servidor de compilación es, en esencia, un sistema que construye software basado en varios desencadenadores.

Hay varios para elegir. En este libro, veremos a Jenkins, que es un servidor de compilación popular escrito en Java.

Jenkins es una bifurcación del servidor de compilación Hudson. Kohsuke Kawaguchi fue el principal contribuyente de Hudson, y en 2010, después de que Oracle adquirió Hudson, continuó trabajando en la bifurcación de Jenkins. Jenkins es claramente la más exitosa de las dos cepas de hoy.

Jenkins tiene soporte especial para construir código Java, pero de ninguna manera se limita a solo construir Java.

Configurar un servidor Jenkins básico no es particularmente difícil desde el principio.” [29]

A continuación podemos apreciar una imagen en la que se crearon varios Jobs generando un pipeline de pruebas de seguridad SAST y DAST.

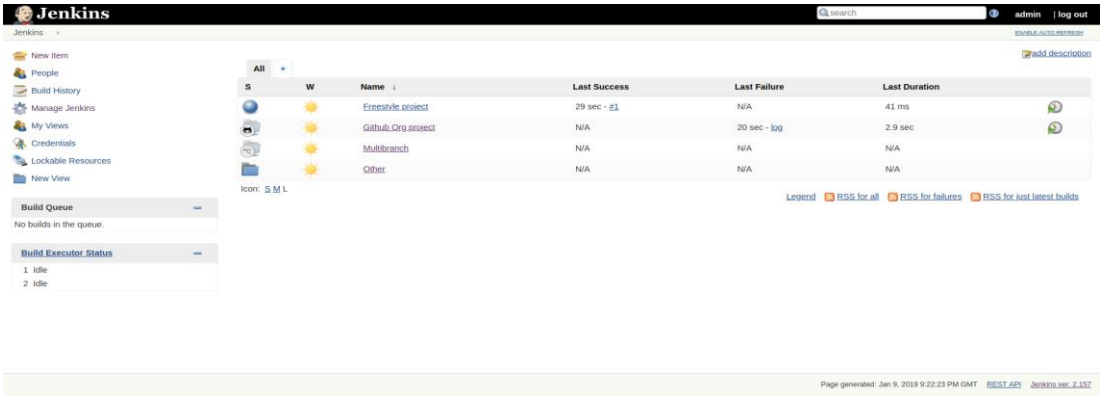


Imagen 19 Jenkins con Jobs generados
Fuente: Imagen Propia

8.4 SonarQube

SonarQube es una Tool de código abierto para análisis estático de código fuente, se basa en otras soluciones que fueron desarrolladas muchos años antes como: Checkstyle, PMD o FindBugs pero agregando muchas otras mejoras a la solución final y ayudando a mejorar el código desarrollado tanto para que sea más eficiente y limpio como también más seguro.

La herramienta Informa sobre:

- Código duplicado
- Incumplimiento en los estándares de codificación
- Pruebas unitarias
- Cobertura de código
- Complejidad ciclomática
- Errores potenciales
- Comentarios y diseño del software.

Se integra con otras soluciones como Maven, Ant y herramientas de integración continua como pueden ser por citar algunas: Atlassian, Bamboo, Jenkins y Hudson.

Más información al respecto, puede encontrarse en:

- Sitio Web [30] de SonarQube https://www.sonarqube.org/

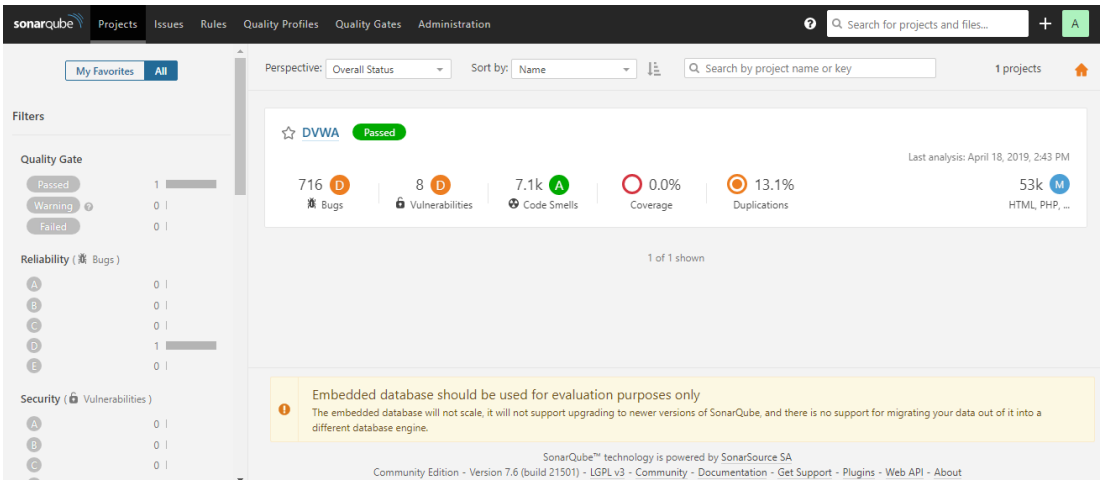


Imagen 20 SonarQube con Análisis estático de la Aplicación DVWA[15]
Fuente: Imagen Propia

8.5 OWASP ZAP

OWASP ZAP es un escáner de seguridad web de código abierto, el cual permite realizar escaneos dinámicos detectando vulnerabilidades del OWASP TOP 10, se debe destacar de esta herramienta que es uno de los proyectos más activos del Proyecto OWASP.

Cuándo se usa como servidor proxy permite a los usuarios manipular todo el tráfico que pasa a través de él, incluyendo el tráfico del protocolo seguro HTTPS.

