UNIVERSIDAD DON BOSCO

DECANATO DE ESTUDIOS DE POSTGRADOS

TRABAJO DE GRADUACIÓN

GUÍA DE IMPLEMENTACIÓN Y BUENAS PRÁCTICAS DE INTEGRACIÓN CONTINUA PARA EL DESARROLLO DE APLICACIONES WEB EN EMPRESAS DEL ÁREA METROPOLITANA DE SAN SALVADOR

PARA OPTAR AL GRADO DE:

MAESTRO EN ARQUITECTURA DE SOFTWARE

ASESOR:

MG. LUIS EDUARDO GARCÍA MOLINA

PRESENTADO POR:

MARÍA LORENA DE LA FUENTE JACOBO

WILLIAM ANTONIO FLORES AYALA

ROBERTO BLADIMIR HERNÁNDEZ GUEVARA

Antiguo Cuscatlán, La Libertad, El Salvador, Centroamérica.

Febrero de 2019

Índice

Resumen

En el presente trabajo de investigación se tuvo en consideración los beneficios que provee la Integración Continua en la optimización de los procesos de construcción de software, como son la retroalimentación temprana, despliegue en menor tiempo y pruebas automatizadas. Se encontró así la oportunidad de brindar lineamientos y mayor conocimiento de Integración Continua a las empresas del Área Metropolitana de San Salvador, que cuentan con desarrollo de aplicaciones Web y no la utilizan, para que puedan optimizar sus procesos de construcción de Software implementando Integración Continua.

Para tener de referencia información del ámbito salvadoreño, se realizó la investigación en una muestra de empresas del Área Metropolitana de San Salvador que ya han implementado y se encuentran utilizando Integración Continua, así conocer su experiencia y desafíos durante el proceso de implementación.

Se entrevistó a expertos nacionales e internacionales que han participado en implementaciones de Integración Continua, para así tomar en consideración sus recomendaciones.

Se levantó un entorno de pruebas de Integración Continua con las herramientas seleccionadas en base a la investigación, dando como resultado la guía de implementación y buenas prácticas que se espera sean los insumos para futuras implementaciones en las empresas salvadoreñas.

Capítulo I: Planteamiento General de la Investigación

1.1 Descripción del tema

El desarrollo de software es crítico para las empresas, este impulsa la innovación y aumenta la ventaja competitiva. En su estudio publicado en 2018, Freeform Dynamics encontró que la importancia del software ha crecido en todas las áreas del negocio, pero sobre todo para mejorar la línea superior donde se impulsa el crecimiento, ayuda al negocio a expandirse y competir de manera efectiva. (Freeform Dynamics, 2018) El estudio encuestó a 1,279 altos funcionarios, tanto gerentes como profesionales, en un rango de medianas a grandes empresas en 15 países, ocho sectores industriales y cinco continentes.

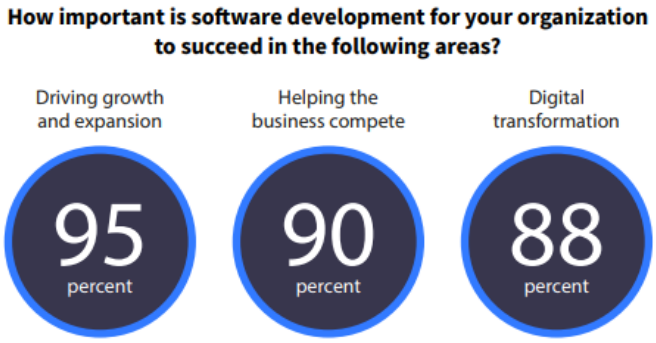


Figura 1.Tomado de: Software Lifecycle Modernization Research

Acorde a las predicciones de la Consultora International Data Corporation, sobre la industria de la tecnología de la información mundial para 2018 y años posteriores se tiene que “para 2020, el 60% de las empresas se encontrarán en el proceso de implementar nuevos cimientos de Tecnología de la Información como parte de una organización totalmente articulada en torno a una estrategia de plataforma digital”. (Torrejon, 2017)

La Tecnología de la Información debe responder oportunamente a las tendencias del consumidor y exigencias del negocio, lo que ha obligado a buscar optimizar los procesos de desarrollo de software. Las metodologías tradicionales han tenido que abrirle camino a la metodología ágil, la cual ha podido resolver muchos de los problemas que ha experimentado el desarrollo de software; se ha pasado de un modelo predictivo a un modelo adaptativo, orientado a las personas y no a los procesos (Navarro Cadavid, Fernández Martínez, & Morales Vélez, 2013).

La metodología ágil ya no es sólo una tendencia, es una necesidad que las nuevas gerencias deben adoptar para enfrentarse al actual mercado competitivo. La brecha entre tecnología y negocio debe ser cada vez más pequeña, en conjunto definen productos mínimos viables para obtener retroalimentación temprana al estar continuamente entregando valor en periodos cortos.

El desarrollo ágil por sí solo no sería capaz de lograr entregables funcionales de manera rápida, sin el apoyo de prácticas y valores culturales que disminuyan la brecha entre desarrollo y operaciones para maximizar la velocidad de entrega de un producto o servicio, desde la idea inicial hasta producción. Este conjunto de prácticas y valores se denomina DevOps.

Una de las prácticas dentro de DevOps, que optimiza el desarrollo de software con un alto grado de calidad, pero que a su vez su adopción presenta cierto grado de complejidad es la denominada Integración Continua, identificada de aquí en adelante como IC. En su popular artículo “Continuous Integration”, Martin Fowler la describe como “una práctica de desarrollo de software donde los miembros del equipo integran sus trabajos constantemente, usualmente diariamente, dando lugar a múltiples integraciones

por día. Cada integración es verificada por un compilado automático, incluyendo pruebas para detectar lo más pronto posible cualquier error en la integración.”

1.2 Problemática a abordar

En su estudio Freeform Dynamics encontró que aunque la mayoría de las empresas están comprometidas con la adopción tanto de DevOps como ágiles, la implementación se encuentra aún en etapas iniciales. (Freeform Dynamics, 2018)

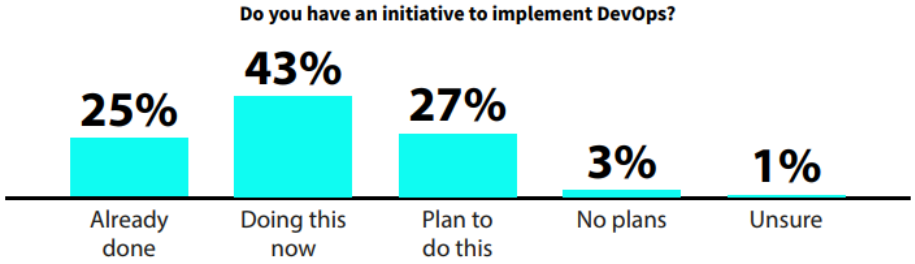


Figura 2. Tomado de: Software Lifecycle Modernization Research

En El Salvador aunque las empresas conocen los beneficios de implementar IC, tienen la dificultad de no disponer de una guía donde se contemplen los desafíos contextualizados a la realidad del desarrollo de software salvadoreño e incluyan buenas prácticas para obtener el máximo provecho de la IC. Inclusive en la investigación de campo se encontró que quienes han utilizado consultoría ha sido con una empresa o consultor extranjero.

En la investigación realizada se encontró tutoriales informales publicados en blogs enfocados a diferentes lenguajes de programación y tecnologías, cabe mencionar que la mayoría de estos se encuentran en idioma inglés.

Antecedentes

Se tiene el caso de éxito en España del banco BBVA, el cual tuvo una transformación digital adoptando metodologías ágiles para la gestión de proyectos para ganar rapidez y

flexibilidad. Francisco Gonzales, presidente de BBVA, está convencido que los bancos se transformarán en empresas de software.

En 2018 la aseguradora Allstate obtuvo un reconocimiento en los premios Gartner Eye on Innovation, por la API que desarrollaron en 21 días utilizando TDD, desarrollo de software orientado a pruebas, con IC y prácticas de desarrollo ágil. Al asociarse con Uber, Allstate tuvo que revolucionar sus productos de seguros comerciales y los procesos de creación y manejo de reclamos. Sólo tenían 3 semanas para el inicio de la vigencia de la póliza y debían tener una solución para la recepción de reclamos. Formaron un equipo de negocio y tecnología para desarrollar la API que permitió integrar su sistema de reclamos con terceros; lo que ha dado camino a nuevos tipos de pólizas de automóviles comerciales y otras formas de economía compartida con modelos de negocio disruptivos.

A nivel mundial la IC como práctica ágil es usada por grandes empresas como Amazon, Netflix, Target, Wal-Mart, Etsy, Facebook, Adobe, Spotify y GoPro, permitiéndoles reducir el tiempo de salida al mercado del desarrollo de sus aplicaciones con código de mayor calidad. Y cada vez se unen más empresas a esta tendencia obteniendo grandes resultados.

Etsy paso de tener publicaciones del sitio completo 2 veces a la semana que tardaban 4 horas a un enfoque más ágil. InfoQ, en el artículo publicado en 2014, amplía como Etsy ha logrado por medio de la IC, junto a otras prácticas de entrega continua, tener 50 implementaciones al día y más de 14,000 series de pruebas con 150 desarrolladores. (Miranda, João 2014)

La Consultora International Data Corporation en sus predicciones indica que se acelerará el ritmo de innovación debido a la adopción de prácticas de DevOps, como IC,

que permiten aumentar el número de lanzamientos anuales de código de aplicación en un 50%. (IDG Network, 2016)

1.3 Justificación

Son ampliamente conocidos los beneficios de aplicar el desarrollo ágil y prácticas de DevOps, como IC, en la optimización de los procesos de la construcción de software así brindando una ventaja competitiva a las empresas donde su estrategia se encuentra impulsada por la tecnología.

En el artículo en línea de Nearshore Americas, el principal recurso de investigación, datos y noticias para tomadores de decisiones de negocios que participan en el suministro de servicios de conocimiento e inversión en América Latina, el Caribe y Canadá, indican que las empresas están aprendiendo a tomar ventaja de las nuevas tecnologías apoyándose del desarrollo ágil para generar valor al negocio y esto está generando una disrupción dentro de la industria de Tecnología de Información.

Antes las empresas Norteamericanas buscaban adquirir sus desarrollos de software en la India, sin embargo el desarrollo ágil necesita de constante comunicación entre tecnología y el negocio, sobre todo para la resolución de bugs; colocando en desventaja a la India y brindándole una ventaja por la cercanía de horario a las empresas de Latinoamérica. (Heffernan & Woollacott, 2015)

Esta ventaja competitiva puede ser aprovechada por las empresas salvadoreñas que se dedican al desarrollo de software, al encontrarse listas aplicando desarrollo ágil y prácticas de DevOps, que les permitan optimizar sus procesos de construcción de software.

Al entregar una guía de implementación, se está apoyando a las empresas salvadoreñas a dar los primeros pasos a montarse en un entorno ágil con IC y así estar más preparados para ser competitivos en el mercado.

1.4 Objetivos

1.4.1 Objetivo General

La investigación tiene como objetivo diseñar una guía de implementación y buenas prácticas de IC para el desarrollo de aplicaciones Web en empresas del Área Metropolitana de San Salvador, AMSS, basada en la experiencia de las empresas de la misma área y expertos en el campo de IC. Además, se pretende sea una referencia para empresas que desarrollan aplicaciones Web y tengan el interés de implementar IC para optimizar sus procesos de construcción de software.

1.4.2 Objetivos Específicos

- Identificar los desafíos más comunes durante la implementación de IC en las empresas en estudio del AMSS.
- Identificar las herramientas más utilizadas en un entorno de IC para las actividades de desarrollo de aplicaciones Web.
- Diseñar una guía de buenas prácticas en las actividades de IC para el desarrollo de aplicaciones Web.
- Diseñar una guía de implementación de IC para el desarrollo de aplicaciones Web, enfocada al entorno de desarrollo Java y .NET que la mayoría de las empresas en estudio del AMSS utilizan.

1.5 Alcances

La investigación contempla una guía de implementación de IC para el desarrollo de aplicaciones Web en empresas del AMSS. Así mismo se incluye una guía de buenas prácticas que le permita a las empresas optimizar las actividades de IC en base a la experiencia de las empresas del AMSS y expertos en IC.

1.6 Limitantes

La investigación incluyó únicamente empresas del AMSS, específicamente aquellas localizadas en los municipios de San Salvador, Santa Tecla y Antiguo Cuscatlán, en donde se encuentran la mayoría que cumplen al perfil de la investigación. Se buscaba que las empresas utilizaran IC en el desarrollo de aplicaciones Web y así garantizar la recolección de toda la información necesaria para la investigación.

El entorno de desarrollo Web al que se enfocó la guía de implementación se seleccionó en base a la investigación realizada en las empresas en estudio del AMSS, dando como resultado dos lenguajes de desarrollo predominantes C# .Net y Java.

Se excluye el proceso de implementar el entorno de IC, y cabe mencionar que la validación de la guía de implementación fue mediante el juicio de expertos en el campo de IC.

1.7 Preguntas de investigación

La investigación es sustentada en base a una muestra de empresas del AMSS que han implementado IC y en el juicio de expertos, donde se buscó responder las siguientes preguntas:

- ¿Cuáles son los desafíos más comunes durante la implementación de IC?

- ¿Cuáles son las herramientas más utilizadas en un entorno de IC para desarrollo Web?
- ¿Cómo implementar un entorno de IC para el desarrollo de aplicaciones Web?
- ¿Qué buenas prácticas aplican en las actividades de IC?

Las respuestas nos permitieron construir las guías en base a las herramientas más utilizadas para la implementación de un entorno de IC.

Capítulo II: Marco Conceptual

2.1 Desarrollo Web

Para poder contextualizar la investigación es importante comprender el significado del proceso de desarrollo Web, así como de todos los elementos relacionados a este. El proceso de desarrollo Web suele incluir diseño Web, escritura de código del lado del cliente y servidor y configuración de seguridad de red, entre otras tareas para poder desplegar la aplicación Web. (Techopedia, 2017).

2.2 Aplicación Web

El desarrollo Web implica la construcción de aplicaciones basadas en los protocolos de internet. Así, una aplicación Web se define como un sistema de software al que se accede a través de Internet o Intranet y que se construyen de acuerdo con ciertas tecnologías y estándares. (Oliveros, del Valle Rojo, Wehbe, & Rousselot, 2011).

Por otro parte, Casteleyn, Dolog y Matera. (2009) en su libro Engineering Web Applications indica que una aplicación Web se encuentra compuesta por una capa de presentación en HTML donde la lógica de la misma tiende a encontrarse en un servidor Web remoto o en algunos casos en múltiples servidores remotos distribuidos.

2.3 Ingeniería de Sistemas

Dado que las aplicaciones Web tienen características particulares en relación a los sistemas de software tradicional es necesario abordarlas desde una perspectiva global hasta lo específico para conocer los diferentes conceptos o áreas que aglutina tales como: arquitectura de la información, ingeniería de software, ingeniería de datos, indexación, entre otros. Para ello se parte definiendo todo alrededor de la ingeniería de sistemas.

Blanchard (1995) define la ingeniería de sistemas como “la aplicación de técnicas científicas y de ingeniería para transformar una necesidad operativa en la descripción de los parámetros de prestaciones de un sistema y en su configuración mediante la utilización de un proceso iterativo de definición, síntesis, análisis, diseño, prueba y evaluación; integrando los parámetros técnicos relacionados y asegurando la compatibilidad de todas las interrelaciones físicas, funcionales y del programa, de forma que se consiga la mejor definición y diseño del sistema completo; también integrando los aspectos de fiabilidad, mantenibilidad, seguridad, supervivencia, de personal y otros similares en el proceso global de ingeniería para conseguir los objetivos técnicos, de coste y de calendario fijados”.

De igual manera la IEEE (1983) coloca su propia definición, diciendo que la "Ingeniería de Sistemas es la aplicación de las ciencias matemáticas y físicas para desarrollar sistemas que utilicen económicamente los materiales y fuerzas de la naturaleza para el beneficio de la humanidad."

2.4 Ingeniería de Software

La Ingeniería de Sistemas involucra técnicas y metodologías para la construcción de hardware, así como de software; su enfoque es más general en los procesos de ambos campos. El enfoque exclusivo para el desarrollo de software está bajo la responsabilidad de la Ingeniería de Software, cuya definición se establece a continuación:

Fritz Bauer (Randell, 1996), definió la ingeniería de software como el establecimiento y uso de principios robustos de ingeniería para obtener económicamente un software confiable y que funcione de modo eficiente en máquinas reales.

Por su parte, la IEEE (1983) menciona que la ingeniería de software es la aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento del software; es decir, la aplicación de la ingeniería de software y el estudio de enfoques sistemático.

Por lo tanto, se puede decir que la ingeniería de software en el desarrollo de aplicaciones Web es la aplicación de un enfoque sistemático, disciplinado y cuantificable.

2.4.1 Ciclo de Vida del Software

El desarrollo de aplicaciones Web se encuentra dentro de la Ingeniería de Software referenciada en el Ciclo de Vida del Software, el cual también es conocido como proceso de desarrollo de software o Ciclo de Vida del Desarrollo de Software.

Algunos autores lo exponen como un marco de referencia que contiene los procesos, las actividades y las tareas involucradas en el desarrollo, así como también la explotación y el mantenimiento de un producto de software, iniciando con la definición de requerimientos y finalizando en el momento en que éste deja de usarse,. A su vez, el Ciclo de Vida del Software debe ser confiable, predecible y eficiente. (Vaca Barahonaa, Hidalgo Poncea, & Arcos Medinaa, 2015).

La Norma ISO/IEC Standard 12207:2008 lo define como “un marco de referencia que contiene los procesos, las actividades y las tareas involucradas en el desarrollo, explotación y mantenimiento de un producto software, abarcando la vida del sistema desde la definición de requisitos hasta que se deja de utilizar”.

2.4.2 Etapas del ciclo de vida del software

El Ciclo de Vida del Software está constituido por etapas o fases las cuales pueden variar según el modelo de desarrollo que se utilice en el proyecto.

Entre autores hay diferencias en cuanto al nombre o cantidad de las etapas por ejemplo para Pressman (2010) hace referencia al ciclo de vida clásico al modelo en cascada y dicta que sus etapas son la especificación de los requerimientos por parte del cliente para continuar con la planeación, modelado, construcción y despliegue, concluyendo con el apoyo del software terminado. Otros autores como Sommerville (2005) menciona que lo constituyen el Análisis y definición de requerimientos, diseño del sistema y del software, Implementación y prueba de unidades, Integración y prueba del sistema, para concluir con el Funcionamiento y mantenimiento.

A pesar del modelo de desarrollo o del autor podemos identificar que por lo general se incluye el análisis, el diseño, la codificación, donde se escribe el código fuente basado en los algoritmos de la etapa de diseño, las pruebas donde se integra el sistema, para luego examinarlo en busca de errores y finalizando con el despliegue o puesta en marcha.

2.4.3 Integración Continua

En las últimas etapas como lo son la codificación, las pruebas y el despliegue siempre se identificado lo recurrente que son los problemas a la hora de intentar integrarse. Esto realmente no es un problema nuevo, existe desde que se comenzaron a construir los primeros desarrollos.

Por lo general la integración del software no se vuelve complicada en proyectos en los que solo existe una persona y donde se realizan pocas modificaciones al código y cuyos sistemas de dependencias externas son mínimos. Ahora qué sucede si ese proyecto requiere cambios considerables y recurrentes, seguramente eso implica contratar más personal, el solo hecho de contratar un desarrollador más ya le incluye un grado de complejidad al momento de realizar la integración.

Para solventar los problemas de la integración del software es que la industria crea una práctica de ingeniería a la que bautizan con el nombre de IC, permitiendo reducir riesgos, procesos repetitivos, haciendo el software desplegable y logrando visualizar el estado actual del proyecto. Pero ¿cuál es la definición de la IC que se manejara en este documento? Fowler, Duvall (2007) la define como una práctica de desarrollo de software donde los miembros del equipo integran sus trabajos constantemente, usualmente diariamente - dando lugar a múltiples integraciones por día. También knowledge Powerhouse (2016) en el libro Microservices Interview Question la define como un enfoque de desarrollo de software en el cual los desarrolladores integrar su código dentro de un repositorio común varias veces al día.

Para efectos de esta investigación por ser una definición más clara y contundente se tomará la definición de IC dada por Martin Fowler considerando también que es el máximo referente en dicho campo.

Lo que busca la IC es automatizar las actividades de construcción por lo cual es importante definir el término automatizado. La RAE lo define como el hecho de aplicar la automática a un proceso o un dispositivo, tal concepto es bastante genérico puesto que puede aplicarse a diferentes contextos. Duvall, Matyas y Glover. (2007) lo define como un proceso que una vez iniciado no requiere ninguna intervención del usuario.

Las actividades de construcción son continuas, acorde a la definición del diccionario Merriam Webster (2016), continuo hace referencia a algo ininterrumpido en el espacio y el tiempo. Tal definición significa que el proceso de construcción correrá todo el tiempo, sin embargo, ese no es el caso de IC. Al tomar la definición de Paul Duvall se obtiene más alineación a IC quien lo relaciona y describe contextualizando las tareas de un servidor

de integración en donde un proceso ejecuta continuamente sondeos o verificaciones de los cambios sucedidos en un repositorio de control de versiones.

2.4.3.1 Proceso de Construcción

Las actividades que IC automatiza se encuentran en el proceso de construcción. La enciclopedia digital Techopedia define proceso de construcción como el proceso por medio del cual el código fuente es convertido en forma autónoma para que pueda ejecutarse en un ordenador, es decir es el código fuente que se convierte en código ejecutable. Duvall (2007) nos proporciona una definición bajo contexto de IC, donde establece que construcción podría incluir un conjunto de actividades para generar, probar, inspeccionar y desplegar software.

2.4.3.2 Componentes de un entorno de IC

Para entender el proceso de IC, es necesario conocer donde inicia realmente dicho proceso. Todo inicia cuando el desarrollador hace confirmaciones de su código fuente a un repositorio. En proyectos donde hay muchas personas con distintos roles pueden hacer cambios que inician el ciclo de IC. Los desarrolladores cambian el código fuente, el o los administradores de base de datos cambian la definición de tablas, los equipos construyen y despliegan cambios en los archivos de configuración, interfaces, etc.

2.4.3.2.1 REPOSITORIO DE CONTROL DE VERSIONES

El primer componente descrito por Duvall (2007) es el relacionado al Repositorio de Control de Versiones. Un repositorio de control de versiones debería ser un requisito usarlo con el fin de realizar la IC. De hecho, incluso si no usamos IC, un repositorio de control de versiones debería ser un estándar para un proyecto de software. El propósito de un repositorio de control de versiones es administrar los cambios del código fuente y

otros activos de software, como por ejemplo la documentación, utilizando un repositorio de acceso controlado.

2.4.3.2.2 SERVIDOR DE INTEGRACIÓN

El segundo componente es el Servidor de IC. Un servidor de integración ejecuta una compilación de integración cada vez que se realiza cada vez que se realiza cambios en el repositorio de control de versiones. Normalmente, se configura el servidor de IC para verificar cambios en el repositorio cada pocos minutos o más. El servidor de integración recibe los archivos fuentes y ejecuta un script o varios scripts de compilación. Los servidores de integración pueden también ser programados para construir con frecuencia, como por ejemplo cada hora, pero eso no significa que estemos aplicando IC. Los servidores de IC normalmente proveen una conveniente pizarra donde los resultados construidos son publicados.

2.4.3.2.3 SCRIPT DE CONSTRUCCIÓN

El tercer componente está relacionado con el Script de Construcción. Un script de construcción básicamente es un simple script o conjunto de scripts que se pueden usar para compilar, realizar pruebas, inspeccionar y desplegar software. Podríamos implementar un script de construcción sin implementar un sistema de IC. Ant, NAnt, make, MSBuild son algunos ejemplos de herramientas de construcción que pueden automatizar la construcción del ciclo de IC.

2.4.3.2.4 MÁQUINA DE INTEGRACIÓN

El cuarto elemento es denominado Mecanismo de Retroalimentación. Debido a que uno de los propósitos claves de la IC es poder entregar retroalimentación en una construcción de integración. Siempre se quiere conocer tanto como sea posible si ha

habido problemas con las últimas construcciones. Así al recibir esta información de forma temprana y puntual es posible solucionar problemas rápidamente.

2.4.3.2.5 SERVIDOR DE PRUEBA

El quinto componente descrito por Duvall (2007) es una Maquina de Integración de construcción. La máquina de integración de construcción es un elemento separado, es exclusivo, cuya responsabilidad es integrar el software. A pesar que Paul no lo menciona en su libro es importante mencionar un componente sumamente importante en un entorno de IC, estamos hablando del servidor de pruebas que servirá para automatizar todas las pruebas necesarias.

2.4.3.3 Características de IC

Además de los componentes, en su libro Duvall (2007) expone que la IC tiene 5 aspectos que la caracterizan, describe la compilación del código fuente, la integración de base de datos, pruebas, inspecciones, etc.

La primera característica está relacionada con la compilación del código fuente esta es básica y una de las comunes de IC y muchas veces se convierte en un sinónimo de IC. La compilación implica crear código ejecutable para que sea legible por los humanos, claro que esto es aplicable a un contexto donde los resultados son binarios, en los entornos Web es ligeramente diferente el proceso de compilación, aunque no se esté generando binarios muchos de ellos proporcionan la capacidad de crear los archivos necesarios que son los que luego serán desplegados en un entorno de producción.

Otra característica de IC es la integración de base de datos. Aunque algunas personas consideran que la integración del código fuente y la integración de base de datos son dos procesos totalmente separados. Lo cierto es que si usamos una base de datos como una

parte integral de nuestra aplicación de software la única forma de asegurar la integración de la base de datos es por medio de una única fuente: el repositorio de control de versiones.

Muchos consideran IC sin las pruebas automáticas y continuas para no ser IC. Y claro es que sin pruebas automáticas es difícil para los desarrolladores tener confianza en los cambios del software. La mayoría de desarrolladores que utilizan IC hacen uso de pruebas utilizando herramientas como JUnit, NUnit u otros framework de pruebas.

El despliegue es otra característica de IC. Duvall (2007) define el despliegue continuo como la capacidad de ofrecer trabajo o software desplegable en cualquier momento. Lo que significa que se cumple uno de los objetivos claves de la IC que es generar artefactos de software empaquetados con los últimos cambios del código fuente y ponerlo a disposición de los entornos de prueba.

Para finalizar, la característica denomina documentación y retroalimentación hace referencia a la utilización de ciertas herramientas que pueden generar documentación como: diagrama de clases, documentación sobre métodos, argumentos u otra información. Por otra parte, tenemos la retroalimentación que básicamente se refiere al hecho que un entorno de IC nos ofrece velocidad, lo que provee rápidamente retroalimentación a los desarrolladores o a los involucrados en el proyecto.

Capítulo III: Marco Metodológico

3.1 Tipo de Investigación

Tomando como punto de partida la definición de Martínez (2006), la investigación sigue una metodología cualitativa con rasgos cuantitativos, dado que se necesita saber la opinión, la experiencia y el conocimiento de empresas y expertos de IC, pero también es necesario clasificar y medir los hallazgos encontrados; y para lograr interpretar, entender y plasmar de mejor manera el análisis, se deben utilizar modelos mentales, que brinden una visión de mayor claridad, permitiendo dar un significado sustancial y relevante para el desarrollo del proyecto. A parte de ello, se carece de hipótesis porque está abierto a todas las posibilidades, y la mejor será evidenciada en la culminación de la investigación.

A su vez, es de naturaleza descriptiva, donde los hechos son explicados y documentados tal cual son observados, sin explicar el porqué de ellos. De igual manera, dado que el estudio es realizado de manera autónoma y, como ya se mencionó en el párrafo anterior, no se dispone de una hipótesis en la que se tenga que apoyar de una correlación entre las variables encontradas. Basándonos en la definición de Arias que establece como estudios descriptivos los que miden de forma independiente las variables y aun cuando no se formulen hipótesis, tales variables aparecen enunciadas en los objetivos de investigación. (Arias, 1999).

El universo del estudio, el cual abarca las empresas del AMSS que utilizan IC, es de carácter variable porque que cada día se pueden crear nuevas empresas en AMSS, que pueden o no tener desarrollo de aplicaciones Web y utilizar IC en sus actividades de construcción.

Además, el universo del estudio es de carácter finito, que abarca las empresas del AMSS que utilizan IC en el desarrollo de aplicaciones Web. Se conoce que actualmente en el AMSS hay pocas empresas que cumplen con el perfil para ser objeto de estudio y también se necesita la autorización para documentar y observar los procesos del área de Tecnología de las empresas.

Por las causas anteriores, y en base a los artículos tanto de Martínez-Salgado (2012) como de Casal y Mateu (2003), el tipo de muestreo que se utilizó fue el no probabilístico y este a su vez es intencional, llamado también por conveniencia u opinático, dado que éste es más flexible y se acopla en gran medida a la complejidad que presenta la investigación. (Martínez-Salgado, 2012)

3.1.1 Población:

Se consideraron las empresas del AMSS cuyas actividades económicas son de programación informática, y sumando a ellas, las que a su vez difieren de dichas actividades económicas pero que desarrollan aplicaciones Web para lograr sus objetivos de negocio.

3.1.2 Muestra:

Para el estudio se utilizó un muestreo intencional, donde las empresas seleccionadas del AMSS se encuentran utilizando IC por un mínimo de 6 meses y desarrollan aplicaciones Web en los lenguajes de C# o Java. Estando estos lenguajes dentro de los 5 más populares acorde al estudio de TIOBE en febrero del 2018, lo que nos permitió abarcar mayor representatividad. (TIOBE software BV, 2018)

Adicional se debía tener la autorización para poder recopilar información de la implementación y prácticas de IC de la empresa, así como tener un área o persona responsable del proceso de IC.

Para poder tener suficiente representatividad se tienen 7 empresas. Por medio de diferentes canales de comunicación, como llamadas, chats o entrevistas, se recolectó información para validar que las empresas seleccionadas cumplan con los criterios definidos.

3.2 Procedimiento

Para poder iniciar con la investigación y alcanzar los objetivos específicos:

1 Se definieron los criterios de selección: Ser una empresa del AMSS que utiliza IC por un mínimo de 6 meses y desarrolla aplicaciones Web en los lenguajes C# o Java. Así como tener autorización de la empresa y un área o persona encargada de IC.

2 Se contactó vía correo electrónico, telefónico o chat para identificar las empresas que cumplían con los criterios establecidos.

3 Se solicitó la autorización al contacto de la empresa para poder entrevistar y recopilar información de la implementación y prácticas de IC dentro de la empresa.

4 Se definió el perfil de los expertos en IC, pudiendo ser nacionales o extranjeros con 3 a 5 años de experiencia en desarrollo Web y 1 año de experiencia en IC.

5 Se contactó vía correo electrónico, telefónico o chat a los expertos que cumplían con el perfil para solicitarles su participación en la investigación.

6 Se creó un directorio con la información de contacto de los expertos que aceptaron participar en la investigación.

Se amplía por cada objetivo específico planteado las actividades que se realizarán para alcanzarlos:

Identificar los desafíos más comunes durante la implementación de IC en las empresas en estudio del AMSS.

1 Se realizó un estudio bibliográfico sobre los desafíos en las implementaciones de IC. Se utilizaron como fuentes bibliográficas investigaciones, artículos de consultoras de software y opiniones de expertos.
2 Se diseñó una guía de referencia para la entrevista con el encargado de IC. Se enfocó en investigar cuales fueron los desafíos que enfrentaron durante la implementación de IC.
3 Se concertó una cita con el encargado de IC de cada una de las empresas en estudio.
4 Se realizó la entrevista al encargado de IC apoyándose en la guía diseñada. Se grabó y redactó una síntesis de cada una de las entrevistas.
5 Se diseñó una guía de referencia para la entrevista con expertos de IC. Se enfocó en investigar cuales han sido los desafíos más comunes que han encontrado durante las implementaciones de IC donde han participado.
6 Se concertó una cita con cada uno de los expertos de IC, que aceptaron participar en la investigación.
7 Se realizó la entrevista con cada uno de los expertos de IC, apoyándose en la guía diseñada. Se grabó y redactó una síntesis de cada una de las entrevistas.
8 Se tabularon las respuestas de las entrevistas.

9 Se analizaron los resultados obtenidos para determinar los desafíos más comunes durante la implementación del entorno de IC.

Identificar las herramientas más utilizadas en un entorno de IC para las actividades de desarrollo de aplicaciones Web.

1 Se realizó un estudio bibliográfico sobre las herramientas y tecnologías más utilizadas en un entorno de IC basado en aplicaciones Web sobre C# y Java. Se utilizaron como fuentes bibliográficas investigaciones, artículos de consultoras de software y opiniones de expertos.
2 Se diseñó una guía de referencia para la entrevista con el área o encargado de IC. Se enfocó en investigar cuales son las herramientas y tecnologías integradas para implementar el entorno de IC, así como comprender la razón de la selección de las mismas.
3 Se concertó una cita con el encargado de IC de cada una de las empresas en estudio.
4 Se realizó la entrevista al encargado de IC, apoyándose en la guía diseñada. Se grabó y redactó una síntesis de cada una de las entrevistas.
5 Se diseñó una guía de referencia para la entrevista con expertos de IC. Se enfocó en investigar cuales son las herramientas y tecnologías que han utilizado para implementar IC y el porqué de su selección.
6 Se concertó una cita con cada uno de los expertos de IC, que han accedido a participar a la investigación.
7 Se realizó la entrevista con los expertos de IC, apoyándose en la guía diseñada. Se grabó y redactó una síntesis de cada una de las entrevistas.

8 Se tabularon las respuestas de las entrevistas.

9 Se analizaron los resultados obtenidos de las entrevistas y observaciones. Se identificaron dos lenguajes predominantes siendo C# .Net y Java, por lo que se seleccionó las herramientas más utilizadas en cada entorno de IC para las actividades de desarrollo de aplicaciones Web en el AMSS.

Diseñar una guía de buenas prácticas en las actividades de IC para el desarrollo de aplicaciones Web.

1 Se realizó un análisis de los informes creados a partir de las entrevistas y así se identificó buenas prácticas en las actividades de IC.

2 Se realizó un estudio bibliográfico sobre buenas prácticas en las actividades de IC. Se consultaron diferentes tipos fuentes como tesis de maestría, artículos de consultoras de software y opiniones de expertos.

3 Se contactó a los expertos de IC, que aceptaron participar en la investigación.

4 Se revisó la guía de buenas prácticas con los expertos en IC donde se validó y abonó a las buenas prácticas.

5 Se recibió la carta de cada experto avalando la guía de buenas prácticas.

Diseñar una guía de implementación de IC para el desarrollo de aplicaciones Web, enfocada al entorno de desarrollo que la mayoría de las empresas en estudio del AMSS utilice.

1 A partir de toda la información recolectada en la investigación, se diseñó la arquitectura del entorno de IC para C# .Net y Java.

2 Se instaló, configuró e integró las herramientas dentro de cada entorno de IC bajo la arquitectura diseñada en servidores locales.

3 Se definió y ejecutaron las pruebas para validar el correcto funcionamiento de cada entorno de IC.

4 Se creó la guía de implementación de cada entorno de IC a partir de la información recolectada y los resultados obtenidos del entorno implementado. La guía contempla la instalación, configuración e integración de las herramientas.

5 Se validó la guía de implementación levantando cada entorno de IC a partir de la misma en la nube.

6 Se ejecutaron las pruebas para validar el correcto funcionamiento de cada entorno de IC en la nube.

7 Se contactó a cada uno de los expertos que accedieron participar en la investigación para solicitar apoyo en validar la guía de implementación bajo los criterios de claridad y fácil comprensión en redacción, selección, configuración e integración de herramientas.

8 Se recibió e incorporó en la guía de implementación la retroalimentación de los expertos.

9 Se recibió la carta de cada experto avalando la guía de implementación.

3.3 Perfil Empresa

Empresas con operaciones en el área metropolitana de San Salvador, que han implementado y estén utilizando IC en sus desarrollos internos de aplicaciones web, con un mínimo de 6 meses.

3.4 Perfil Experto

Profesionales en Ciencias de la Computación con un mínimo de 3 años de experiencia en desarrollo de aplicaciones web y un mínimo de un año de experiencia en IC. Debe tener conocimientos generales sobre herramientas de versionado, administración de repositorios, pruebas unitarias, integración y construcción de Software. Así como conocimiento de estándares y buenas prácticas en desarrollo de aplicaciones web.

Capítulo IV: Administración de la Investigación

4.1 Entregables del proyecto

4.1.1 Guía de buenas prácticas

Detalle de buenas prácticas en las actividades de IC para poder optimizar el uso de la misma basada en las experiencias de las empresas en estudio y el juicio de expertos.

4.1.2 Guía de implementación del entorno de integración continua para el desarrollo de aplicaciones web en Java

Documento que incluye el diseño de la arquitectura, instalación y configuración de las herramientas que conforman el entorno de IC para el desarrollo de aplicaciones web en Java. Basado en la investigación realizada en las empresas en estudio del AMSS y juicio de expertos. Se espera sea un insumo para dar los primeros pasos en la construcción de un entorno de desarrollo ágil utilizando integración continua.

4.1.3 Guía de implementación del entorno de integración continua para el desarrollo de aplicaciones web en C# .Net

Documento que incluye el diseño de la arquitectura, instalación y configuración de las herramientas que conforman el entorno de IC para el desarrollo de aplicaciones web en C# .Net. Basado en la investigación realizada en las empresas en estudio del AMSS y juicio de expertos. Se espera sea un insumo para dar los primeros pasos en la construcción de un entorno de desarrollo ágil utilizando integración continua.

Siendo revisadas y avaladas por 4 expertos en integración continua. Las cartas de validación se encuentran en el Anexo 5.1.3.

4.2 Entrevistas con las empresas en estudio

4.2.1 Kadevjo con Nelson Chicas, Gerente General.

Kadevjo es una empresa de desarrollo de soluciones digitales creativas, desde la creación de prototipos hasta su implementación. En 2013 implementaron en 2 semanas IC, logrando automatizar la construcción, análisis de código y automatización de pruebas. Lo que los motivo a implementar IC fue buscar ahorrar tiempo, así como optimizar esfuerzo en el proceso de integrar y crear el build continuamente al utilizar metodología ágil, "Scrum". Principalmente implementaron IC para mejorar la calidad ya que al ser automatizado se estandariza el proceso.

La implementación de IC se realizó con esfuerzo interno basándose en la documentación oficial de Jenkins, pero el mantenimiento de la configuración de Jenkins era complejo. Tuvieron una segunda implementación de IC creando sus scripts de integración continua y utilizando Git. Al tener el 80% de sus desarrollos en .Net, encontraron que Azure tenía Mobile App Center, una herramienta de IC con las herramientas integradas sin tener que administrar las configuraciones y es la que están utilizando.

Entre los desafíos que tuvieron durante la implementación fue determinar en dónde se originaba el error, si en Jenkins o en las herramientas que utiliza Jenkins. La solución de los errores de las herramientas no la encontraron en la documentación oficial de Jenkins, si no en foros.

Otro desafío utilizando Jenkins se presentó cuando varios equipos querían compilar al mismo tiempo y se llegaba al límite de uso del CPU del servidor, lo que ocasionaba que se perdiera el proceso. Se solucionó utilizando una cola de operaciones.

Actualmente dentro del flujo de IC tienen la creación de build, análisis de código estático, pruebas unitarias y despliegue. Al tener IC es clave el control de versiones que se está utilizando, el más común es Git. Utilizan el modelo de ramas GitFlow, donde tienen dos ramas principales “master” y “develop”; así como por cada funcionalidad se crea una rama. Cuando se termina la funcionalidad está se une a la rama de desarrollo y por lo menos cada 3 días se hace push al repositorio lo que ejecuta automáticamente el flujo de IC; sí hay alguna falla los procesos generan reportes para identificarla. Al terminar el flujo en la rama de desarrollo y que todo está estable se une a la rama de producción. En producción solamente se construye el software y se realiza el despliegue. Por el momento las pruebas funcionales y de estrés se realizan de forma manual.

Entre las buenas prácticas que han identificado es utilizar Git. SVN como control de versiones es funcional pero al momento de implementar IC es mejor utilizar Git, porque tiene funcionalidades que le permite acoplarse mejor al flujo. Así como utilizar GitFlow, la regla que utilizan es hacer “pull” y “push” todos los días y ejecutar el flujo de IC cada 3 días.

En la actividad de pruebas, re utilizan scripts de pruebas que son comunes entre proyectos para optimizar tiempo.

4.2.2 Telus con Carlos Claramount, Desarrollador de Software

Carlos es un experto en tecnologías Java. Ha trabajado en roles principales como líder de desarrollo, director de proyecto, desarrollador senior, analista de sistemas. Actualmente es uno de los líderes de la implementación de Integración Continua y Entrega Continua en Telus International.

Telus International es una empresa que provee servicios de centro de llamadas multi-idioma y servicios digitales de tecnologías de la información a clientes globales. Es importante aclarar que Telus International es una subsidiaria de Telus, una de las más grandes compañías de telecomunicaciones en Canadá.

Carlos ha tenido relación y experiencias en ambas compañías y de acuerdo a él, Telus es una empresa que desde hace años han estado utilizando Integración Continua.

Para Carlos, Integración Continua es un ciclo de etapas que permite que los desarrolladores utilicen herramientas que integran, construyen y prueben el código para finalmente desplegarlo libre de errores. Dentro de la lista de herramientas de Integración Continua menciona a Git como sistema de control de versiones, Jenkins como servidor de integración, Nexus como administrador de artefactos y Selenium para pruebas funcionales automatizadas, argumenta que las herramientas han sido seleccionadas basándose en algunos factores como, si son de código abierto y multiplataforma.

Dentro de las etapas de Integración Continua menciona, la construcción; la cual describe como el momento en que los desarrolladores escriben el código y un sistema de control de versiones controla cada uno de los cambios, seguido de una segunda etapa llamada publicación, que es cuando el desarrollador publica sus cambios y donde el servidor de integración se encarga de construir el software para posteriormente desplegarlo en algún ambiente de desarrollo.

En Telus se ha definido como buena práctica la definición de ramas: “master”, “develop” y “hotfix”. En cuanto al tema de la frecuencia de integrar el código no se tiene una política, se basan en la duración del sprint o es a discreción del desarrollador pero siempre dando por sentado que la parte que se está publicando es funcional.

En la etapa denominada despliegue, la cual al momento de realizar esta síntesis, en Telus International se realiza de forma manual ya que no existe un software que automatice dicha tarea. Y la verificación como última fase que involucra un equipo de control de calidad o un conjunto de pruebas automatizadas construidas a través de software que prueba software.

Telus hace uso de metodologías ágiles, hay equipos que utilizan "Scrum" y otros usan "Kanban" porque su foco es en hacer entregas continuas basadas en entregables o características funcionales y con valor para el negocio.

La alta gerencia se ha involucrado no solo en temas de Integración Continua, sino en todos los proyectos de innovación para la compañía, no hay bloqueo alguno, por el contrario es desde ese nivel que se presiona y empuja para que esos cambios se hagan realidad, argumenta Carlos.

Hay muchos desafíos que las compañías enfrentan al implementar integración continua y Telus no es la excepción. La falta de capacitación o el desconocimiento generalizado de Integración Continua es uno de ellos y este aumenta las probabilidades que la implementación fracase. Una de las recomendaciones que Carlos nos comparte para minimizar ese riesgo es apoyarse de consultores experimentados en Integración Continua para hacer más fácil el proceso de implementación.