Más información al respecto, puede encontrarse en:

- Sitio Web del Proyecto OWAS ZAP https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project

8.6 Diferencia entre SonarLint y SonarQube

Se debe realizar una aclaración sobre las dos herramientas que fueron presentadas en los apartados 9.1 y 9.4 ya que ambas herramientas son utilizadas en diferentes momentos del pipeline.

[15] DVWA: Maldita aplicación web vulnerable (DVWA por sus siglas en inglés, Damn Vulnerable Web App)

En el caso de SonarLint es una herramienta que será utilizada en el IDE del desarrollador al momento que se está desarrollando la pieza de código, por lo tanto el análisis estático sobre el código realizado por la herramienta es sin que se haya enviado a un repositorio y sin comenzar la Integración Continua (CI) del pipeline.

En cambio SonarQube también es una herramienta para el análisis estático de código fuente (SAST), pero pudiendo descargar el código fuente en la Integración Continua (CI) del pipeline, desde un repositorio para su posterior análisis y reporte de los errores detectados.

9 Laboratorio de DevSecOps

9.1 Arquitectura del Laboratorio

Se presenta una solución que se realizó en forma de laboratorio utilizando las Tools OpenSource que fueron detalladas en el apartado "8 Herramientas OpenSource".

El laboratorio fue implementado en una máquina virtual VMWare con sistema operativo Debían 9, de forma monolítica[16], se instalaron los siguientes servicios:

- **SonarQube**
- **Jenkins**
- **OWASP ZAP**

También se utilizó Pycharm que es un IDE para desarrollar aplicaciones en el lenguaje de programación Python al cual se le instalo el plugin SonarLint con el cual se desarrolló una aplicación web de demostración utilizando el microframework Flask.

Se creó una cuenta en el servicio Github, la cual se utilizó como repositorio y control de versiones de código fuente.

Quedando la arquitectura como se muestra en la siguiente imagen:

[16] Monolítica: Todas las aplicaciones están corriendo debajo de la misma máquina virtual.

ARQUITECTURA

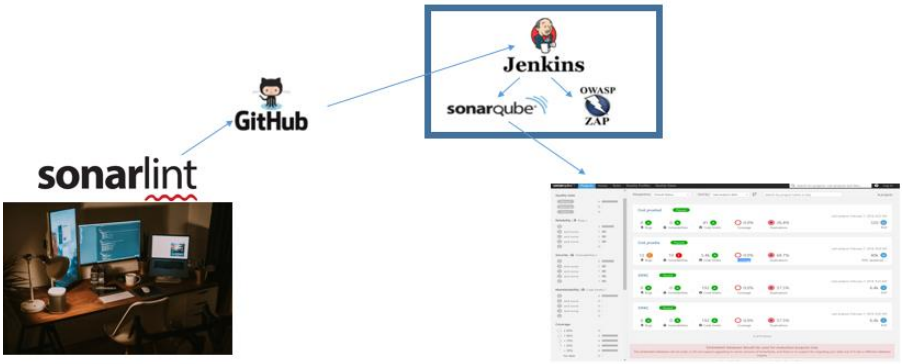


Imagen 21 Arquitectura Laboratorio DevSecOps
Fuente: Imagen Propia

9.2 Desarrollo del Proceso de Automatización

Para comenzar con la automatización de las pruebas de seguridad se generó un pipeline a través de Jenkins para ello se crearon tres jobs.

- Primer Job: Este job que se le dio el nombre de “DVWA_Job” es el encargado de detectar si se realiza un commit en el repositorio de Github, en caso de detectarse un commit nuevo se conecta al Github y descarga el código al servidor local.
- Segundo Job: Este job que se le dio el nombre de “SonarQube_Job” desencadena una vez finalizado correctamente el primer Job un análisis estático de la aplicación DVWA.
- Tercer Job: este último job que se le dio el nombre de “OWASPZAP_Job” se ejecuta si el análisis de código estatico realizado en la etapa anterior fue con éxito, se ejecuta la aplicación para luego realizar un análisis dinámico de código con OWASP ZAP

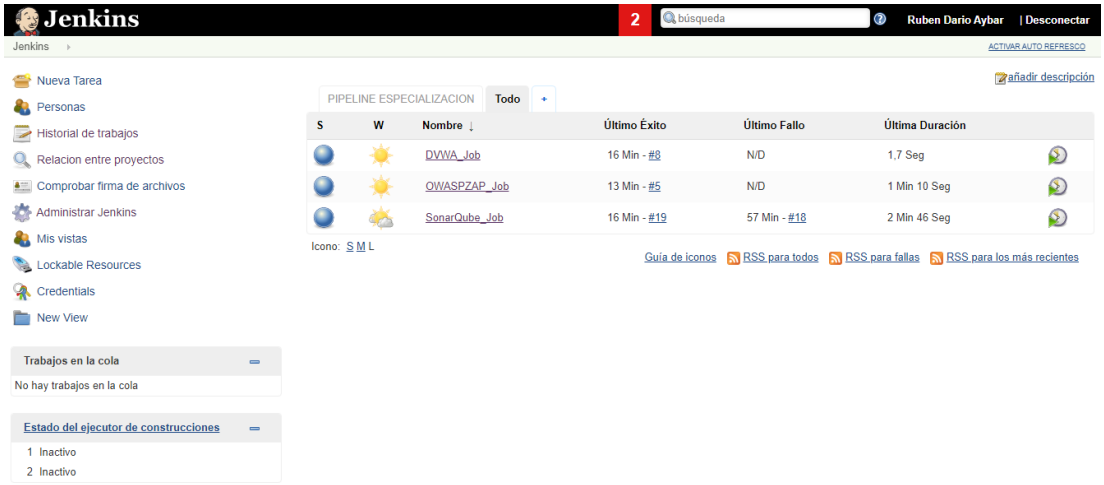


Imagen 22 Jenkins con Jobs del Pipeline Especialización
Fuente: Imagen Propia

Para la Prueba de Concepto (PoC) se ejecuta el Job DVWA_Job y se monitorea el correcto desarrollo de cada una de las etapas del pipeline.

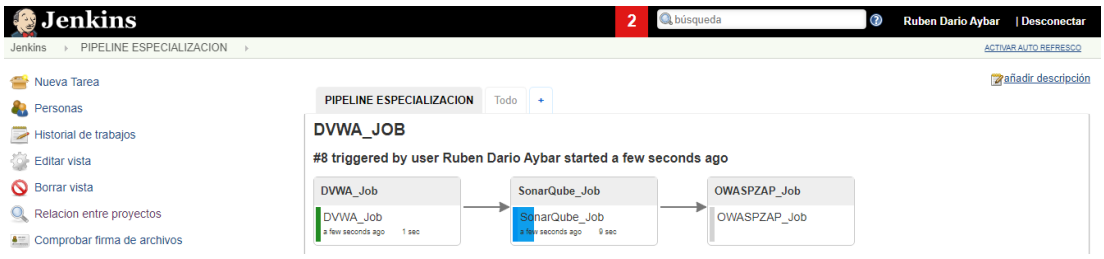


Imagen 23 Pipeline Especialización en ejecución
Fuente: Imagen Propia

Si se van monitoreando las diferentes salidas de cada uno de los Jobs que se van ejecutando, se puede apreciar como los resultados van quedando registrados con la leyenda “Finishid: SUCCESS”.

Por ello tras la ejecución correcta de cada uno de los Jobs se ejecuta a continuación el siguiente.

Como queda registrado en las capturas de pantalla que fueron tomadas por cada Job:

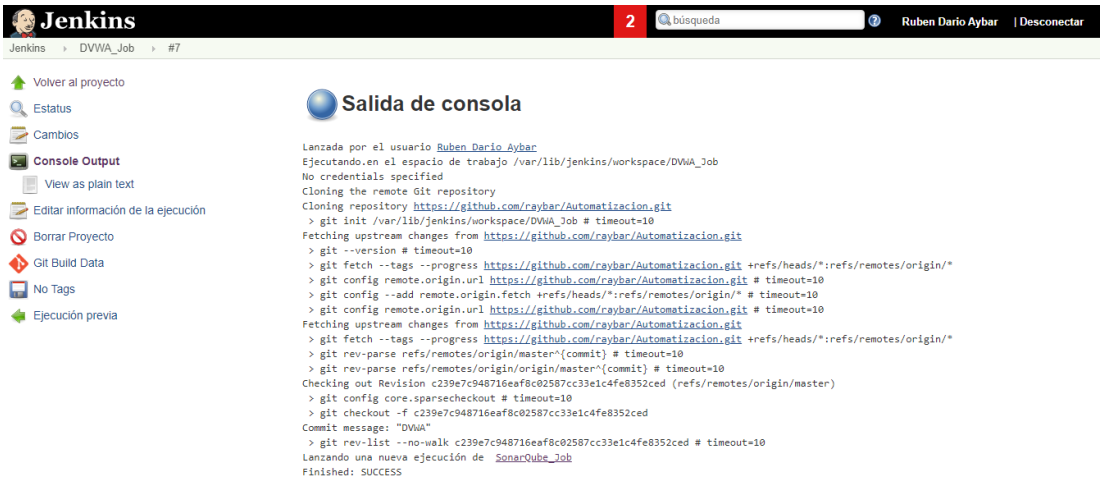


Imagen 24 Finishid: SUCCESS Job DVWA_Job
Fuente: Imagen Propia

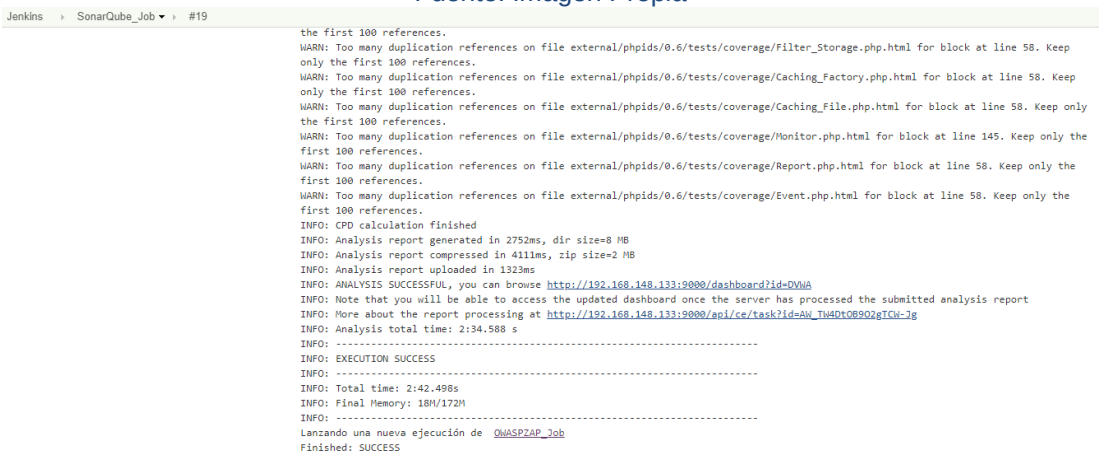


Imagen 25 Finishid: SUCCESS Job SonarQube_Job
Fuente: Imagen Propia

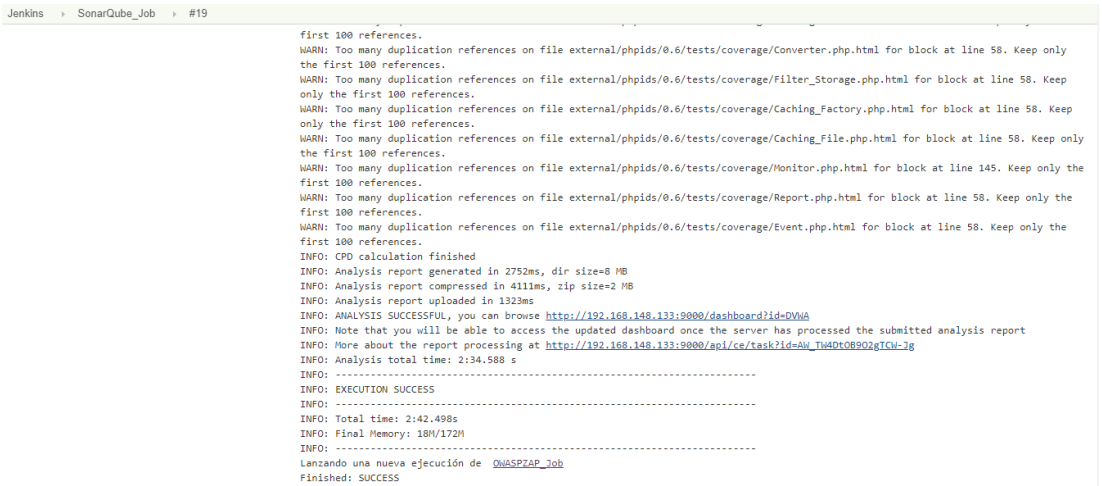


Imagen 26 Finishid: SUCCESS Job OWASPZAP_Job
Fuente: Imagen Propia

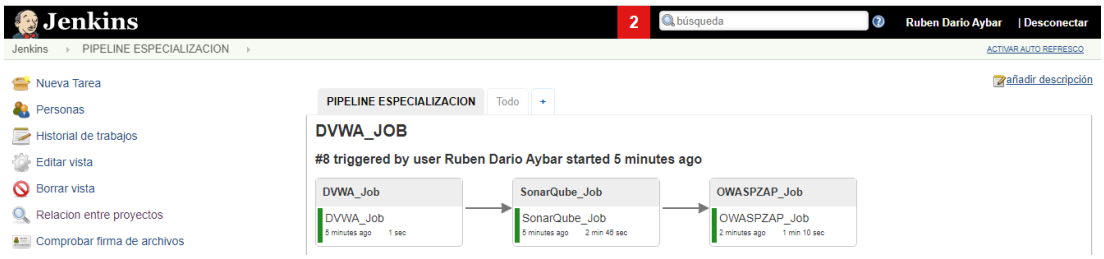


Imagen 27 Pipeline Especialización finalizado correctamente
Fuente: Imagen Propia

Si entramos en la consola de SonarQube, ya nos muestra el análisis de código estático que fue realizado sobre la aplicación DVWA.

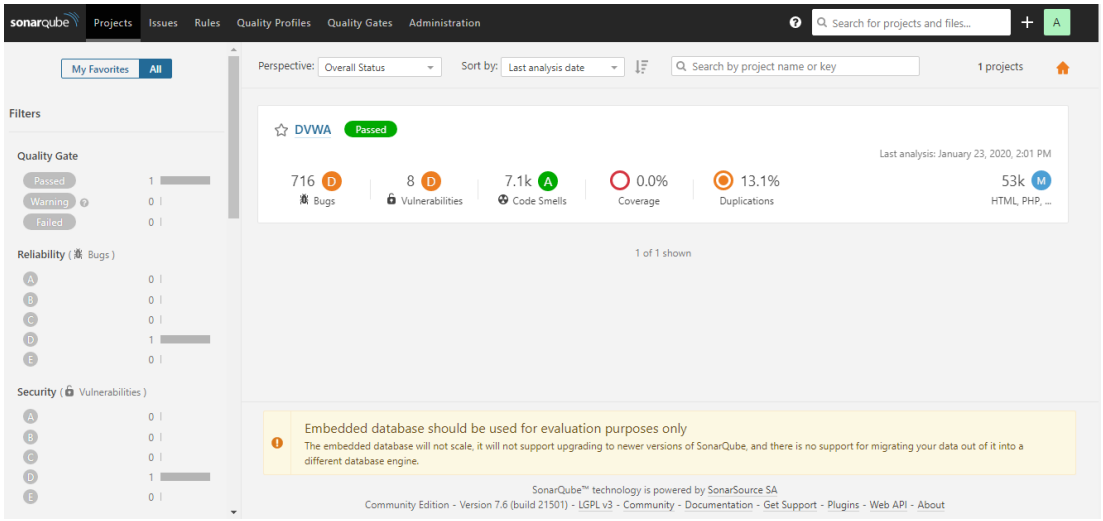


Imagen 28 GUI de SonarQube con el resultado del Análisis Estático de Código
Fuente: Imagen Propia

Por ultimo podemos buscar el resultado del análisis estático de código en el Job llamado OWASPZAP_Job, que como se puede ver en la imagen siguiente se adjuntan al Job como unos archivos en formato html o xml.

Si abrimos el archivo llamado "JENKINS_ZAP_VULNERABILITY_REPORT_new_reports.html" nos muestra el reporte.

Imagen 29 Resultado del Job OWASPZAP_Job con el reporte generado
Fuente: Imagen Propia

ZAP Scanning Report

Summary of Alerts

Risk Level	Number of Alerts
High	0
Medium	1
Low	3
Informational	0

Alert Detail

Medium (Medium)	X-Frame-Options Header Not Set
Description	X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.
URL	http://localhost:7272/error.html
Method	GET
URL	http://localhost:7272/
Method	GET
URL	http://localhost:7272/demo.css
Method	GET
URL	http://localhost:7272/welcome.html
Method	GET
Instances	4
Solution	Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).
Other information	At "High" threshold this scanner will not alert on client or server error responses.