4.2.3 AFP Crecer con Oscar Dinarte

Oscar Dinarte es un experto en desarrollo de software en la administradora de pensiones AFP Crecer. Esta compañía ofrece a los trabajadores administrar sus ahorros previsionales de forma ética y eficiente con el objetivo de proteger sus ahorros y asegurarles una pensión futura.

La implementación de Integración Continua en esta administradora de pensiones nace como respuesta a la exigencia de construir software de calidad y de forma rápida. Dado que, de acuerdo a Óscar, habían venido experimentando una serie de problemas que iban desde la mala calidad hasta problemas de integrar el código y por tanto salidas a producción costosas en tiempo y dinero, sumado al desafío de cómo gestionar los problemas de la integración del código. Se tuvo que cambiar el modelo de desarrollo que hasta ese momento había sido un modelo en cascada.

Para hacer la implementación de Integración Continua se apoyaron de consultores especializados en ello.

Además se realizó todo un análisis y levantamiento de todos los requerimientos haciendo uso de entrenamiento especializado en metodologías ágiles como "Scrum" y XP. Durante este proceso se involucró a todo el equipo de desarrollo, esto con el objetivo que todos estuvieran consciente y sobre todo que se pudieran formar en esa metodología, de tal forma que se convirtiera en una cultura de trabajo más adelante.

La empresa en ese momento estaba invirtiendo en proyectos de innovación, como el que se mencionó anteriormente. Con ello se buscaba entregas cortas, entregables que le dieran al cliente un avance y visión más real así como también la posibilidad de hacer cambios rápidos si el cliente descubre que no es el producto que se pensó inicialmente.

Para Oscar, el gran problema de implementar Integración Continua en las empresas no está en las tecnologías que se deben instalar, integrar y configurar sino más bien en el reto de convertir ese proceso en una cultura. Solo para poner en contexto, ellos experimentaron errores en producción aun cuando ya se estaba utilizando Integración Continua porque no se hicieron todas las pruebas unitarias necesarias. Para él, hacer IC

una cultura es el más grande reto a superar dado que las personas están acostumbradas a un flujo específico que han venido haciendo durante años y cambiar ese flujo genera resistencia en ellos.

Al consultar sobre la frecuencia de las integraciones, menciona que se hacen constantemente siempre y cuando los desarrolladores se hayan asegurado que su código es funcional para luego hacer la confirmación respectiva. Una vez publicados los cambios entra en acción el "pipeline" que se tiene configurado en el servidor de integración que para la AFP es Jenkins.

Las herramientas de Integración Continua, como Jenkins, SonarQube, GitLab fueron seleccionadas basándose principalmente en el factor de licenciamiento, además para ellos era importante que se ajustaran a sus necesidades y que fuese funcional y multiplataforma.

Oscar, argumenta que la información sobre implementar Integración Continua fue escasa, dedicando e invirtiendo muchas horas a la investigación y solución de problemas. Para él fue muy complicado encontrar fuentes completas que garantizaran una implementación exitosa.

Como buenas prácticas nos comparte, que siempre se debe buscar entregas cortas que garantice la una implementación simple y detección y solución rápida de errores así como también definir estándares o al menos un líneas guías que definan el flujo que se debe seguir. En sus palabras lo define como: "Ser ordenado".

4.2.4 Aerolínea con Carlos Flores, Desarrollador Java y .Net

Carlos Flores es un experto desarrollador Java y .Net, cuenta con más de 7 años trabajando en el área de desarrollo de software y ahora desempeñando el rol de coordinador de equipos de desarrollo. Carlos trabaja en una aerolínea, una marca

comercial que representa a las aerolíneas latinoamericanas, cuenta con un número de colaboradores de más de 19,000. Es especializada en transporte de pasajeros y carga, además tiene una alianza con Star Alliance con la cual pueden ofrecer una conectividad de más de 1,300 destinos.

De acuerdo a Carlos la compañía tiene una estructura jerárquica con alrededor de 400 colaboradores solo en el área de TI. Actualmente están utilizando dos metodologías ágiles, "Scrum" y "Kanban". La primera de ellas la utilizan en desarrollos evolutivos mientras que la segunda la utilizan para desarrollos relacionados con el soporte y mantenimiento, en otras palabras este último se convierte en el día a día.

Al hablar del personal que labora en los equipos de tecnología menciona que la mayoría tiene un promedio de 3 a 5 años y las edades de las personas ronda el promedio de los 28 años.

Una de las razones por la que implementaron integración continua debido que tenían muchos problemas de versionamiento, con las publicaciones y las pruebas, como resultado las salidas a producción eran tardías y muchas veces con errores, además comenta que en algunos casos aplicaciones tenían y otras no, lo cual representaba un verdadero desafío para ellos.

Por lo general cuando la alta gerencia no tiene claridad en el impacto que puede generar un proyecto de integración continua, primero no garantiza el apoyo para potenciar ese tipo de iniciativas y en segundo lugar no tiene intención de presupuestar para tal iniciativa. Ese no fue el caso de esta aerolínea porque de acuerdo a Carlos, ellos sí recibieron el apoyo de la alta gerencia y también un presupuesto para ello.

Una vez se contaba con el apoyo y presupuesto para ello, comenzaron a hacer una investigación de las tecnologías, identificar los problemas y aplicaciones que debían alinearse a un entorno de IC. Al momento de hacer la investigación Carlos señala que encontrar información específica y detallada que les guiará para implementar o poner a correr todo el pipeline de integración continua fue en realidad muy escasa. De hecho ellos contrataron a expertos en Integración Continua quien les ayudó a implementarlo

Antes, durante y después de la implementación aún siguieron utilizando Visual Studio para todas las aplicaciones .NET y Eclipse para las aplicaciones Java. Por otro lado, antes de implementar IC habían implementado un servidor de construcción pero no existía un pipeline completo sumando al hecho que las integraciones no se hacían diarias ni basadas en alguna política de integración continua.

Dentro de las herramientas de IC seleccionaron a Jenkins como servidor de integración, TFS como administrador de repositorios .NET, MSBuild como constructor, Git como sistema de versionamiento y Artefactory como servidor de artefactos. Algunas de estas herramientas se seleccionaron por su robustez, flexibilidad e integración con muchas tecnologías.

Carlos menciona algunos desafíos que tuvieron que superar al inicio del proceso de implementación. Uno de ellos fue cómo lograr adecuar los procesos de desarrollo a integración continua dado que no todos los equipos estaban preparados, además al inicio probablemente no se tiene control de versiones ni mucho menos las personas están preparadas para asumir el cambio en el flujo de trabajo. La construcción de pruebas unitarias representó un gran reto técnico la cual a medida avanzaba el proceso lograron superar.

Carlos finaliza compartiendo algunas buenas prácticas como versionar el código, confirmarlo y desplegarlo en el servidor de versionamiento periódicamente, cuya misión es bajar la complejidad del código y mantener el repositorio actualizado así como la construcción de pruebas unitarias que nos aseguren que el código hace lo que estamos pensando que debe hacer.

4.2.5 Institución Gubernamental

Esta institución, es una dependencia de la Presidencia de la República de El Salvador cuya misión es proporcionar a los ciudadanos los mejores servicios aduaneros y tributarios.

En esta institución gubernamental utilizan un ALM, por sus siglas en inglés Application Lifecycle Management o gestión del ciclo de vida de la aplicación, con el cual gestionan toda la integración continua. Les permite levantar documentos de planeación y gestionar todas las actividades que requiere la metodología “Scrum”. En efecto “Scrum” es la metodología agile que ellos utilizan.

Uno de los problemas con las que ellos se enfrentaron fue el hecho que para las pruebas unitarias no había alguien designado que las diseñará para que luego los desarrolladores las implementaran en su código. Quizás el problema principal que también ellos experimentaron fue que los desarrolladores se resistían al nuevo flujo de trabajo porque esto implicaba una nueva metodología y además crear pruebas unitarias e integrar su código periódicamente.

Tuleap es el software ALM que ellos utilizan integrado con SonarQube para análisis de código estático, Git como sistema de control de versiones, Jenkins como servidor de integración y SmarGit para gestión de confirmaciones.

Para ellos siempre y cuando se tenga presupuesto se debe comprar un ALM ya que gestiona e integra un entorno de IC y adquirir software que se alinee con la metodología de desarrollo que se está utilizando y el lenguaje de programación. Además recomiendan contratar expertos en integración continua que faciliten y aceleren el proceso.

Actualmente hay cerca de 80 desarrolladores utilizando Tuleap y las integraciones se hacen diariamente, para ellos el más grande reto fue que los desarrolladores escribieran pruebas unitarias.

4.2.6 Tecoloco con Roberto Yudice, Arquitecto de Software.

Trabaja para SaonGroup, que es una empresa que se dedica a tener bolsas de trabajo similares a LinkedIn, solo que no son redes sociales. SaonGroup, para El Salvador bajo Tecoloco, a parte se tienen sitios en Irlanda, Reino Unido y en Luxemburgo.

Los roles desempeñados a la fecha son de, principalmente Arquitecto de Software y parcialmente de Gerente de Desarrollo, liderando un equipo de 35 personas, con un perfil sobre todo técnico, a cargo de asegurar la visión en cuanto al área técnica de dónde quiere estar la empresa, con una experiencia de un año en IC y posteriormente experiencia en Entrega Continua EC.

Lo que motivó a la empresa a implementar IC fue sobre todo la eficiencia, en la cual se empezó con el servidor “Team Foundation” y luego implementaron Bamboo. IC es muy útil e indispensable cuando se tienen varios programadores haciendo cambios continuos a la misma base de código porque por ejemplo, se usa para el flujo de Git, GitFlow, entonces se tiene una rama “master” que es para producción, y como hay varios equipos trabajando en un mismo sitio, entonces a veces, un equipo hace un “release”, entonces cuando éste está listo le hacen un “merge” al “master”, lo que realiza un “push”

automático de “master” a todos las otras ramas de ese sitio por lo que se asegura que todos los equipos cuenten con los mismo cambios.

La empresa implementa IC desde hace dos años, y los desafíos encontrados al momento de la implementación fueron, primeramente la cultura, porque pasa que cada desarrollador tiene control total y pueden incluso dar “commit” a su código que ni siquiera compila, y de repente, con IC tiene que esperar a que el servidor de construcción ejecute las pruebas unitarias, compile el código.

En la parte del financiamiento no hubo problema; lo que sí fue un desafío es lo referido a los servidores de construcción, ya que estos se saturaban, lo que ocasionaba que los equipos tuvieran que esperar a éste respondiera. Para resolver lo anterior, entrando más en la parte de DevOps, se empezó a migrar a Amazon Web Service (AWS), y con ello se empezó a utilizar Terraform con lo que toda infraestructura se puede programar en código, entonces se realizaba un “script” de Terraform para construir “build servers”, es decir, cuando se saturan dichos servidores, ejecutan el “script” de Terraform y con eso automáticamente se tiene otro “build server”.

Todas las herramientas son Atlassian, entonces todas están interconectadas, de hecho, la herramienta Jira que es lo que utilizan para conectar con Bamboo, que por ejemplo, este último detecta el problema o “issue” de Jira que por política interna se ha dejado en el “commit” del ticket, y se le avisa a Jira que se hizo un cambio con dicho ticket, y automáticamente se hace una transición en el “issue” de Jira hacia otro estado, entonces si el compilado es exitoso, se pasa a otro estado, por lo que el desarrollador no tiene que hacer acción alguna, es decir, no tiene que ir a Jira a actualizar el “issue”, sino que solo hace un “commit” y se realiza todo.

La maduración del proceso de IC tomó aproximadamente un año, es decir solo configurar el servidor de construcción tomó alrededor de un mes, pero la implementación como tal, fue lo que llevó mayor tiempo.

Los ambientes con los que se cuentan son de desarrollo, “Unit Integration Testing” (UIT) y producción. Dentro de las actividades relacionadas con IC, lo único que se tiene manual es el despliegue a producción, los despliegues como a los ambientes de desarrollo son automáticos. Lo anterior es porque las pruebas unitarias no llegan a cubrir todo los escenarios posibles que garanticen que no ocurrirá falla alguna en ambiente productivo, por lo que debe de haber una intervención manual que sea responsable y garante de dicho proceso. La integración de código y la ejecución de las tareas de IC se realizan en cada “commit” que realizan los desarrolladores.

También, los lineamientos sobre el uso de IC fueron mejorando conforme el tiempo, en lo que se observaban necesidades o se solventaban inconvenientes, como el hecho de colocar el “issue” de Jira en cada “commit”.

Para la implementación, únicamente se contó con el conocimiento del personal interno, tomando como fuentes de información libros o buscadores web.

Hablando sobre el flujo, primero se crean “issue” en Jira, porque sin uno, no se pueden hacer “commit” ya que el controlador de versiones Stash lo rechaza. Luego este “issue” lo toma el desarrollador, y cuando ya está terminado hace un “commit” con una descripción que incluya dicho “issue”, así cuando se haga “push” al repositorio remoto se dispara la ejecución de Bamboo que automáticamente realiza la construcción, que compila el código, ejecuta las pruebas unitarias y las pruebas de código estático con SonarQube, y si todo sale bien, se realiza de manera automática el despliegue.

Cabe mencionar que antes de pasar los cambios a producción se pasan las pruebas manuales de QA, luego, ya que se cuenta con balanceo entre dos nodos, primero se coloca un cambio en un nodo que no está expuesto al público, entonces en eso nodo es que QA pasa las pruebas funcionales, y si todo resulta satisfactorio, una persona operativa se encarga de aprobar el cambio al nodo expuesto.

Para el "build server" se utiliza Bamboo de Atlassian, el control de versiones es Bitbucket, la administración de los "issues" en Jira, para wikis se tiene Confluence, para métricas SonarQube, para cubrir las pruebas unitarias se usa DotCover y como lenguaje de programación se utiliza .NET.

Para las fuentes de información o guías de IC, de manera documental para las herramientas si había información, pero para saber cuáles herramientas y buenas prácticas de IC seguir, no se contaba, por lo que todo fue hecho en base a experiencia. Una de las buenas prácticas que se implementó mediante la experiencia es el uso de GitFlow.

Tampoco se cuenta con información documental del proceso de implementación de IC debido a que al inicio el equipo era de alrededor de 7 personas, y era todo como bien informal, con ideas suscitadas de manera espontánea para implementarse en producción. Tampoco se contaba con apoyo de expertos u otras empresas para la implementación.

Para la complejidad de los proyectos, cada equipo cuenta con 7 u 8 personas, entre ellos 4 o 5 desarrolladores, 1 o 2 de QA y otro de operaciones, tampoco hay ningún proyecto que incluya más de eso. Los proyectos están expuestos al público, con una concurrencia de 100 peticiones por segundo.

Las herramientas recomendables serían Jenkins sobre Bamboo, debido al licenciamiento. Otro que no es muy recomendable es Jira para lo cual se está evaluando

Pivotal Tracker, esto debido a que en Jira es demasiado complicado, pese a que se ha ido simplificado con el tiempo, es decir, se tienen muchas características las cuales en lugar de sacarles todo el provecho, tienden a confundir y reducir la claridad en el uso.

4.2.7 DAI con Victor Salazar, Especialista en Tecnología.

DAI es una empresa de desarrollo de soluciones en el ámbito financiero y fiscal. Se encuentran en El Salvador desde 2017 con el apoyo de USAID. Cuenta con un equipo de tecnología de 35 personas entre 20 y 45 años, distribuidos en equipos siguiendo la metodología PMBOK y se encuentran evaluando integrarla con "Scrum". Las soluciones se están desarrollando en Java y Oracle 12c.

Durante 2 semanas configuraron e implementaron el entorno de IC con el apoyo del Ministerio de Hacienda, quienes ya habían implementado IC.

El entorno de IC se encuentra en un solo servidor SUSE, el cual se respalda diariamente. Utilizan Jenkins como orquestador, SonarQ para análisis de código estático

La infraestructura que utilizan es Red Hat y Jboss EAP 6. En Jenkins tienen una tarea para desarrollo, pruebas y producción que corresponde a cada una de las ramas configuradas. Adicional como sugerencia del Ministerio de Hacienda tienen dos tareas que se ejecutan semanalmente. Una para la rama de desarrollo y otra para la rama de pruebas, donde se obtiene todo lo que se encuentra en la rama para automáticamente ejecutar SonarQ e identificar los "bugs" y vulnerabilidades a corregir. Para construir el software utilizan Maven 3. Se seleccionaron las herramientas por las recomendaciones de Ministerio de Hacienda. Gradle es más rápido para hacer el build que Maven, pero se seleccionó Maven por que ya habían utilizado esta herramienta. Falta incluir las pruebas unitarias, por conexión en la red.

4.3 Entrevistas con expertos

4.3.1 Juliana Ceballos, Innovation Expert

Es un consultor experto en arquitecturas empresariales, desarrolladora de software y experta en prácticas de innovación para transformar las compañías. Ha desempeñado roles como director, manager y líder de proyectos de innovación.

Con una experiencia en Integración Continua de más de 10 años, de los cuales algunos de ellos fueron durante el tiempo que desempeño el rol de desarrolladora de software, pasando por líder de proyecto y terminando esos años con como uno de los principales impulsores e implementadores de proyectos de integración continua.

Ha trabajado en al menos dos grandes proyectos relacionados con compañías internacionales, ayudándoles a crear el gobierno de las aplicaciones, las buenas prácticas y toda la evangelización de Integración Continua e inmersión de los empleados en dicho proceso.

Por otro lado, asevera que en todos los proyectos de implementación una de las principales razones que motiva a las empresas a implementar IC es porque están buscando una forma de mejorar la calidad del código, la eficiencia, el manejo y versionamiento de código, las salidas a ambientes productivos y la rápida detección de errores.

Además agrega que al momento de implementar un proceso de IC, primeramente se debe definir el alcance del proyecto, que nos ayude a visualizar las fronteras de nuestro proyecto, identificar qué aplicaciones migrar y porque y por último, pero no menos importante basado en los lenguajes que la empresa tenga al momento de implementar se debe seleccionar la suite de IC idónea que cumplan con el alcance y las aplicaciones seleccionadas. De acuerdo a ella, se deben seleccionar herramientas que soporten

diferentes plataformas, por ejemplo Jenkins como servidor de integración es multiplataforma así que nos permite montarlo en entornos Linux y Window y además está basado en plugins para cada plataforma.

Al hablar de desafíos al momento de implementar IC asegura que es un proceso duro y largo, no de lado de la implementación de las tecnologías de IC en sí misma, sino más bien el trabajo que implica definir o redefinir procesos y manejar la resistencia al cambio del lado de los desarrolladores, dado que estos piensan que están bajo control. Para ella hay tres tipos de desafíos, culturales, tecnológicos y técnicos. Además comparte que los directores o managers de los equipos de desarrollo deben ser los principales impulsores del proceso.

Por último, nos enseña una serie de buenas prácticas en un proceso de implementación de IC. Como ya mencionamos al inicio, es el hecho de definir el alcance y que aplicaciones serán parte del proceso, además de definir estándares de desarrollo, y las herramientas ajustadas a estos estándares, definir estrategias de ramas, realizar pruebas con soluciones antes de completar el proceso global de IC así como unir y comprometer a los equipos de infraestructura y desarrollo.

4.3.2 Martin Salías, Kleer Coach & Trainer

Martin es un experto en diseño y desarrollo de software con más de 30 años de experiencia en la industria del software. Su especialidad está en la arquitectura de software, metodologías ágiles, flujos de trabajos con un enfoque orientado al liderazgo de equipos y automatización de procesos.

Actualmente es un entrenador y consultor en Kleer, la cual es una compañía especializada en entrenamientos y capacitación de metodologías ágiles, además es la primera compañía latinoamericana en ser reconocida y formar parte de "Scrum" Alliance.

Dentro del proceso de implementación de IC, Martin ha desempeñado diferentes roles, pasando por el rol de desarrollador, arquitecto de software y consultor en metodologías ágiles e implementador. En todo este tiempo aprendió que es mucho más importante entender los procesos que el uso de las herramientas en sí misma. Para Martín las empresas están enfocando mucho el esfuerzo en las tecnologías y no están alineando a sus empleados para que abracen los cambios y los conviertan en sus nuevos hábitos y flujos de trabajo.

Recomienda que un proceso de integración continua debe ser un proceso evolutivo e incremental lo que significa que el proceso se vaya introduciendo poco a poco, paso a paso. Para él, es más importante ir incorporando las prácticas de IC a medida que avanza el tiempo porque las plataformas grandes traen mucha configuración que al final no se utilizara, pero las personas en ese momento están buscando que las herramientas hagan las cosas por sí solas olvidando que son las personas las primeras que deben cambiar su mentalidad, ser, equipos muy comunicados, colaborativos, que hagan confirmaciones frecuentes y que tienen una forma estándar de trabajo.

Obviamente que las herramientas son parte del entorno de IC y en base a esto también explica que lo mejor sería implementar herramientas independientes o que cumplan con alguna de las actividades de IC y no hacer uso de herramientas o "frameworks" completos que al final se subutilizaran y por tanto no estará optimizado para las necesidades de una empresa que está adaptándose a un proceso nuevo y complejo. Recomienda Jenkins como

motor de integración, Cucumber para pruebas de aceptación y para pruebas de carga y de performance JMeter.

En cuanto al tamaño de los equipos cree que lo importante es que estén bien comprometidos y que la información sea efectiva entre ellos. Utilizando estándares de desarrollo.

4.3.3 Otto Girón, Arquitecto de Software en Xoom Guatemala

Xoom es una empresa miembro de la familia PayPal, ofrece maneras rápidas, fáciles y seguras de enviar dinero, recargar teléfonos y pagar facturas para la familia y amigos alrededor del mundo.

Otto Girón es un experto en desarrollo de software y arquitecturas empresariales, tiene una amplia experiencia como desarrollador de software, líder de equipo, arquitecto de software en grandes compañías como Xoom y PayPal.

Fue uno de los impulsores y ejecutores de IC en Xoom. Otto nos comparte todo el proceso inicial que les requirió como por ejemplo definir las tecnologías, lograr que las personas adoptaran IC y el manejo de la resistencia al cambio. A medida avanzaba el tiempo fueron agregando mejoras buscando optimizar el proceso.

Inicialmente lo que deseaban era que los desarrolladores pudieran abrazar el proceso y hacerlo su día a día, pero eso no fue una tarea fácil dado que nadie quería hacer cosas nuevas basada en estándares exigidas por la industrias como por ejemplo las pruebas unitarias, hacer planes de pruebas, automatizarlas o uno de los desafíos que tuvieron que vencer hacer confirmaciones de código, hacer uniones ramas y correr el servidor de construcción que en su caso era Jenkins.

Actualmente en Xoom tiene implementado todas las actividades de Integración Continua pero además están utilizando otras que forman parte de a lo que los expertos denominan Entrega Continua, la cual no es parte del alcance de esta síntesis. Al hablar de las tecnologías que implementaron podemos mencionar Jenkins, SonarQube, Bitbucket y Docker. Decidieron usar Makefile como herramienta de automatización de tareas como, compilar, ejecución de pruebas unitarias y reléase de las aplicaciones.

De acuerdo a Otto, en Xoom utilizan metodologías agiles, específicamente "Scrum". Las integraciones se hacen dependiendo la duración del "sprint", pero para ellos no representa una política, pero generalmente lo hacen basado en partes funcionales.

Otto nos comparte como al inicio del proceso de implementación encontrar documentación no fue una tarea fácil. La documentación existente era escasa y la existente no estaba detallada y en muchos casos desactualizada, argumenta Otto.

Por último, nos comparte una serie de buenas prácticas, para él es importante que los desarrolladores tengan cierta libertad para elegir librerías y la lógica de la aplicación, pero enfatiza en la necesidad de definir los estándares que se deben usar independiente de la tecnología que se esté utilizando. También para él es importante definir estándares de infraestructura o lo que también se denomina infraestructura como código, por ejemplo utilizar herramientas que nos ayuden a configurar infraestructura de forma declarativa para definir configuración. Menciona herramientas como Terrafont, Puppet.

4.3.4 Hernán Arteaga, Arquitecto de Software en Telus International

Con experiencia de un año y medio, dentro de la implementación y uso de integración continua, tratando de mejorar el proceso de esta práctica. Se tiene una participación en

dos implementaciones, tanto como desarrollador y arquitecto, definiendo ambientes, “pipeline”, tecnologías y siendo parte del uso de IC.

Hay varios puntos por los que se ha considerado la implementación de IC, algunos buenos y malos. Primeramente, lo que motivó el uso, es tratar de mejorar la productividad en el desarrollo, de hacer el tema de “merge” que se puedan descargar del repositorio, que sea mucho más ágil, el poder encontrar errores en el transcurso de lo que se va desarrollando, pues una de las etapas de IC son las pruebas unitarias, a través de ella se puede evaluar el código y sirven para la evaluación de pruebas de regresión, para que al ejecutar un cambio, se constate que nada se ha arruinado. Otro, el famoso tiempo de salida al mercado, el poder entregar lo más rápido los productos al cliente. Dentro de los puntos malos en los cuales se ha implementado IC se debe al uso de esta práctica por moda, y sin tener claros los objetivos que realmente se quieren alcanzar.

Durante el proceso de implementación IC, se han identificado las fases siguientes: primero el tema de la cultura, ya que, sin ella, aunque se tengan las mejores tecnologías no va a servir de nada, por lo que se debe indagar y preparar al equipo sobre todo lo que implica dicha práctica, como marcos de trabajo ágiles como “Kanban” o “Scrum”, y que se convenga.

Luego de lo anterior, siempre dentro de los pasos a seguir, se deben definir estrategias de ramas, como manejo de nuevos cambios, actualizaciones con la “master”, ambientes de desarrollo, etc. A parte, la estrategia de cómo hacer el “pipeline”, si se va a quedar en pruebas o producción, o se va a subir únicamente al repositorio de artefactos. Luego de eso se puede definir qué aplicaciones se van a incluir en este camino, por ejemplo, las que sufren cambios con mayor frecuencia, que son las que más se están haciendo despliegues

a producción, ello para ver el valor de esta práctica. Lo más difícil de implementarlo, es llevar todo este camino de estrategias.

Sobre las herramientas de IC, se han utilizado primeramente como servidor de integración, Jenkins, sobre todo por la facilidad de uso, compatibilidad de otras herramientas como Maven y por ser código abierto. Para el administrador de repositorios se recomienda GitLab porque es código abierto, se puede tener en una infraestructura propia, ya sea en la nube o en sitio, lo que sobre pasa otras herramientas como Github o Bitbucket, y de igual manera por la compatibilidad con Git.

Para la herramienta de análisis de código estático, es muy recomendable SonarQube porque tiene una gran cantidad de reglas, que a veces se pasan desapercibidas, el tema de catálogos es muy bueno. Se pueden escoger reglas que tengan sentido para el negocio, que puedan tener un gran impacto, para realizar un análisis más certero y coherente a las metas, por ejemplo, se puede definir una regla para la aceptación porcentual de vulnerabilidades, que puede variar según el tipo de proyecto y objetivo a lograr. Como repositorio de artefactos es muy recomendable Nexus, sobre otros como JFrog Artifacory, primero porque su versión de código abierto, y provee construcciones con Greadle o Maven, y en pueden ser en Docker.

Dentro de los desafíos más encontrados en implementación es la realización de pruebas unitarios, incluyendo la capacitación de los desarrolladores, hablando de TDD o el tema de la frecuencia de los “commit”. También es un problema el inculcar a los desarrolladores el cumplimento de reglas como las que conlleva el uso de análisis estático. El tema cultural y conocimiento de todo el equipo de desarrollo es importante, el hecho de cambiar las diversas formas de hacer las cosas para integrarse con IC sin confusiones.

Dentro de la experiencia, en aproximadamente un año con aplicaciones web, se han realizado equipos de trabajo de 5 personas, con cantidad de usuarios por aplicación de mil personas, por lo que pueden ser consideradas como grandes.

Al hablar del proceso de IC, primero el desarrollado realiza un "push", se ejecuta, por ejemplo, cada día en la noche, la verificación de nuevos cambios y con ello la ejecución de las tareas de Jenkins, luego se realizan las pruebas unitarias, luego se pasan las pruebas de código estático en las cuales se evalúan las métricas definidas con anterioridad. Si cumple todo lo anterior, se puede subir al repositorio de artefactos. Se puede tener un "pipeline" por aplicación

En resumen, lo recomendable para una implementación efectiva, lo primero es trabajar el tema cultural, y dado que es un tema bastante grande, se puede segmentar y digerirlo poco a poco. También se debe preparar al equipo en temas como el agilísimo, luego de eso se puede empezar a definir estrategias y tecnologías. También es importante incluir a la parte operativa para que entiendan el trabajo de los equipos de desarrollo y también poder contar con el apoyo futuro de ellos.

4.3.5 Pablo Portillo, Arquitecto de Software en Telus International

Cuenta con una experiencia alrededor de 6 años en IC, en parte siendo consumidor de procesos de IC, es decir, enfocada en su uso, y en otros momentos de implementando "front scratch", entrega continua, DevOps y algunos tipos de herramientas que se encuentran en ese tipo de sistemas. Cuenta con dos implementaciones de IC, y diversos temas relacionados con esta práctica. Los roles desempeñados han sido de desarrollador, también, como al inicio de una implementación no existen roles como tal, entonces siendo

"Scrum Master", se ha tenido participación de otros roles para una correcta infraestructura de DevOps, lo cual con llevó a ejercer un rol de arquitecto.

En una de las implementaciones, se estableció y formo todo un equipo para el área de DevOps, con arquitectos de DevOps, que definían cómo se iba a integrar la IC y la Entrega Continua EC al principio de los proyectos, inclusive, se definía si el proyecto aplicaba a ese tipo de prácticas, porque por ejemplo, si uno de los proyectos no iba a cambiar en el tiempo, entonces no era muy conveniente o necesario realizar el esfuerzo de incluirlo en IC o EC, porque a lo mejor solo se van a hacer cambios cada año o cada 6 meses. Eran necesarios 4 o 5 encargados de disponer ese tipo de tecnología, de estar monitoreando y darle mantenimiento al servidor Jenkins, entre ellos el arquitecto de DevOps y el ingeniero de DevOps.

El objetivo que las empresas persiguen al utilizar IC es poder garantizar la calidad del software en desarrollos colaborativos en cada "sprint" o iteración, utilizando de manera evidente estrategias de ramas o de almacenado de código, pero al final lo que se busca con IC es que el código pueda compilarse y esté listo como para poder convertirse en un "release" en algunos casos.

La otra razón de gran relevancia de la implementación y uso de IC que de igual forma va ligado a EC es la reducción del tiempo de salida al mercado, es decir dar la solución al cliente en el menor tiempo posible.

Hablando sobre las fase de la IC, dado que en una empresa, siempre se quieren ver resultados lo más pronto posible, primero se deben seleccionar las aplicaciones que más necesitan de IC, para iniciar es muy conveniente apoyase de una empresa que ya maneje herramientas o que tenga experiencia en este tema, como por ejemplo RedHat, todo ello

abonando para poder entregar valor de forma rápida como primera fase, evitando que los desarrolladores estuvieran compilando de manera manual el código o haciendo “merges” manuales. Con lo anterior se le da sentido y orientación al uso de herramientas como Jenkins, SonarQube o Nexus.

Para una Fase 2, darle es necesario moverse a herramientas que preparen infraestructura como Chef o Ansible, y moverse hacia otro tipo de tecnología como Docker o Kubernetes, y talvez tratar de resolver problemas de consistencia, como en la parte de homologación de ambientes.

La selección de herramientas depende mucho de la realidad de la empresa, porque no hay una fórmula mágica que solvente todo, todo de ir basado en el negocio, en una pueden ser adaptadas a Java o a .Net, otras en PHP, pero uno de los servidores más estándar es Jenkins, ya que tiene mucha compatibilidad.

Para el manejo de versiones, se tiene la experiencia con SVN y Git, con Tortoise, utilizando GitLab, GitHub, etc. Dentro de todo ello es muy bueno el uso de Git, ya que el manejo de ramas es muy bueno, de fácil uso y maduro, y permite que el desarrollo colaborativo sea más sencillo. Hay que hacer mención que GitLab como tal puede ser utilizado para IC pero en proyectos pequeños.

Para el análisis de código estático, es recomendable el uso de SonarQube ya que permite poder manejar o definir distintos tipos de reglas para diferentes lenguajes de programación; además de ellos definir niveles de validación de código. Esta herramienta, al estar en IC, es capaz de manejar, en la parte de arquitectura de aplicaciones, diversos niveles de tolerancia según cada aplicación, lo que quiere decir que, según lo que se defina

en SonarQube, otras herramientas actuaran de tal forma que avalaran o desaprobaran las construcciones de los artefactos de manera automática.

Entre las métricas se debe mencionar que es recomendable comenzar con el rendimiento, siguiendo luego con las vulnerabilidades, luego de ello se podría ir según las demás necesidades de la empresa.

Para la parte del repositorio de artefactos, se tiene experiencia con Nexus, Artifactory y JFrog, para lo cual este último sobresale por su interfaz amigable y también que aparte de tener la opción de código abierto, también se puede descargar la versión empresarial que cuenta con soporte, lo cual es muy importante. Se puede empezar con una versión gratuita y después se puede actualizar a una versión con soporte.

También se menciona la vanguardia sobre el uso de registros de contenedores, los cuales son utilizados para la carga de imágenes de por ejemplo Docker.

Dentro de los desafíos se tienen, como primer punto la cultura seguido del agilísimo. Ambos van apegados que a que el cliente esté inmerso en todo el proceso de desarrollo. De igual forma el “Status Quo”, para lo cual a veces no se saben vender los beneficios de las prácticas de IC.

Los proyectos que se han trabajado incluyen rutas de buses de servicio, aplicaciones tanto para usuarios internos como externos, dirigidos a grandes cantidades de personas, Business Rules Management System (BRMS), JBOSS Fuse, aplicaciones de integración, entre otras.

Se pueden invocar tareas de herramientas de “Infrastructure-as-Code “o (IaC) como Terraform o Ansible, para que, por ejemplo, se verifiquen los paquetes instalados y

para la ejecución de los artefactos previamente cargados desde el repositorio, en la arquitectura orientada a servicios definida.

En resumen, en conveniente enfocarse, no tanto en las tecnologías, sino más bien en entender las cualidades y experiencia de los desarrolladores, ya que si se cuenta con personal con experticia en estos temas. También se deben definir estrategias de ramas, como, por ejemplo, puede haber una rama por desarrollador por sprint que son los "features-branches", luego se puede hacer el "merge" del código. Después tomar el tiempo para colocar las reglas. No realizar un pipeline genérico, es mejor enfocarse en casos particulares según los objetivos.

4.4 Hallazgos

Se describen todos los hallazgos encontrados durante la investigación realizada en las empresas del Área Metropolitana de San Salvador (AMSS), así como también a expertos nacionales e internacionales.

El total de empresas investigadas fue de siete, lo cual representa la muestra de la investigación, los porcentajes mostrados en los hallazgos han sido obtenidos basado en dichos datos recopilados.

4.4.1 Motivaciones

4.4.1.1 Lo que más motivó a todas las empresas a implementar IC fueron sobre todo los beneficios de la detección temprana de errores y la reducción en los tiempos de entrega, ya que por la forma que venían trabajando, no realizaban integraciones frecuentes de código, lo que les generaba conflictos y salidas difíciles a producción porque acumulaban requerimientos, mostrando a la vez errores en los aplicativos.

4.4.2 Tecnología y metodologías de desarrollo

4.4.2.1 En la investigación se pudo evidenciar que las metodologías ágiles son las más compatibles para trabajar en un entorno de IC, esto fue expresado tanto por las empresas como por expertos. El 72% de las empresas investigadas utilizaban, tanto antes como después de implementar IC, “Scrum” como marco de trabajo; mientras que el restante estaba utilizando metodologías tradicionales, en específico la metodología en cascada, pero después de implementar IC, la mitad de éstas adoptaron una metodología ágil, y las demás se encuentran en proceso de implementación.

4.4.2.2 El 86% de las empresas consultadas tienen preferencia por el uso de herramientas de bajo costo de implementación y mantenimiento, y que sean multiplataforma, principalmente en servidor de integración, repositorio de versionamiento y plataforma de análisis estático. El resto de los casos, decidió utilizar una herramienta que incluyera el ciclo completo de IC, a la cual tenían acceso según el licenciamiento, dado que este se alineaba de manera adecuada a los objetivos del negocio. Es decir dentro de la implementación de IC, el uso de los recursos tecnológico depende de los objetivos del negocio y del acceso que se posea.

4.4.2.3 Se identificaron dos entornos de desarrollo predominantes y que marcan tendencia en las empresas del AMSS. Uno de ellos es el desarrollo de aplicaciones web basadas en .NET C#, en este aspecto el 40% de la muestra expresaron que el desarrollo web basado en .NET es el que utilizaban antes y después de la implementación de IC. El restante se basa en Java para sus desarrollos. Es

importante resaltar que las empresas que actualmente están utilizando .NET expresaron que están evaluando o se encuentran migrando a un entorno Java.

4.4.2.4 Según el estudio, el total de las empresas aseguran que, si bien es cierto, tuvieron que integrar al proceso de IC herramientas propias de esta práctica, eso no significó un cambio radical en las tecnologías que utilizaban, por ejemplo, los lenguajes de programación siguen siendo los mismos, de igual manera con el servidor de despliegues. Contrario a lo que se pudiera llegar a pensar, de que uno de los requerimientos debería de ser actualizar su pila de tecnologías, más bien, las empresas y expertos destacan que la implementación efectiva de IC es sobre todo un cambio cultural.

4.4.3 Desafíos

4.4.3.1 Tanto empresas como expertos coinciden que el desafío más importante es el convertir IC en una cultura, dado que las personas están acostumbradas a un flujo con prácticas de desarrollo que han venido haciendo durante años, donde se sienten cómodos, lo que puede generar una resistencia al cambio. La adopción de IC implica definir o redefinir procesos y gestionar el cambio a nivel organizacional, especialmente del lado de los desarrolladores.

4.4.3.2 Durante la implementación se encuentra el desafío de la falta de capacitación o el desconocimiento generalizado de IC, que implica dedicar horas de investigación y solución de problemas. Para superarlo, una de las recomendaciones que comparten, tanto expertos como empresa, es apoyarse o referenciarse con consultores expertos en IC.

4.4.3.3 El 43% de las empresas tuvo como desafío técnico, la construcción de pruebas unitarias porque los desarrolladores no estaban realizándolas antes de la implementación.

4.4.4 Implementación

4.4.4.1 Se identificó que la mitad los casos en estudio se apoyó con un consultor o empresa consultora para poder realizar la implementación de IC, y la mitad restante hizo uso de otros recursos como documentación oficial, foros y sitios web. El apoyarse de un consultor experto en IC no solo les facilitó el proceso de implementación, sino que también lo aceleró, argumentan las empresas consultadas. Las empresas que no lo hicieron tuvieron que invertir más tiempo y una curva de aprendizaje mayor.

4.4.4.2 Todos los casos estudiados reflejan que no existe un proceso totalmente automatizado, desde la construcción hasta el despliegue en ambiente productivo, más bien, siempre hay una intervención manual que asegura, responsabiliza y avala los nuevos cambios cargados en los repositorios de desarrollo y pruebas, ya que según las empresas, se dificulta y se generan más costos en aspectos como el tiempo y dinero, al intentar cubrir todos los escenarios funcionales y técnicos posibles y probables.

4.4.5 Buenas prácticas

4.4.5.1 Dentro del estudio, los expertos mencionan que al iniciar la implementación se deben definir de manera priorizada las reglas, métricas y aspectos a medir según

el objetivo del negocio, de lo contrario el resultado de la evaluación podría quedar deficiente o sobreestimado. Claro ejemplo es el uso de Selenium o SonarQube, para los cuales, se deben seleccionar reglas mayormente alineadas a los objetivos de la empresa, para que la evaluación presente más efectividad al momento de medir la calidad del código y funcionamiento de la aplicación, lo que dará como resultado que se corrijan las vulnerabilidades o fallos de manera priorizada y eficaz.

4.4.5.2 Por lo general las empresas que no utilizan IC tienden a realizar sus integraciones quincenal o mensualmente, algunas veces dándole libertad al desarrollador para que decida por sí mismo cuándo integrar el código. El estudio muestra cómo las empresas aumentaron la frecuencia de integración del código. El 86% de los casos consultados, antes de implementar IC integraban el código mensualmente o a juicio del desarrollador, del restante, la mitad se basaba en “Sprints” de dos semanas. Después de implementar IC todas las empresas en estudio, comenzaron hacer integraciones más frecuentes, por lo general diarias.

4.4.5.3 Los expertos también mencionan que para la implementación efectiva de IC, se debe tener en consideración las características del equipo de desarrollo que estará inmerso en el nuevo proceso. Las herramientas y metodologías deben ser seleccionadas de tal manera que se acoplen a su experiencia para un mayor aprendizaje y una mejor adopción, siempre teniendo en cuenta los objetivos del negocio.

Capítulo V: Conclusiones

Dentro de los desafíos se logra concluir, que el de mayor relevancia es la adopción de las prácticas de IC dentro de la cultura organizacional, por lo que es importante dar mayor énfasis a las personas y a la reingeniería de procesos desde el inicio de la implementación.

La selección de las herramientas también conlleva un alto impacto en la implementación IC, por lo que éstas deben fáciles y sencillas utilizar, que soporten múltiples plataformas, con una alta compatibilidad, sin que provoquen ataduras tecnológicas considerables, ya que el uso de herramientas complejas y poco flexibles, podrían hacer confusión en el proceso de implementación al contar con opciones subutilizadas según las metas de las empresas. En sí, las herramientas y el entorno de la infraestructura giran en torno a lo que se quiere lograr, así como los recursos económicos con lo que se cuente, pero sobre todo, de cómo esto se adapte de mejor manera al cambio en la cultura y a los objetivos del negocio.

Se debe hacer énfasis en que tanto la guías de implementación, como la de buenas prácticas, realizadas y descritas en este documento, las cuales están basadas en la experiencia de empresas del AMSS y expertos en IC, son un insumo para que otras organizaciones lo tomen como un punto de partida en la construcción de un entorno de desarrollo ágil de aplicaciones web utilizando IC, para con ello aprovechar los beneficios que proporciona esta práctica**.**

Como conclusión final, se determinó que la IC no es de uso exclusivo de las empresas como Telus o Tecoloco, más bien, la implementación efectiva dentro del contexto salvadoreño depende de los objetivos del negocio, seleccionando las

herramientas que mejor se acoplen, y de la adopción de las prácticas ágiles de IC, de esta forma, el entorno de IC se encontrará en continua evolución y en sintonía con los objetivos de la empresa.

Capítulo VI: Anexos

6.1 Instrumentos

6.1.1 Guía de Entrevista

Objetivo: Recopilar información de la experiencia de la implementación de IC.

1 ¿Qué metodología utilizan en sus desarrollos?

2 ¿Qué les motivó a implementar IC?

3 ¿En qué año realizaron la implementación de IC?

4 ¿Qué tanto apoyo se recibió de alta gerencia durante todo el proceso de Implementación?

5 ¿Cuáles fueron sus fuentes de información en las que se basaron para la implementación del entorno IC?

6 ¿Qué tan accesible fue la información que utilizaron?

¿En qué lenguajes de programación desarrollaban antes de implementar IC?

7 ¿Qué IDE de desarrollo utilizaban antes de implementar IC?

8 ¿Cuáles de las actividades de IC han implementado?

9 ¿Continúan desarrollando en los mismos lenguajes de programación?

10 ¿Con cuánta frecuencia realizan el proceso de integración con IC?

11 ¿Cuál herramienta para la construcción de compilados implementaron? ¿Por qué fue seleccionada?

12 ¿Cuál herramienta de control de versiones implementaron? ¿Por qué fue seleccionada?

13 ¿Cuál servidor de integración continua implementaron? ¿Por qué fue seleccionado?

14 ¿Cuál herramienta para análisis de código estático implementaron? ¿Por qué fue seleccionada?

15 ¿Cuáles considera que han sido los desafíos técnicos en la implementación del entorno de IC?

16 ¿Qué otros desafíos se les presentaron en la implementación de IC?

17 ¿Qué buenas prácticas implementaron al adoptar IC?

6.2 Tabulación de entrevistas con las empresas en estudio

Pregunta planteada:	**¿Qué metodología utilizaban en sus desarrollos?**	
Empresas \ Respuestas	**"Scrum"**	**Cascada**
KADEVJO	X	
AFP CRECER		X
TECOLOCO	X	
AEROLÍNEA	X	
INSTITUCIÓN GUBERNAMENTAL	X	
TELUS		X
DAI		X

Pregunta planteada:	**¿Qué les motivó a implementar IC?**				
	Reducción Errores / Fallas	**Reducción de Tiempo**	**Poca satisfacción del usuario final**	**Reducción de costos por cambios**	**Baja / Mejorar calidad del software**
KADEVJO		X			X
AFP CRECER	X	X	X	X	X
TECOLOCO	X	X			
AEROLINEA	X	X		X	X
INSTITUCIÓN GUBERNAMENTAL					X
TELUS		X			X
DAI					X

Pregunta planteada:	¿En qué año realizaron la implementación de IC?		
	Antes del 2015	2016	2018
KADEVJO	X		
AFP CRECER		X	
TECOLOCO		X	
AEROLINEA		X	
INSTITUCIÓN GUBERNAMENTAL	X		
TELUS	X		
DAI			X

Pregunta planteada:	¿Qué tanto apoyo se recibió de alta gerencia durante todo el proceso de Implementación?		
	0 - 25%	25 - 75%	75 - 100%
KADEVJO			X
AFP CRECER	X		
TECOLOCO			X
AEROLÍNEA			X
INSTITUCIÓN GUBERNAMENTAL		X	
TELUS			X
DAI			X

Pregunta planteada:	¿Cuáles fueron sus fuentes de información en las que se basaron para la implementación del entorno IC?			
	Consultor / Experto	Empresa Consultora	Documentación Oficial	Foros
KADEVJO			X	X
AFP CRECER	X	X		
TECOLOCO				
AEROLINEA		X		
INSTITUCIÓN GUBERNAMENTAL	X		X	
TELUS	X			
DAI		X		

Pregunta planteada:	**¿En qué lenguajes de programación desarrollaban antes de implementar IC?**		
	C# /.NET	**Java**	**PHP**
KADEVJO	X		X
AFP CRECER	X		
TECOLOCO	X	X	
AEROLINEA	X	X	
INSTITUCIÓN GUBERNAMENTAL		X	
TELUS		X	
DAI		X	

Pregunta planteada:	**¿Qué IDE de desarrollo utilizaban antes de implementar IC?**		
	Visual Studio	**Eclipse**	**Netbeans**
KADEVJO	X		
AFP CRECER	X		
TECOLOCO	X		
AEROLÍNEA	X	X	
INSTITUCIÓN GUBERNAMENTAL		X	X
TELUS	X	X	
DAI		X	

Pregunta planteada:	**¿Cuáles de las actividades de IC han implementado? ¿Cómo es el proceso de IC?**			
	Construcción	**Pruebas Unitarias**	**Análisis de Código Estático**	**Despliegue**
KADEVJO	X	X	X	X
AFP CRECER	X	X	X	
TECOLOCO	X	X	X	
AEROLINEA	X		X	
INSTITUCIÓN GUBERNAMENTAL	X		X	X
TELUS	X	X	X	X
DAI	X	X	X	

Pregunta planteada:	¿Continúan desarrollando en los mismos lenguajes de programación? En caso que no, en ¿En cuáles lenguajes de programación desarrollan utilizando IC?
	Si
KADEVJO	X
AFP CRECER	X
TECOLOCO	X
AEROLINEA	X
INSTITUCIÓN GUBERNAMENTAL	X
TELUS	X
DAI	X

Pregunta planteada:	¿Qué IDE están utilizando en la actualidad con IC?		
	Visual Studio	Eclipse	Netbeans
KADEVJO	X		
AFP CRECER	X		
TECOLOCO	X		
AEROLÍNEA	X	X	
INSTITUCIÓN GUBERNAMENTAL		X	X
TELUS	X	X	
DAI		X	

Pregunta planteada:	¿Con cuánta frecuencia realizan el proceso de integración con IC?			
	Diariamente	Cada 2 días	Por Sprint	Por Error
KADEVJO		X		
AFP CRECER	X			
TECOLOCO				X
AEROLINEA	X			
INSTITUCIÓN GUBERNAMENTAL			X	
TELUS			X	
DAI	X			

Pregunta planteada:	¿Cuál herramienta para la construcción de compilados implementaron?	
	Maven	Nuget
KADEVJO		X
AFP CRECER		X
TECOLOCO		X
AEROLINEA	X	X
INSTITUCIÓN GUBERNAMENTAL	X	
TELUS	X	
DAI	X	

Pregunta planteada:	¿Cuál herramienta de control de versiones implementaron?
	Git
KADEVJO	X
AFP CRECER	X
TECOLOCO	X
AEROLINEA	X
INSTITUCIÓN GUBERNAMENTAL	X
TELUS	X
DAI	X

Pregunta planteada:	¿Cuál servidor de integración continua implementaron?				
	Jenkins	Terrrafront	Bambu	Tuleap	Azure MobileCenter
KADEVJO	X				X
AFP CRECER	X				
TECOLOCO		X	X		
AEROLÍNEA	X				
INSTITUCIÓN GUBERNAMENTAL				X	
TELUS	X				
DAI	X				

Pregunta planteada:	**¿Cuál herramienta para análisis de código estático implementaron?**		
	SonarQube	**Gerrit**	**Azure MobileCenter**
KADEVJO			X
AFP CRECER	X		
TECOLOCO	X		
AEROLINEA	X		
INSTITUCIÓN GUBERNAMENTAL	X	X	
TELUS			
DAI	X		

Pregunta planteada:	**¿Por qué fueron seleccionadas?**				
	Open Source	**Documentación**	**Soporte**	**Fácil configuración**	**Multiplataforma**
KADEVJO		X	X	X	X
AFP CRECER	X		X		X
TECOLOCO	X				
AEROLINEA	X				
INSTITUCIÓN GUBERNAMENTAL	X	X			
TELUS	X				X
DAI	X	X			

Pregunta planteada:	**¿Cuáles considera que han sido los desafíos técnicos en la implementación del entorno de IC?**						
	Aplicaciones Heredadas	**Adopción de Estándares**	**Adopción TDD**	**Pruebas automatizadas**	**Desarrollo Ágil**	**Soporte múltiples lenguajes**	**Experiencia en El Salvador**
KADEVJO						X	X
AFP CRECER		X	X	X	X		
TECOLOCO							
AEROLÍNEA	X	X					
INSTITUCIÓN GUBERNAMENTAL			X	X	X		
TELUS							X
DAI				X			X

Pregunta planteada:	¿Qué otros desafíos se les presentaron en la implementación de IC?	
	Cambio de Paradigma / Resistencia al Cambio	**Cultural**
KADEVJO		X
AFP CRECER	X	X
TECOLOCO	X	
AEROLINA	X	
INSTITUCIÓN GUBERNAMENTAL	X	
TELUS		X
DAI	X	

Pregunta planteada:	**¿Qué buenas prácticas implementaron al adoptar IC?**					
	Estrategia de Ramas	**Manejo de errores**	**Orden**	**Integraciones Frecuentes**	**Pruebas en Pre-Producción**	**Adopción de estándares**
KADEVJO	X			X		
AFP CRECER			X	X	X	X
TECOLOCO	X	X		X		
AEROLÍNEA	X	X		X		
HACIENDA	X			X		X
TELUS	X			X		
DAI				X		X

6.3 Cartas de validación de expertos

6.3.1 Martin Salías, Kleer Coach& Trainer

San Salvador, Diciembre de 2018

Universidad Don Bosco

Respetados señores:

Yo, Martín Salías, conozco y doy por validada la Guía de Implementación de Integración Continua para los entornos de .Net y JAVA, que incluye la instalación y configuración de las herramientas; así como la Guía de Buenas Prácticas, que brinda apoyo para implementar y adoptar Integración Continua; realizadas por María Lorena De La Fuente Jacobo, William Antonio Flores Ayala y Roberto Bladimir Hernández Guevara, haciendo denotar el aporte que brinda para la comunidad de profesionales en el área de tecnología en El Salvador.

MARTIN SALIAS
SOCIO GERENTE
KLEER S.R.L.

6.3.2 Juliana Ceballos, Innovation Expert

JSan Salvador, Diciembre de

2018 Universidad Don Bosco Respetados señores:

Yo, conozco y doy por validada la Guía de Implementación de Integración Continua para los entornos de .Net y JAVA, que incluye la instalación y configuración de las herramientas; así como la Guía de Buenas Prácticas, que brinda apoyo para implementar y adoptar Integración Continua; realizadas por María Lorena De La Fuente Jacobo, William Antonio Flores Ayala y Roberto Bladimir Hernández Guevara, haciendo denotar el aporte que brinda para la comunidad de profesionales en el área de tecnología en El Salvador.

Juliana Ceballos De Ortiz

Gerente de innovación tecnológica

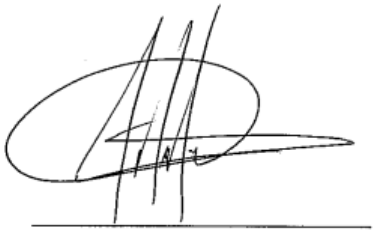
6.3.3 Hernán Arteaga, Arquitecto de Software en Telus International

San Salvador, Diciembre de 2018

Universidad Don Bosco

Respetados señores:

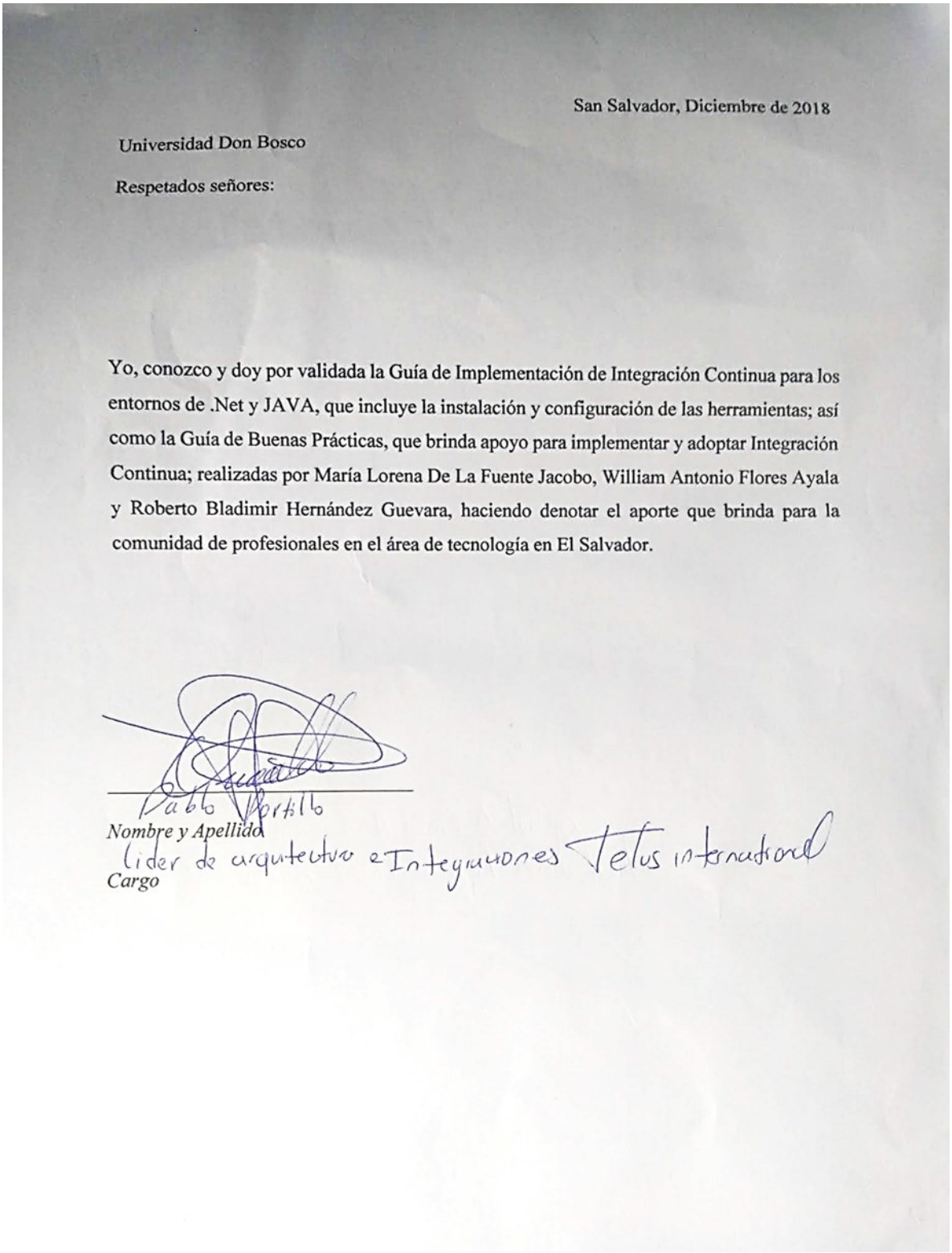
Yo, conozco y doy por validada la Guía de Implementación de Integración Continua para los entornos de .Net y JAVA, que incluye la instalación y configuración de las herramientas; así como la Guía de Buenas Prácticas, que brinda apoyo para implementar y adoptar Integración Continua; realizadas por María Lorena De La Fuente Jacobo, William Antonio Flores Ayala y Roberto Bladimir Hernández Guevara, haciendo denotar el aporte que brinda para la comunidad de profesionales en el área de tecnología en El Salvador.



Nombre y Apellido José Hernán Arteaga

Cargo Application Development Architect

6.3.4 Pablo Portillo, Arquitecto de Software en Telus International



San Salvador, Diciembre de 2018

Universidad Don Bosco

Respetados señores:

Yo, conozco y doy por validada la Guía de Implementación de Integración Continua para los entornos de .Net y JAVA, que incluye la instalación y configuración de las herramientas; así como la Guía de Buenas Prácticas, que brinda apoyo para implementar y adoptar Integración Continua; realizadas por María Lorena De La Fuente Jacobo, William Antonio Flores Ayala y Roberto Bladimir Hernández Guevara, haciendo denotar el aporte que brinda para la comunidad de profesionales en el área de tecnología en El Salvador.

Pablo Portillo
Nombre y Apellido
Lider de arquitectura e Integraciones Telus International
Cargo

6.4 Guía de buenas prácticas desarrollo de aplicaciones web utilizando integración continua

UNIVERSIDAD DON BOSCO

Vicerrectoría de Estudios de Postgrados

Maestría en Arquitectura de Software

Guía de buenas prácticas desarrollo de aplicaciones web utilizando integración continua

Presentado por:

María Lorena De La Fuente Jacobo

William Antonio Flores Ayala

Roberto Bladimir Hernández Guevara

Asesor:

Luis García

Antiguo Cuscatlán, La Libertad, El Salvador

Enero 2019

Introducción

Integración continua, conocida de acá en adelante como IC, ya no es un concepto nuevo en nuestros días, se ha vuelto cada vez un término mucho más familiar en los departamentos de desarrollo de software. De hecho, ya hay empresas en el Área Metropolitana de San Salvador que están utilizando esta práctica ágil reportando beneficios significativos.

La implementación del entorno de IC puede ser un proceso desafiante por varias razones. Como apoyo para afrontar los desafíos, se han recolectado buenas prácticas para la implementación de IC en base a la experiencia de las empresas del Área Metropolitana de San Salvador, que han implementado y han estado utilizándola por lo menos durante 6 meses en sus desarrollos de aplicaciones Web. Además de incorporar recomendaciones de expertos en el área de IC bajo entornos de desarrollo Web.

Esta guía pretende ser un insumo para las empresas interesadas en implementar un entorno de desarrollo ágil para aplicaciones Web utilizando IC.

Desarrollo Ágil

En un estudio realizado por la Consultora Forrester (2013), se identificó que hay empresas que intentan aumentar su velocidad de desarrollo implementando modelos ágiles de desarrollo, para responder oportunamente a la necesidad del negocio de innovación. Este es el comienzo para tener entregas tempranas y continuas, sin embargo no es suficiente si no se optimiza y mejora los procesos de integración, pruebas y despliegue. El equipo podría dedicar demasiado tiempo en el despliegue de la solución y en reparar defectos post implementación.

Integración y entrega continua están alineados a los principios del desarrollo ágil, cómo es el de la entrega continua, "*Entregamos software funcional frecuentemente, entre dos semanas y un mes, con preferencia por periodos de tiempo lo más corto posibles*".

Las empresas en estudio identificaron la relación de integración y entrega continua con el desarrollo ágil, por lo que quienes se encontraban utilizando modelos tradicionales, implementaron modelos ágiles para aprovechar los beneficios de la integración y entrega continua; siendo Scrum el modelo más utilizado. Este es un cambio organizacional, por lo que requiere el apoyo de la alta gerencia, así como la capacitación y definición de nuevos roles dentro de equipos multidisciplinarios.

Adopción de IC

Un cambio organizacional, ya sea que se trate de agregar un nuevo proceso o modificar uno existente, puede generar resistencia por lo que al iniciar un proceso de implementación de IC debe existir una correcta gestión del cambio.

La consultora KPMG define el objetivo de la gestión del cambio como "*buscar facilitar y conseguir la implementación exitosa de los procesos de transformación, lo que implica trabajar con y para las personas en la aceptación y asimilación de los cambios y en la reducción de la resistencia; facilitando la aceptación y asimilación de los cambios, producto de una nueva forma de operación*."

Se ha coincidido con las empresas y expertos que la gestión del cambio y la cultura ágil son claves para la adopción de IC. Se debe invertir más tiempo en conocer a los equipos, su experiencia y conocimiento, para poder hacerlos parte del nuevo proceso que en la selección y configuración de herramientas. Se busca crear una cultura organizacional de IC que abrace el cambio y la innovación.

Lianping Chen, que es el Jefe de Ingeniería de Software del Departamento de Sistemas de R&D y Calidad de Procesos en Huawei Technologies, expone las siguientes estrategias para superar los retos de adopción de integración y entrega continua:

Venderlo como un analgésico

Adoptar prácticas ágiles como integración y entrega continua requieren cambios en diferentes áreas, desde las actividades de arquitectura, prácticas de desarrollo de software, estructura organizativa y cultura. Para poder lograrlo se necesita el apoyo de los interesados, tanto de tecnología como del negocio.

La estrategia es identificar los dolores que tiene cada uno de los interesados y presentarle como y porque IC puede apoyar a solventar ese dolor. Una vez los interesados comprenden como IC les puede beneficiar, se vuelven promotores de la adopción. Los directores de tecnología deben ser los principales impulsores.

Establecer un equipo dedicado con miembros multidisciplinarios

El tener un equipo dedicado permite invertir el esfuerzo, enfoque y tiempo necesario para construir el entorno de IC y montar a cada uno de los equipos. Además que sí no hay un equipo dedicado, existe el riesgo de no tener avances o no darle continuidad al proyecto de implementación.

Es un equipo multidisciplinario, porque implementar IC no requiere solamente de conocimientos de desarrollo, sino también de administración de sistemas, redes y operaciones. Durante la implementación, antes de gozar de los beneficios de IC, los equipos deben mejorar sus prácticas y procesos de desarrollo de software existentes por lo que se tener experiencia en mejora de procesos puede ser beneficioso.

La selección de los miembros del equipo es clave para generar sinergia, así como ser entusiastas por la innovación. Ellos serán los primeros en adoptar el cambio y transmitirlo a los demás equipos.

El tener un equipo multidisciplinario también permite establecer una mejor conexión y empatía al momento de realizar la sensibilización con los diferentes equipos de tecnología, lo que puede ayudar a suavizar la adopción.

Entrega continua de entrega continua

Para mantener el apoyo de los interesados y evitar que la iniciativa muera se debe ir demostrando el valor de la integración y entrega continua al negocio de manera temprana y continua. Esta estrategia también puede ayudar a reducir el riesgo y la dificultad de la adopción por parte de los equipos de desarrollo. Así como poder obtener retroalimentación de los equipos de desarrollo sobre las herramientas y procesos, para ir ajustándolo y ayudar a la adopción.

Martin Salas, consultor y entrenador en Kleer, recomienda que el proceso de adopción de IC debe ser evolutivo e incremental. Lo más importante es que los equipos de desarrollo cambien su mentalidad, sean equipos colaborativos, que hagan confirmaciones frecuentes y que tienen una forma estándar de trabajo.

Iniciar con aplicaciones sencillas pero importantes

Seleccionar las primeras aplicaciones que se montarán en integración y entrega continua es importante, sí se selecciona la aplicación equivocada no se podrá demostrar el valor y podrá perder apoyo de los interesados al no cumplir los tiempos o expectativas. La mejor estrategia es seleccionar aplicaciones sencillas de migrar, pero que sean importantes para el negocio.

Sencilla porque si la primera aplicación que se va a migrar es difícil de migrar esta tomará más tiempo y sí falla pondrá en riesgo la iniciativa, además del impacto en el impulso y la moral del equipo. Usualmente las aplicaciones más fáciles de migrar son las nuevas con equipos de mente abierta, que tienden a utilizar patrones de diseño modernos y tienen menos dependencia de sistemas heredados, porque les será más fácil re diseñar o adaptar la arquitectura de la aplicación.

Importante porque permitirá darle mayor visibilidad a los beneficios y el valor que brinda al negocio la integración y entrega continua.

Esqueleto Visual del Pipeline

Tener un diagrama visual de cómo será el pipeline final de la aplicación es útil para mantener el impulso y motivación de los equipos con aplicaciones que toma más tiempo la migración, por los cambios que se deben hacer en la arquitectura y prácticas de desarrollo.

Instalación y Configuración de Herramientas

Como se ha expuesto, construir un entorno de IC no es un proceso sencillo y corto, donde hay que mantener apoyo de los interesados demostrando desde temprano el beneficio que brinda a la compañía. En un proceso de implementación de IC es necesario:

Definir las aplicaciones que serán migradas al entorno de IC

Al hacer lo anterior, se define el alcance del proyecto de implementación en cuanto al catálogo de aplicaciones que serán parte de dicho cambio. Esto dará una ruta crítica en cómo avanzar para lograr que dichas aplicaciones puedan utilizarse bajo un entorno de IC.

Apoyarse de expertos en IC que facilite y apoye en el proceso

Es un buen ejercicio referenciarse con otras empresas que ya estén utilizando IC en sus desarrollos, sí se tiene filiales o empresas hermanas se pueden apoyar con la experiencia de las mismas. Esto servirá para tener un mejor contexto antes de iniciar.

Es recomendable buscar apoyo con expertos en IC, esto puede ahorrar tiempo por el conocimiento y orientación que puedan aportar en la implementación. Sin embargo, no todas las implementaciones han sido apoyadas por expertos o consultores desde el inicio, muchas han sido iniciativas de Tecnología y por el perfil de los miembros del equipo han podido implementar un primer entorno de IC.

Gestión de cuentas centralizada

Debido a la cantidad de herramientas a utilizar, es recomendable manejar un solo repositorio de cuentas ya sea por base de datos, LDAP u otro contenedor, ya que de esta manera hay un mayor control y orden sobre la información.

Selección de herramientas

Algunas de las herramientas utilizadas en un entorno de IC ya cuentan con características necesarias para implementarlas por completo sin la necesidad de depender de otras herramientas que permitan sostener dicho entorno. Se sugiere que se implemente el ambiente de IC con herramientas que ya cuentan con esas características y así evitar la complejidad de mantener diferentes herramientas. Por ejemplo Jenkins es un servidor de integración que provee la mayoría de características necesarias para todo lo que se necesita de IC.

Además se hace especial énfasis en los criterios para seleccionar las herramientas. Uno de ellos es considerar el soporte, así como también estar consciente sobre el lenguaje de programación que se tiene en ese momento y la facilidad que puede tener de integración con las demás herramientas.

No es conveniente para la implementación que dichas herramientas sean de alta complejidad y de baja flexibilidad, ya que el estar a la vanguardia o seguir el ritmo del crecimiento y evolución del negocio podría resultar en un alto coste económico y de tiempo. Sería más oportuno contar con herramientas de uso fácil y sencillo, y que además soporten múltiples plataformas, así como que no provoquen ataduras tecnológicas considerables.

En base a la realidad salvadoreña de las empresas, se ha encontrado una preferencia notable a las herramientas de código abierto y al ser integradas de manera interna por sobre herramientas complejas y costosas, que podrían ser en un principio subutilizadas. Así también esto permite ir completando el flujo de IC conforme los equipos de desarrollo se van adaptando a las prácticas de IC y estándares de código.

Flujo de Trabajo

Estrategia de Ramas y Versionamiento

Es recomendable definir la estrategia, norma o política de cómo nombrar las ramas, los commit y los lanzamientos, por ejemplo los lanzamientos podrían llevar como prefijo "*v*" seguido por el número de versión (v1.1.0); los commits deberían incluir información sobre las funcionales o cambios que se están incorporando; también la política de ramas es recomendable basarse en estrategias estrictas y validadas como por ejemplo las que define GitFlow.

Para integración y entrega continua se recomienda usar GitFlow, el cual es un conjunto de extensiones que facilita la gestión de ramas y flujos de trabajo para organizar el trabajo del equipo. El centro en torno al cual gira GitFlow son las ramas. El trabajo se organiza en dos ramas principales:

1. **master**: Es la rama que contiene el código que está en producción. Se recomienda que esta rama se encuentre protegida y solamente se integrará cuando el código sea estable y con la calidad requerida para salir a producción, se puede tener una aprobación manual por medio de un "*merge request*".
2. **develop**: La rama que contiene las funcionalidades a incluir en una próxima salida a producción. Todos los desarrollos que se realizan se guardarán en develop, a la espera de salir a producción.

Al crear el repositorio, se recomienda inicializar el proyecto con el archivo "*README*", para que cree la rama master y a partir de esta se crea la rama develop.

Además de las dos ramas principales, el modelo GitFlow propone las siguientes ramas de apoyo:

1. **feature**: Es una rama con una copia exacta del contenido de develop. El propósito de las features es desarrollar nuevas funcionalidades del proyecto, en un ambiente controlado, de forma tal de que, una vez que estén listos, se agreguen primero a la rama develop.
2. **hotfix**: Es una rama con una copia exacta del contenido de master. Para poder realizar cambios sobre lo que está en master y corregir algún error en producción, lo que hacemos es generar una copia exacta de su contenido y lo metemos en una nueva rama que llamamos hotfix. De esta forma, evitamos trabajar directamente sobre el código que está en master, y volvemos a trabajar en un ambiente controlado.
3. **release**: Es una rama con una copia exacta del contenido de develop, que está lista para ser enviada a producción. Alguien del equipo genera la rama release para publicar en el ambiente de pruebas. El objetivo es someterla a unas últimas revisiones antes de enviarla a producción.

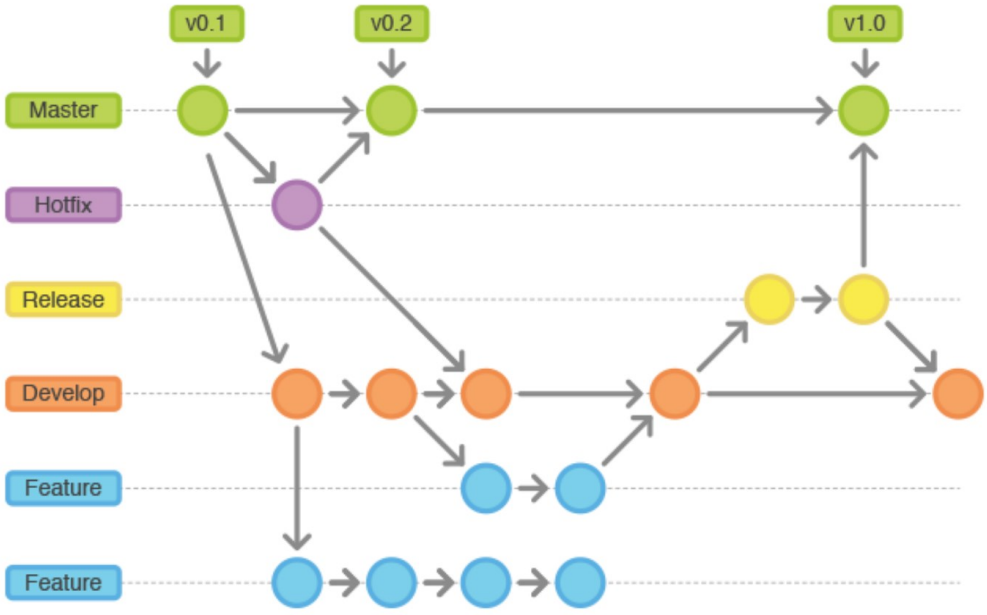


Figura 1. GitFlow.

Integraciones y Gestión de Conflictos

IC basa su filosofía en poder hacer que los equipos integren su código diariamente. Así que, es una buena práctica cuando se inicia un proceso de implementación definir la frecuencia con la que se harán tales integraciones. Si se quiere que un entorno de IC trabaje efectivamente, los desarrolladores deberán cambiar sus hábitos de desarrollos de software, esta estrategia los obliga a realizar integraciones, "*pull*" y "*push*", a la rama develop diariamente o cuando se tenga un componente terminado.

El proceso de IC puede iniciarse automáticamente después de un "*push*" a la rama develop o con una tarea automática que se ejecute diariamente al finalizar el día o cada semana o manualmente. Es importante que la ejecución se realice lo más pronto posible para tener una retroalimentación de cualquier posible fallo.

Además es importante definir una estrategia para la resolución de conflictos que seguramente se van a presentar cuando se realizan las integraciones. Se puede configurar y ajustar Git para que siga una estrategia predefinida cuando se hagan las integraciones.

Es recomendable configurar notificaciones cuando suceden errores, agregando a la lista de interesados de la tarea del servidor de integración continua a personas como desarrolladores, lideres o encargados de proyectos, entre otros; los cuales deben ser definidos según las necesidades, el tamaño del equipo, conveniencia u otros factores.

Pipeline

Estrategia de Pipeline

El pipeline define cómo será el flujo que seguirá todo el proceso de IC, se debe dedicar tiempo para definir una estrategia que permita asegurar que el proceso es el más óptimo para el proyecto, basado en las necesidades y objetivos de la empresa.

Un flujo base puede ser:

1. El desarrollador al haber completado una funcionalidad y está haber pasado exitosamente las pruebas unitarias, versiona el código en la rama develop en el repositorio.
2. El administrador de repositorios mediante un "*webhook*" o un proceso manual se comunica con el servidor de integración.
3. El servidor de integración inicia el conjunto de tareas configuradas:
    a. Descargar las dependencias del repositorio de artefactos.
    b. Compilar el código.
    c. Ejecutar las pruebas unitarias.
    d. Ejecutar las pruebas de código estático.
    e. Desplegar en un entorno de pruebas.
    f. Ejecutar pruebas de regresión.
    g. Ejecutar pruebas de rendimiento.
    h. Sí cumplió con la calidad esperada, integrar a la rama release.
    i. Sí se realizará el despliegue automático, integrar a la rama master y desplegar en el entorno de pre producción o producción.
    j. Sí se realizará el despliegue con una aprobación manual, dejar el "*merge request*" para ser aprobado e integrado a la rama master.

Cabe mencionar que el flujo antes descrito puede tener variantes, ya que por ejemplo algunas empresas, realizan las pruebas de código estático en una tarea aparte del

pipeline del proyecto, que puede ser ejecutada en un momento posterior y con diferente periodicidad.

En la mayoría de los casos, el despliegue automático a producción no está considerado, porque se considera más apropiado colocar a una persona encargada de validar que dicho cambio sea acorde al requerimiento y que cumpla con la documentación necesaria.

Pruebas

Es recomendable el uso de pruebas unitarias, funcionales y de regresión automatizadas, porque mitigan el riesgo de errores como consecuencia de un cambio. Así mismo sería ideal incluir pruebas de rendimiento.

Respecto a las pruebas unitarias, es importante que se realice un plan de pruebas utilizando alguno de los enfoques de pruebas unitarias como "*Test Driven Developmen*t" (TDD) o "*Behavior Driven Development (BDD)*".

Estándares de Codificación y Métricas

SonarQube como herramienta de análisis estático tiene una configuración por defecto, sin embargo se recomienda implementar las métricas gradualmente dependiendo del enfoque y meta del proyecto para facilitar la adopción.

Por ejemplo una métrica de calidad puede ser que dentro del resultado de las pruebas de código estático, las vulnerabilidades y bugs se encuentren en cero, ya que pueden representan un peligro alto para el negocio; y por otro lado que no se mida la utilización de código innecesario, porque son aplicaciones que se han venido trabajando sin métricas y seguramente no cumplirá con el estándar.

Al principio los márgenes de tolerancia puede que estén muy altos o bajos, pero consolidando diferentes métricas y con la experiencia de cada iteración se irán ajustando, se encontrarán en continua mejora. Inicialmente se puede recomendar centrarse en las siguientes métricas: Cobertura del código por pruebas unitarias, duplicidad de código, complejidad ciclomática y los comentarios, ya que son las más fáciles de interpretar y tienen repercusión directa en la deuda técnica.

Referencias

KPMG Ltda. (1 de Diciembre de 2018). *home.kpmg.* Obtenido de Gestión del Cambio: https://home.kpmg/co/es/home/services/advisory/management-consulting/capitalhumano-y-gestion-del-cambio/gestion-del-cambio.html

Duvall, P. M., Matyas, S., & Glover, A. (2008). *Continuous Integration, Improving Software Quality and Reducing Risk.* Dorling Kindersley.

Elsevier Inc. (2017). Continuous Delivery: Overcoming adoption challenges. *L. Chen / The Journal of Systems and Software*, 72-86.

Forrester Consulting. (Marzo de 2013). *info.thoughtworks.com.* Obtenido de Continuous Delivery: A Maturity Assessment Model: http://info.thoughtworks.com/rs/thoughtworks2/images/continuous_delivery_a_maturity_assessment_modelfinal.pdf

6.5 Guía de implementación del entorno de integración continua para el desarrollo de aplicaciones web en Java

UNIVERSIDAD DON BOSCO

Vicerrectoría de Estudios de Postgrados

Maestría en Arquitectura de Software

Guía de implementación del entorno de integración continua para el desarrollo de aplicaciones web en Java

Presentado por:

María Lorena De La Fuente Jacobo

William Antonio Flores Ayala

Roberto Bladimir Hernández Guevara

Asesor:

Luis García

Antiguo Cuscatlán, La Libertad, El Salvador

Enero 2019

Índice

Introducción

La presente guía es un insumo para dar los primeros pasos en la construcción de un entorno de desarrollo ágil para el desarrollo de aplicaciones Web basadas en Java, utilizando integración continua, conocida de aquí en adelante como IC. Se incluye el diseño de una arquitectura, la instalación y configuración de las herramientas que conforman el entorno de IC.

Se explica paso a paso la instalación y configuración del servidor de integración Jenkins, quien es el responsable de orquestar y ejecutar todo el "pipeline" de IC, construcción y ejecución de pruebas unitarias. Las pruebas unitarias representan uno de los pilares de IC y no son más que pruebas que el desarrollador escribe para probar cada pieza o componente del software. (Martin Fowler). Además, aprenderá cómo instalar y configurar el sistema de versionamiento GIT, un sistema que controla y permite llevar un seguimiento de todos los cambios hechos en el código, así como también instalar y configurar GitLab uno de los administradores de repositorios basados en GIT más populares. Para el análisis de código estático aprenderá cómo instalar y configurar el servidor de SonarQube. El análisis de código estático se refiere a la capacidad de analizar el software sin necesidad de ejecutar el programa que va a ser analizado y para ello se hace uso de SonarQube.

Las herramientas fueron seleccionadas en base al estudio realizado a un conjunto de empresas del área metropolitana de San Salvador, que han implementado y están utilizando integración continua por lo menos durante 6 meses en sus desarrollos de aplicaciones Web. Así mismo se validó con los expertos a finales del 2018.

Preparación

Antes de comenzar la implementación del entorno de IC hay que evaluar el modelo de desarrollo que se está utilizando y la arquitectura de las aplicaciones Web. Hay una relación de integración y entrega continua con el desarrollo ágil, por lo que es recomendable implementar modelos ágiles para aprovechar los beneficios; siendo Scrum el modelo más utilizado según el estudio. Este es un cambio organizacional, por lo que requiere el apoyo de la alta gerencia, así como la capacitación y definición de nuevos roles dentro de equipos multidisciplinarios.

Un cambio organizacional, ya sea que se trate de agregar un nuevo proceso o modificar uno existente, puede generar resistencia por lo que al iniciar un proceso de implementación de IC debe existir una correcta gestión del cambio.

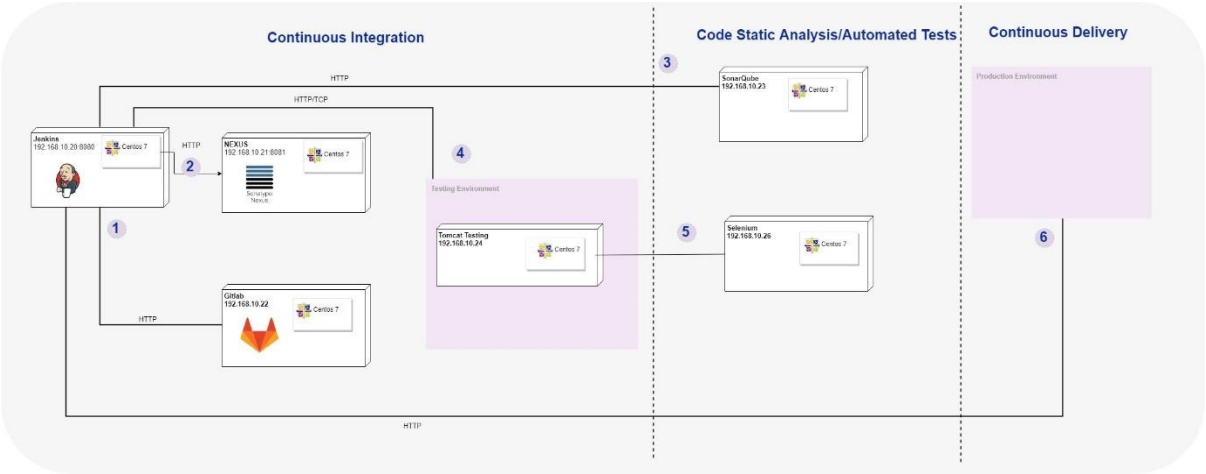
Se ha coincidido con las empresas y expertos que la gestión del cambio y la cultura ágil son claves para la adopción de IC. Se debe invertir más tiempo en conocer a los equipos, su experiencia y conocimiento, para poder hacerlos parte del nuevo proceso que en la selección y configuración de herramientas.

Es importante definir las aplicaciones que serán migradas al entorno de IC, iniciando con aplicaciones sencillas pero importantes que den valor al negocio a corto plazo y demuestren los beneficios de IC manteniendo así el apoyo a la iniciativa. Es recomendable también, buscar apoyo de expertos en IC, esto puede ahorrar tiempo por el conocimiento y orientación que puedan aportar en la implementación.

Es muy apropiado definir la estrategia de ramas, versionamiento, integraciones, gestión de conflictos, “pipelines” y pruebas. De igual forma, es relevante el establecimiento de estándares de desarrollo y métricas de calidad que cada una de las herramientas debe de verificar para validar una nueva funcionalidad.

Para mayor detalle de las estrategias y recomendaciones para la implementación y adopción de IC, referenciarse con la guía de buenas prácticas adjunta.

Arquitectura del Entorno de IC para JAVA



Herramientas del Entorno de IC

Compilador de código

JDK

Es un entorno de desarrollo para crear aplicaciones, applets y componentes utilizando el lenguaje de programación Java. El JDK incluye herramientas útiles para desarrollar y probar programas escritos en el lenguaje de programación Java y que se ejecutan en la plataforma Java.

Control de Versiones

Git

Es un sistema de control de versiones distribuidas de código abierto y gratuito diseñado para manejar todo, desde proyectos pequeños a muy grandes, con velocidad y eficiencia.

Administrador de repositorios

GitLab

Es un administrador de repositorios Git, además tiene la funcionalidad de ser una aplicación individual para todas las etapas del ciclo de vida de DevOps.

Repositorio de artefactos

Nexus Repository OSS

Nexus proporciona un repositorio para empresas, por lo tanto, sirve como alojamiento propio de repositorios. Nexus puede servir como proxy para los repositorios públicos. Con dicho proxy, el tiempo para recibir un artefacto se reduce y ahorra ancho de banda. Nexus le permite alojar sus artefactos de construcción privados. Nexus está disponible como distribución comercial y de código abierto.

Construcción de proyecto

Apache Maven

Es una herramienta de gestión y comprensión de proyectos de software. Basándose en el concepto de un modelo de objeto de proyecto (POM), Maven puede gestionar la compilación, los informes y la documentación de un proyecto a partir de una pieza central de información.

Servidor de Integración

Jenkins

Es un servidor autónomo de código abierto que se puede usar para automatizar todo tipo de tareas relacionadas con la construcción, prueba y entrega o implementación de software. Este puede ser considerado como la piedra angular para implementar de manera efectiva IC en una organización

Evaluador de código estático

SonarQube

Es una plataforma de código abierto para realizar revisiones automáticas con análisis estático de código para detectar errores, olores de código y vulnerabilidades de

seguridad en más de 20 idiomas de programación, incluidos Java, C #, JavaScript, TypeScript, C / C ++, COBOL y más.

Pruebas unitarias

JUnit

Es un marco simple para escribir pruebas repetibles. Es una instancia de la arquitectura xUnit para marcos de prueba de unidades.

Servidor de despliegue

Apache Tomcat

Es un software con una implementación de código abierto de tecnologías Java Servlet, JavaServer Pages, Java Expression Language and Java WebSocket. Las especificaciones de Java Servlet, JavaServer Pages, Java Expression Language y Java WebSocket se desarrollan bajo el Java Community Process. El software Apache Tomcat se desarrolla en un entorno abierto y participativo y se lanza bajo la versión 2 de la Licencia Apache.

Proceso de instalación y configuración inicial del Servidor de Integración Jenkins

Prerrequisitos

Hardware

- 2 GB de RAM
- 1 CPU
- 1 Interfaz de red
- Disco duro de 10 GB

Software

- CentOS 7, 64bit
- Usuario con privilegios de super administrador.
- Tener instalado el comando wget y firewall-cmd en el sistema operativo.

Nota: Se está utilizando para el sistema operativo el usuario "*jenkinsuser*" como usuario administrador.

Paso 1. Actualizar paquetes del sistema

Una buena práctica de un administrador de sistemas Linux es mantener el sistema actualizado, por lo que se recomienda actualizar los paquetes a la última versión estable por medio de la terminal, ejecutando las siguientes líneas de comando con el usuario administrador.