Imagen 30 reporte del Análisis Dinámico de Código con OWASP ZAP
Fuente: Imagen Propia

10 Conclusiones

La presente obra surgió como una idea cuando estaba cursando la materia del Profesor Lic. Julio Cersar Ardita, de ahí en más no fue solo una idea sino que se fue transformando en un proyecto real.

Considero que los temas tratados en el TFE son de total actualidad, ya que empresas de la envergadura de Mercado Libre o Al Mundo trabajan con una cultura organizacional DevSecOps ya en un estado de madurez muy alto.

DevOps y DevSecOps no son solo Buzzware, sino es un término que quedo para acompañar la transformación digital de las organizaciones, y cuando hablamos de organizaciones que están llevando adelante la transición no nos referimos a cambios en lo tecnológico solamente, ya que no son términos que nos estén hablando de herramientas o metodologías, sino a términos que nos hablan de cambios culturales organizacionales.

Aunque falta mucho camino por recorrer ya que el mismo recién se inicia, las organizaciones ya tomaron el puntapié inicial para llevarlo adelante.

Para mí fue una gran experiencia y espero que para los lectores también y les aporte un valor para sus vidas personales y profesionales, todo lo que se expuso en la obra lo pude llevar adelante como proyecto en un organismo del Estado Argentino, introduciendo un pipeline a su Ciclo de Vida del Desarrollo de Software y transformando la organización de forma Cultural.

También me dio grandes satisfacciones ya que de la idea de esta obra pude presentar una charla en varias conferencias a nivel Nacional como Internacional.

A nivel Nacional fue presentada en OWASP Latam Tour 2019 Capitulo Buenos Aires, ParanaConf, PampaSeg 2019 y a nivel Internacional en el evento Security BSides Latam 2019 en Lima, Perú.

Bibliografía

[1] O. S. Initiative, «Open Source Initiative,» 23 Julio 2007. [En línea]. Available: https://opensource.org/osd-annotated. [Último acceso: 12 Enero 2020].

[2] B. D. S. LATINOAMÉRICA, «Salesforce Blog,» 11 Diciembre 2017. [En línea]. Available: https://www.salesforce.com/mx/blog/2017/12/Que-es-la-transformacion-digital.html. [Último acceso: 12 Enero 2020].

[3] W. O. C. Carlos Cabral, «Trilhas em Seguranca da Informacao: Caminhos e Ideias Para a Protecao de Dados (1st Edition),» Rio de Janeiro, BRASPORT, 2015, p. 121.

[4] L. Redondo, «MTP Blog,» 15 Febrero 2018. [En línea]. Available: https://www.mtp.es/paso-para-cumplir-con-la-gdpr-desarrolla-software-seguro/. [Último acceso: 13 Enero 2020].

[5] «ecured.cu,» [En línea]. Available: https://www.ecured.cu/Ciclo_de_vida_del_software. [Último acceso: 21 Enero 2020].

[6] D. Cantone, «La biblia del programador. Implementacion Y Debugging,» Buenos Aires, MP Ediciones, 2006, p. 17.

[7] «Fases del Ciclo de Vida del Desarrollo de Software,» [En línea]. Available: https://addontraining.com/wp-content/uploads/2016/10/sdlc.png. [Último acceso: 12 Enero 2020].

[8] F. B. Galiano, «FCO. JAVIER GARCÍA CASTELLANO DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN E INTELIGENCIA ARTIFICIAL,» [En línea]. Available: http://flanagan.ugr.es/docencia/2005-2006/2/apuntes/ciclovida.pdf. [Último acceso: 12 Enero 2020].

[9] «Evoluciòn de las Metodologìas de Desarrollo,» [En línea]. Available: https://www.timetoast.com/timelines/evolucion-de-las-metodologias-de-construccion-de-software. [Último acceso: 12 Enero 2020].

[10] «Costo de Reparar Error por Etapa,» [En línea]. Available: https://www.sallychan.gallery/ssdlc-seeking-roi-2015/2vwvqws7k8759lcp0hasuus2c4pgz5. [Último acceso: 13 Enero 2020].

[11] T. Janca, «dev.to,» 9 Abril 2019. [En línea]. Available: https://dev.to/azure/pushing-left-like-a-boss-part-1-4d9i. [Último acceso: 19 Enero 2020].

[12] G. Kim, J. Humble, P. Debois y J. & willis, The DevOps Handbook:: How to Create World-Class Agility, Reliability, and Security in Technology Organizations, Portland: IT Revolution, 2016.

[13] «Logo de la Agile 2008 Conference,» [En línea]. Available: https://leadinganswers.typepad.com/photos/uncategorized/2008/01/20/agile2008.jpg. [Último acceso: 13 Enero 2020].

[14] A. É. Villamarín, «Universitat Oberta de Catalunya,» Junio 2019. [En línea]. Available: http://openaccess.uoc.edu/webapps/o2/bitstream/10609/97626/6/aelezTFG0619memoria.pdf. [Último acceso: 17 Enero 2020].

[15] R. Hens, «El Blog de Roberto Hens,» 12 Septiembre 2019. [En línea]. Available: http://www.robertohens.com/2019/09/devops-aclarando-el-concepto.html. [Último acceso: 17 Enero 2020].

[16] «Afiche de la Conferencia Velocity 09,» [En línea]. Available: http://assets.en.oreilly.com/1/event/29/velocity2009_336x280.gif. [Último acceso: 15 Enero 2020].

[17] «Logo del Primer DevOpsDays 2009,» [En línea]. Available: https://devopsdays.org/blog/2009/11/11/devopsdays-2009-belgium-a-great-success/. [Último acceso: 15 Enero 2020].