1. Instalar el repositorio epel-release ejecutando:

sudo yum install epel-release

```
[jenkinsuser@mas ~]$ sudo yum install epel-release
[sudo] password for jenkinsuser:
Complementos cargados:fastestmirror, langpacks
Loading mirror speeds from cached hostfile
epel/x86_64/metalink                                    |  14 kB     00:00
```

2. Actualizar los paquetes a la última versión estable ejecutando:

sudo yum update

```
[jenkinsuser@mas ~]$ sudo yum update
Complementos cargados:fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: mirror.teklinks.com
 * epel: mirror.coastal.edu
 * extras: www.gtlib.gatech.edu
 * updates: mirror.cs.vt.edu
```

3. Reiniciar el servidor para que tome las actualizaciones:

sudo reboot

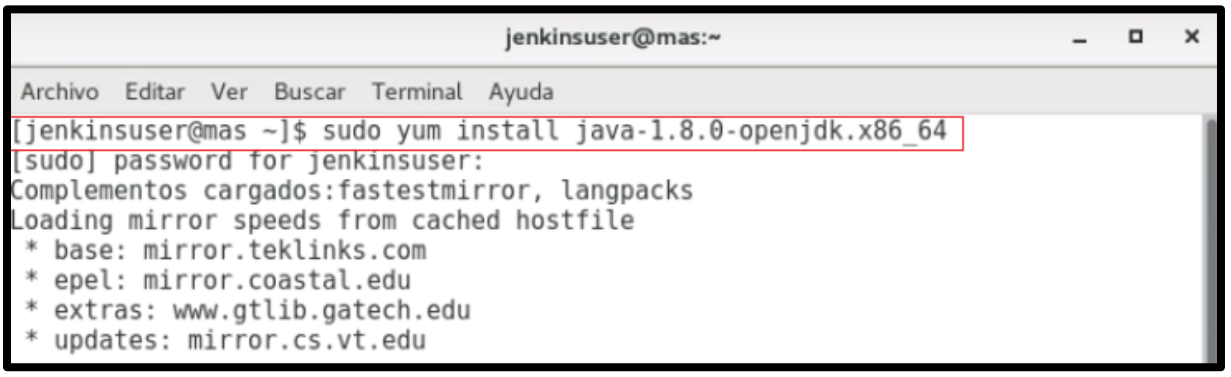
```
[jenkinsuser@mas ~]$ sudo reboot
[sudo] password for jenkinsuser:
```

Paso 2. Instalación Java

Se necesita instalar y configurar una máquina virtual Java, para este caso se usará la versión del OpenJDK Runtime Environment 1.8.0 mediante el manejador de paquetes YUM.

1. Instalar el OpenJDK Runtime Enviroment 1.8.0 ejecutando:

sudo yum install java-1.8.0-openjdk.x86_64

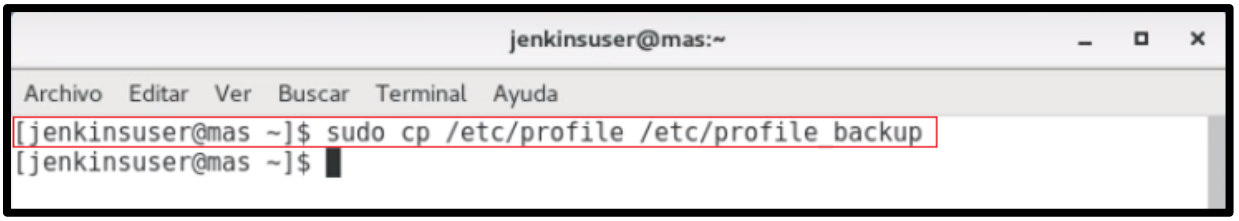


2. Comprobar la correcta instalación ejecutando el siguiente comando, el cual brindará la información del entorno java:

java –version

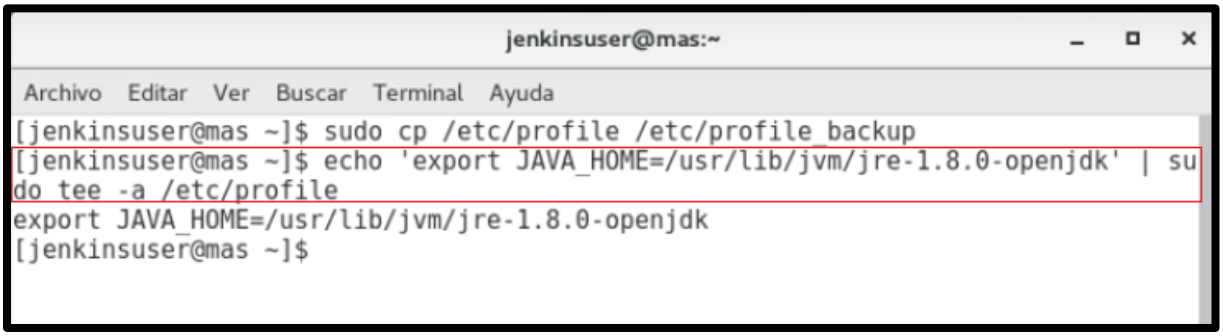
3. Antes de iniciar la configuración de las variables de entorno, es recomendable hacer un respaldo de las mismas ejecutando:

sudo cp /etc/profile /etc/profile_backup



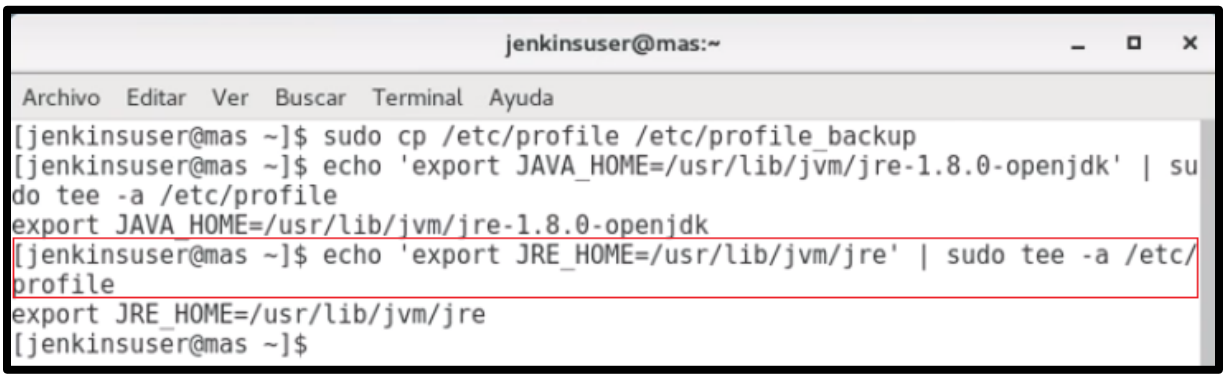
4. Asignarle valor a la variable de entorno “JAVA_HOME” ejecutando:

echo 'export JAVA_HOME=/usr/lib/jvm/jre-1.8.0-openjdk' | sudo tee -a /etc/profile



5. Asignarle valor a la variable de entorno “JRE_HOME” ejecutando:

echo 'export JRE_HOME=/usr/lib/jvm/jre' | sudo tee -a /etc/profile



6. Actualizar la terminal con los nuevos valores de las variables de entorno:

source /etc/profile

7. Asegurarse de la correcta definición de las variables de entorno imprimiéndolas en la terminal con el comando:

echo $JAVA HOME
echo $JRE HOME

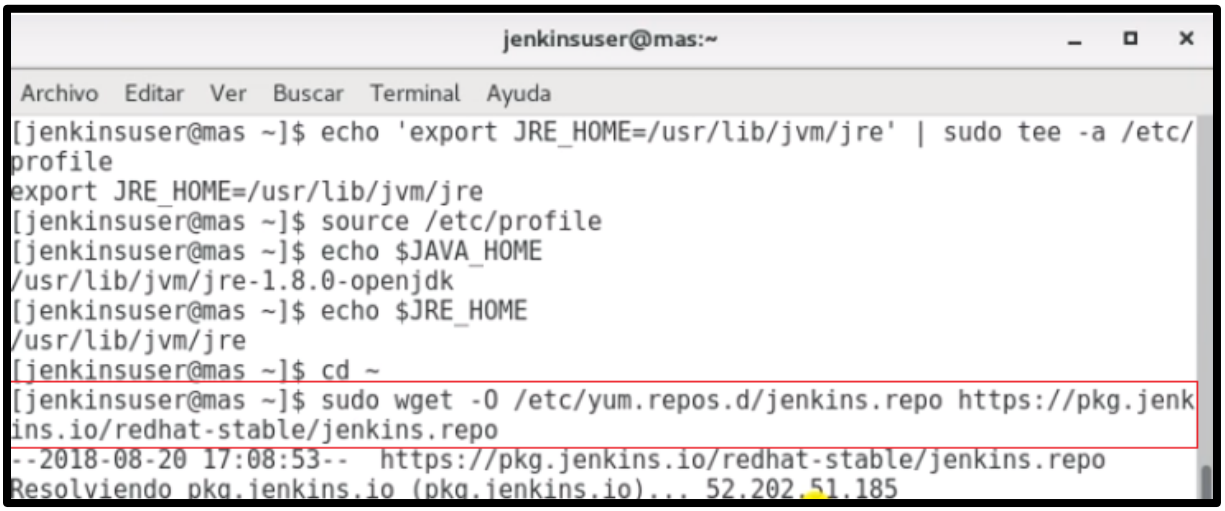
Paso 3. Instalación de Jenkins

Se continúa con la instalación de Jenkins utilizando el repositorio oficial de YUM, como recomendación instalar la última versión estable.

1. Ejecutar las siguientes líneas de comando para encontrarse en la carpeta raíz, descargar e importar Jenkins:

cd ~

sudo wget -O /etc/yum.repos.d/Jenkins.repo https://pkg.Jenkins.io/redhat-stable/Jenkins.repo



2. Importar las llaves oficiales ejecutando:

sudo rpm --import https://pkg.Jenkins.io/redhat-stable/Jenkins.io.key

3. Instalar Jenkins ejecutando:

sudo yum install Jenkins

```
[jenkinsuser@mas ~]$ yum install jenkins
Complementos cargados:fastestmirror, langpacks
Necesita ser usuario root para poder ejecutar este comando.
[jenkinsuser@mas ~]$ sudo yum install jenkins
Complementos cargados:fastestmirror, langpacks
```

4. Configurar el servicio de Jenkins para que se ejecute al iniciar el sistema con los siguientes comandos:

sudo systemctl start jenkins.service

sudo systemctl enable jenkins.service

```
Instalado:
  jenkins.noarch 0:2.121.3-1.1

¡Listo!
[jenkinsuser@mas ~]$ sudo systemctl start jenkins.service
[jenkinsuser@mas ~]$ sudo systemctl enable jenkins.service
jenkins.service is not a native service, redirecting to /sbin/chkconfig.
Executing /sbin/chkconfig jenkins on
```

5. Configurar debidamente el cortafuego para permitir el tráfico entrante al puerto destinado para el servidor web, en este caso se ocupará el puerto 8080:

sudo firewall-cmd --zone=public --permanent --add-port=8080/tcp

```
Transaction test succeeded
Running transaction
  Instalando    : jenkins-2.121.3-1.1.noarch                                1/1
  Comprobando   : jenkins-2.121.3-1.1.noarch                                1/1

Instalado:
  jenkins.noarch 0:2.121.3-1.1

¡Listo!
[jenkinsuser@mas ~]$ sudo systemctl start jenkins.service
[jenkinsuser@mas ~]$ sudo systemctl enable jenkins.service
jenkins.service is not a native service, redirecting to /sbin/chkconfig.
Executing /sbin/chkconfig jenkins on
[jenkinsuser@mas ~]$ sudo firewall-cmd --zone=public --permanent --add-port=8080
/tcp
success
[jenkinsuser@mas ~]$
```

sudo firewall-cmd --reload

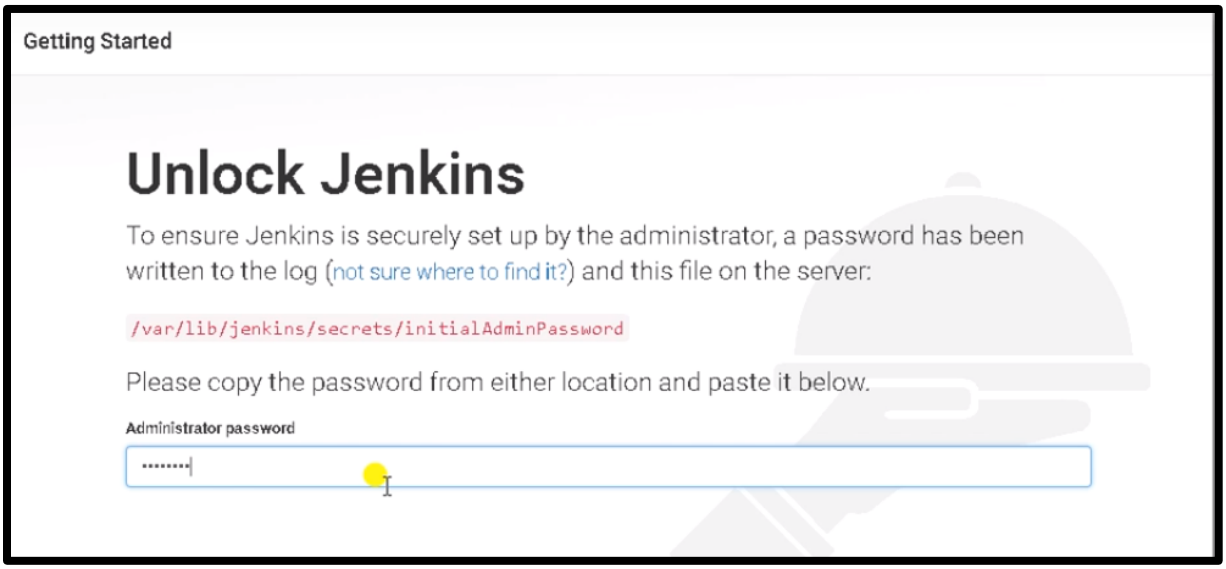
6. Para comprobar el correcto funcionamiento del servidor, desde un navegador web acceder a la siguiente dirección:

http://<HOST-IP>:8080

7. Nos indicará la ruta donde ha generado la contraseña de administrador en un archivo log que se encuentra en el servidor. Utilizar la siguiente línea de comando para ver el archivo:

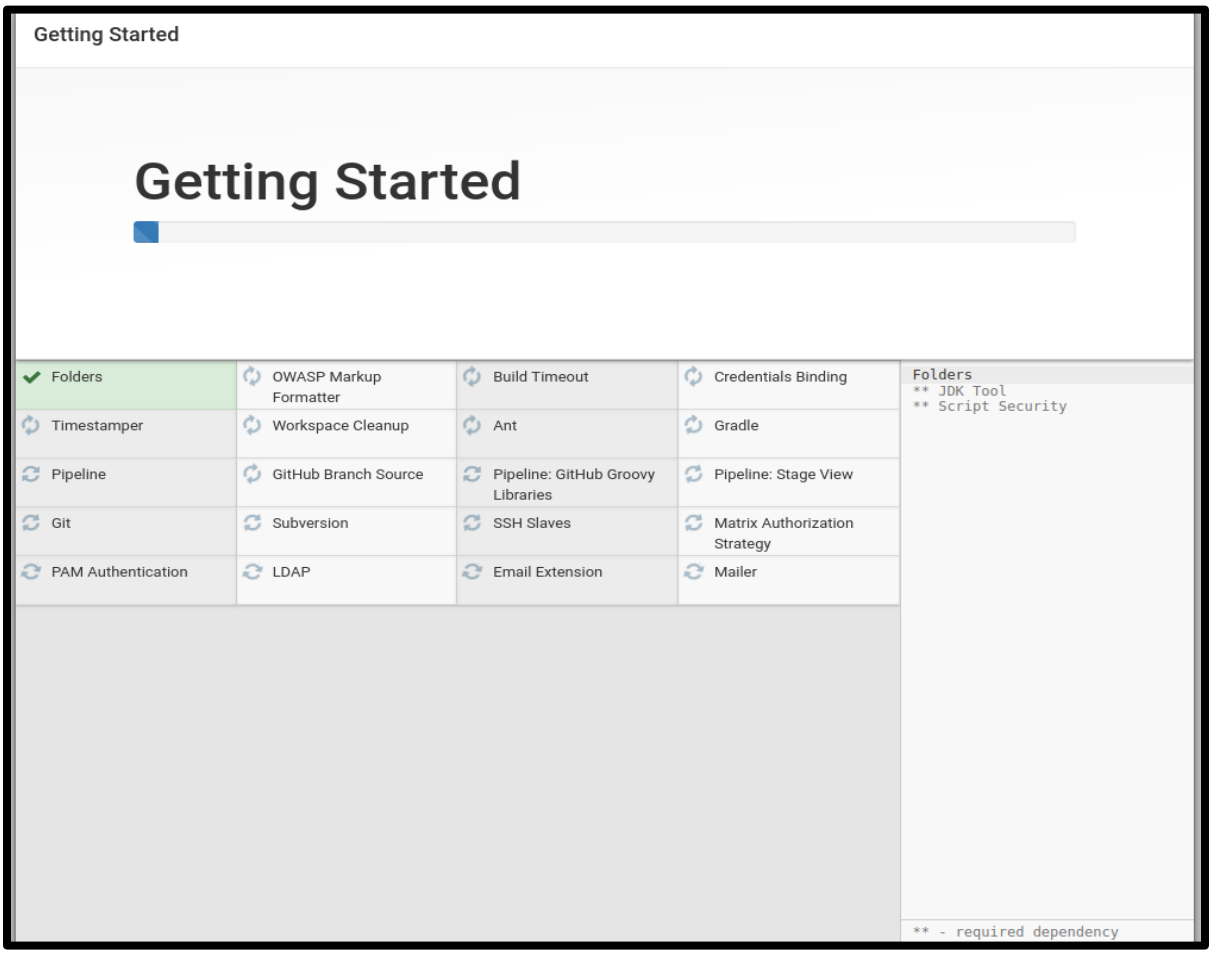
sudo vi /var/lib/Jenkins/secrets/initialAdminPassword

8. Copiar la contraseña e ingresarla en la dirección que se tiene abierta en el navegador web, para desbloquear el servidor de Jenkins.



Paso 4. Configuración inicial de Jenkins

1. Se inicia instalando los complementos sugeridos.



2. Se continúa creando el usuario administrador.

3. Se está listo para ingresar al panel de Jenkins.

4. Para poder identificar los servidores, en caso de no tener un servidor de nombres de dominio (DNS), se recomienda añadir la dirección IP del host servidor Jenkins al archivo "*hosts*":

sudo vim /etc/hosts

5. Se agrega la línea de comando con la dirección IP y el dominio definidos, para lo cual se deberá cambiar la información según convenga, por ejemplo:

192.168.1.XX Jenkins.mas.com

Paso 5. Instalación Git

1. Para poder acceder a los repositorios de código es necesario instalar Git en el servidor Jenkins, ejecutando la siguiente línea de comando:

sudo yum -y install Git

Proceso de instalación y configuración inicial del Servidor de Repositorios GitLab

Prerrequisitos

Hardware

- 3 GB de RAM
- 1 CPU
- 1 Interfaz de red
- Disco duro de 20 GB

Software

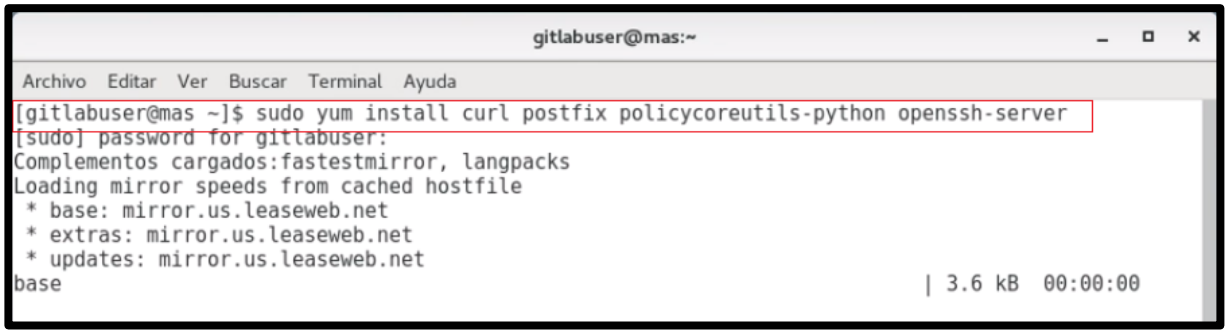
- CentOS 7, 64bit
- GitLab 11.2
- Usuario con privilegios de super administrador.
- Tener instalado el comando wget y firewall-cmd en el sistema operativo.

Nota: Se está utilizando para el sistema operativo el usuario "gitlabuser" como usuario administrador.

Paso 1: Instalación de Paquetes

1. Instalar los paquetes requeridos por GitLab, usando CURL para descargar el instalador del repositorio. Ejecutar el siguiente comando con el usuario administrador:

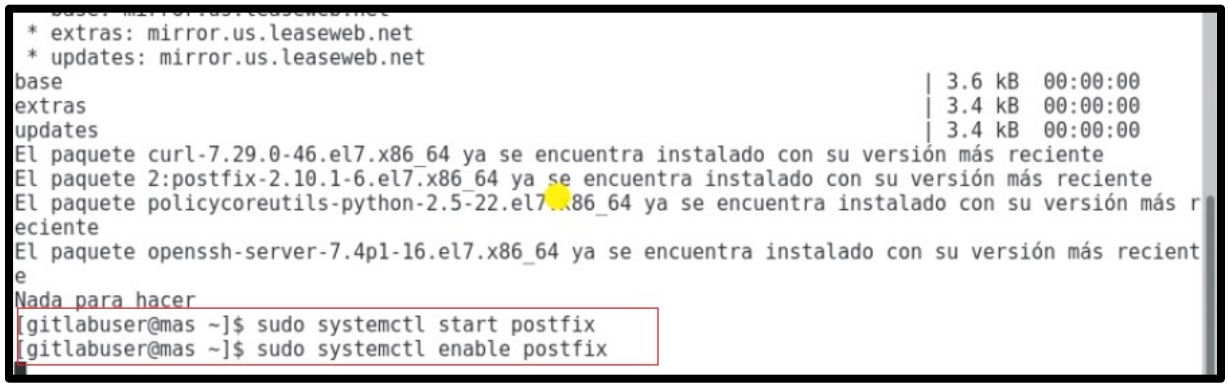
sudo yum -y install curl policycoreutils openssh-server openssh-clients postfix



2. Se recomienda activar los servicios para que se ejecuten de forma automática cuando inicie el sistema, ejecutando los siguientes comandos:

sudo systemctl enable postfix

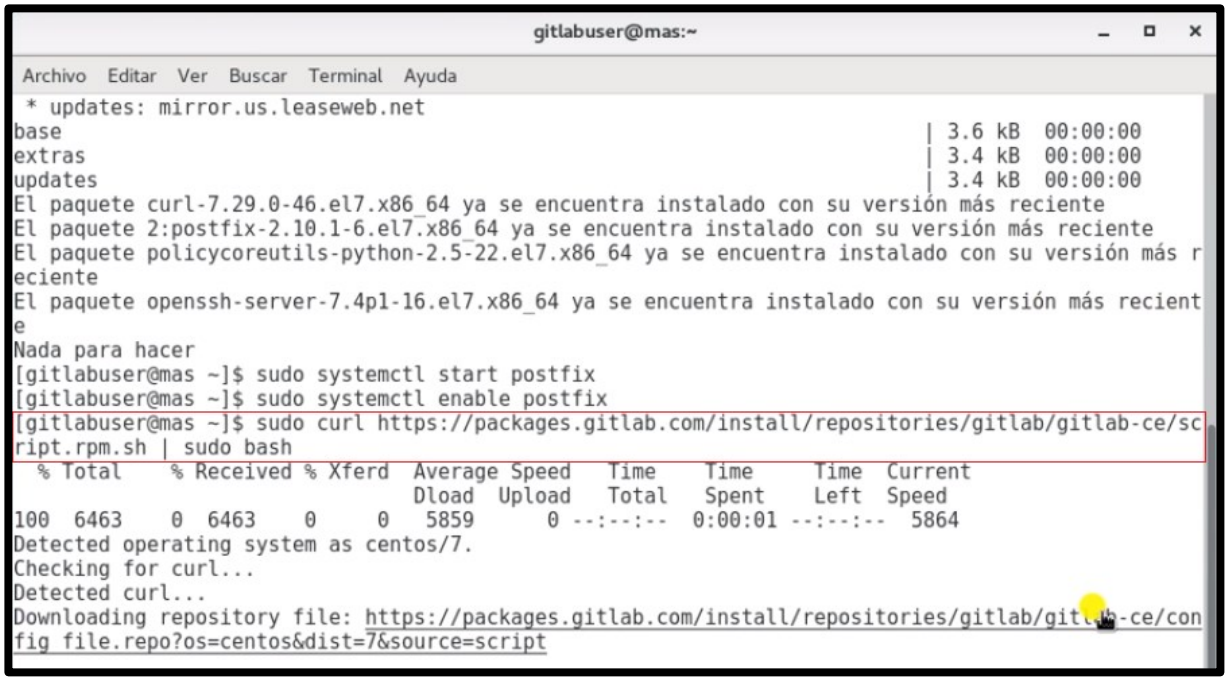
sudo systemctl enable postfix



Paso 2: Instalar GitLab

1. GitLab provee un instalador para agregar el repositorio oficial de GitLab. Lo único que se debe de hacer es descargar el instalador con CURL y ejecutar el script, tal y como se muestra a continuación:

sudo curl https://packages.gitlab.com/install/repositories/gitlab/gitlab-ce/script.rpm.sh | sudo bash



2. Para instalar GitLab se debe ejecutar la siguiente línea de comando y esperar a que termine el proceso:

sudo yum -y install gitlab-ce

3. Cuando la instalación se haya completado, se debe ser capaz de ver como resultado una pantalla similar a la que se muestra a continuación:

```
Thank you for installing GitLab!
GitLab was unable to detect a valid hostname for your instance.
Please configure a URL for your GitLab instance by setting `external_url`
configuration in /etc/gitlab/gitlab.rb file.
Then, you can start your GitLab instance by running the following command:
  sudo gitlab-ctl reconfigure

For a comprehensive list of configuration options please see the Omnibus GitLab readme
https://gitlab.com/gitlab-org/omnibus-gitlab/blob/master/README.md

  Comprobando   : gitlab-ce-11.3.6-ce.0.el7.x86_64                                                                1/1

Instalado:
  gitlab-ce.x86_64 0:11.3.6-ce.0.el7

¡Listo!
```

Paso 3: Configurar un Nombre de Dominio para GitLab

Configurar un dominio que permita identificar el servidor GitLab, en este caso se usará el dominio "*gitlab.mas.com*".

1. Ir al directorio "*etc/gitlab*", ejecutando la siguiente línea de comando en el editor VIM:

cd /etc/gitlab/

2. Abrir el archivo de configuración "*gitlab.rb*".

sudo vi gitlab.rb

3. Para completar la configuración en el archivo de configuración "*gitlab.rb*", cambiar la línea **external_url** por la dirección del dominio.

```
## GitLab configuration settings
##! This file is generated during initial installation and **is not** modified
##! during upgrades.
##! Check out the latest version of this file to know about the different
##! settings that can be configured by this file, which may be found at:
##! https://gitlab.com/gitlab-org/omnibus-gitlab/raw/master/files/gitlab-config-template/gitlab.rb.template


## GitLab URL
##! URL on which GitLab will be reachable.
##! For more details on configuring external_url see:
##! https://docs.gitlab.com/omnibus/settings/configuration.html#configuring-the-external-url-for-gitlab
external_url 'http://gitlab.mas.com'

## Roles for multi-instance GitLab
##! The default is to have no roles enabled, which results in GitLab running as an all-in-one instance.
##! Options:
##!   redis_sentinel_role redis_master_role redis_slave_role geo_primary_role geo_secondary_role
##! For more deatils on each role, see:
##! https://docs.gitlab.com/omnibus/roles/README.html#roles
##!
# roles ['redis_sentinel_role', 'redis_master_role']

## Legend
##! The following notations at the beginning of each line may be used to
##! differentiate between components of this file and to easily select them using
##! a regex.
##! ## Titles, subtitles etc
##! ##! More information - Description, Docs, Links, Issues etc.
##! Configuration settings have a single # followed by a single space at the
##! beginning; Remove them to enable the setting.

##! **Configuration settings below are optional.**
##! **The values currently assigned are only examples and ARE NOT the default
##!   values.**


################################################################################
################################################################################
##                Configuration Settings for GitLab CE and EE                 ##
################################################################################
################################################################################

################################################################################
## gitlab.yml configuration
##! Docs: https://gitlab.com/gitlab-org/omnibus-gitlab/blob/master/doc/settings/gitlab.yml.md
################################################################################
# gitlab_rails['gitlab_ssh_host'] = 'ssh.host_example.com'
# gitlab_rails['time_zone'] = 'UTC'
```

Paso 4: Configuración del Cortafuegos:

1. Se deben de agregar los puertos pertinentes al cortafuegos para acceder al servicio, generalmente el puerto ssh viene agregado por defecto. Para lo anterior ejecutar los siguientes comandos:

sudo systemctl start firewalld

sudo systemctl enable firewalld

sudo firewall-cmd --permanent --add-service ssh

sudo firewall-cmd --permanent --add-service http

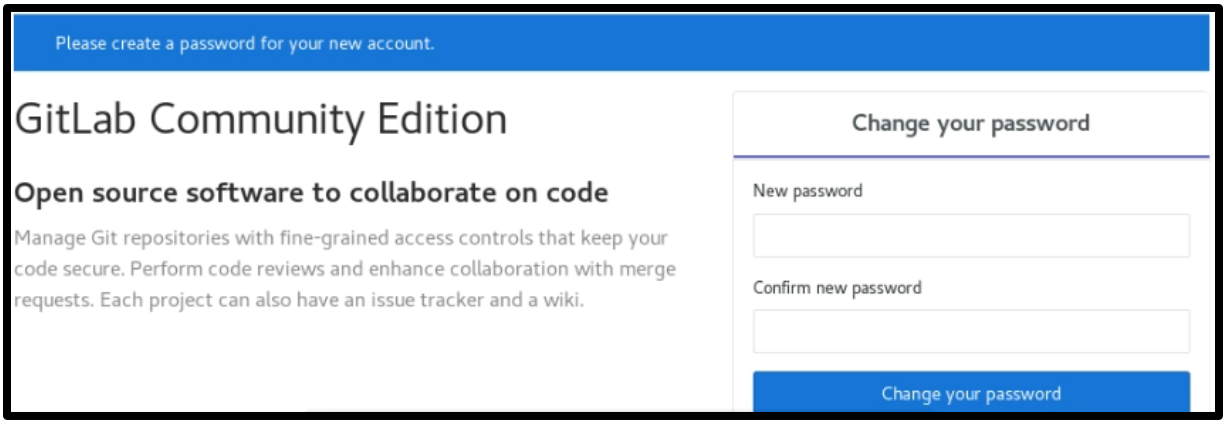
```
[gitlabuser@centos-s-1-mas-gitlab gitlab]$ sudo systemctl start firewalld
[gitlabuser@centos-s-1-mas-gitlab gitlab]$ sudo systemctl enable firewalld
[gitlabuser@centos-s-1-mas-gitlab gitlab]$ sudo firewall-cmd --permanent --add-service ssh
Warning: ALREADY_ENABLED: ssh
success
[gitlabuser@centos-s-1-mas-gitlab gitlab]$ sudo firewall-cmd --permanent --add-service http
success
[gitlabuser@centos-s-1-mas-gitlab gitlab]$
```

2. Para que se termine la instalación y asegurar que todas las configuraciones queden realizadas correctamente, ejecutar el siguiente comando:

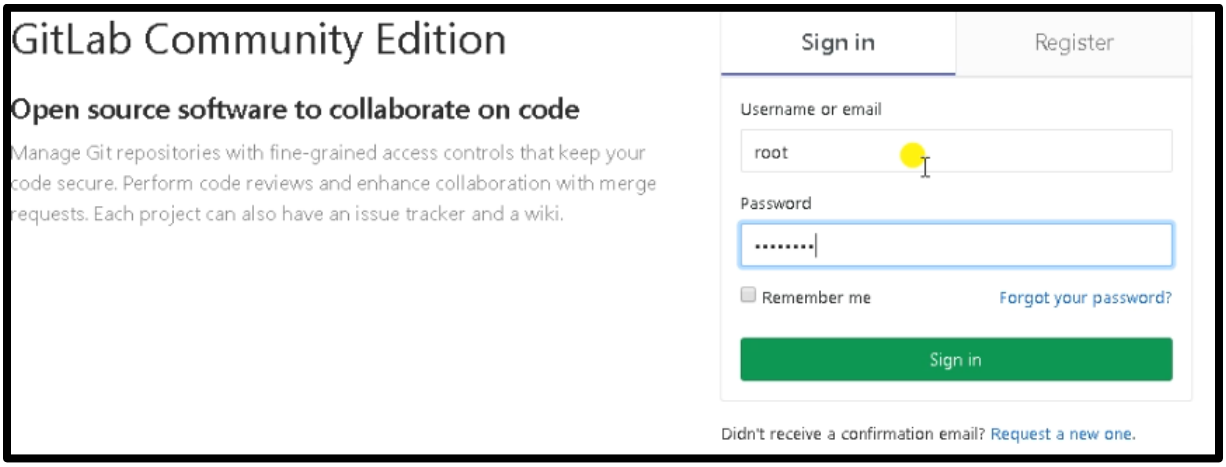
sudo gitlab-ctl reconfigure

Paso 5: Configuraciones post-instalación

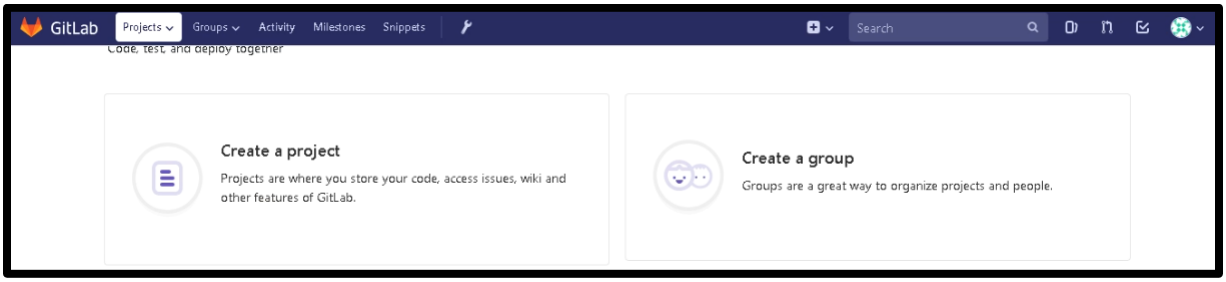
1. Restablecer la contraseña de usuario "*root*" del servidor GitLab, ingresando al servidor GitLab por medio de un navegador y la dirección, en este caso "*gitlab.mas.com*", lo cual abrirá la página web:



2. Ingresar una contraseña segura, que serán las credenciales para ingresar en el inicio de sesión:



3. Por último, se tendrá acceso a la consola administrativa de GitLab, desde la cual se pueden crear grupos, proyectos, configuraciones especiales, entre otras funcionalidades.


GitLab
Projects
Groups
Activity
Milestones
Snippets
Search
Code, test, and deploy together
Create a project
Projects are where you store your code, access issues, wiki and other features of GitLab.
Create a group
Groups are a great way to organize projects and people.

Proceso de instalación y configuración inicial de SonarQube

Prerrequisitos:

Hardware

- 2 GB de RAM
- 1 CPU
- 1 Interfaz de red
- Disco duro de 20 GB

Software

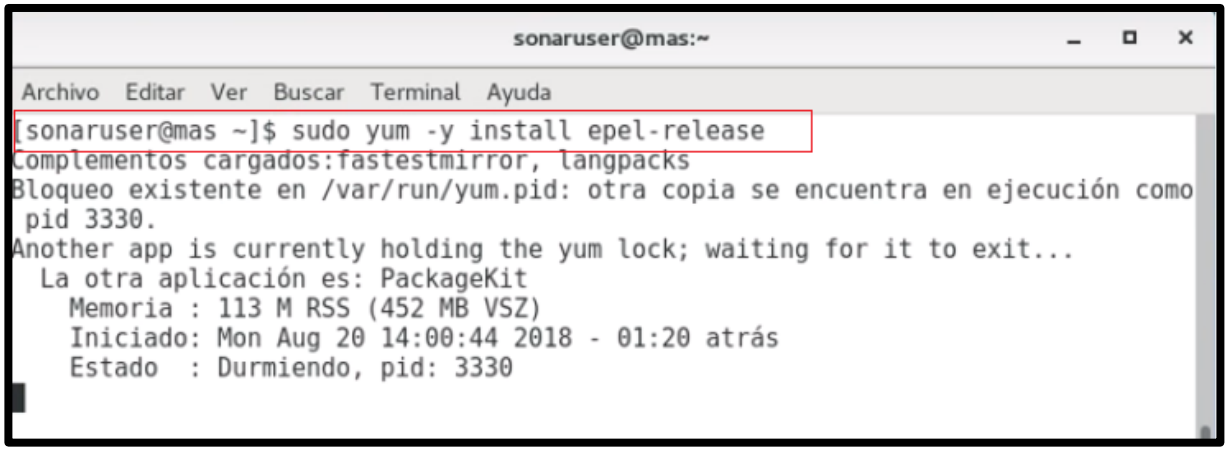
- CentOS 7, 64bit
- SonarQube 6.4
- Comando unzip instalado
- Usuario con privilegios de super administrador.
- Tener instalado el comando wget y firewall-cmd en el sistema operativo.

Nota: Se está utilizando para el sistema operativo el usuario "*sonaruser*" como usuario administrador.

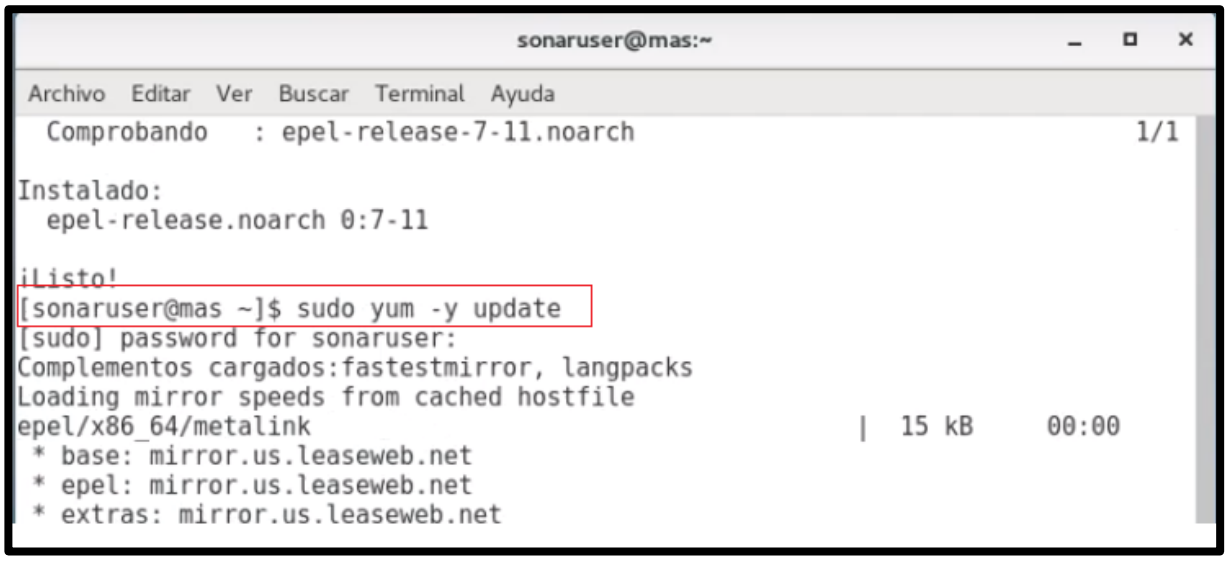
Paso 1: Instalación de Paquetes

1. En una terminal ejecutar las siguientes líneas de comandos, con el usuario administrador:

sudo yum -y install epel-release



sudo yum -y update



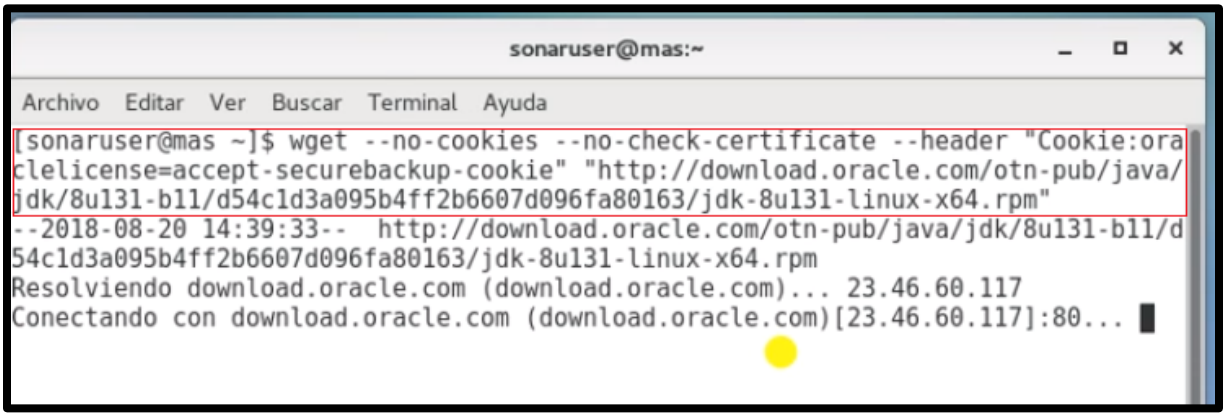
sudo shutdown -r now


systemtap-runtime.x86_64 0:3.2-8.el7_5
targetcli.noarch 0:2.1.fb46-6.el7_5
tzdata.noarch 0:2018e-3.el7
tzdata-java.noarch 0:2018e-3.el7
vdo.x86_64 0:6.1.0.168-18
xorg-x11-drv-wacom.x86_64 0:0.34.2-5.el7
yum-plugin-fastestmirror.noarch 0:1.1.31-46.el7_5
yum-utils.noarch 0:1.1.31-46.el7_5
¡Listo!
[sonaruser@mas ~]$ sudo shutdown -r now
[sudo] password for sonaruser:

Paso 2: Instalación Java

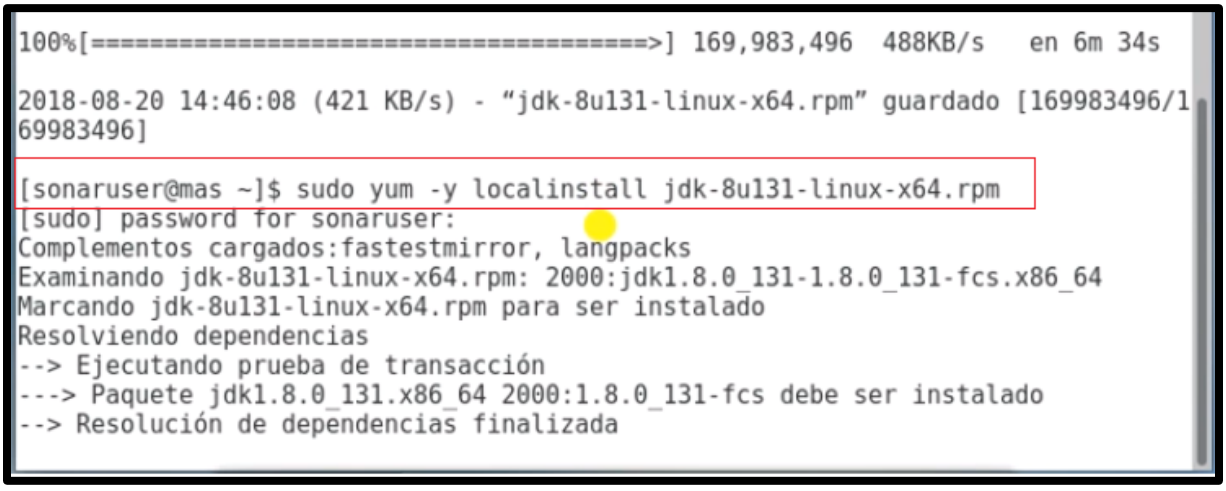
1. Descargar Java, para este caso se usará la versión del SE 1.8.0.131 del mismo, mediante el comando wget. Ejecutar el siguiente comando:

wget --no-cookies --no-check-certificate --header "Cookie:oraclelicense=accept-securebackup-cookie" "http://download.oracle.com/otn-pub/java/jdk/8u131-b11/d54c1d3a095b4ff2b6607d096fa80163/jdk-8u131-linux-x64.rpm"



2. Instalar el archivo descargado de la siguiente manera:

sudo yum -y localinstall jdk-8u131-linux-x64.rpm



3. Para finalizar se puede verificar la información de la instalación de Java mediante el siguiente comando:

java -version

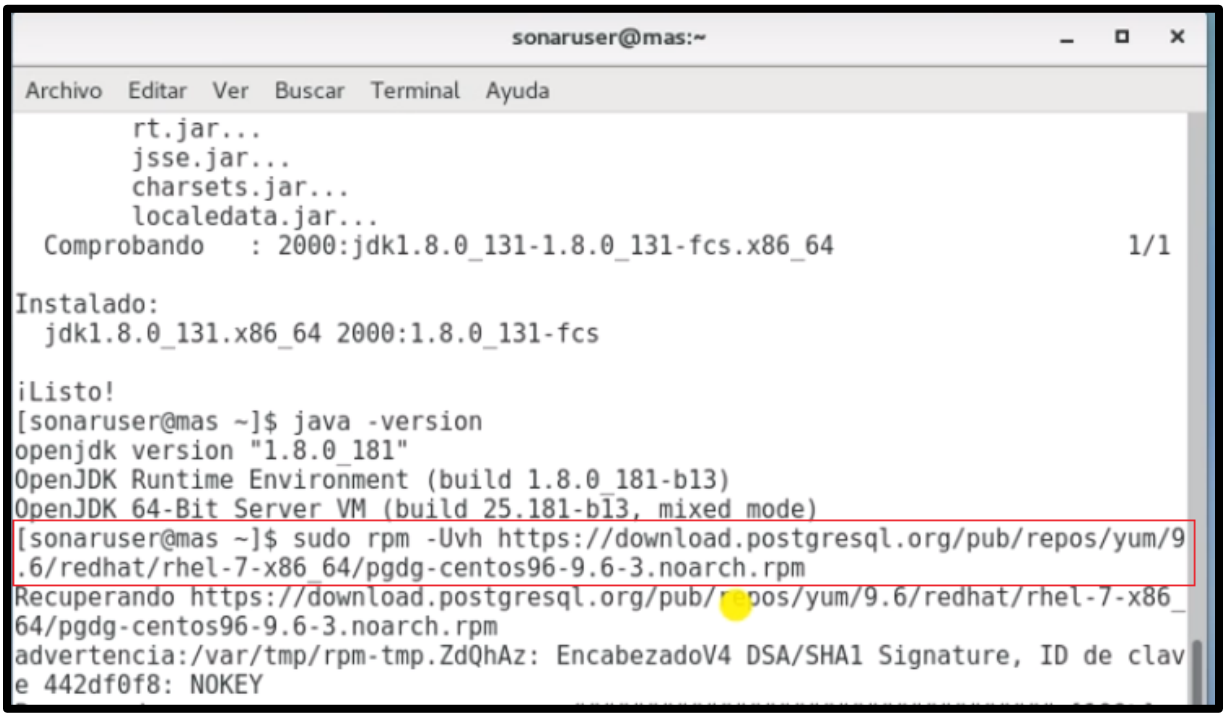

[sonaruser@mas ~]$ java -version
openjdk version "1.8.0_181"
OpenJDK Runtime Environment (build 1.8.0_181-b13)

Paso 3: Instalación de sistema de gestión de bases de datos

Se debe instalar un sistema de gestión de bases de datos que se adecue tanto a las metas como a los recursos de la empresa, el cual podría ser MySQL, Oracle, PostgreSQL, Microsoft SQL Server, etc. En este caso se utilizará PostgreSQL.