[18] J. Garzas, «JAVIERGARZAS.COM,» 8 Junio 2016. [En línea]. Available: https://www.javiergarzas.com/2016/06/la-historia-del-proyecto-phoenix-devops-agilidad-salvar-proyecto-fracaso.html. [Último acceso: 19 Enero 2020].

[19] «Linea de Tiempo DevOps,» [En línea]. Available: https://danielvillahermosa.files.wordpress.com/2019/11/devops_timeline.jpg?w=656. [Último acceso: 17 Enero 2020].

[20] «development -desarrolo- y operations -operaciones,» [En línea]. Available: https://gotodigital.es/devops-metodologia/. [Último acceso: 19 Enero 2020].

[21] «DevOps Comunicaciòn en tiempo real,» [En línea]. Available: https://www.autentia.com/2018/08/17/entendiendo-devops-en-5-minutos/. [Último acceso: 19 Enero 2020].

[22] RedHat, «Blog RedHat,» [En línea]. Available: https://www.redhat.com/es/topics/devops/what-is-ci-cd. [Último acceso: 19 Enero 2020].

[23] «Modelo Tradicional vs Modelo DevOps,» [En línea]. Available: https://go.insitech.com.mx/hs-fs/hubfs/blog/banner/Comparison.png?width=543&name=Comparison.png. [Último acceso: 19 Enero 2020].

[24] M. Soni, «DevOps Bootcamp,» Birmingham, Packt Publishing Ltd., 2017, p. 18.

[25] «Integración Continua - Despliegues Continuos - Implementaciones Continuos,» [En línea]. Available: https://www.redhat.com/es/topics/devops/what-is-ci-cd. [Último acceso: 19 Enero 2020].

[26] «Pipeline,» [En línea]. Available: https://testautomationresources.com/ci-cd/ci-cd-pipeline/. [Último acceso: 19 Enero 2020].

[27] D. Insights, «DeVSecOps and the cyber imperative. Elevating, embedding, and evolving your risk response,» *Tech Trends 2019. Beyond the digital frontier,* pp. 102-117, 2019.

[28] W. Ashford, «ComputerWeekly.com,» 20 Marzo 2018. [En línea]. Available: https://www.computerweekly.com/news/252437170/Firms-need-to-move-from-DevOps-to-DevSecOps-says-expert. [Último acceso: 21 Enero 2020].

[29] T. Hsu, Hands-On Security in DevOps, Birmingham: Packt Publishing Ltd, 2018.

[30] «Cuadernos de Seguridad,» [En línea]. Available: https://cuadernosdeseguridad.com/2019/01/cuida-la-seguridad-desde-el-diseno-con-el-modelado-de-amenazas/. [Último acceso: 22 Enero 2020].

[31] «Modelado de Amenazas en el SDLC,» [En línea]. Available: https://es.slideshare.net/borismurillo/threat-modeling-24711655. [Último acceso: 22 Enero 2020].

[32] G. E. B. Olivares, «MODELADO DE AMENAZAS, UNA TÉCNICA DE ANÁLISIS Y GESTIÓN DE RIESGO ASOCIADO A SOFTWARE Y APLICACIONES.,» [En línea]. Available: http://polux.unipiloto.edu.co:8080/00004067.pdf. [Último acceso: 21 Enero 2020].

[33] J. Encinar, «Boomernix,» 3 Diciembre 2018. [En línea]. Available: https://www.boomernix.com/2018/03/ciclo-de-vida-de-desarrollo-se-software.html. [Último acceso: 21 Enero 2020].

[34] Garner, «Garner Glossary,» [En línea]. Available: https://www.gartner.com/en/information-technology/glossary/static-application-security-testing-sast. [Último acceso: 22 Enero 2020].

[35] Garner, «Garner Glossary,» [En línea]. Available: https://www.gartner.com/en/information-technology/glossary/dynamic-application-security-testing-dast. [Último acceso: 22 Enero 2020].

[36] Firma-e, «Firma-e Consultoria & Desarrollo TI,» 11 Marzo 2013. [En línea]. Available: https://www.firma-e.com/blog/que-es-un-test-de-intrusion-1a-parte/. [Último acceso: 22 Enero 2020].

[37] A. Kah, «www.question-defense.com,» 7 Octubre 2019. [En línea]. Available: https://www.question-defense.com/2019/10/07/what-are-the-owasp-top-10-vulnerabilities. [Último acceso: 22 Enero 2020].

[38] J. Verona, Practical DevOps, Birmingham: Packt Publishing Ltd., 2016.

[39] L. Rendek, «linuxconfig.org,» 10 Junio 2017. [En línea]. Available: https://linuxconfig.org/how-to-install-jenkins-on-debian-9-stretch-linux. [Último acceso: 22 Enero 2020].

[40] «Imagina Formación,» [En línea]. Available: https://www.imaginaformacion.com/tutorial/como-instalar-sonarqube-en-linux/. [Último acceso: 23 Enero 2020].

[41] SonarQube, «SonarQube Docs 8.1,» [En línea]. Available: https://docs.sonarqube.org/latest/setup/get-started-2-minutes/. [Último acceso: 23 Enero 2020].

[42] D. Gonzalez, Implementing Modern DevOps, Birmingham: Packt Publishing Ltd., 2017.

[43] O. Nguyen, «Test Automation Resources,» 28 Agosto 2019. [En línea]. Available: https://testautomationresources.com/ci-cd/ci-cd-pipeline/. [Último acceso: 19 Enero 2020].

ANEXO 1

Instalación básica de Jenkins [39]

Para la integración continua en el pipeline del laboratorio de DevSecOps se debe instalar un servidor de integración continua que en este caso es Jenkins en el sistema operativo Debían 9.