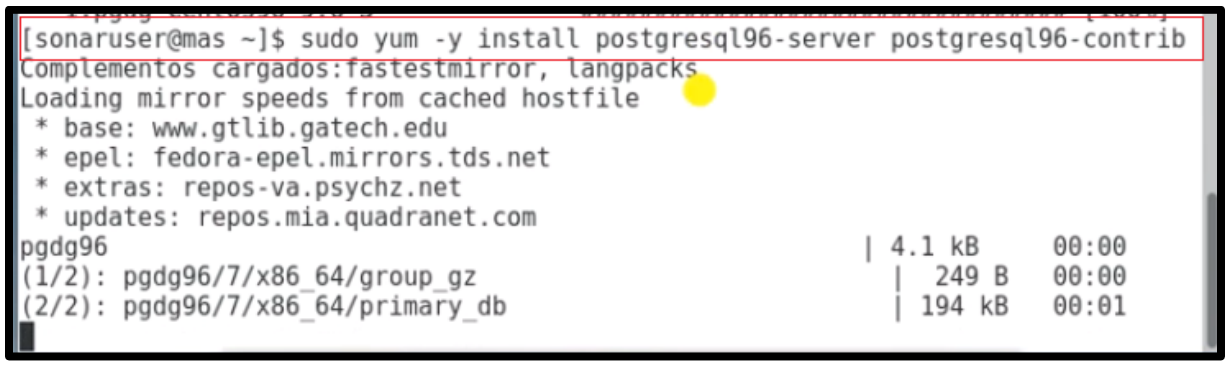
1. Instalar el repositorio para PostgreSQL:

sudo rpm -Uvh https://download.postgresql.org/pub/repos/yum/9.6/redhat/rhel-7-x86_64/pgdg-centos96-9.6-3.noarch.rpm



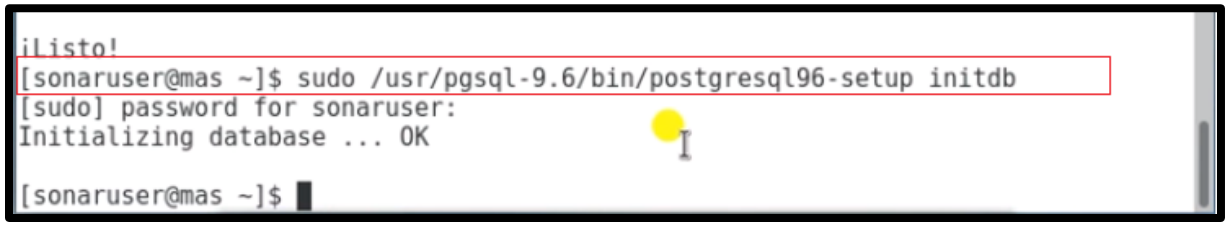
2. Instalar PostgreSQL con la ejecución del siguiente comando:

sudo yum -y install postgresql96-server postgresql96-contrib



3. Una vez instalado se debe inicializar la base de datos:

sudo /usr/pgsql-9.6/bin/postgresql96-setup initdb

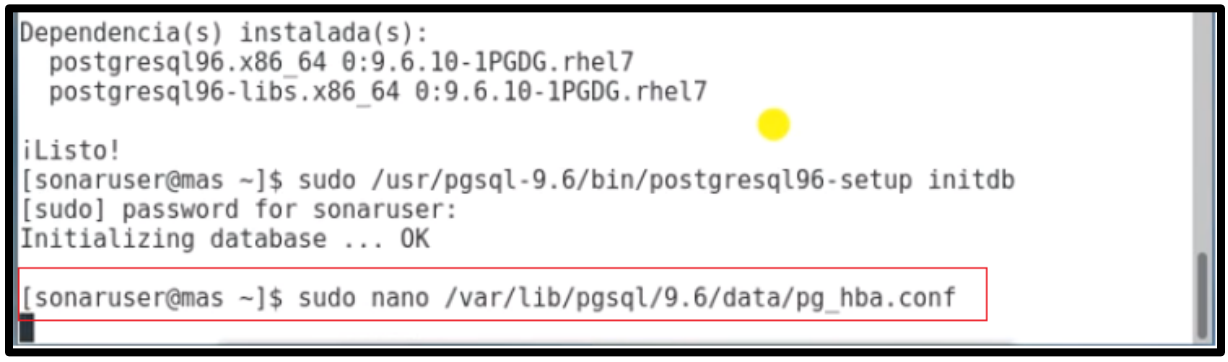


4. Si no se tiene el editor nano instalado, instalar con el siguiente comando:

yum install nano

5. Se debe realizar la edición del archivo "*/var/lib/pgsql/9.6/data/pg_hba.conf*" para habilitar la autenticación basada en MD5.

sudo nano /var/lib/pgsql/9.6/data/pg_hba.conf



6. Se debe identificar y cambiar las líneas que se refieran a peer, trusty e idnet a MD5 de la siguiente forma:

Pasar de:

```
# TYPE  DATABASE        USER            ADDRESS                 METHOD

# "local" is for Unix domain socket connections only
local   all             all                                     peer
# IPv4 local connections:
host    all             all             127.0.0.1/32            ident
# IPv6 local connections:
host    all             all             ::1/128                 ident
# Allow replication connections from localhost, by a user with the
# replication privilege.
#local   replication     postgres                                peer
#host    replication     postgres        127.0.0.1/32            ident
#host    replication     postgres        ::1/128                 ident
```

Cambiar a:

```
# listen on a non-local interface via the listen_addresses
# configuration parameter, or via the -i or -h command line switches.



# TYPE  DATABASE        USER            ADDRESS                 METHOD

# "local" is for Unix domain socket connections only
local   all             all                                     trust
# IPv4 local connections:
host    all             all             127.0.0.1/32            md5
# IPv6 local connections:
host    all             all             ::1/128                 md5
# Allow replication connections from localhost, by a user with the
```

7. Se debe iniciar el servidor PostgreSQL y es conveniente también permitir que éste se inicie automáticamente en el momento del arranque ejecutando:

sudo systemctl start postgresql-9.6

sudo systemctl enable postgresql-9.6

```
[sonaruser@mas ~]$ sudo /usr/pgsql-9.6/bin/postgresql96-setup initdb
[sudo] password for sonaruser:
Initializing database ... OK

[sonaruser@mas ~]$ sudo nano /var/lib/pgsql/9.6/data/pg_hba.conf
[sonaruser@mas ~]$ sudo nano /var/lib/pgsql/9.6/data/pg_hba.conf
[sonaruser@mas ~]$ sudo systemctl start postgresql-9.6
[sonaruser@mas ~]$ sudo systemctl enable postgresql-9.6
Created symlink from /etc/systemd/system/multi-user.target.wants/postgresql-9.6.
service to /usr/lib/systemd/system/postgresql-9.6.service.
```

8. Se necesita de cambiar la contraseña para el usuario predeterminado de PostgreSQL:

sudo passwd postgres

```
[sonaruser@mas ~]$ sudo passwd postgres
Cambiando la contraseña del usuario postgres.
Nueva contraseña:
```

9. Ahora se debe de cambiar al usuario "*postgres*":

su - postgres

```
[sonaruser@mas ~]$ su - postgres
Contraseña:
-bash-4.2$
```

10. En PosgreSQL se debe de crear un nuevo usuario tal como se muestra a continuación:

createuser sonar

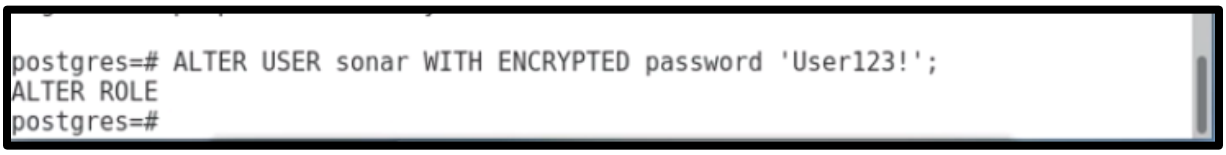
```
-bash-4.2$ createuser sonar
-bash-4.2$
```

11. Luego se debe de cambiar a la terminal de PostgreSQL con el uso del siguiente comando:

psql

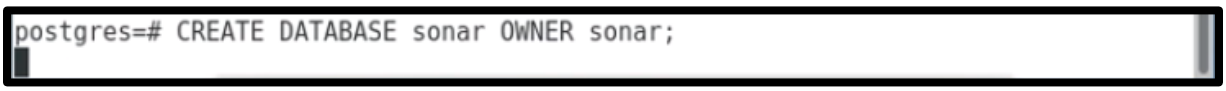
12. Establecer una contraseña segura para el usuario recién creado para la base de datos que se usará para SonarQube. Con fines didácticos se colocará la clave "*User123!"*.

ALTER USER sonar WITH ENCRYPTED password 'User123!';



13. Se crea una nueva base de datos para la base de datos PostgreSQL ejecutando:

CREATE DATABASE sonar OWNER sonar;



14. Habiendo realizado lo anterior, se debe de salir de la terminal psql ejecutando lo siguiente:

\q

15. Para seguir con la instalación de SonarQube se necesita volver al usuario sudo ejecutando el comando **exit**.

Paso 4: Descarga y configuración inicial de SonarQube

Es recomendable descargar la última versión estable de SonarQube, ya que estas versiones ya han solventado diversos bugs, y han sido probadas y garantizadas sobre gran cantidad de escenarios.

1. Descargar el archivo de instalación de SonarQube, para lo cual se utiliza el comando siguiente:

sudo wget https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-6.4.zip

```
[sonaruser@mas ~]$ wget https://sonarsource.bintray.com/Distribution/sonarqube/s
onarqube-6.4.zip
```

2. Si no se cuenta instalado unzip, se debe ejecutar el siguiente comando:

sudo yum install unzip

```
[sonaruser@mas ~]$ sudo yum -y install unzip
```

3. Descomprimir el archivo de SonarQube con el comando unzip:

sudo unzip sonarqube-6.4.zip -d /opt

```
[sonaruser@mas ~]$ sudo unzip sonarqube-6.4.zip -d /opt
```

4. De igual forma, se debe renombrar el directorio:

sudo mv /opt/sonarqube-6.4 /opt/sonarqube

```
[sonaruser@mas ~]$ sudo mv /opt/sonarqube-6.4 /opt/sonarqube
[sonaruser@mas ~]$
```

5. Abrir el archivo de configuración de SonarQube ”sonar.properties” con algún editor de texto; para este caso se utiliza el comando nano:

sudo nano /opt/sonarqube/conf/sonar.properties

```
[sonaruser@mas ~]$ sudo nano /opt/sonarqube/conf/sonar.properties
```

6. En este archivo, se deben de identificar las líneas siguientes:

#sonar.jdbc.username=

#sonar.jdbc.password=

7. Quitar el comentario y colocar las credenciales de la base de datos de PostGreSQL configuradas previamente:

sonar.jdbc.username=sonar

sonar.jdbc.password=User123

```
  GNU nano 2.3.1   Fichero: /opt/sonarqube/conf/sonar.properties    Modificado

# User credentials.
# Permissions to create tables, indices and triggers must be granted to JDBC us$
# The schema must be created first.
sonar.jdbc.username=sonar
sonar.jdbc.password=User123!

#----- Embedded Database (default)
# H2 embedded database server listening port, defaults to 9092
#sonar.embeddedDatabase.port=9092
#----- MySQL 5.6 or greater
# Only InnoDB storage engine is supported (not myISAM).
# Only the bundled driver is supported. It can not be changed.
#sonar.jdbc.url=jdbc:mysql://localhost:3306/sonar?useUnicode=true&characterEnco$


#----- Oracle 11g/12c
# - Only thin client is supported
# - Only versions 11.2.x and 12.x of Oracle JDBC driver are supported
# - The JDBC driver must be copied into the directory extensions/jdbc-driver/or$

^G Ver ayuda ^O Guardar   ^R Leer Fich ^Y Pág Ant   ^K CortarTxt ^C Pos actual
^X Salir     ^J Justificar^W Buscar    ^V Pág Sig   ^U PegarTxt  ^T Ortografía
```

8. Luego, se debe identificar la porción siguiente:

#sonar.jdbc.url=jdbc:postgresql://localhost/sonar

9. Quitar comentario a la línea, guardar el archivo y salir.

```
#----- Oracle 11g/12c
# - Only thin client is supported
# - Only versions 11.2.x and 12.x of Oracle JDBC driver are supported
# - The JDBC driver must be copied into the directory extensions/jdbc-driver/or$
# - If you need to set the schema, please refer to http://jira.sonarsource.com/$
#sonar.jdbc.url=jdbc:oracle:thin:@localhost:1521/XE


#----- PostgreSQL 8.x/9.x
# If you don't use the schema named "public", please refer to http://jira.sonar$
sonar.jdbc.url=jdbc:postgresql://localhost/sonar


#----- Microsoft SQLServer 2012/2014 and SQL Azure
```

Paso 5: Configurar como servicio

A conveniencia propia, SonarQube se puede iniciar directamente utilizando una secuencia de comandos de inicio que va incluida en el paquete del instalador.

1. Editar el archivo *"sonar.service"* con un editor de texto; para este caso se utiliza de nuevo el comando nano:

sudo nano /etc/systemd/system/sonar.service

```
[sonaruser@mas ~]$ sudo nano /etc/systemd/system/sonar.service
```

2. Agregar al archivo las siguientes líneas:

```
  GNU nano 2.3.1    Fichero: /etc/systemd/system/sonar.service      Modificado

[Unit]
Description=SonarQube service
After=syslog.target network.target

[Service]
Type=forking

ExecStart=/opt/sonarqube/bin/linux-x86-64/sonar.sh start
ExecStop=/opt/sonarqube/bin/linux-x86-64/sonar.sh stop

User=root
Group=root
Restart=always

[Install]
WantedBy=multi-user.target

^G Ver ayuda ^O Guardar   ^R Leer Fich ^Y Pág Ant   ^K CortarTxt ^C Pos actual
^X Salir     ^J Justificar^W Buscar    ^V Pág Sig   ^U PegarTxt   ^T Ortografía
```

Paso 6: Iniciar SonarQube

1. Iniciar SonarQube ejecutando la siguiente línea de comando:

sudo systemctl start sonar

```
[sonaruser@mas ~]$ sudo systemctl start sonar
[sonaruser@mas ~]$
```

2. Habilitar también el servicio SonarQube para que se inicie automáticamente en el momento del arranque:

sudo systemctl enable sonar

```
[sonaruser@mas ~]$ sudo systemctl enable sonar
Created symlink from /etc/systemd/system/multi-user.target.wants/sonar.service t
o /etc/systemd/system/sonar.service.
```

3. Verificar que el servicio se está ejecutando correctamente, lo cual se puede saber mediante el siguiente comando:

sudo systemctl status sonar

```
[sonaruser@mas ~]$ sudo systemctl enable sonar
Created symlink from /etc/systemd/system/multi-user.target.wants/sonar.service t
o /etc/systemd/system/sonar.service.
[sonaruser@mas ~]$ sudo systemctl status sonar
● sonar.service - SonarQube service
   Loaded: loaded (/etc/systemd/system/sonar.service; enabled; vendor preset: di
sabled)
   Active: active (running) since lun 2018-08-20 15:27:45 CST; 48s ago
 Main PID: 15212 (wrapper)
   CGroup: /system.slice/sonar.service
           ├─15212 /opt/sonarqube/bin/linux-x86-64/./wrapper /opt/sonarqube/b...
           ├─15214 java -Dsonar.wrapped=true -Djava.awt.headless=true -Xms8m ...
           ├─15246 /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.181-3.b13.el7_5.x86_...
           └─15287 /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.181-3.b13.el7_5.x86_...

ago 20 15:27:44 mas systemd[1]: Starting SonarQube service...
ago 20 15:27:44 mas sonar.sh[15165]: Starting SonarQube...
ago 20 15:27:45 mas sonar.sh[15165]: Started SonarQube.
ago 20 15:27:45 mas systemd[1]: Started SonarQube service.
[sonaruser@mas ~]$
```

Paso 7: Configurar el proxy inverso

Debido a que SonarQube de manera predeterminada resuelve por el puerto 9000 en localhost, es muy recomendable utilizar un proxy inverso para que se pueda acceder a la aplicación a través del puerto HTTP estándar. En este caso se utiliza el servidor web Apache.

1. Instalar el servidor web Apache se debe ejecutar lo siguiente:

sudo yum -y install httpd

```
[sonaruser@mas ~]$ sudo yum -y install httpd
```

2. Crear un nuevo host virtual editando el archivo sonar.yourdomain.com.conf, se abre el archivo con el comando nano:

sudo nano /etc/httpd/conf.d/sonar.yourdomain.com.conf

```
[sonaruser@mas ~]$ sudo nano /etc/httpd/conf.d/sonar.yourdomain.com.conf
```

3. Agregar al archivo lo siguiente:

```
<VirtualHost *:80>
   ServerName sonar.yourdomain.com
   ServerAdmin me@yourdomain.com
   ProxyPreserveHost On
   ProxyPass / http://localhost:9000/
   ProxyPassReverse / http://localhost:9000/
   TransferLog /var/log/httpd/sonar.yourdomain.com_access.log
   ErrorLog /var/log/httpd/sonar.yourdomain.com_error.log
</VirtualHost>
```

```
  GNU nano 2.3.1   Fichero: /etc/httpd/conf.d/sonar.mas.com.conf    Modificado

<VirtualHost *:80>
    ServerName sonar.mas.com
    ServerAdmin me@mas.com
    ProxyPreserveHost On
    ProxyPass / http://localhost:9000/
    ProxyPassReverse / http://localhost:9000/
    TransferLog /var/log/httpd/sonar.mas.com_access.log
    ErrorLog /var/log/httpd/sonar.mas.com_error.log
</VirtualHost>
```

4. Iniciar el servidor Apache, y se debe de habilitar el inicio automático en el momento del arranque ejecutando lo siguiente:

sudo systemctl start httpd

```
[sonaruser@mas ~]$ sudo systemctl start httpd
[sonaruser@mas ~]$
```

sudo systemctl enable httpd

```
[sonaruser@mas ~]$ sudo systemctl enable httpd
Created symlink from /etc/systemd/system/multi-user.target.wants/httpd.service t
o /usr/lib/systemd/system/httpd.service.
[sonaruser@mas ~]$
```

Paso 8: Configuración del Cortafuego

1. Es necesario configurar el cortafuego para que permita acceder a la aplicación mediante el puerto HTTP:

sudo firewall-cmd --add-service=http --permanent

```
[sonaruser@mas ~]$ sudo firewall-cmd --add-service=http --permanent
success
```

sudo firewall-cmd --reload

```
[sonaruser@mas ~]$ sudo firewall-cmd --reload
success
```

2. Iniciar el servicio de SonarQube

sudo systemctl start sonar

```
[sonaruser@mas ~]$ sudo systemctl start sonar
[sonaruser@mas ~]$
```

3. Se debe colocar el módulo de seguridad para el Kernel Linux conocido como SELinux en modo permisivo, para que no interfiera con la instalación:

sudo setenforce 0

```
[sonaruser@mas ~]$ sudo setenforce 0
[sonaruser@mas ~]$
```

Nota: Se deben monitorear los mensajes del registro de acciones de SELinux que por defecto se escriben dentro de /var/log/audit/audit.log, para determinar si se presentan errores de seguridad o de contexto que requieran corrección.

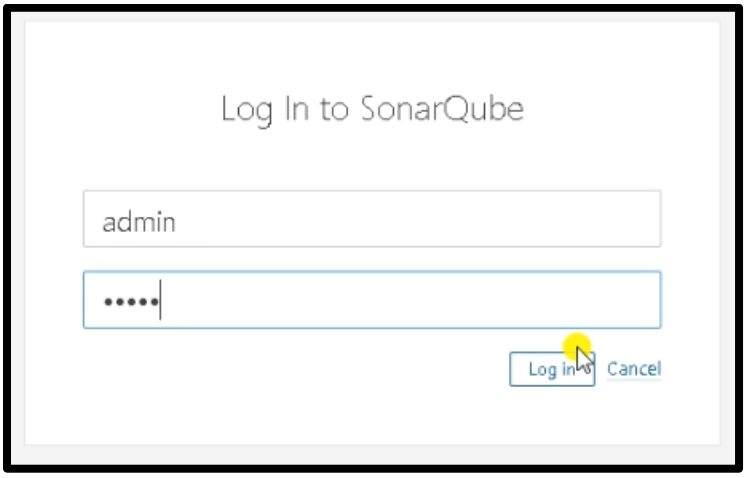
4. Luego de todo lo anterior, SonarQube está instalado correctamente en el servidor, y se accede mediante la dirección siguiente:

http://sonar.yourdomain.com

5. Hay que recordar que si se está trabajando en ambiente local, se debe configurar el dominio en el archivo hosts del sistema operativo.

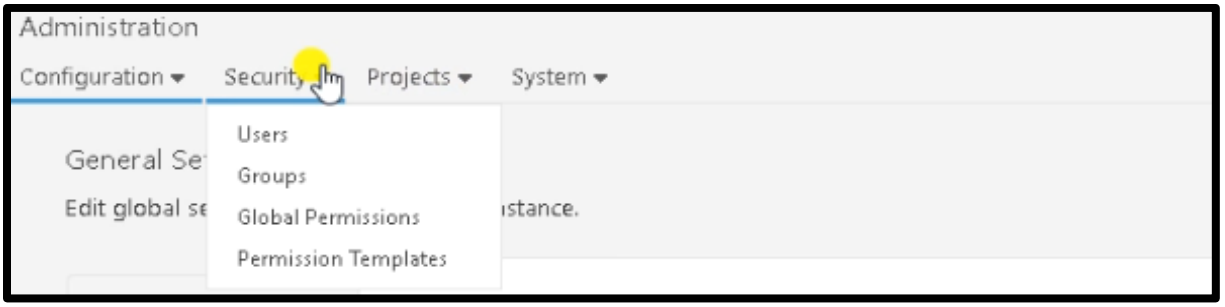
Paso 9: Crear Usuario

1. Iniciar sesión mediante la cuenta de administrador por defecto, la cual es con el usuario "*admin*" y clave "admin.

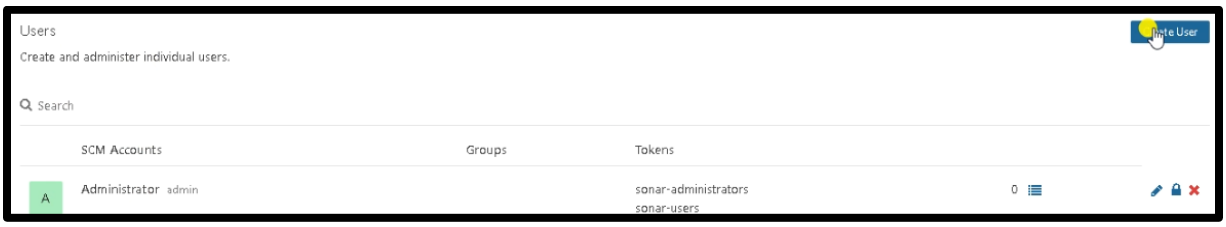


2. Ir a la opción "*Administration*":

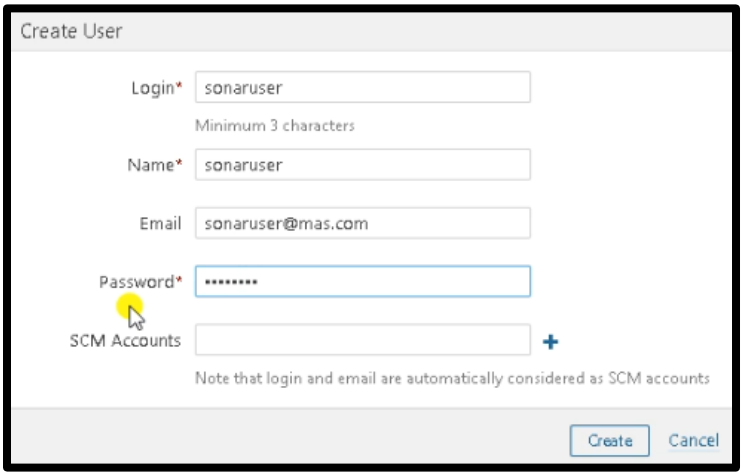
3. Seleccionar Security e ingresar a la opción "*Users*":



4. Dar clic en la opción "*Create User*":



5. Para finalizar llenar los datos correspondientes y hacer clic en "*Create*":



Proceso de instalación y configuración del Repositorio de Artefactos Sonatype Nexus

Prerrequisitos

Hardware

- 2 GB de RAM
- 1 CPU
- 1 Interfaz de red
- Disco duro de 10 GB

Software

- CentOS 7, 64bit
- Usuario con privilegios de super administrador.
- Tener instalado el comando wget y firewall-cmd en el sistema operativo.

Nota: Se está utilizando para el sistema operativo el usuario "*Nexususer*" como usuario administrador.

Paso 1. Actualizar paquetes del sistema

1. Para actualizar los paquetes a la última versión estable ejecutar:

sudo yum update -y

```
[nexususer@centos-s-1-mas-nexus ~]$ sudo yum update -y
```

Paso 2: Instalar Java

1. Después se debe instalar Java, por fines didácticos se utiliza OpenJDK Runtime Environment 1.8:

sudo yum install java-1.8.0-openjdk.x86_64

```
[nexususer@centos-s-1-mas-nexus ~]$ sudo yum install java-1.8.0-openjdk.x86_64
```

Paso 3: Instalación Nexus

1. Crear y acceder a un directorio llamado app:

sudo mkdir /app

```
[nexususer@centos-s-1-mas-nexus ~]$ sudo mkdir /app
[nexususer@centos-s-1-mas-nexus app]$
```

2. Descargar Nexus, para este caso Nexus-3.0.2-02:

sudo wget https://sonatype-download.global.ssl.fastly.net/Nexus/3/Nexus-3.0.2-02-unix.tar.gz

```
[nexususer@centos-s-1-mas-nexus ~]$ sudo wget https://sonatype-download.global.s
sl.fastly.net/nexus/3/nexus-3.0.2-02-unix.tar.gz
```

3. Después se debe descomprimir el archivo descargado:

sudo tar -xvf Nexus-3.0.2-02-unix.tar.gz

```
[nexususer@centos-s-1-mas-nexus ~]$ sudo tar -xvf nexus-3.0.2-02-unix.tar.gz
```

4. Después se debe cambiar el nombre del archivo a Nexus y ser colocado en la carpeta /app creada anteriormente:

sudo mv Nexus-3.0.2-02 /app/Nexus

```
[nexususer@centos-s-1-mas-nexus ~]$ sudo mv nexus-3.0.2-02 /app/nexus
[nexususer@centos-s-1-mas-nexus ~]$
```

Paso 4: Crear usuario Nexus

1. Como buena práctica de seguridad, no se recomienda ejecutar el servicio Nexus con ningún usuario sudo, por lo que se crea un nuevo usuario llamado Nexus.

sudo adduser Nexus

```
[nexususer@centos-s-1-mas-nexus app]$ sudo adduser nexus
[nexususer@centos-s-1-mas-nexus app]$
```

2. Después se debe cambiar la propiedad del archivo Nexus al usuario Nexus, y para ello se utiliza la siguiente línea de comando:

sudo chown -R Nexus:Nexus /app/Nexus

```
[nexususer@centos-s-1-mas-nexus app]$ sudo chown -R nexus:nexus /app/nexus
[nexususer@centos-s-1-mas-nexus app]$
```

3. Después se debe abrir el archivo /app/Nexus/bin/Nexus.rc para quitar comentario al parámetro run_as_user y configurar de la siguiente manera:

sudo vi /app/Nexus/bin/Nexus.rc

```
[nexususer@centos-s-1-mas-nexus app]$ sudo vi  /app/nexus/bin/nexus.rc
```

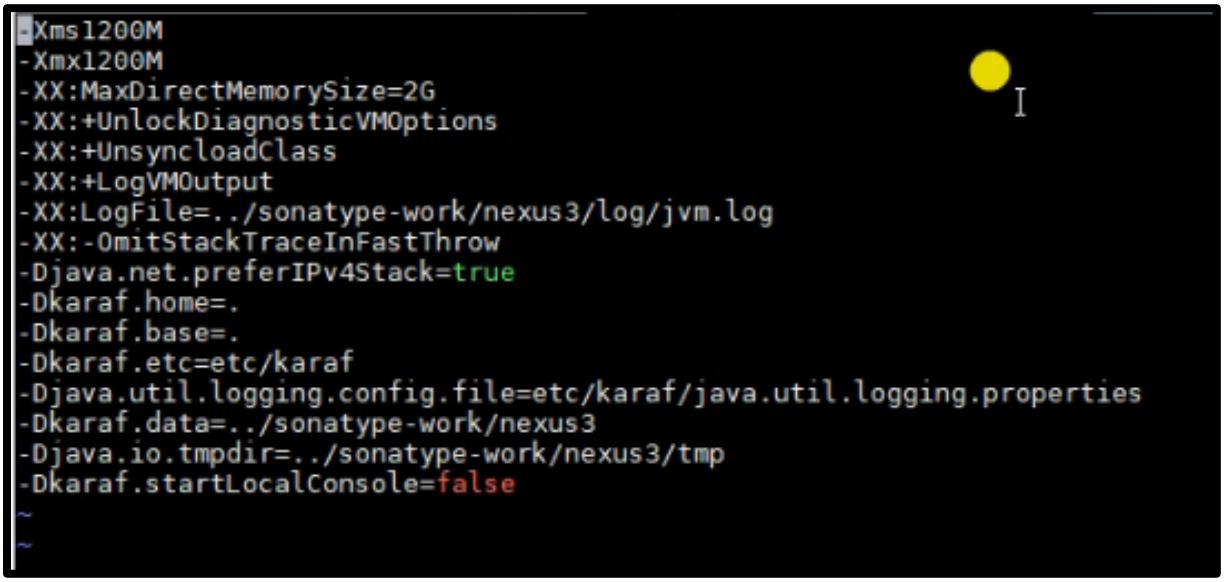
run_as_user="Nexus"

```
run_as_user="nexus"
~
```

Paso 5: Directorio de datos

1. Si desea cambiar el directorio de datos Nexus predeterminado, abra el archivo de propiedades de Nexus y cambie el parámetro "*-Dkaraf.data*" del directorio de datos a una ubicación preferida como se muestra a continuación:

sudo vi /app/Nexus/bin/Nexus.vmoptions



Paso 6: Ejecutar Nexus como un servicio

Es mejor tener una entrada init.d para administrar el servicio de Nexus usando el comando de servicio de Linux. Siga los pasos que se indican a continuación para la configuración.

1. Crear un enlace simbólico para el script de servicio Nexus a la carpeta /etc/init.d.

sudo ln -s /app/Nexus/

```
[nexususer@centos-s-1-mas-nexus app]$ sudo ln -s /app/nexus/bin/nexus /etc/init.d/nexus
[nexususer@centos-s-1-mas-nexus app]$
```

2. Ejecutar los siguientes comandos para agregar el servicio de Nexus al inicio.

sudo chkconfig --add Nexus

```
[nexususer@centos-s-1-mas-nexus app]$ sudo chkconfig --add nexus
[nexususer@centos-s-1-mas-nexus app]$
```

sudo chkconfig --levels 345 Nexus on

```
[nexususer@centos-s-1-mas-nexus app]$ sudo chkconfig --levels 345 nexus on
[nexususer@centos-s-1-mas-nexus app]$
```

3. Para iniciar el servicio Nexus en el puerto 8081, se utiliza el siguiente comando:

sudo service Nexus start

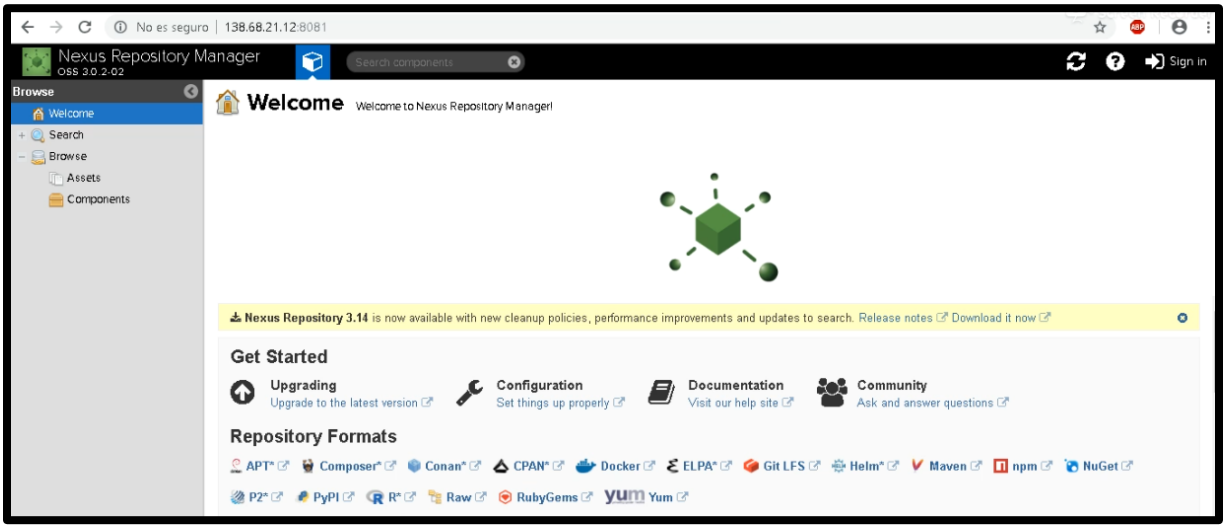
```
[nexususer@centos-s-1-mas-nexus app]$ sudo service nexus start
Starting nexus
[nexususer@centos-s-1-mas-nexus app]$
```

4. También se puede verificar el estado del servicio de la siguiente manera:

sudo service Nexus status

```
[nexususer@centos-s-1-mas-nexus ~]$ sudo service nexus status
Last login: Wed Oct 31 21:17:28 UTC 2018 on pts/0
nexus is running.
```

5. Para acceder al panel de control de Nexus, ingresar a http://localhost:8081. Podrá ver la página de inicio de Nexus como se muestra a continuación:



Las credenciales por defecto de administrador son usuario "*admin*" y contraseña "*admin123*".

Nota: Para ingresar al servidor desde otro sitio, se debe agregar el puerto 8081 al firewall.

Integración de Herramientas

Antes de comenzar

Para la fase de integración entre las herramientas, primeramente se debe tener completa certeza de la comunicación a nivel de red y de puertos de los componentes de los servidores. También es necesario tener una cuenta en Oracle para la descarga del JDK.

Paso 1. Repositorio de artefactos (Nexus) y Construcción automatizado (Jenkins)

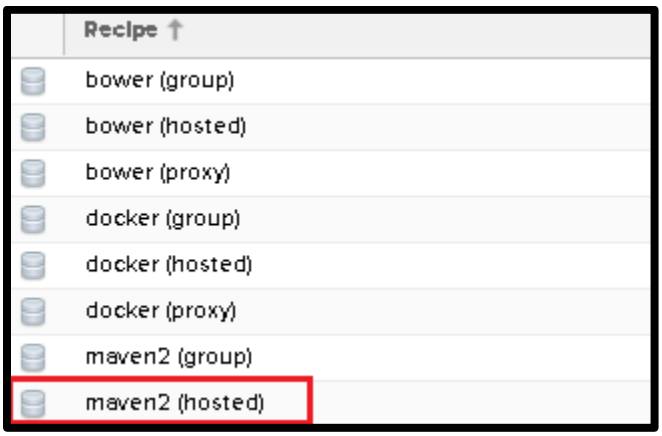
Primeramente, para el uso de Nexus, se debe crear un repositorio en dicho servidor de la forma siguiente:

1. Iniciar sesión con usuario administrador.

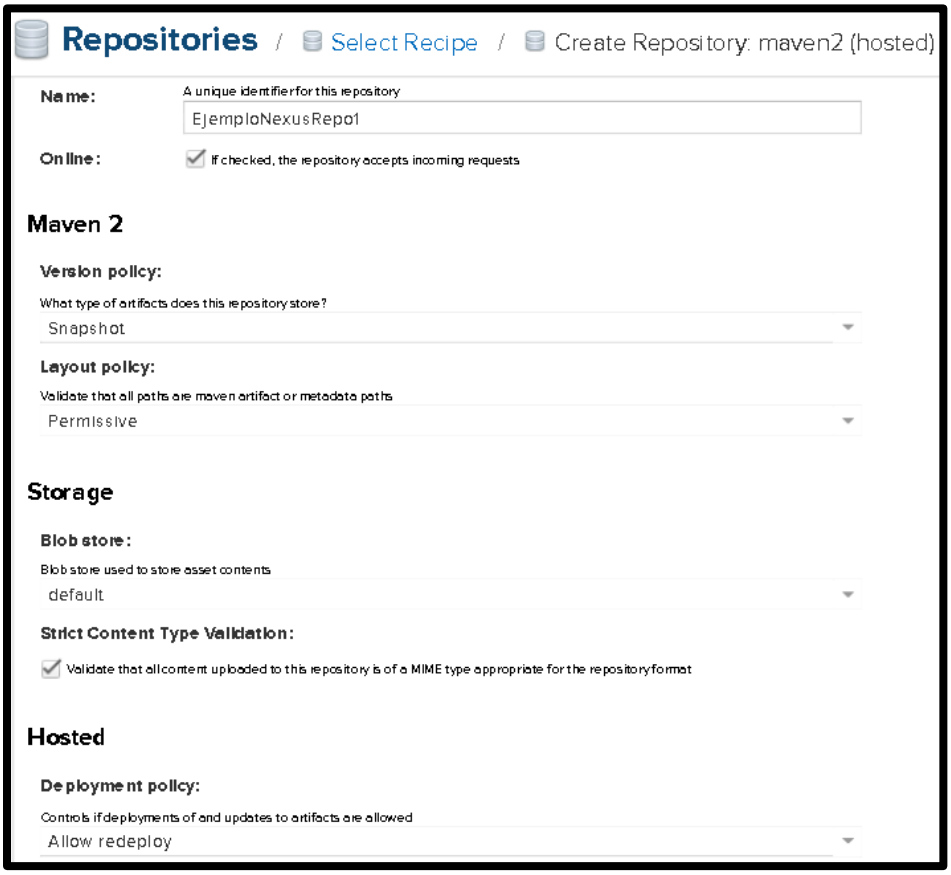
2. Luego ir a la opción "*Administration*".

3. Dirigirse a "Repository" y hacer clic en "*Create repository*":

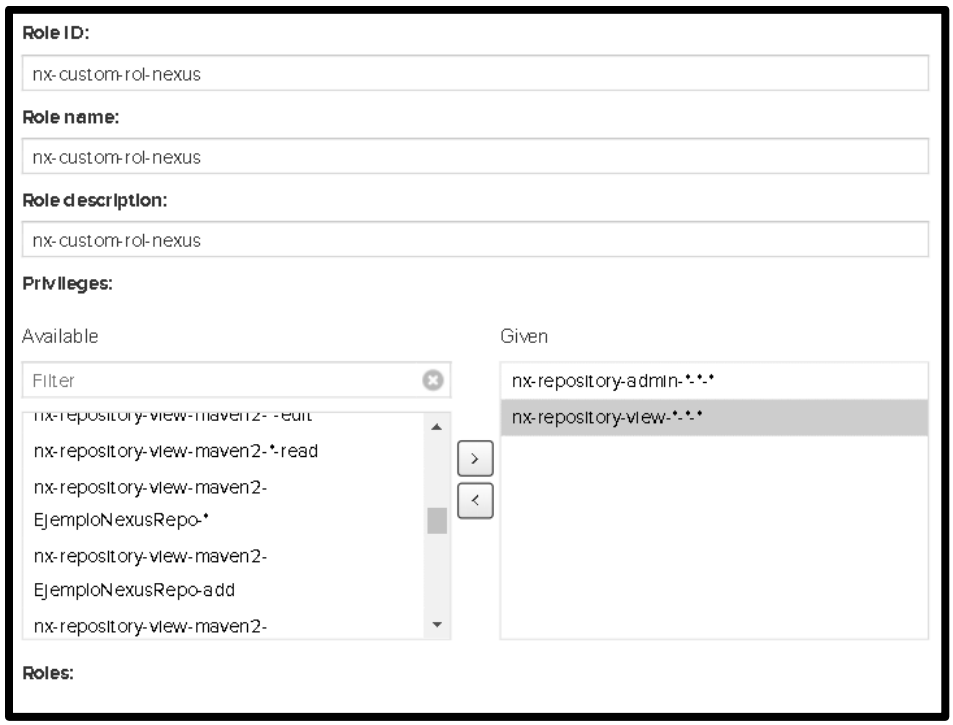
4. Seleccionar maven2 (hosted)



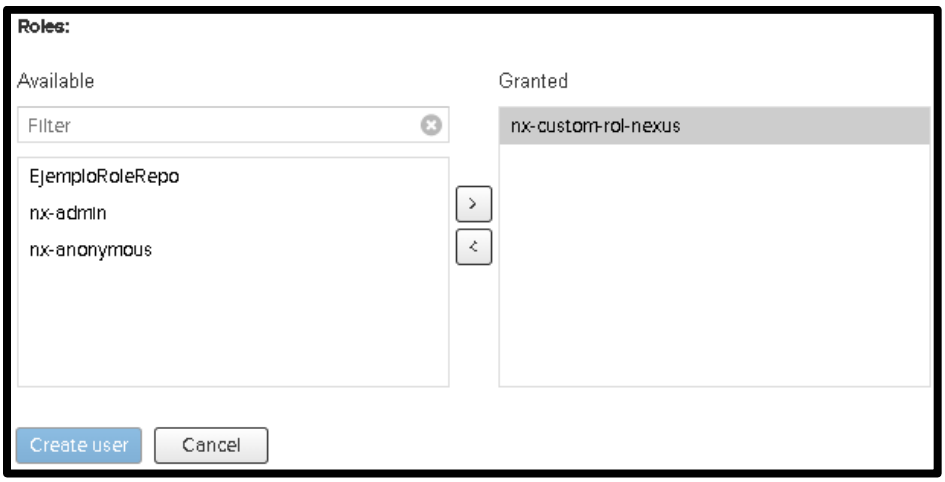
5. Crear el repositorio de la manera que se describe a continuación:



6. Crear un rol y agregar los privilegios "*nx-repository-admin*" y "*nx-repository-view*".



7. Es necesario crear un usuario en Nexus y asignarle el rol creado anteriormente:



8. Si no existe el archivo settings.xml de Maven ubicado en la carpeta "*.m2*", se crea, o si en efecto existe, se modifica. Se colocan las credenciales del usuario de Nexus, así como también el repositoryId del repositorio creado.

<servers>
<server>
<id>repositoryId</id>
<username>userNexus</username>
<password>pass</password>
</server>
</servers>

Como ejemplo se coloca lo siguiente:

```
<servers>
<server>
  <id>EjemploNexusRepo</id>
  <username>nexususer</username>
  <password>User123!</password>
</server>
</servers>
```

9. Se crean los archivos compilados pertinentes y se suben al repositorio de Nexus mediante el siguiente comando a través de una consola de comandos:

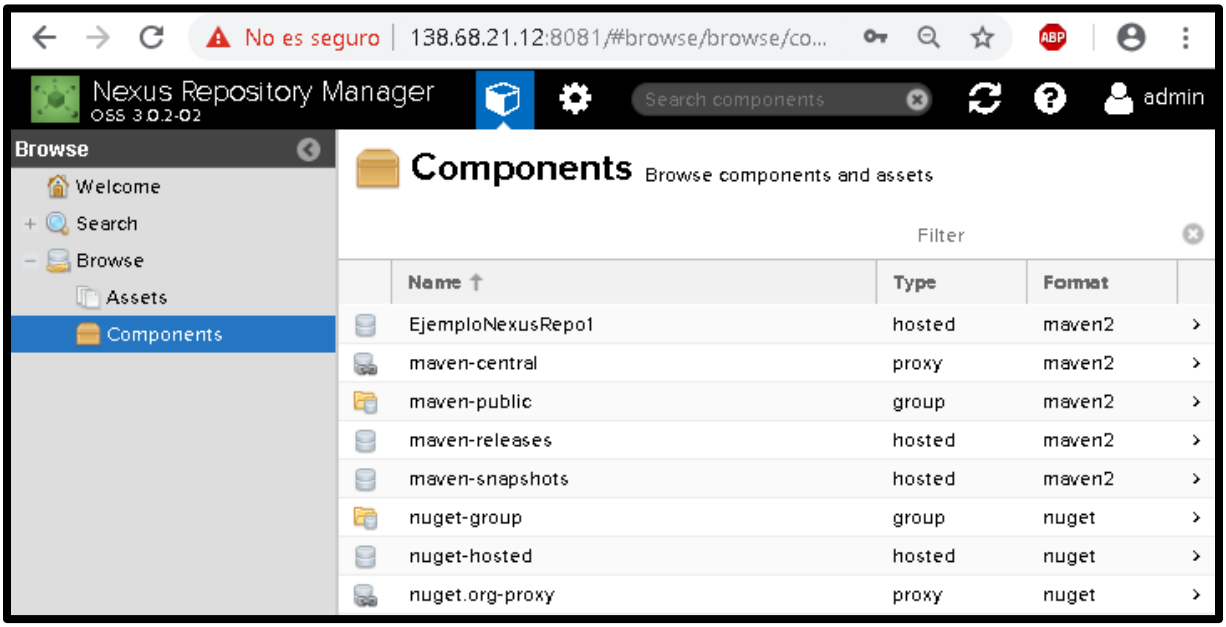
mvn deploy:deploy-file -DgroupId=<groupid> -DartifactId=<artifactid> -Dversion=<version> -Dpackaging=<packaging> -Dfile=<ubicación y nombre del archivo> -DrepositoryId=<repositoryId> -Durl=<URL del repositorio>

Como ejemplo se coloca lo siguiente:

```
C:\>mvn  deploy:deploy-file -DgroupId=com.mas -DartifactId=mas2 -Dversion=1.1.0.0-SNAPSHOT
 -Dpackaging=jar -Dfile=C:\Users\W2BFlores\Documents\MAS\EjemploNexus2.jar -DrepositoryId=
EjemploNexusRepo -Durl=http://138.68.21.12:8081/repository/EjemploNexusRepo1
```

```
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
                                                        [INFO] Total time: 9.345 s
[INFO] Finished at: 2018-10-31T19:01:26-06:00
[INFO] ------------------------------------------------------------------------
```

10. Después de eso, el artefacto ya se podrá ver desde la consola de Nexus en la parte de componentes:



11. Para hacer uso del artefacto recién creado, se añade el repositorio al archivo "*pom.xml*" del proyecto trabajado, de la siguiente manera:

<repositories>

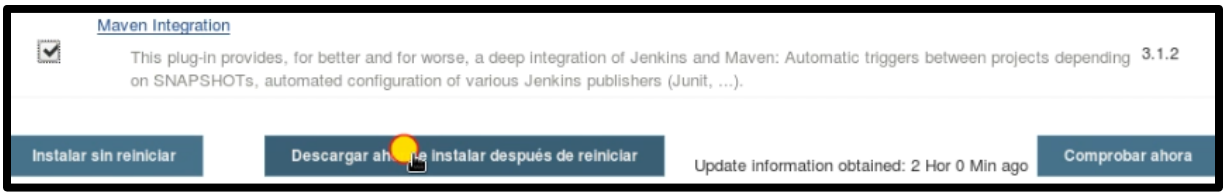
<repository>

```
    <id>EjemploNexusRepo</id>
    <name>EjemploNexusRepoName</name>
    <url>http://138.68.21.12:8081/repository/EjemploNexusRepo1</url>
  </repository>
</repositories>
```

12. Jenkins mediante el archivo "*pom.xml*" el proyecto, reconocerá inmediatamente el repositorio que se ha creado.

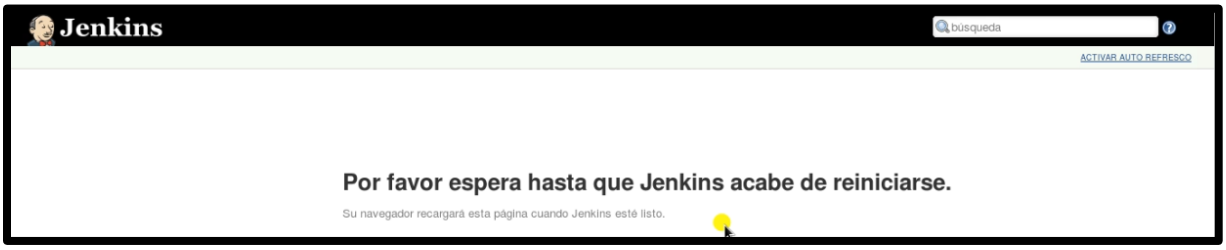
Paso 2. Construcción (Maven) y Servidor de Integración (Jenkins)

1. Ingresar a Jenkins con el usuario administración.
2. Buscar y descargar el complemento de Jenkins llamado "*Maven Integration*" en "*Administrar Jenkins > Administrar Plugins*".
3. Esperar que la descarga e instalación del complemento finalice.



4. Reiniciar el servidor Jenkins, lo recomendable es reiniciarlo de manera segura, colocando /safeRestart en la barra de navegación del explorador:

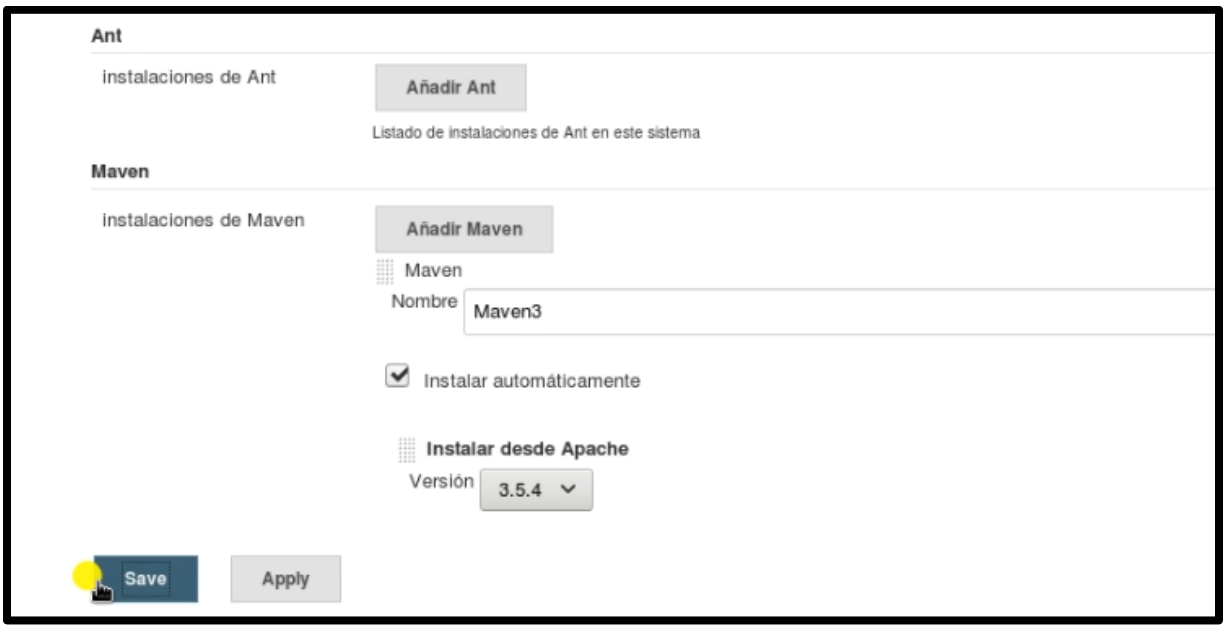
localhost:8080/safeRestart



5. En el servidor Jenkins, en la opción "*Administrar Jenkins > Global Tool Configuration*", se debe añadir Java al servidor de Jenkins.

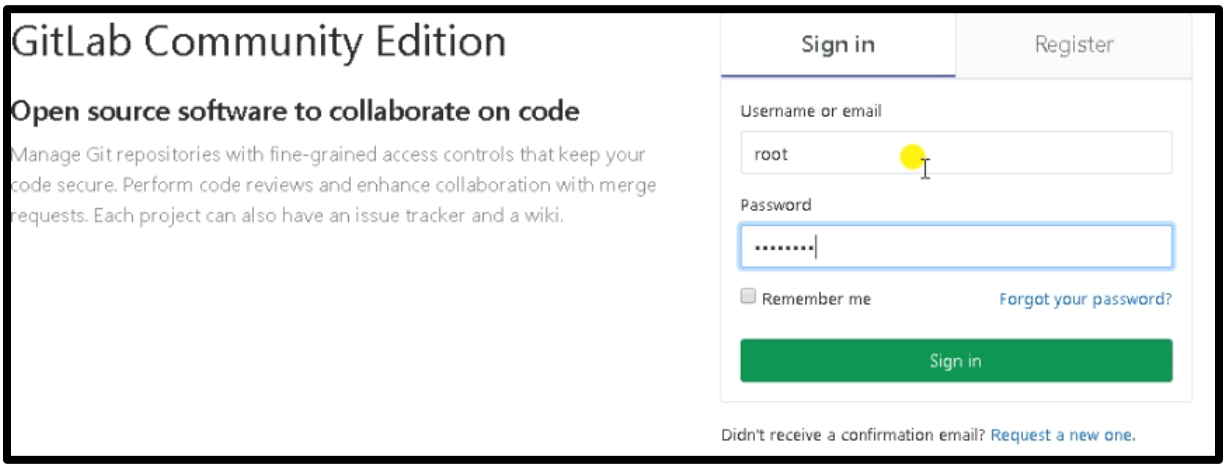
6. Ingresar una cuenta válida de Oracle:

7. Configurar el complemento de Maven y guardar:

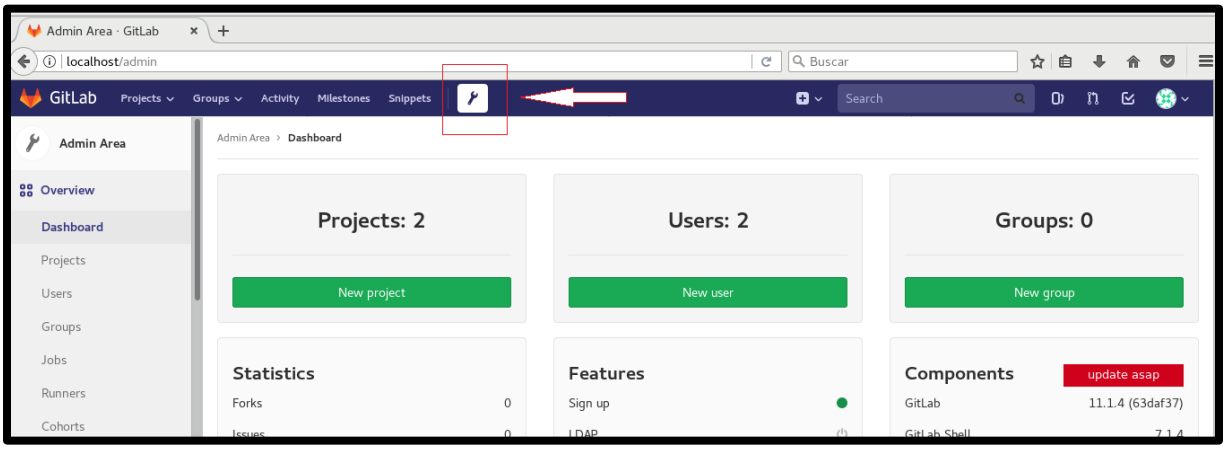

Ant
instalaciones de Ant
Añadir Ant
Listado de instalaciones de Ant en este sistema
Maven
instalaciones de Maven
Añadir Maven
Maven
Nombre
Maven3
Instalar automáticamente
Instalar desde Apache
Versión
3.5.4
Save
Apply

Paso 3. Administrador de repositorios (GitLab) y Servidor de integración (Jenkins)

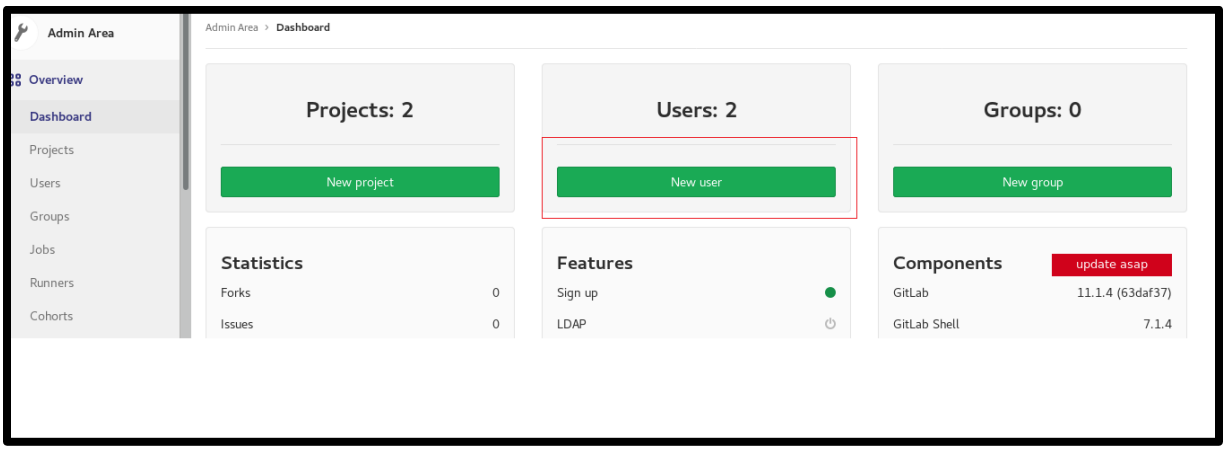
1. Ingresar al servidor GitLab utilizando el usuario "*root*":



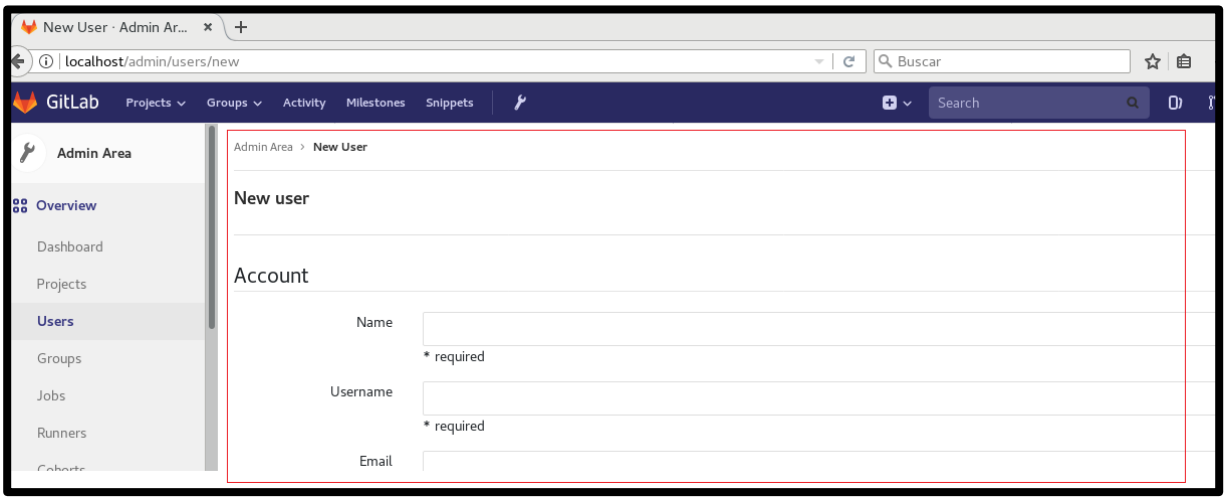
2. Ir a la opción "*Área Administrativa*", identificado con el botón de herramienta, para crear un usuario administrador:



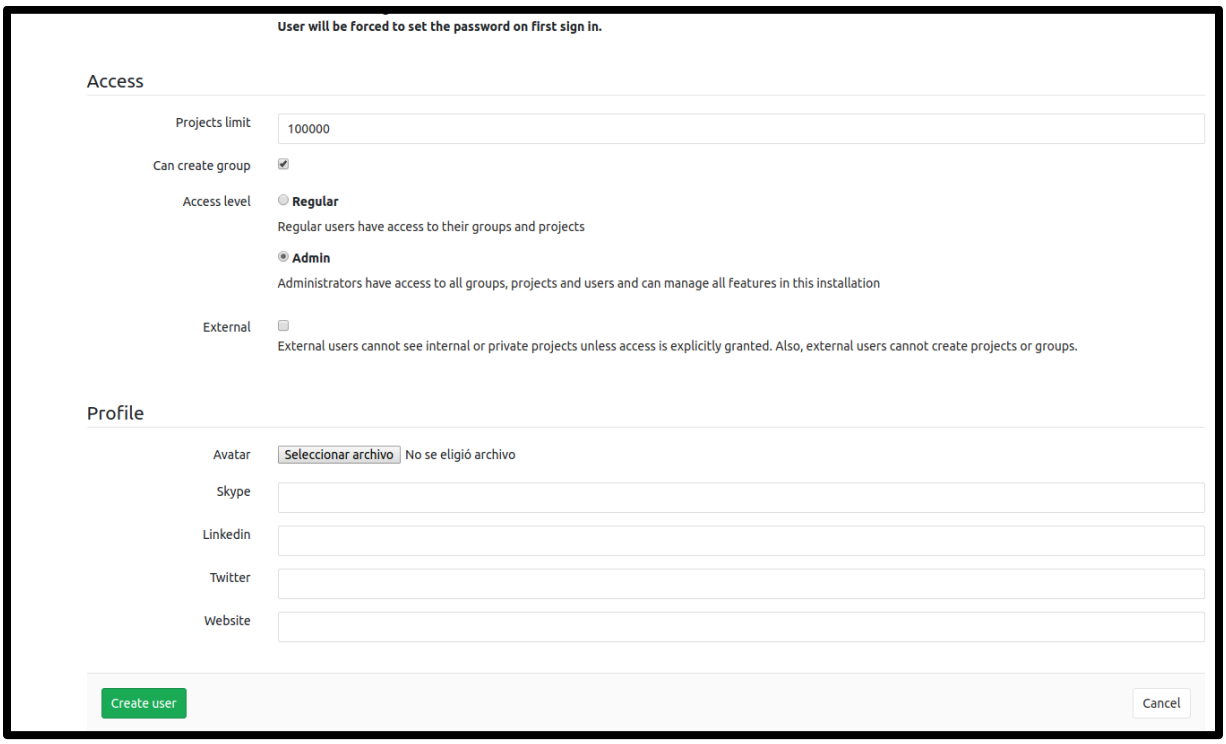
3. Hacer clic en "*New User*":



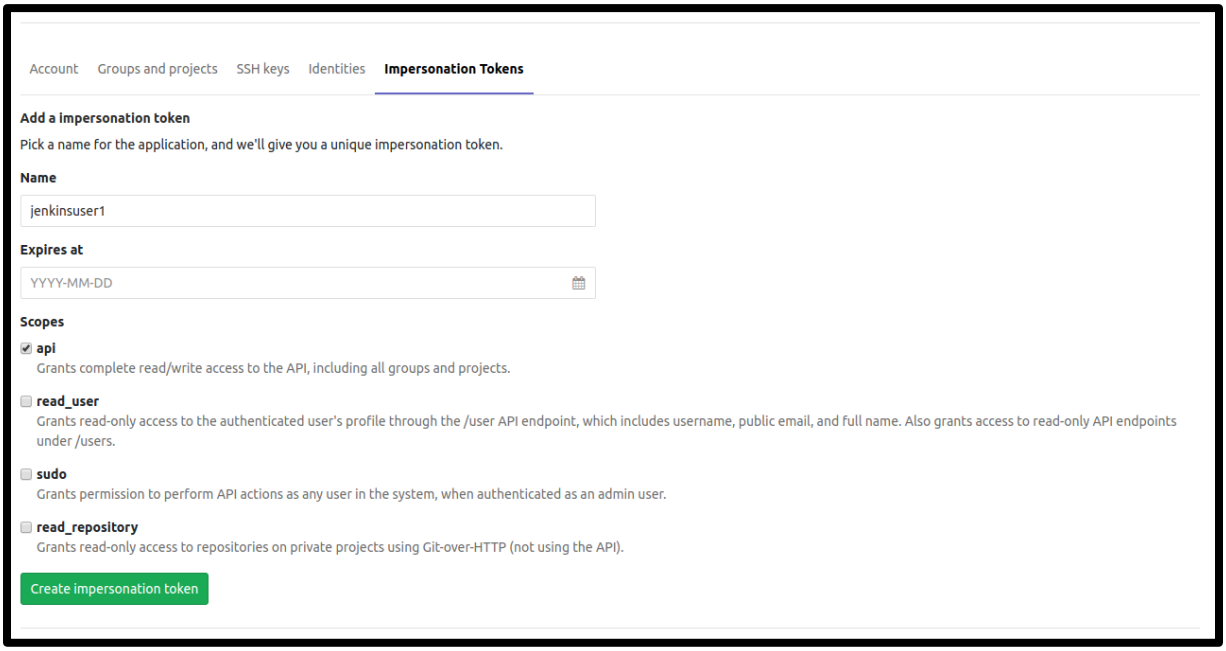
4. Completar el formulario:



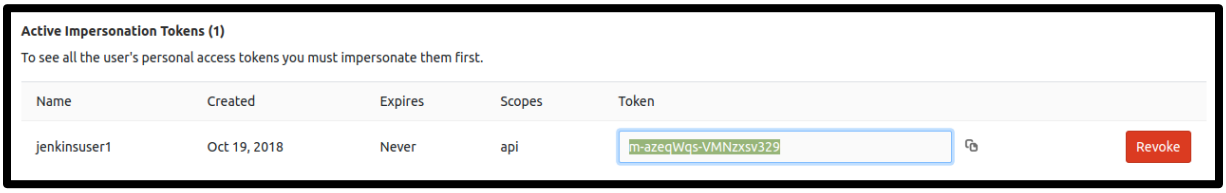
5. Se procede a la creación del usuario con nivel de acceso Admin y se clic en "*Create user*":



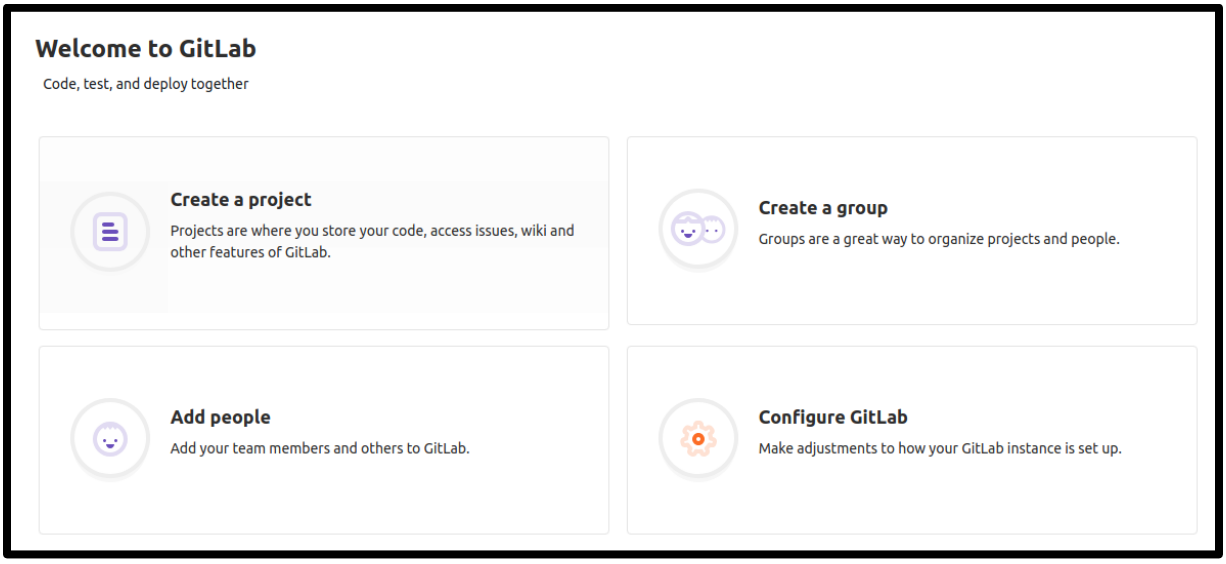
6. Ir a la opción "*Impersonation Tokens*" dentro de la configuración del usuario, para crear un "*Token*" de autenticación con alcance "*api*", permitiendo acceso completo, y fecha de caducidad indefinida:



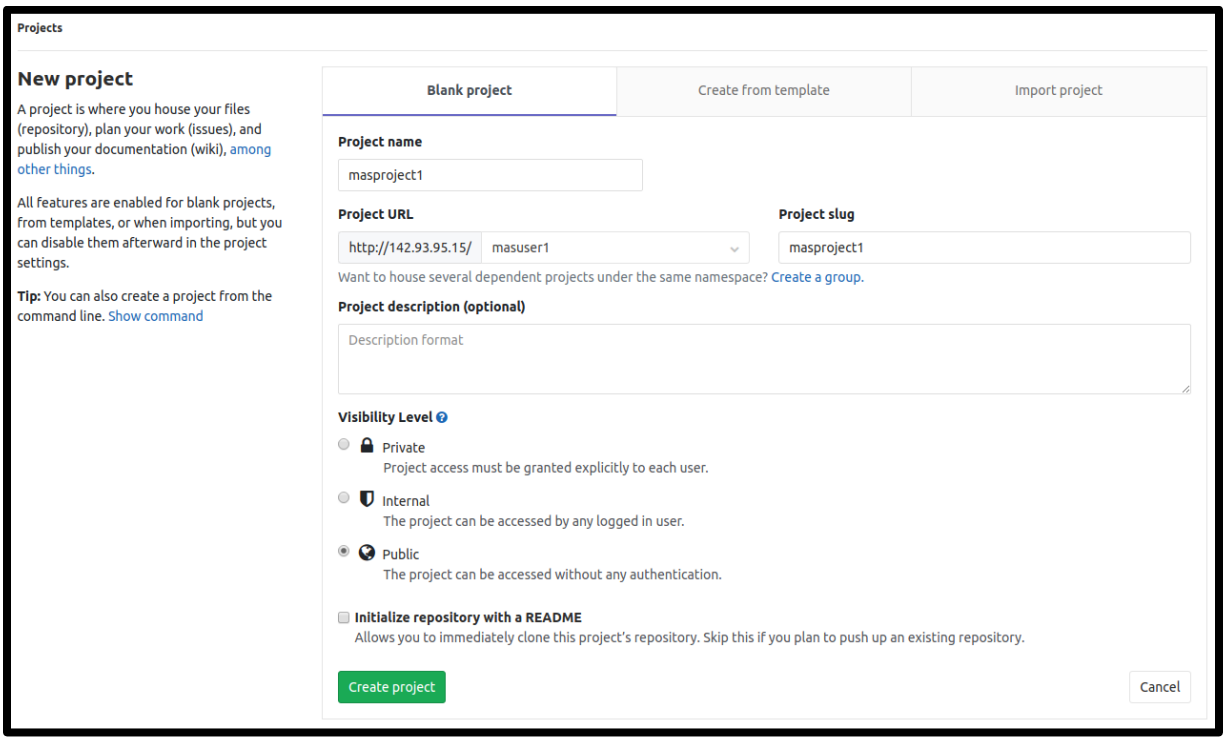
7. Crear, copiar y guardar el "*Token*", este servirá para la integración con el servidor Jenkins:



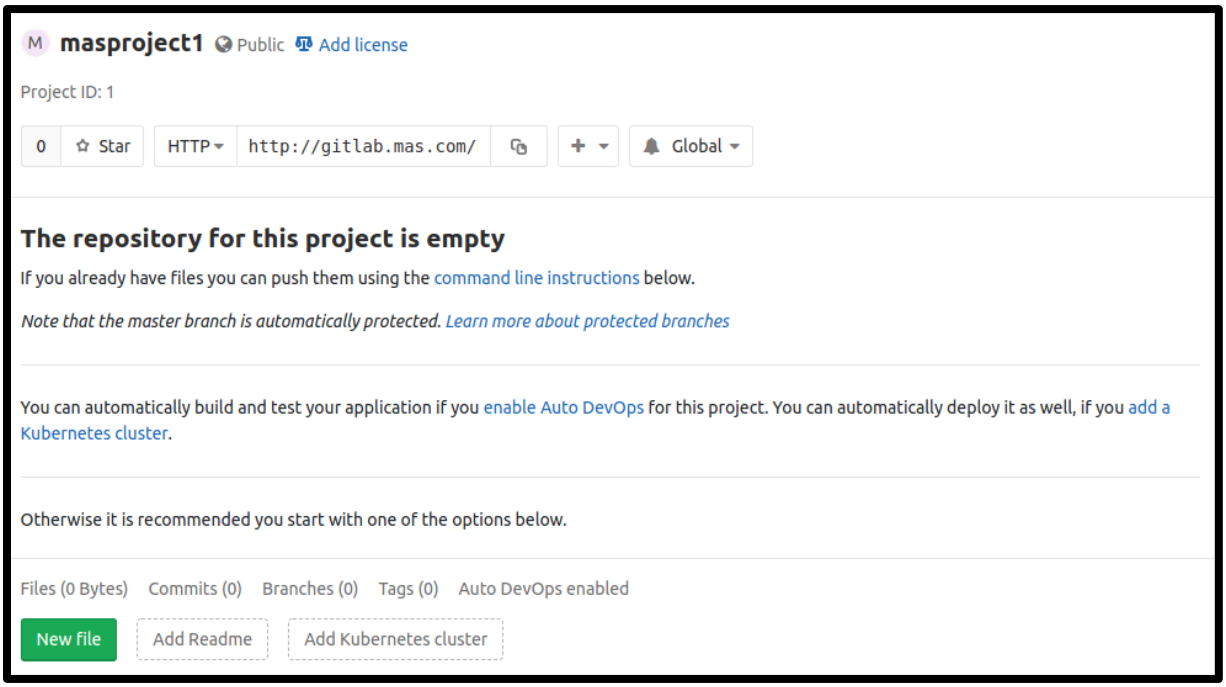
8. Ingresar a GitLab con el usuario administrador y crear un proyecto nuevo:



9. En la creación del proyecto se debe ingresar el nombre del proyecto y seleccionar el nivel de visibilidad, para este caso será "*public*":



10. Al darle clic en "*Create project*" se tendrá la instalación del repositorio Git:



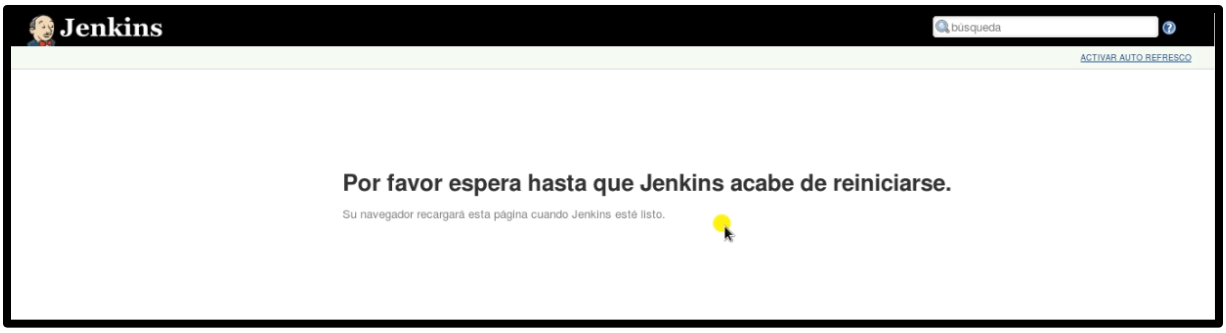
11. Ingresar al servidor Jenkins para colocar los complementos correspondientes para GitLab.

12. Buscar y seleccionar el complemento para GitLab en la opción "*Administrar Jenkins > Administrar Plugins*":

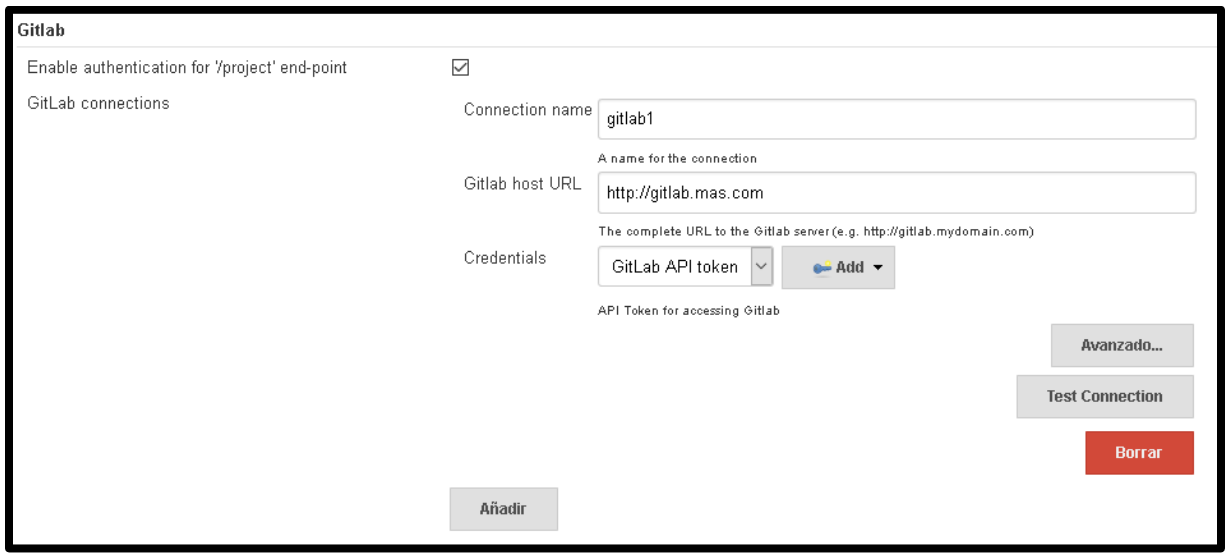
13. Descargar e instalar el complemento:

14. Reiniciar el servidor Jenkins, lo recomendable es reiniciarlo de manera segura, colocando /safeRestart en la barra de navegación del explorador.

localhost:8080/safeRestart

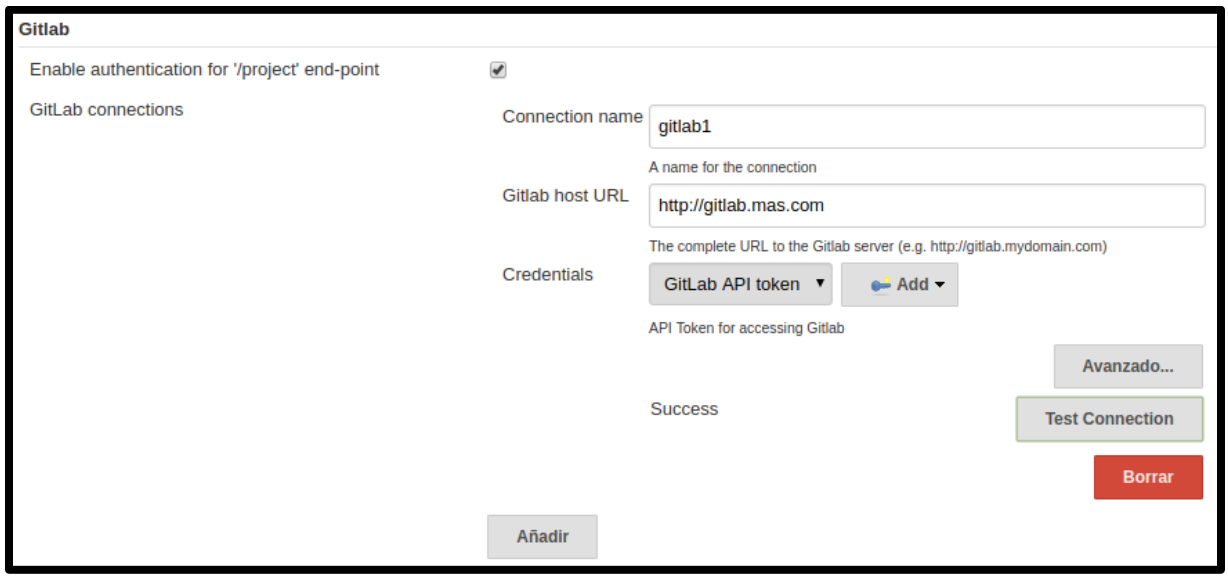


15. En "*Administrar Jenkins > Configurar el Sistema*", se debe añadir la conexión para GitLab, colocando el nombre la URL del servidor y las credenciales correspondientes:

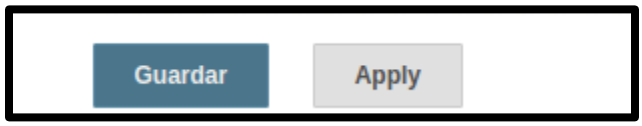


16. Colocar el "*Token*" generado en GitLab:

17. Probar conexión con GitLab:



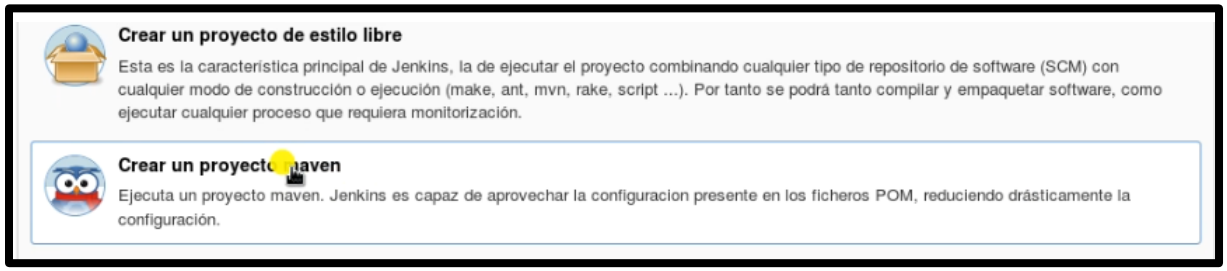
18. Luego darle clic en "*Apply*" y "*Guardar*" los cambios en la parte inferior:



Paso 4. Crear proyecto en Jenkins

1. Ingresar a Jenkins con usuario administrador.
2. Crear un proyecto nuevo:

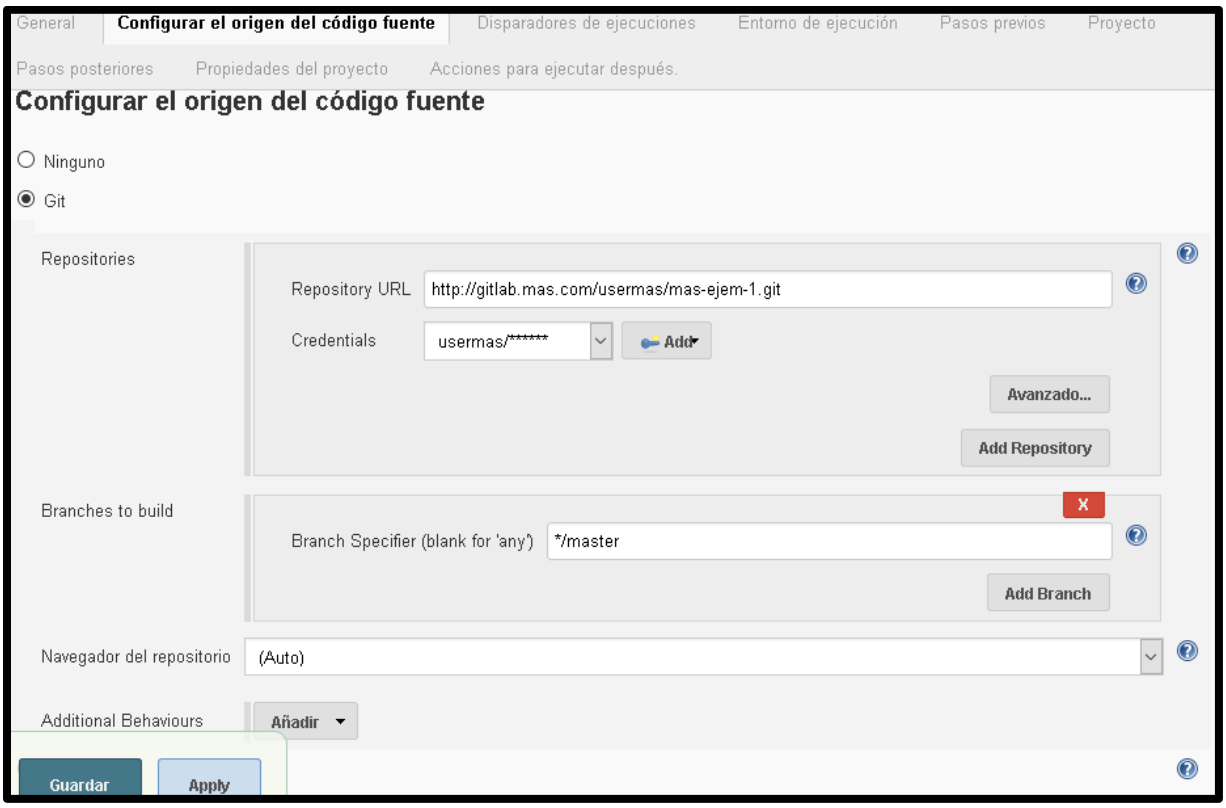
3. Seleccionar "*Crear un proyecto maven*":



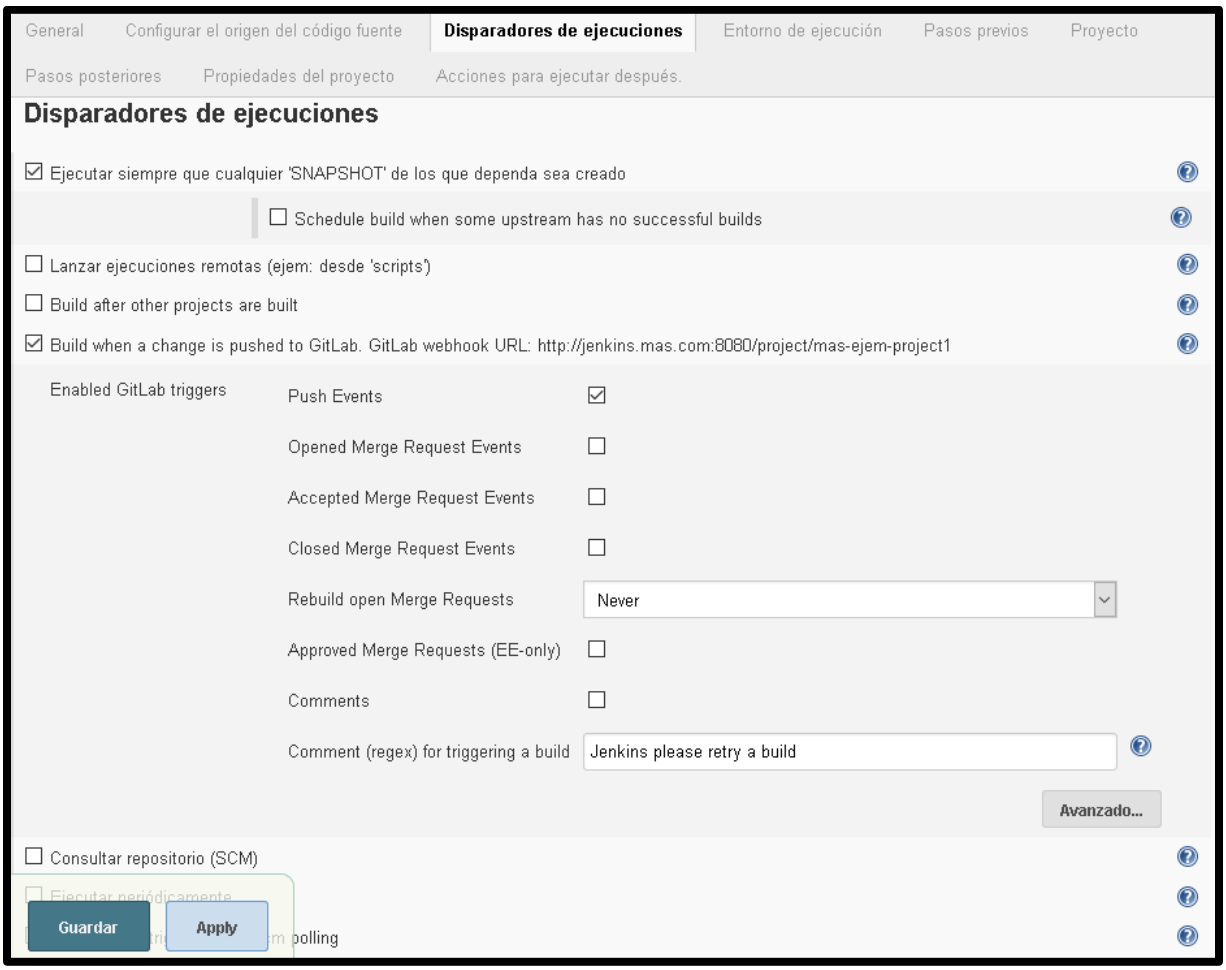
4. Hacer clic en "*ok*" en la parte inferior:

5. En la opción configurar, en la pestaña Propiedades del Proyecto, se debe especificar la conexión de GitLab.

6. En la pestaña de Configurar el origen del código fuente, es necesario colocar la URL del repositorio de Git del proyecto configurado en GitLab, junto con las credenciales correspondientes y la rama a utilizar:



7. Para ver el uso automático de la construcción de código, en la parte de Disparadores de ejecuciones, se pueden colocar los eventos que desencadenan las ejecuciones, por ejemplo, que se realice la construcción del proyecto en el momento que se realice un "*push*" al servidor GitLab:

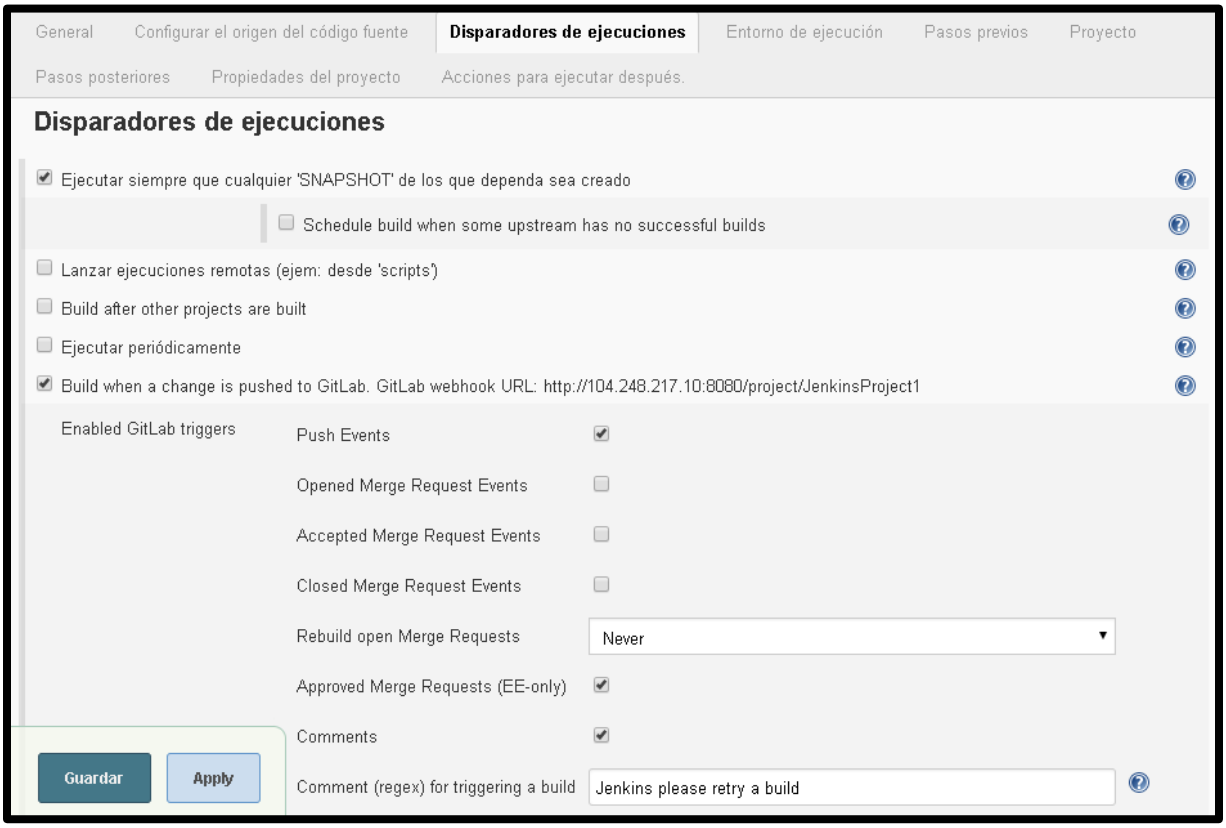


8. En la parte de entorno de ejecución > Proyecto, siempre en la configuración del proyecto, en Fichero POM raíz se debe colocar el nombre del proyecto / la ubicación del archivo pom:

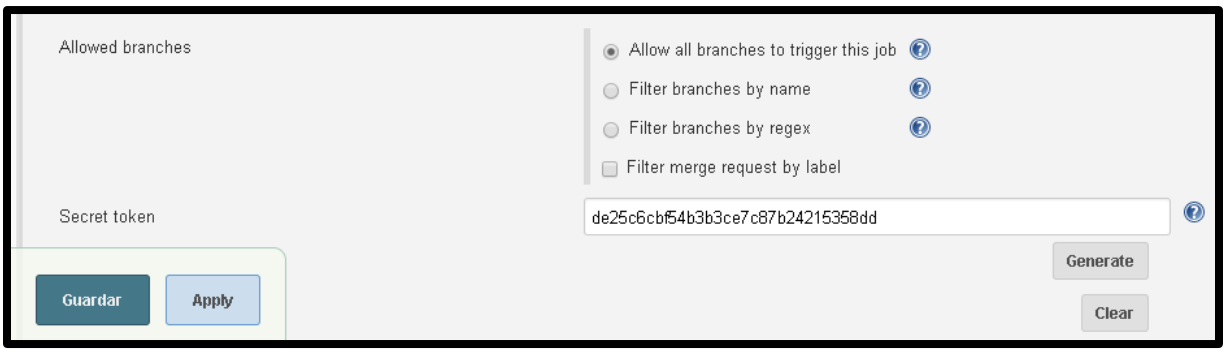
9. Cabe mencionar que el repositorio por defecto es:

/var/lib/Jenkins/workspace/<ProjectJenkins>/

10. Es necesario marcar la opción "*Build when a change is pushed to GitLab. GitLab webhook URL*" en el apartado "Disparadores de ejecuciones" del proyecto de Jenkins, esto servirá para vincular eventos cuando algo está sucediendo dentro del proyecto de GitLab.



11. Dar clic en el botón "*Avanzado*" en el mismo apartado, y generar un *"Token"* secreto:

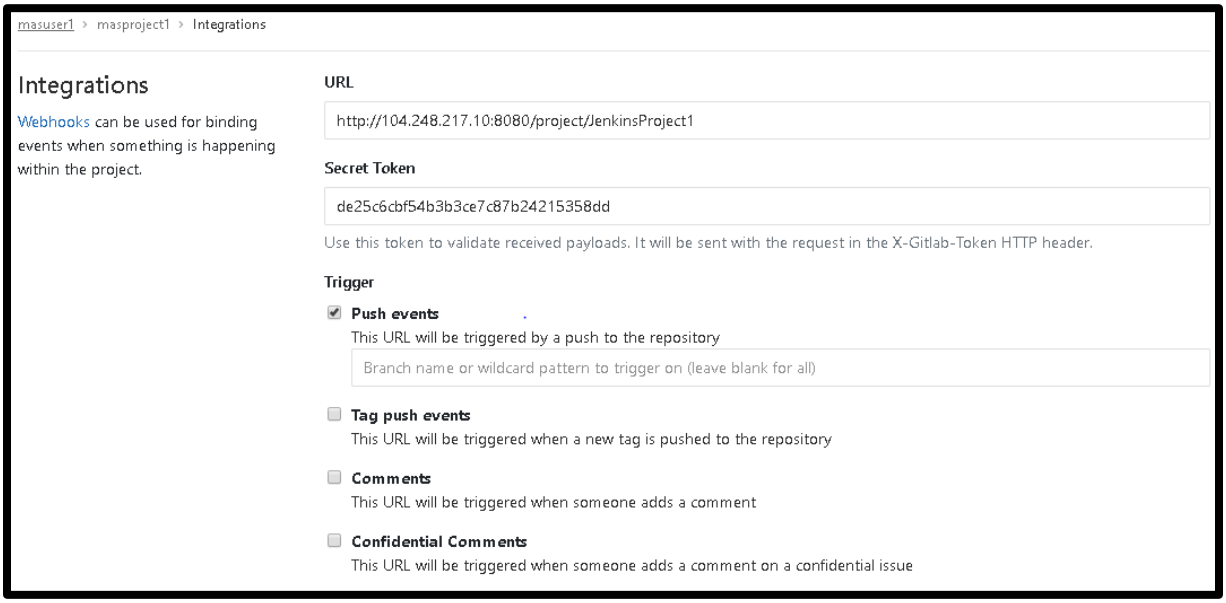


12. Se da clic en "*Apply*" y "*Guardar*" los cambios en Jenkins, para luego continuar con la configuración del *"webhook"* en GitLab.

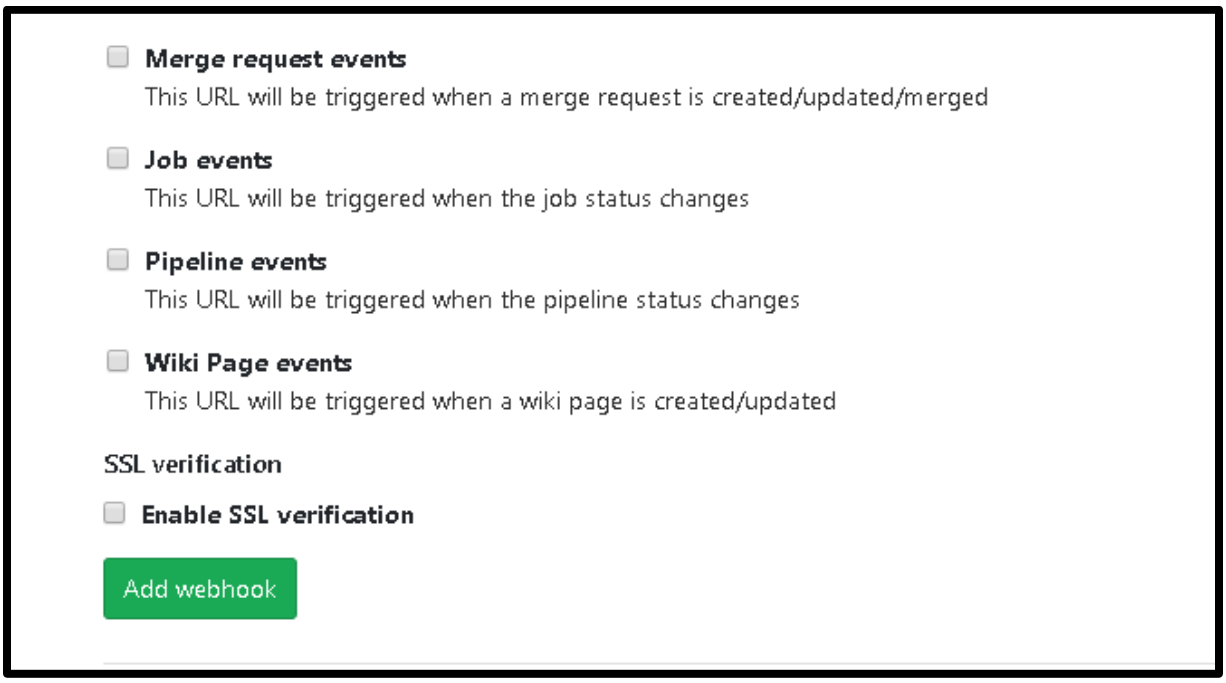
13. En el servidor GitLab se debe de ir al proyecto creado anteriormente.

14. En la opción "*Admin area*", luego dirigirse a Settings > Integrations, en la parte de "*webhooks*" colocar la URL y *"Token"* secreto proporcionado por Jenkins.

15. Seleccionar el disparador "*Push event*":



16. Dar clic en el botón "*Add webhook*":



17. Por último, se puede comprobar la correcta comunicación y configuración del "*webhook*" recién creado, al realizar un test de los eventos:

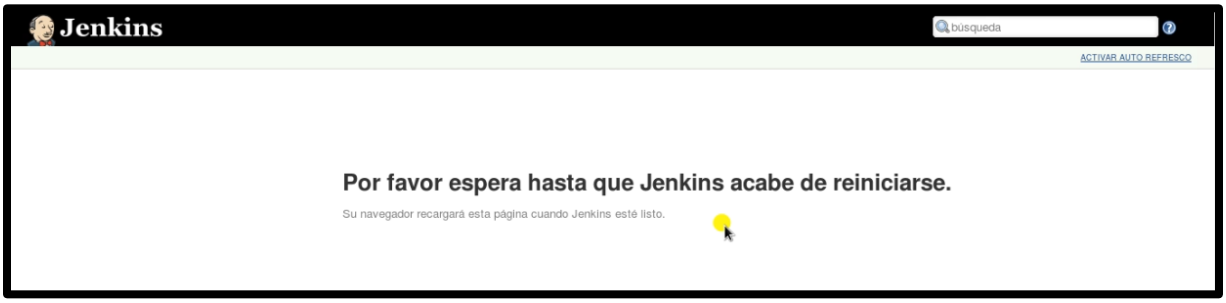
Paso 5. Evaluador de código estático (SonarQube) y Servidor de integración (Jenkins)

1. Buscar el complemento correspondiente para escanear proyectos con SonarQube, para ello hay que dirigirse a la opción de Jenkins: "*Administrar Jenkins> Administrar Plugins*"

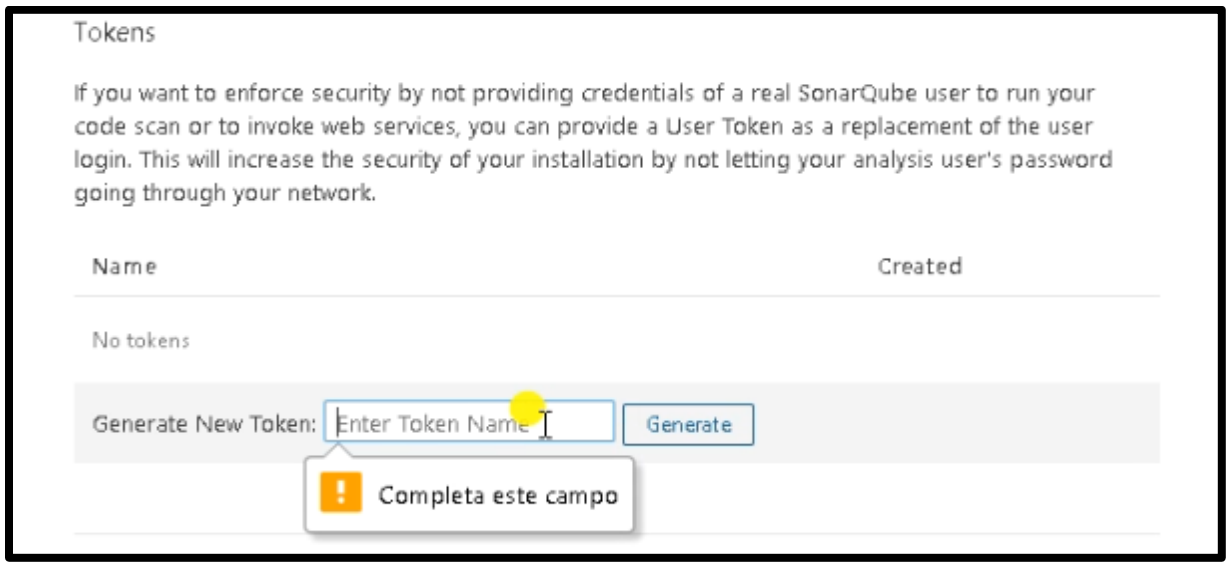
2. Dar clic en descargar e instalar el complemento:

3. Reiniciar el servidor Jenkins, lo recomendable es reiniciarlo de manera segura, colocando /safeRestart en la barra de navegación del navegador.

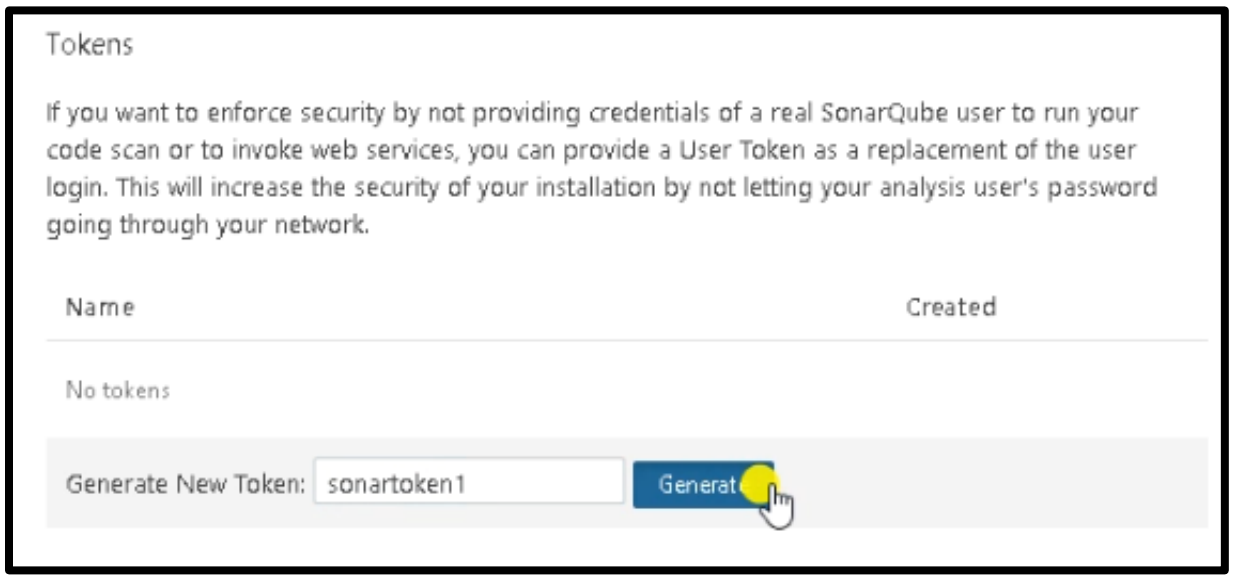
localhost:8080/safeRestart



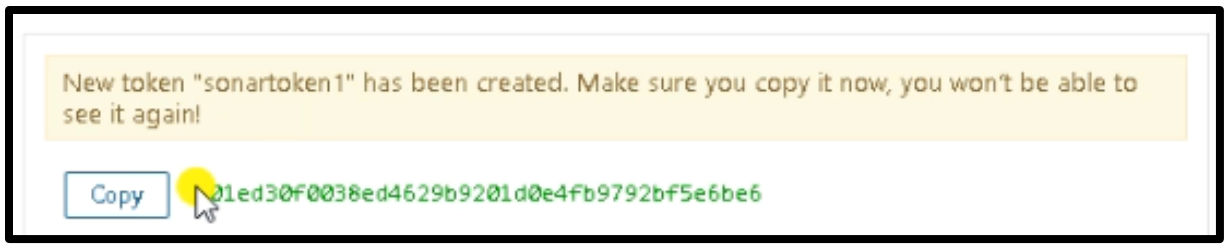
4. Es necesario ingresar al servidor de SonarQube y en el usuario creado luego de la instalación, se debe de agregar un *"Token"* en la opción security > Token:



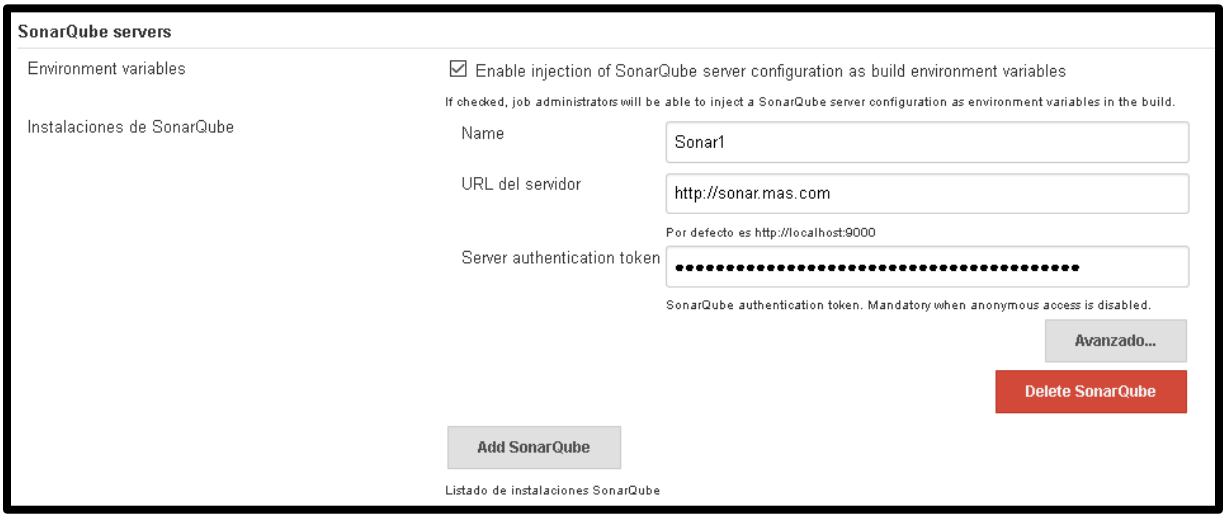
5. Dar clic en "*Generate*":



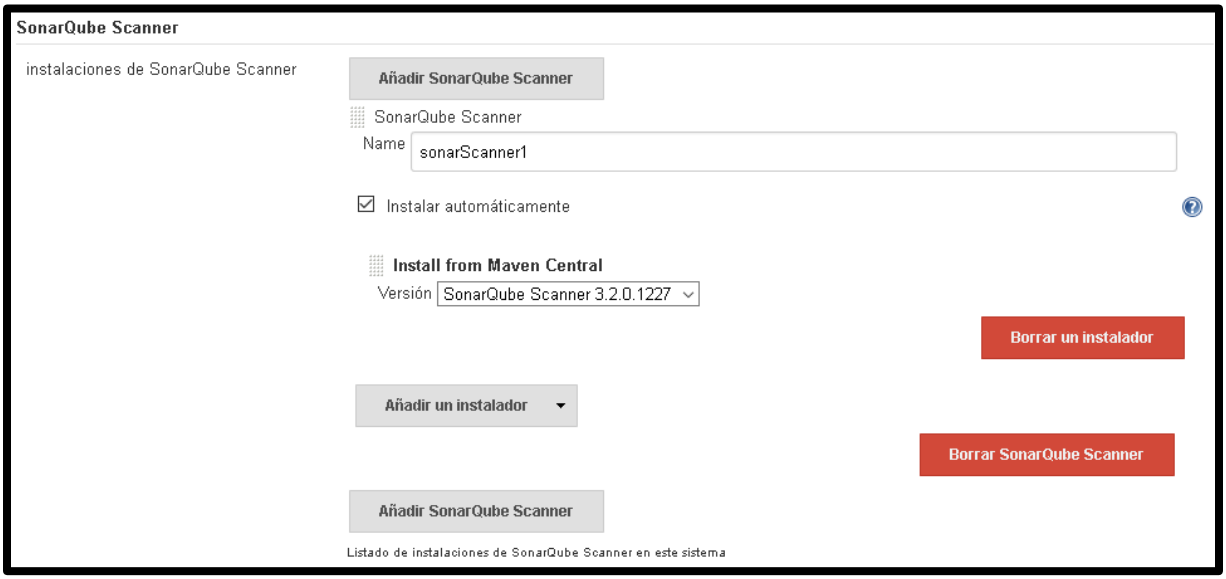
6. Guardar el *"Token"* generado.



7. Ingresar al servidor en Jenkins, para lo cual se debe de ir a "*Administrar Jenkins > Configurar el Sistema*" y en la sección de SonarQube ingresar la siguiente configuración:



8. Configurar el servidor Jenkins para SonarQube en: "*Administrar Jenkins > Global Tool Configuration*":



9. Dentro del proyecto de Jenkins añadir como paso previo la ejecución de SonarQube Scanner, seleccionando la opción "*Execute SonarQube Scanner*".

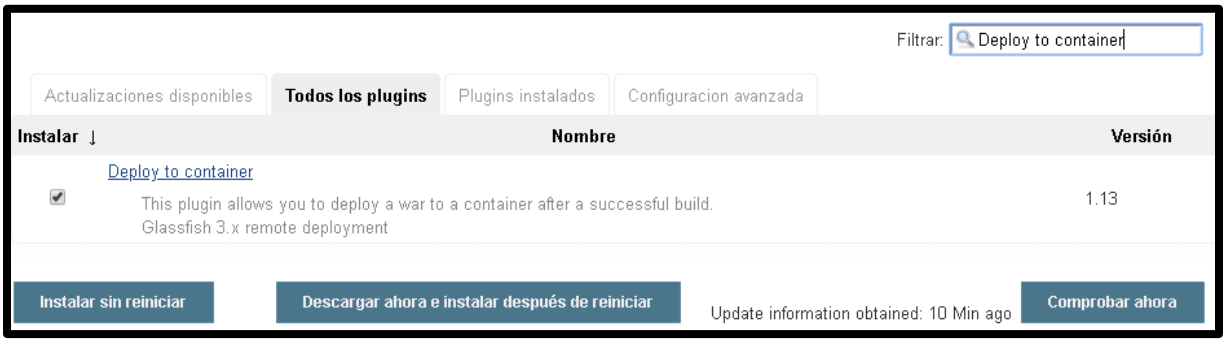
10. Configurar la ejecución de la siguiente manera como mínimo:


Execute SonarQube Scanner
Task to run
JDK
Java8
JDK to be used for this SonarQube analysis
Path to project properties
Analysis properties
sonar.projectKey=un
sonar.sources=.
Additional arguments
JVM Options
Añadir un paso previo

Automatización de pruebas con Servidor de integración

Paso 1. Servidor de integración (Jenkins) y Servidor de aplicaciones / contenedor de servlets (Tomcat)

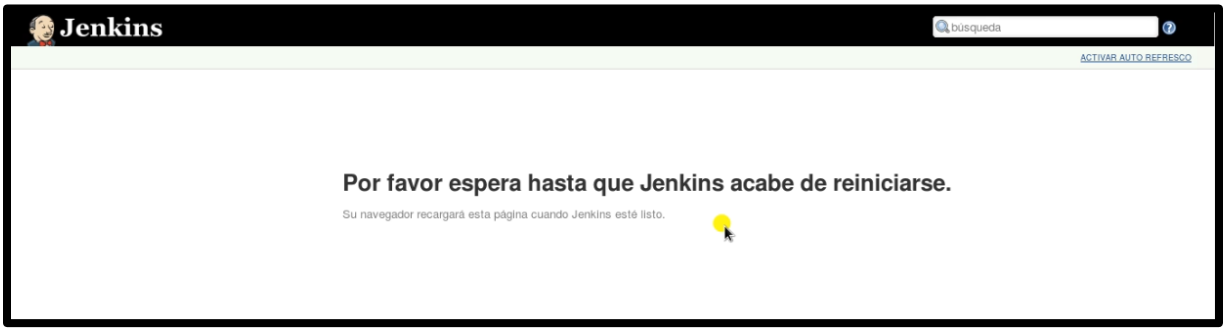
1. Lo primero que se debe hacer para integrar Jenkins con el servidor Tomcat, es buscar complemento de Jenkins llamado "*Deploy to container*" en *"Administrar Jenkins > Administrar Plugins*".



2. Descargar e instalar el complemento.

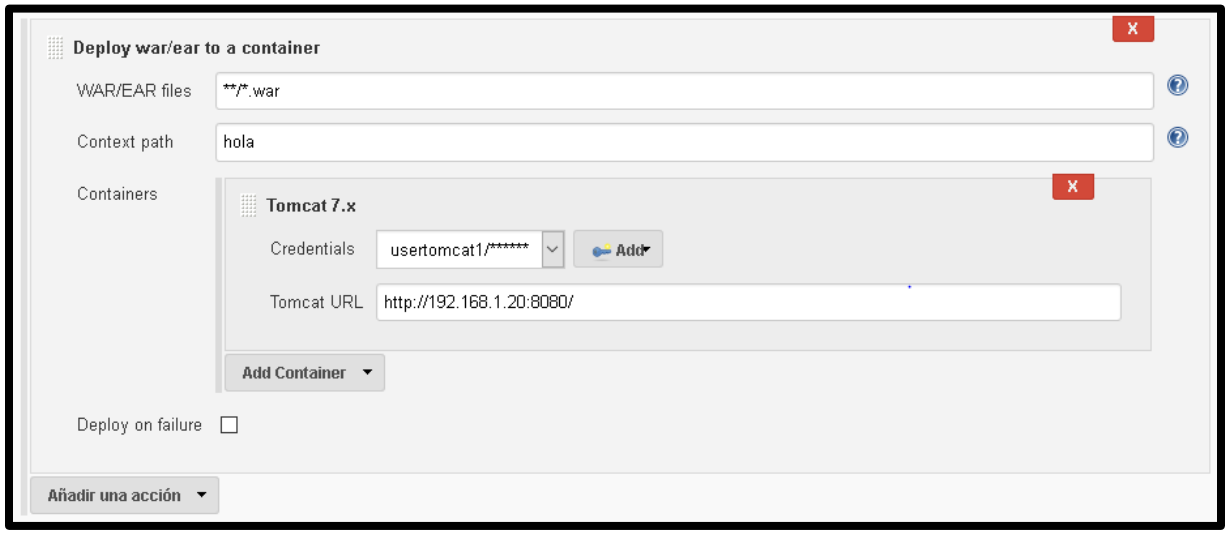
3. Reiniciar el servidor Jenkins, lo recomendable es reiniciarlo de manera segura, colocando /safeRestart en la barra de navegación del navegador.

localhost:8080/safeRestart



4. Ingresar al proyecto creado en Jenkins, y en la opción de "*configurar*", en la parte de "*Añadir una acción*" se debe adicionar "*Deploy war/ear to a container*":

5. Se debe configurar y guardar la ejecución de la siguiente manera como mínimo:



Paso 2. Servidor de Integración (Jenkins), pruebas unitarias y herramienta de automatización de pruebas funcionales y regresión (Selenium)

1. Dentro del servidor Jenkins, crear un directorio para la instalación del driver "*gueckodriver*" para el navegador Mozilla Firefox, ejecutando el siguiente comando:

sudo mkdir /home/geckinstall

2. Posicionar en la carpeta antes creada:

cd /home/geckinstall/

3. Descargar el driver desde la siguiente URL, cabe mencionar que para este caso se utilizará la versión geckodriver-v0.23.0 para Linux 64 bits.

sudo wget https://Github.com/mozilla/geckodriver/releases/download/v0.23.0/geckodriver-v0.23.0-linux64.tar.gz

4. Descomprimir el archivo descargado:

sudo tar -zxvf geckodriver-v0.23.0-linux64.tar.gz

5. Después de tener listo el driver, se debe instalar el navegador Mozilla Firefox, esto en caso de no tenerlo en el servidor Jenkins:

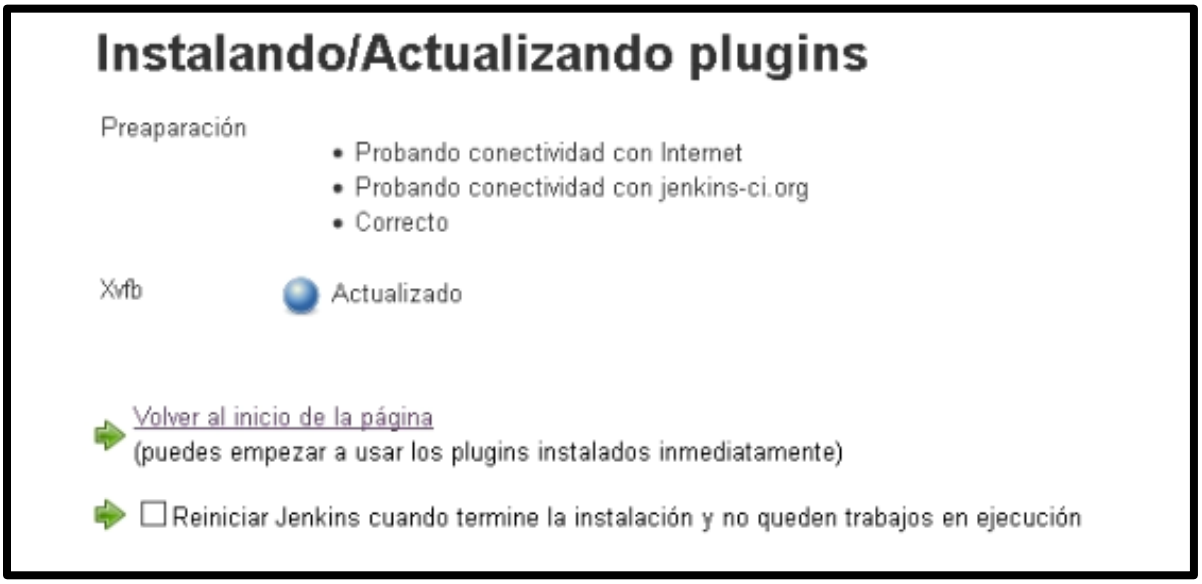
sudo yum install firefox -y

6. Instalar un servidor X11 que emule salidas de video, para este caso se utiliza el Xvfb o también conocido como X virtual framebuffer:

sudo yum install xorg-x11-server-Xvfb

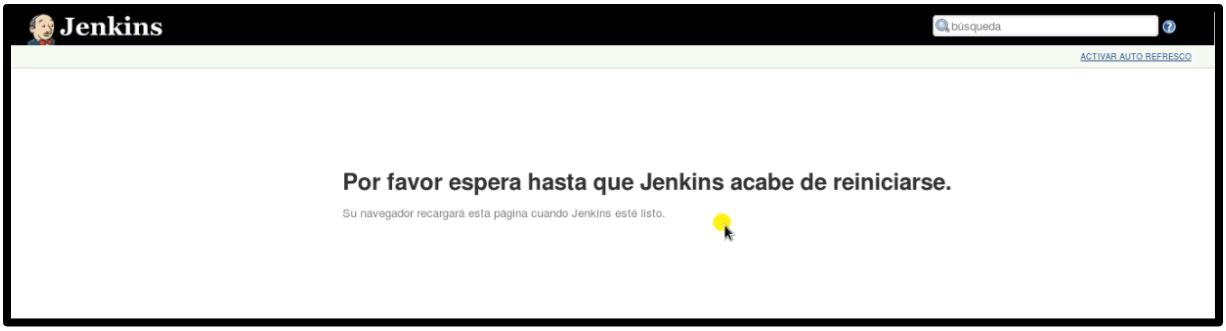
7. Ir a la consola administrativa de Jenkins, en "*Administrar Jenkins* > *Administrar Plugins*", en "*Todos los Plugins*", buscar el complemento correspondiente de Xvfb, en:

8. Descargar e instalar complemento:

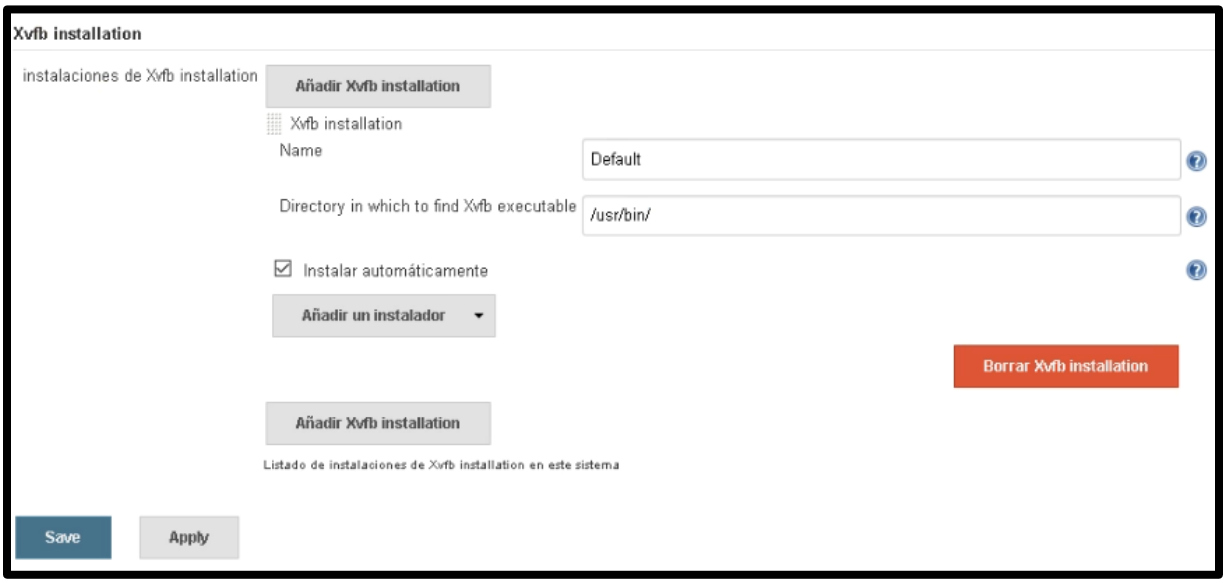


9. Después para asegurarse que todo quede correctamente instalado, se debe reiniciar de manera segura el servidor Jenkins, colocando /safeRestart en la barra de navegación del navegador.