Para su instalación se deben seguir los siguientes pasos:

1. **Incluir el repositorio APT de Jenkins**

El primer paso es incluir el repositorio y la clave APT de Jenkins. Comencemos agregando una nueva clave:

```
# wget -q -O - http://pkg.jenkins-ci.org/debian-stable/jenkins-
ci.org.key | apt-key add


OK
```

A continuación se debe agregar un repositorio. Aquí tenemos dos opciones. Primero, podemos agregar un repositorio estable de LTS Jenkins:

```
# echo "deb http://pkg.jenkins-ci.org/debian-stable binary/" >
/etc/apt/sources.list.d/jenkins.list
```

Otra opción es incluir el último repositorio de la versión de Jenkins:

```
# echo "deb http://pkg.jenkins-ci.org/debian binary/" >
/etc/apt/sources.list.d/jenkins.list
```

Una vez hecho esto, actualice el índice del repositorio:

```
# apt update
```

2. **Instalar Jenkins**

Una vez que se tienen los repositorios APT con las líneas de Jenkins, se puede proceder a su instalación:

```
# apt install jenkins
```

3. Accediendo a Jenkins

Con los comandos ejecutados, ya se instaló Jenkins, por lo tanto ya se puede acceder a la Interface Gráfica de Administración, para poder conocer la clave para el primer ingreso a Jenkins se debe ejecutar el siguiente comando:

```
# cat /var/lib/jenkins/secrets/initialAdminPassword

b9b1649e223843e29cf17812a2322a3f
```

Una vez que ya tenemos la clave para el primer ingreso, se debe acceder a la siguiente URL `http://jenkins-server-IP:8080` desde un browser y seguir los pasos para finalizar la instalación.

Instalación básica de SonarQube [40] [41]

A continuación se detallan los pasos para realizar la instalación de SonarQube en un servidor debían 9:

1. **Prerrequisitos**

- Se debe contar con al menos 4 Gb de memoria Ram
- Tener actualizado los repositorios (apt-get)
- Se debe tener instalado un JDK para que SonarQube funcione correctamente
- Algun motor de base de datos, puede ser MySQL, PostgresSQL, Oracle, etc

No es parte de este anexo la explicación de la instalación y configuración de los prerrequisitos, pero se deben cumplir para poder realizar la instalación correcta de SonarQube que se detallara a continuación.

2. **Instalando desde el archivo ZIP**

Para comenzar deberemos descargar del sitio oficial la última versión de SonarQube a día de hoy la última versión estable es: Versión: 8.1 Release: December 2019 en su versión Free y OpenSource Comunnity.

1. Descarga SonarQube Community Edition.
2. Como usuario no root descomprimir el zip en la siguiente ruta:

```
/opt/sonarqube
```

3. Como usuario no root, arranque el servidor de SonarQube:

```
Como usuario no root ejecute:
/opt/sonarqube/bin[OS]/sonar.sh console
```

Si su instancia falla al arrancar, verifique sus logs para buscar la causa.

4. Loguese en la siguiente URL http://localhost:9000 con las credenciales de administrador de sistema (login=admin, password=admin).

Integrando SonarQube con Jenkins [24]

Para la realización del análisis dinámico de código durante la ejecución del pipeline del laboratorio de DevSecOps se utilizó la herramienta SonarQube, la cual se debe integrar con el servidor de integración continua Jenkins.

A continuación se describen los pasos que se deben seguir para la integración entre las dos herramientas:

1. **Instalación de Plugin de SonarQube en Jenkins**

1. Se debe acceder al apartado “Administrar Jenkins”

Imagen Administrar Jenkins
Fuente: Imagen Propia

2. Se debe hacer clic en “Administrar Plugins”, y luego clic en la pestaña “Disponibles”.

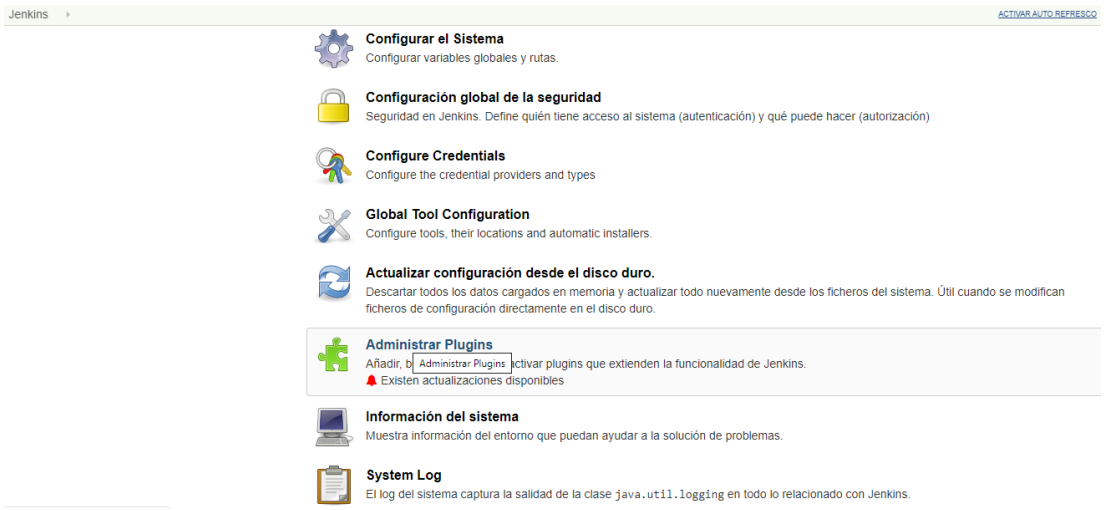


Imagen Administrar Plugins
Fuente: Imagen Propia

3. Encuentre el complemento SonarQube Scanner e instálelo

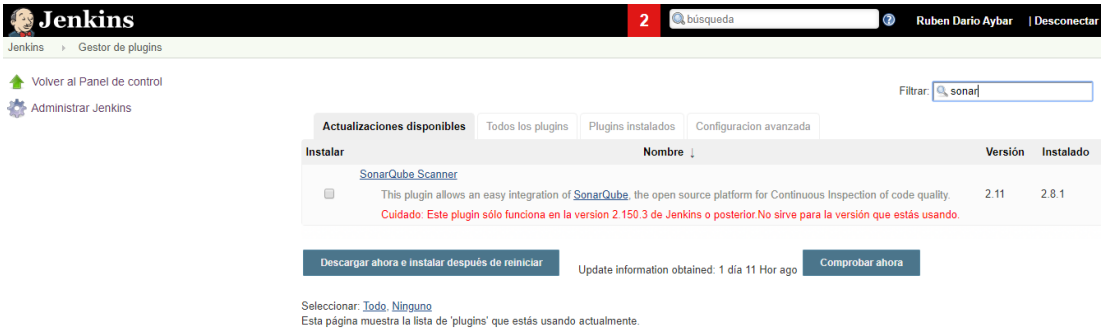


Imagen Plugin de SonarQube para Jenkins
Fuente: Imagen Propia

2. Configuración de Plugin de SonarQube en Jenkins

Una vez instalado el plugin de SonarQube para Jenkins se debe configurar, para poder configurarlo se deben seguir los pasos que se detallan a continuación:

1. Se debe acceder al apartado “Administrar Jenkins”

Imagen Administrar Jenkins
Fuente: Imagen Propia

2. luego haga clic en “Configurar el Sistema”.

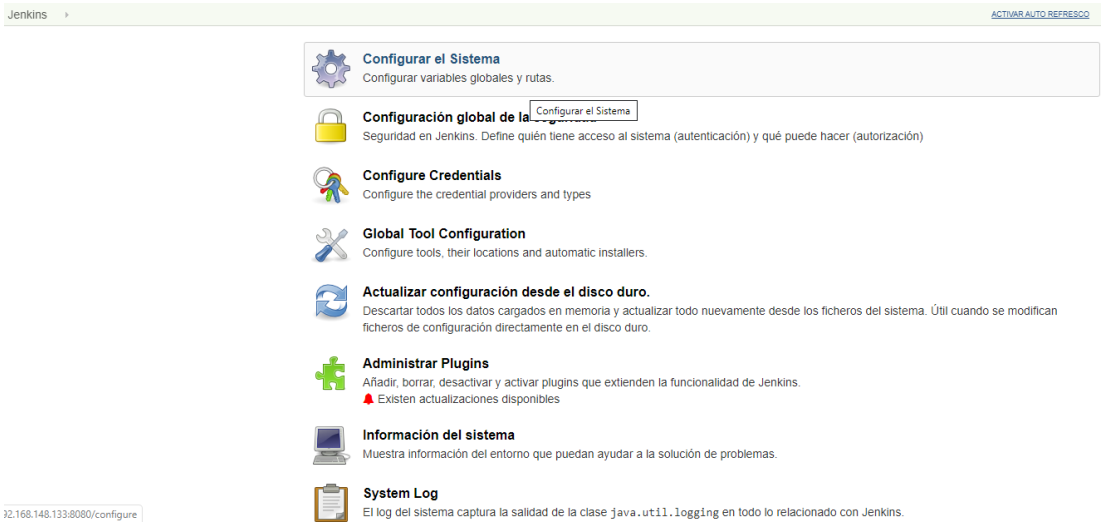


Imagen Configurar el Sistema
Fuente: Imagen Propia

3. Busque la sección de servidores SonarQube y haga clic en Agregar servidor SonarQube. Proporcione la URL y las credenciales del servidor. Obtenga un token de autenticación del servidor SonarQube.

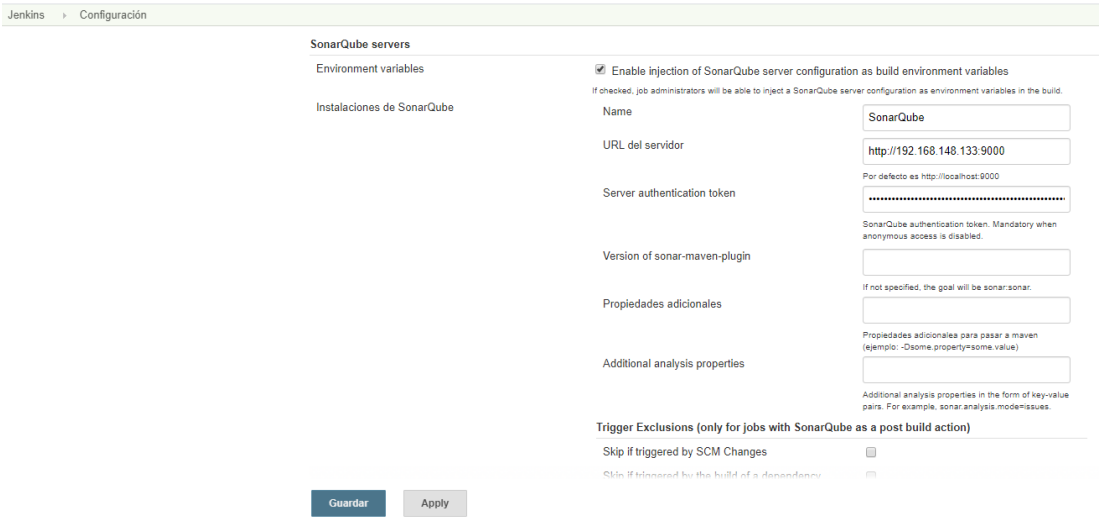


Imagen Configuración del Plugin SonarQube
Fuente: Imagen Propia