localhost:8080/safeRestart

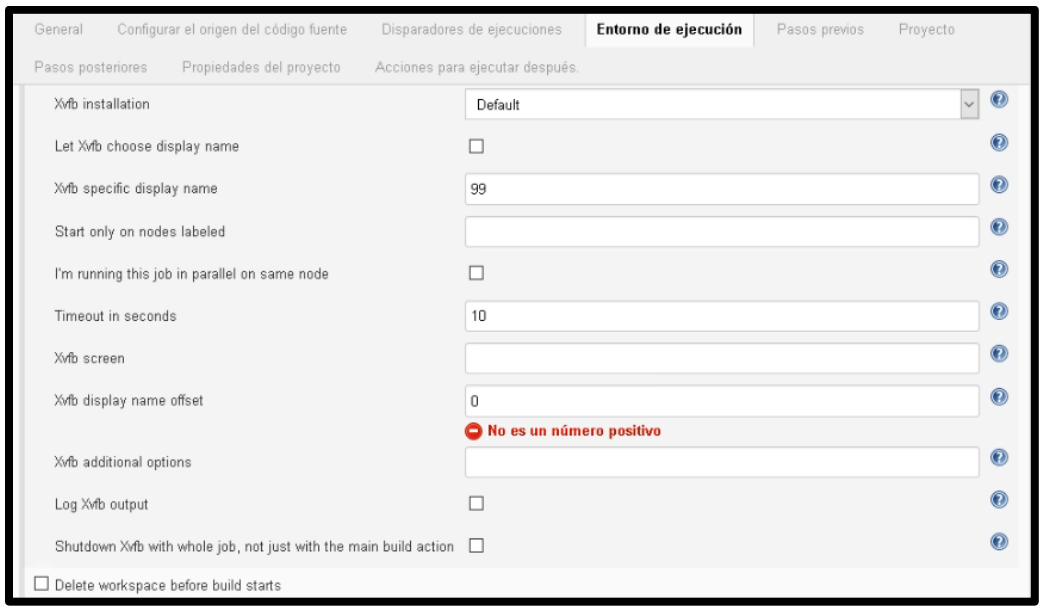


10. Configurar el complemento en el servidor Jenkins por lo que es necesario dirigirse a Jenkins en "*Administrar Jenkins* > *Global Tool Configuration*" y colocar la siguiente configuración:



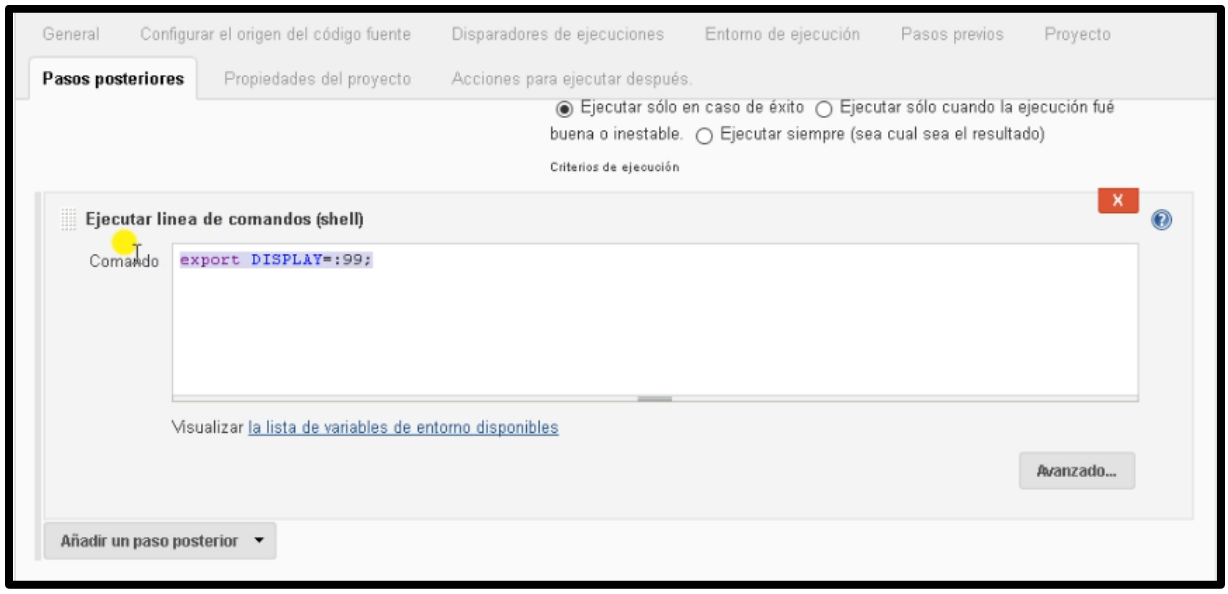
11. Configurar Xvfb en el proyecto de Jenkins, en entorno de ejecución se debe seleccionar "*Start Xvfb before the build, and shut it down after*" y hacer clic en "*Avanzado*":

12. Colocar la siguiente configuración:



13. En "*Añadir un paso posterior*", se debe añadir "*Ejecutar línea de comandos (Shell)*" y colocar:

export DISPLAY=:99;



14. Incluir pruebas unitarias con JUnit y Selenium, añadiendo las siguientes dependencias en el archivo "pom.xml":

<dependencies>

```
...
<dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.12</version>
        <scope>test</scope>
    </dependency>
<dependency>
        <groupId>org.seleniumhq.selenium</groupId>
        <artifactId>selenium-java</artifactId>
        <version>3.5.3</version>
    </dependency>
...
</dependencies>
```

15. Dentro de los directorios de prueba, generar una clase que simule una funcionalidad de la aplicación a desarrollar:

Para los "*import*":

```
import static org.junit.Assert.assertTrue;

import org.junit.Test;

import com.prueba1.app.model.Suma;
```

Para la clase:

```java
public class SumaTest {

    @Test
    public void sumaPositivos() {
        System.out.println("Suma de numeros positivos: ");
        Suma S = new Suma(5, 7);
        assertTrue(S.sumar() == 12);
    }

    @Test
    public void sumaNegativos() {
        System.out.println("Suma de numeros negativos: ");
        Suma S = new Suma(-5, -7);
        assertTrue(S.sumar() == -12);
    }

    @Test
    public void sumaPositivoNegativo() {
        System.out.println("Sum de positivos y negativos: ");
        Suma S = new Suma(5, -7);
        assertTrue(S.sumar() == -2);
    }

}
```

16. incorporar Selenium en el código fuente, por lo que se utilizará de igual forma JUnit para ejecutar los eventos en ambiente de pruebas:

Para los “import”:

```java
import static org.junit.Assert.assertTrue;

import java.io.IOException;
import java.util.concurrent.TimeUnit;

import org.apache.http.client.ClientProtocolException;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
```

Para la clase:

```
public class SeleniumTest {

    private WebDriver driver;
    private String baseUrl;

    public SeleniumTest() {
        System.setProperty("webdriver.gecko.driver", "/home/geckinstall/geckodriver");
        driver = new FirefoxDriver();
        baseUrl = "";
    }

    @Before
    public void setUp() {
        baseUrl = "http://104.248.67.100:8080/exampleTest/hello/index.xhtml";
        driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);
    }

    @Test
    public void test() throws ClientProtocolException, IOException {
        driver.get(baseUrl);
        String actualString = driver.findElement(By.id("form1:btn1")).getAttribute("value");
        assertTrue(actualString.equals("Realizar OP"));
    }

    @After
    public void tearDown() throws Exception {
        driver.close();
    }

}
```

Validación entorno de Integración Continua

1. Se verificó el estado de cada uno de los servicios a utilizar:

Jenkins

```
[jenkinsuser@centos-s-1-mas-jenkins ~]$ sudo systemctl status jenkins
[sudo] password for jenkinsuser:
● jenkins.service - LSB: Jenkins Automation Server
   Loaded: loaded (/etc/rc.d/init.d/jenkins; bad; vendor preset: disabled)
   Active: active (running) since Mon 2018-11-12 04:04:31 UTC; 1min 42s ago
     Docs: man:systemd-sysv-generator(8)
  Process: 1102 ExecStart=/etc/rc.d/init.d/jenkins start (code=exited, status=0
/SUCCESS)
   CGroup: /system.slice/jenkins.service
           └─1275 /etc/alternatives/java -Dcom.sun.akuma.Daemon=daemonized -...

Nov 12 04:04:28 centos-s-1-mas-jenkins systemd[1]: Starting LSB: Jenkins Aut...
Nov 12 04:04:28 centos-s-1-mas-jenkins runuser[1126]: pam_unix(runuser:sessi...
Nov 12 04:04:31 centos-s-1-mas-jenkins jenkins[1102]: Starting Jenkins [  OK  ]
Nov 12 04:04:31 centos-s-1-mas-jenkins systemd[1]: Started LSB: Jenkins Auto...
Hint: Some lines were ellipsized, use -l to show in full.
```

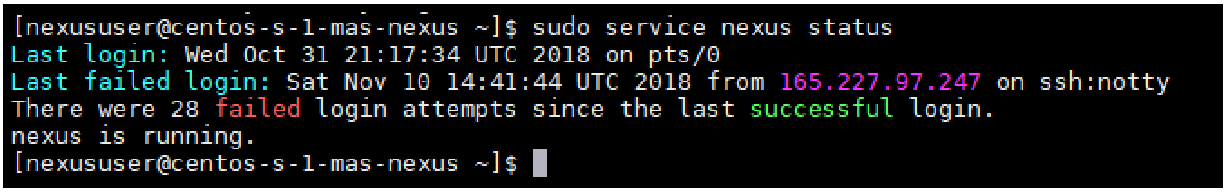
GitLab

```
[gitlabuser@centos-s-1-mas-gitlab ~]$ sudo systemctl status postfix
● postfix.service - Postfix Mail Transport Agent
   Loaded: loaded (/usr/lib/systemd/system/postfix.service; enabled; vendor preset: disabled)
   Active: active (running) since Wed 2018-10-24 23:51:09 UTC; 2 weeks 4 days ago
  Process: 916 ExecStart=/usr/sbin/postfix start (code=exited, status=0/SUCCESS)
  Process: 914 ExecStartPre=/usr/libexec/postfix/chroot-update (code=exited, status=0/SUCCESS)
  Process: 910 ExecStartPre=/usr/libexec/postfix/aliasesdb (code=exited, status=0/SUCCESS)
 Main PID: 1039 (master)
   CGroup: /system.slice/postfix.service
           ├─ 1039 /usr/libexec/postfix/master -w
           ├─ 1041 qmgr -l -t unix -u
           └─15230 pickup -l -t unix -u
```

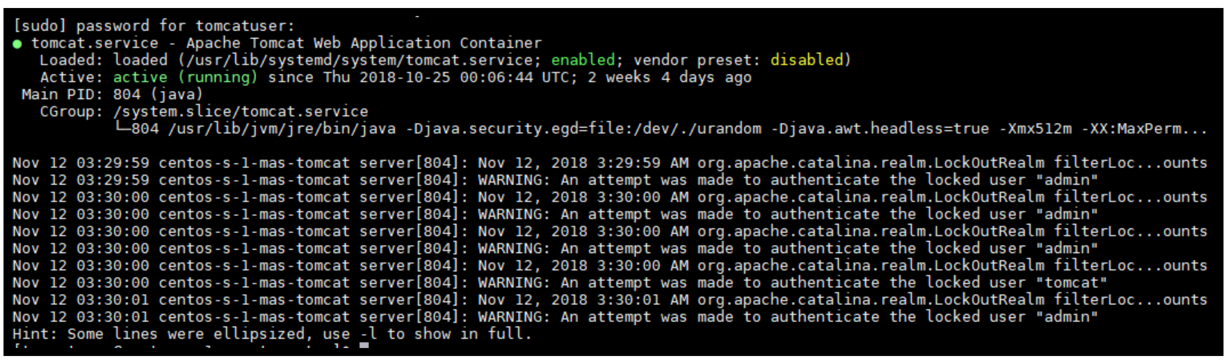
SonarQube

```
[sudo] password for sonaruser:
● sonar.service - SonarQube service
   Loaded: loaded (/etc/systemd/system/sonar.service; enabled; vendor preset: disabled)
   Active: active (running) since Sun 2018-10-28 21:12:09 UTC; 2 weeks 0 days ago
 Main PID: 20769 (wrapper)
   CGroup: /system.slice/sonar.service
           ├─20769 /opt/sonarqube/bin/linux-x86-64/./wrapper /opt/sonarqube/bin/linux-x86-64/../../conf/wrapper.conf wrapper.sys...
           ├─20771 java -Dsonar.wrapped=true -Djava.awt.headless=true -Xms8m -Xmx32m -Djava.library.path=./lib -classpath ../../...
           ├─20791 /usr/java/jdk1.8.0_131/jre/bin/java -Djava.awt.headless=true -Xmx1G -Xms256m -Xss256k -Djna.nosys=true -XX:+U...
           ├─20832 /usr/java/jdk1.8.0_131/jre/bin/java -Djava.awt.headless=true -Dfile.encoding=UTF-8 -Xmx512m -Xms128m -XX:+Hea...
           └─20999 /usr/java/jdk1.8.0_131/jre/bin/java -Djava.awt.headless=true -Dfile.encoding=UTF-8 -Xmx512m -Xms128m -XX:+Hea...
```

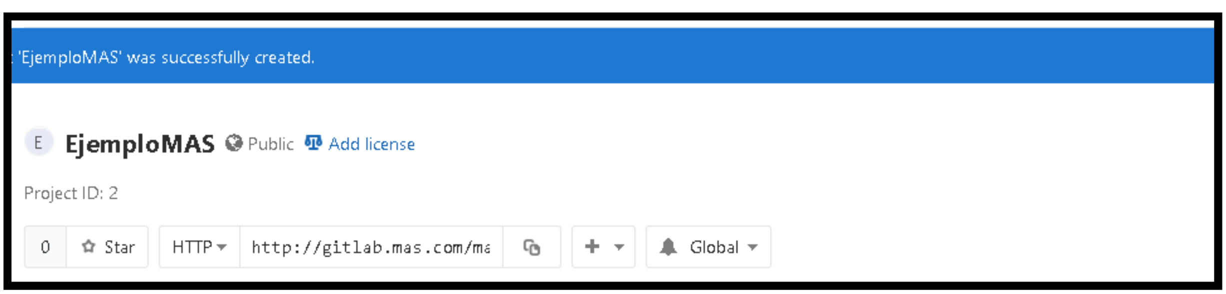
Nexus



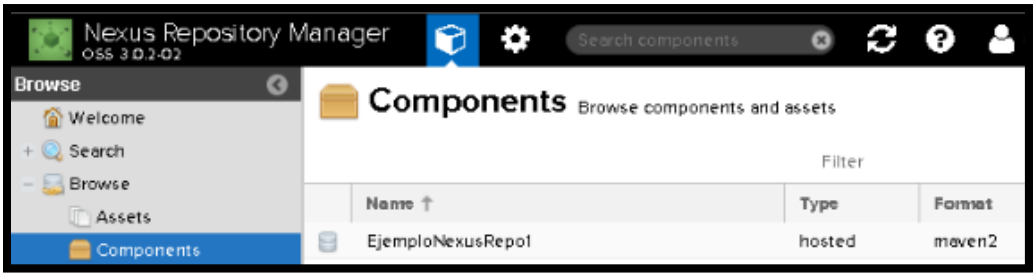
Tomcat



2. Se verificó la correcta comunicación entre los servidores mediante protocolo telnet.
3. Se creó un repositorio Git en GitLab para almacenar el código fuente:



4. Se creó un repositorio en el servidor Nexus, en el cual se cargó una librería creada a manera de ejemplo:



5. Para cargar la librería se ejecutó en consola:

```
mvn deploy:deploy-file -DgroupId=com.mas -DartifactId=mas1
-Dversion=1.1.0.0-SNAPSHOT -Dpackaging=jar -
Dfile=C:\Users\W2BFlores\Documents\MAS\EjemploNexus.jar -
DrepositoryId=EjemploNexusRepo -
Durl=http://138.68.21.12:8081/repository/EjemploNexusRepo1
```

```
C:\Users>mvn  deploy:deploy-file -DgroupId=com.mas -DartifactId=mas1 -Dversion=1.1.0.0-
SNAPSHOT -Dpackaging=jar -Dfile=C:\Users\W2BFlores\Documents\MAS\EjemploNexus.jar -Drep
ositoryId=EjemploNexusRepo -Durl=http://138.68.21.12:8081/repository/EjemploNexusRepo1
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------< org.apache.maven:standalone-pom >-------------------
[INFO] Building Maven Stub Project (No POM) 1
[INFO] --------------------------------[ pom ]---------------------------------
[INFO]
[INFO] --- maven-deploy-plugin:2.7:deploy-file (default-cli) @ standalone-pom ---
Downloading from EjemploNexusRepo: http://138.68.21.12:8081/repository/EjemploNexusRepo
1/com/mas/mas1/1.1.0.0-SNAPSHOT/maven-metadata.xml
Downloaded from EjemploNexusRepo: http://138.68.21.12:8081/repository/EjemploNexusRepo1
/com/mas/mas1/1.1.0.0-SNAPSHOT/maven-metadata.xml (767 B at 1.3 kB/s)
Uploading to EjemploNexusRepo: http://138.68.21.12:8081/repository/EjemploNexusRepo1/co
m/mas/mas1/1.1.0.0-SNAPSHOT/mas1-1.1.0.0-20181112.043518-2.jar
Uploaded to EjemploNexusRepo: http://138.68.21.12:8081/repository/EjemploNexusRepo1/com
/mas/mas1/1.1.0.0-SNAPSHOT/mas1-1.1.0.0-20181112.043518-2.jar (1.4 kB at 1.7 kB/s)
Uploading to EjemploNexusRepo: http://138.68.21.12:8081/repository/EjemploNexusRepo1/co
m/mas/mas1/1.1.0.0-SNAPSHOT/mas1-1.1.0.0-20181112.043518-2.pom
Uploaded to EjemploNexusRepo: http://138.68.21.12:8081/repository/EjemploNexusRepo1/com
/mas/mas1/1.1.0.0-SNAPSHOT/mas1-1.1.0.0-20181112.043518-2.pom (395 B at 514 B/s)
Downloading from EjemploNexusRepo: http://138.68.21.12:8081/repository/EjemploNexusRepo
1/com/mas/mas1/maven-metadata.xml
Downloaded from EjemploNexusRepo: http://138.68.21.12:8081/repository/EjemploNexusRepo1
/com/mas/mas1/maven-metadata.xml (273 B at 1.0 kB/s)
Uploading to EjemploNexusRepo: http://138.68.21.12:8081/repository/EjemploNexusRepo1/co
m/mas/mas1/1.1.0.0-SNAPSHOT/maven-metadata.xml
Uploaded to EjemploNexusRepo: http://138.68.21.12:8081/repository/EjemploNexusRepo1/com
/mas/mas1/1.1.0.0-SNAPSHOT/maven-metadata.xml (767 B at 982 B/s)
Uploading to EjemploNexusRepo: http://138.68.21.12:8081/repository/EjemploNexusRepo1/co
m/mas/mas1/maven-metadata.xml
Uploaded to EjemploNexusRepo: http://138.68.21.12:8081/repository/EjemploNexusRepo1/com
/mas/mas1/maven-metadata.xml (273 B at 347 B/s)
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 5.246 s
[INFO] Finished at: 2018-11-11T22:35:22-06:00
[INFO] ------------------------------------------------------------------------
```

6. Se creó un proyecto Java desde eclipse, que tiene como objetivo, dar el resultado de la suma de 2 números. El proyecto incluye de manera relevante, el archivo de configuración de Maven:

pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.prueba1.app</groupId>
    <artifactId>com.prueba1.app</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>war</packaging>

    <url>http://maven.apache.org</url>

    <properties>
        <!-- This property will be set by the Maven Dependency plugin -->
        <annotatedJdk>${org.checkerframework:jdk8:jar}</annotatedJdk>
    </properties>
<repositories>
    <repository>
        <id>EjemploNexusRepo</id>
        <name>EjemploNexusRepoName</name>
        <url>http://138.68.21.12:8081/repository/EjemploNexusRepo1</url>
    </repository>
</repositories>
    <dependencies>
```

```
    <dependency>
        <groupId>org.seleniumhq.selenium</groupId>
        <artifactId>selenium-java</artifactId>
        <version>3.5.3</version>
    </dependency>
    <dependency>
        <groupId>com.mas</groupId>
        <artifactId>mas1</artifactId>
        <version>1.1.0.0-20181101.010119-1</version>
    </dependency>
```

Para la capa de negocio las clases:

```java
package com.mas.app.controller;

import javax.annotation.PostConstruct;

@ManagedBean
@ViewScoped
public class SumaController {

    private Integer numero1 = 0;
    private Integer numero2 = 0;
    private Integer resultado = 0;

    @PostConstruct
    public void init() {
        this.numero1 = 0;
        this.numero2 = 0;
        this.resultado = 0;
    }

    public void sumarNumeros() {
        this.resultado = this.numero1 + this.numero2;
    }

    public Integer getNumero1() {
        return numero1;
    }

    public void setNumero1(Integer numero1) {
        this.numero1 = numero1;
    }
```

Para el modelo:

```java
package com.mas.app.model;

public class Suma {

    private int n1;

    private int n2;

    public Suma(int n1, int n2) {
        this.n1 = n1;
        this.n2 = n2;
    }

    public int sumar() {
        int resultado = n1 + n2;
        return resultado;
    }
}
```

7. Se creó para la parte Front-End, mediante jsf, un formulario con 2 campos de texto con sus respectivas etiquetas, un botón y una etiqueta para el resultado de la operación:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:ui="http://xmlns.jcp.org/jsf/facelets"
    xmlns:h="http://xmlns.jcp.org/jsf/html"
    xmlns:f="http://xmlns.jcp.org/jsf/core">
<head></head>
<h:form id="form1">
    <br />
        SUMA DE NUMEROS cambio 1:
        <br />
    <br />
    <br />
    <h:panelGrid columns="2">
        <h:column>
            <h:outputLabel value="Digite el primer numero entero: "></h:outputLabel>
        </h:column>
        <h:column>
            <h:inputText value="#{sumaController.numero1}"></h:inputText>
        </h:column>
        <h:column>
            <h:outputLabel value="Digite el segundo numero entero: "></h:outputLabel>
        </h:column>
        <h:column>
            <h:inputText value="#{sumaController.numero2}"></h:inputText>
        </h:column>
    </h:panelGrid>
    <br />
    <br />
    <h:commandButton value="Realizar OP" type="submit" id="btn1"
        actionListener="#{sumaController.sumarNumeros}" />
    <br />
    <br />
    <h:outputLabel value="Resultado: #{sumaController.resultado}" id="r1"></h:outputLabel>
</h:form>
</html>
```

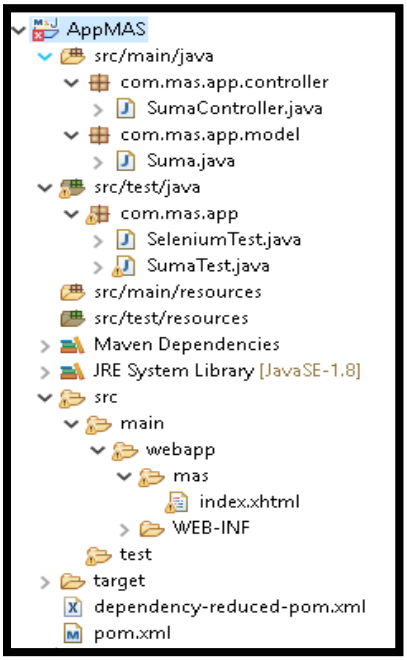
8. Se realizó la parte de las pruebas unitarias con JUnit:

```java
package com.mas.app;

import static org.junit.Assert.assertTrue;

import org.junit.Test;

import com.mas.app.model.Suma;
import com.mas2.Ejemplo2;

public class SumaTest {

    @Test
    public void sumaPositivos() {
        Ejemplo2 ejemplo2 = new Ejemplo2();
        System.out.println("Suma de numeros positivos: ");
        Suma S = new Suma(5, 7);
        assertTrue(S.sumar() == 12);
    }

    @Test
    public void sumaNegativos() {
        System.out.println("Suma de numeros negativos: ");
        Suma S = new Suma(-5, -7);
        assertTrue(S.sumar() == -12);
    }

    @Test
    public void sumaPositivoNegativo() {
        System.out.println("Sum de positivos y negativos: ");
        Suma S = new Suma(5, -7);
        assertTrue(S.sumar() == -2);
    }

}
```

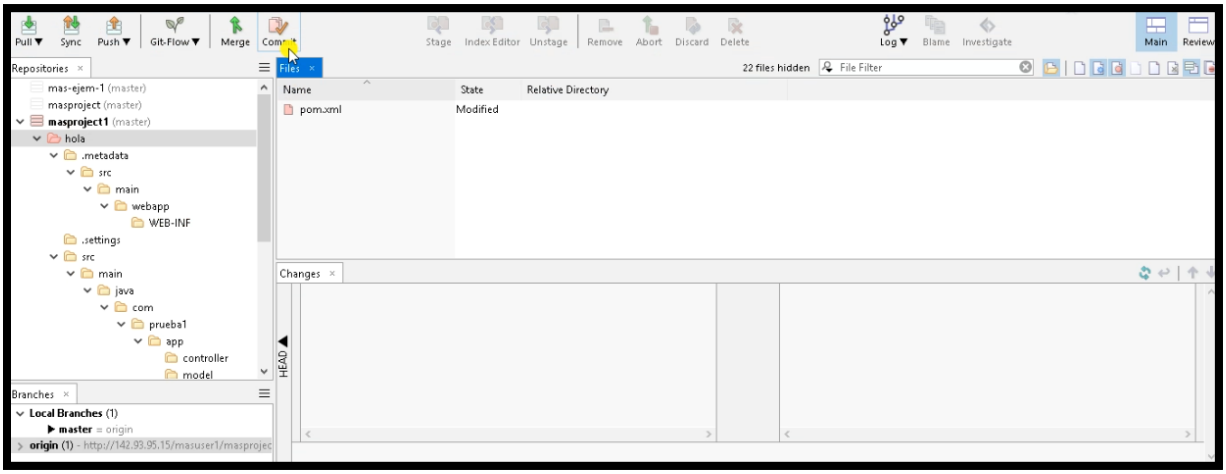
9. Se creó la parte de las pruebas con Selenium:

```java
package com.mas.app;

import static org.junit.Assert.assertTrue;
import java.io.IOException;
import java.util.concurrent.TimeUnit;
import org.apache.http.client.ClientProtocolException;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;

public class SeleniumTest {

    private WebDriver driver;
    private String baseUrl;

    public SeleniumTest() {
        System.setProperty("webdriver.gecko.driver", "/home/geckinstall/geckodriver");
        driver = new FirefoxDriver();
        baseUrl = "";
    }
    @Before
    public void setUp() {
        baseUrl = "http://104.248.67.100:8080/holaTest/hello/index.xhtml";
        driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);
    }
    @Test
    public void test() throws ClientProtocolException, IOException {
        driver.get(baseUrl);
        String actualString = driver.findElement(By.id("form1:btn1")).getAttribute("value");
        assertTrue(actualString.equals("Realizar OP"));
    }
    @After
    public void tearDown() throws Exception {
        driver.close();
    }
}
```

Quedando todo estructurado de la manera siguiente:



10. Mediante el aplicativo SmarGit se cargó el código fuente al repositorio antes creado:



11. Se configuró un "*webhook*" en el GitLab, de la misma forma que se indicó en la guía, para que desde el *"push"* se ejecutara la primera tarea dentro de Jenkins.

```
Started by GitLab push by masuser1
Building in workspace /var/lib/jenkins/workspace/JenkinsProject1
 > git rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
 > git config remote.origin.url http://gitlab.mas.com/masuser1/masproject1.git # timeout=10
Fetching upstream changes from http://gitlab.mas.com/masuser1/masproject1.git
 > git --version # timeout=10
using GIT_ASKPASS to set credentials
 > git fetch --tags --progress http://gitlab.mas.com/masuser1/masproject1.git +refs/heads/*:refs/remotes/origin/*
skipping resolution of commit remotes/origin/master, since it originates from another repository
 > git rev-parse refs/remotes/origin/master^{commit} # timeout=10
 > git rev-parse refs/remotes/origin/origin/master^{commit} # timeout=10
Checking out Revision 78a2fbd1110288d871fe9c600a6c2a7e95911f31 (refs/remotes/origin/master)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f 78a2fbd1110288d871fe9c600a6c2a7e95911f31
Commit message: "Commit 16 311018"
```

12. Se extraen las librerías del repositorio de Nexus configurado tal cual lo menciona la guía:

```
[INFO] ------------------< com.prueba1.app:com.prueba1.app >-------------------
[INFO] Building com.prueba1.app 0.0.1-SNAPSHOT
[INFO] --------------------------------[ war ]---------------------------------
[INFO] Downloading from EjemploNexusRepo: http://138.68.21.12:8081/repository/EjemploNexusRepo1/com/mas/mas2/1.1.0.0-SNAPSHOT/mas2-1.1.0.0-20181101.010119-1.pom
[INFO] Downloaded from EjemploNexusRepo: http://138.68.21.12:8081/repository/EjemploNexusRepo1/com/mas/mas2/1.1.0.0-SNAPSHOT/mas2-1.1.0.0-20181101.010119-1.pom (395 B at 2.2 kB/s)
[INFO] Downloading from EjemploNexusRepo: http://138.68.21.12:8081/repository/EjemploNexusRepo1/com/mas/mas2/1.1.0.0-SNAPSHOT/mas2-1.1.0.0-20181101.010119-1.jar
[INFO] Downloaded from EjemploNexusRepo: http://138.68.21.12:8081/repository/EjemploNexusRepo1/com/mas/mas2/1.1.0.0-SNAPSHOT/mas2-1.1.0.0-20181101.010119-1.jar (1.3 kB at 33 kB/s)
[INFO]
```

13. Para que luego de manera automática se observara la realización de las pruebas de código estático mediante SonarQube:

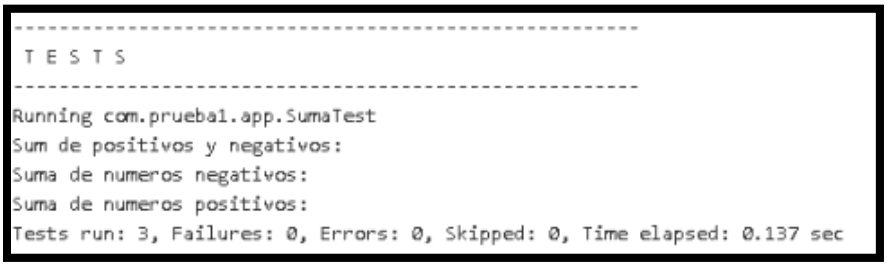
```
 > git rev-list --no-walk acd5bd4668ad7f35a686d3786cfe41b3122748b5 # timeout=10
Xvfb starting$ /usr/bin//Xvfb :99 -fbdir /var/lib/jenkins/xvfb-51-..fbdir3484455352671052595
Unpacking https://repo1.maven.org/maven2/org/sonarsource/scanner/cli/sonar-scanner-cli/3.2.0.1227/sonar-scanner-cli-3.2.0.1227.zip to
/var/lib/jenkins/tools/hudson.plugins.sonar.SonarRunnerInstallation/sonarScanner1 on Jenkins
[JenkinsProject1] $ /var/lib/jenkins/tools/hudson.plugins.sonar.SonarRunnerInstallation/sonarScanner1/bin/sonar-scanner -
Dsonar.host.url=http://104.248.217.66 ******** -Dsonar.projectKey=un -Dsonar.sources=. -
Dsonar.projectBaseDir=/var/lib/jenkins/workspace/JenkinsProject1
INFO: Scanner configuration file: /var/lib/jenkins/tools/hudson.plugins.sonar.SonarRunnerInstallation/sonarScanner1/conf/sonar-
scanner.properties
INFO: Project root configuration file: NONE
INFO: SonarQube Scanner 3.2.0.1227
INFO: Java 1.8.0_192 Oracle Corporation (64-bit)
```

14. Luego llamaba al compilador del JDK:

```
[INFO] ------------------< com.prueba1.app:com.prueba1.app >-------------------
[INFO] Building com.prueba1.app 0.0.1-SNAPSHOT
[INFO] --------------------------------[ war ]---------------------------------
[INFO]
[INFO] --- maven-dependency-plugin:2.3:properties (default) @ com.prueba1.app ---
[INFO]
```

```
https://jenkins.io/redirect/serialization-of-anonymous-classes/
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
```

15. Después pasaba a la parte de pruebas unitarias de JUnit:



16. Después desplegaba el archivo war en el servidor Tomcat en el contexto de test:

17. Después realizada las pruebas definidas con Selenium a esta aplicación de Pruebas, y si al final de todo el ciclo, no se presentaba algún fallo, la aplicación era desplegada en el contexto de Producción:

```
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 30.335 s
[INFO] Finished at: 2018-10-28T22:03:35Z
[INFO] ------------------------------------------------------------------------
Waiting for Jenkins to finish collecting data
[JENKINS] Archiving /var/lib/jenkins/workspace/Tarea2PruebaSelenium/hola/dependency-reduced-pom.xml to
com.prueba1.app/com.prueba1.app/0.0.1-SNAPSHOT/com.prueba1.app-0.0.1-SNAPSHOT.pom
[JENKINS] Archiving /var/lib/jenkins/workspace/Tarea2PruebaSelenium/hola/target/com.prueba1.app-0.0.1-SNAPSHOT.war to
com.prueba1.app/com.prueba1.app/0.0.1-SNAPSHOT/com.prueba1.app-0.0.1-SNAPSHOT.war
channel stopped
[Tarea2PruebaSelenium] $ /bin/sh -xe /tmp/jenkins5292217023669600845.sh
+ export DISPLAY=:99
+ DISPLAY=:99
Xvfb stopping
unlink: No such file or directory
unlink /var/lib/jenkins/xvfb-34-..fbdir358908896242150057l/Xvfb_screen0 failed, Invalid argumentDeploying
/var/lib/jenkins/workspace/Tarea2PruebaSelenium/hola/target/com.prueba1.app-0.0.1-SNAPSHOT.war to container Tomcat 7.x Remote with
context hola
  Redeploying [/var/lib/jenkins/workspace/Tarea2PruebaSelenium/hola/target/com.prueba1.app-0.0.1-SNAPSHOT.war]
  Undeploying [/var/lib/jenkins/workspace/Tarea2PruebaSelenium/hola/target/com.prueba1.app-0.0.1-SNAPSHOT.war]
  Deploying [/var/lib/jenkins/workspace/Tarea2PruebaSelenium/hola/target/com.prueba1.app-0.0.1-SNAPSHOT.war]
Deploying /var/lib/jenkins/workspace/Tarea2PruebaSelenium/hola/target/original-com.prueba1.app-0.0.1-SNAPSHOT.war to container Tomcat 7.x
Remote with context hola
  Redeploying [/var/lib/jenkins/workspace/Tarea2PruebaSelenium/hola/target/original-com.prueba1.app-0.0.1-SNAPSHOT.war]
  Undeploying [/var/lib/jenkins/workspace/Tarea2PruebaSelenium/hola/target/original-com.prueba1.app-0.0.1-SNAPSHOT.war]
  Deploying [/var/lib/jenkins/workspace/Tarea2PruebaSelenium/hola/target/original-com.prueba1.app-0.0.1-SNAPSHOT.war]
Finished: SUCCESS
```

Referencias

HugeServer. (5 de Noviembre de 2018). *HugeServer Knowledgebase*. Obtenido de How to Install GitLab on CentOS 7: https://www.hugeserver.com/kb/how-install-gitlab-centos7/

Coveros Inc. (18 de Mayo de 2016). Obtenido de Run Headless Selenium Tests From Jenkins: https://www.coveros.com/run-headless-selenium-tests-from-jenkins/

DevopsCube. (23 de Septiembre de 2016). *DevopsCube*. Obtenido de How To Install Latest Sonatype Nexus 3 On Linux: https://devopscube.com/how-to-install-latest-sonatype-nexus-3-on-linux/

DigitalOcean Inc. (15 de Junio de 2015). *DigitalOcean Tutorials*. Obtenido de How To Install Apache Tomcat 7 on CentOS 7 via Yum: https://www.digitalocean.com/community/tutorials/how-to-install-apache-tomcat-7-on-centos-7-via-yum

Kili, A. (25 de Octubre de 201). *Tecmint: Linux Howtos, Tutorials & Guides*. Obtenido de How to Fix “firewall-cmd: command not found” Error in RHEL/CentOS 7: https://www.tecmint.com/fix-firewall-cmd-command-not-found-error/

Sonatype Inc. (08 de Enero de 2018). *Sonatype Support*. Obtenido de How do I configure the Nexus Jenkins Plugin: https://support.sonatype.com/hc/en-us/articles/227256688-How-do-I-configure-the-Nexus-Jenkins-Plugin

Stack Exchange Inc. (27 de Marzo de 2018). *Stackoverflow*. Obtenido de Open firewall port on CentOS 7: https://stackoverflow.com/questions/24729024/open-firewall-port-on-centos-7

Vultr Holdings Corporation. (19 de Mayo de 2016). *Vultr Docs*. Obtenido de How to Install Jenkins on CentOS 7: https://www.vultr.com/docs/how-to-install-jenkins-on-centos-7

Vultr Holdings Corporation. (30 de Enero de 2017). *Vultr Docs*. Obtenido de How to Install SonarQube on CentOS 7: https://www.vultr.com/docs/how-to-install-sonarqube-on-centos-7

6.6 Guía de implementación del entorno de integración continua para el desarrollo de aplicaciones web en .NET

UNIVERSIDAD DON BOSCO

Vicerrectoría de Estudios de Postgrados

Maestría en Arquitectura de Software

Guía de implementación del entorno de integración continua para el desarrollo de aplicaciones web en .NET

Presentado por:

María Lorena De La Fuente Jacobo

William Antonio Flores Ayala

Roberto Bladimir Hernández Guevara

Asesor:

Luis García

Antiguo Cuscatlán, La Libertad, El Salvador

Enero 2019

Índice

Introducción

La presente guía es un insumo para dar los primeros pasos en la construcción de un entorno de desarrollo ágil para aplicaciones Web en .Net, utilizando integración continua, conocida de acá en adelante como IC. Se incluye el diseño de la arquitectura, la instalación y configuración de las herramientas que conforman el entorno de IC.

Se explica paso a paso la instalación y configuración del servidor de integración Jenkins, quien es el responsable de orquestar y ejecutar todo el "pipeline" de IC, construcción y ejecución de pruebas unitarias. Las pruebas unitarias representan uno de los pilares de IC y no son más que pruebas que el desarrollador escribe para probar cada pieza o componente del software. [Martin Fowler]. Además, aprenderá cómo instalar y configurar el sistema de versionamiento GIT, un sistema que controla y permite llevar un seguimiento de todos los cambios hechos en el código, así como también instalar y configurar GitLab uno de los administradores de repositorios basados en GIT más populares. Para el análisis de código estático aprenderá cómo instalar y configurar el servidor de SonarQube. El análisis de código estático se refiere a la capacidad de analizar el software sin necesidad de ejecutar el programa que va a ser analizado y para ello se hace uso de SonarQube.

Las herramientas fueron seleccionadas en base al estudio realizado a un conjunto de empresas del área metropolitana de San Salvador, que han implementado y están utilizando integración continua por lo menos durante 6 meses en sus desarrollos de aplicaciones Web. Así mismo se validó con los expertos a finales del 2018.

Preparación

Antes de comenzar la implementación del entorno de IC hay que evaluar el modelo de desarrollo que se está utilizando y la arquitectura de las aplicaciones Web. Hay una relación de integración y entrega continua con el desarrollo ágil, por lo que es recomendable implementar modelos ágiles para aprovechar los beneficios; siendo Scrum el modelo más utilizado según el estudio. Este es un cambio organizacional, por lo que requiere el apoyo de la alta gerencia, así como la capacitación y definición de nuevos roles dentro de equipos multidisciplinarios.

Un cambio organizacional, ya sea que se trate de agregar un nuevo proceso o modificar uno existente, puede generar resistencia por lo que al iniciar un proceso de implementación de IC debe existir una correcta gestión del cambio.

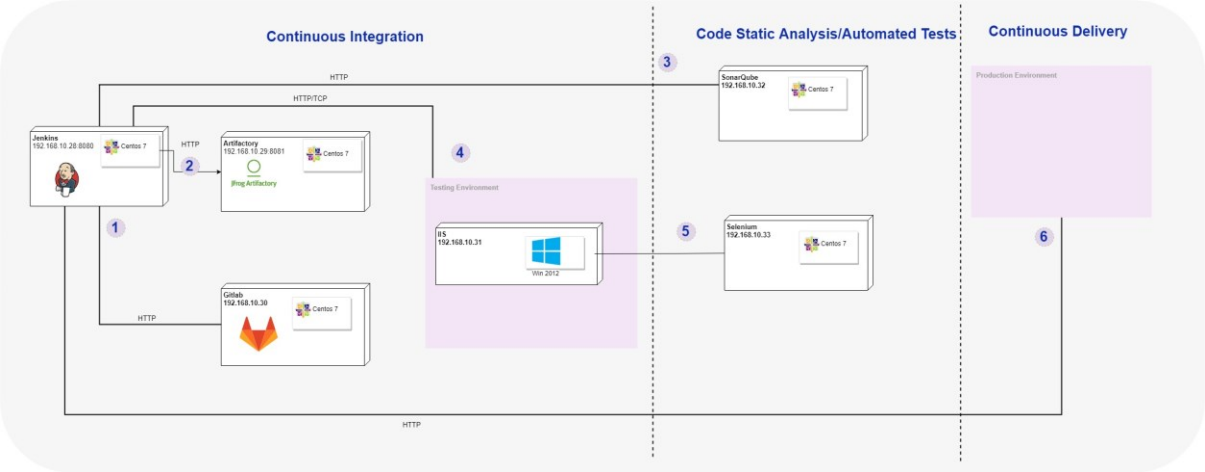
Se ha coincidido con las empresas y expertos que la gestión del cambio y la cultura ágil son claves para la adopción de IC. Se debe invertir más tiempo en conocer a los equipos, su experiencia y conocimiento, para poder hacerlos parte del nuevo proceso que en la selección y configuración de herramientas.

Es importante definir las aplicaciones que serán migradas al entorno de IC, iniciando con aplicaciones sencillas pero importantes que den valor al negocio a corto plazo y demuestren los beneficios de IC manteniendo así el apoyo a la iniciativa. Es recomendable también, buscar apoyo de expertos en IC, esto puede ahorrar tiempo por el conocimiento y orientación que puedan aportar en la implementación.

Es muy apropiado definir la estrategia de ramas, versionamiento, integraciones, gestión de conflictos, “pipelines” y pruebas. De igual forma, es relevante el establecimiento de estándares de desarrollo y métricas de calidad que cada una de las herramientas debe de verificar para validar una nueva funcionalidad.

Para mayor detalle de las estrategias y recomendaciones para la implementación y adopción de IC, referenciarse con la guía de buenas prácticas adjunta.

Arquitectura del Entorno de IC para .NET



Herramientas del Entorno de IC

Compilador de código

MSBuild

Es una plataforma para compilar aplicaciones. Este motor proporciona un esquema XML de un archivo de proyecto que controla cómo la plataforma de compilación procesa y compila el software. Visual Studio utiliza MSBuild, pero no depende de Visual Studio.

Al invocar msbuild.exe en el archivo de proyecto o de solución, puede orquestar y compilar productos en entornos donde no está instalado Visual Studio.

Control de Versiones

Git

Es un sistema de control de versiones distribuidas de código abierto y gratuito diseñado para manejar todo, desde proyectos pequeños a muy grandes, con velocidad y eficiencia.

Administrador de repositorios

Gitlab

Es un administrador de repositorios Git, además tiene la funcionalidad de ser una aplicación individual para todas las etapas del ciclo de vida de DevOps.

Repositorio de artefactos

JFrog Artifactory

Es un repositorio de artefactos universal que permite administrar cualquier pieza de software, binarios o dependencias, en cualquier lenguaje. Además, es otra de las aplicaciones que se pueden instalar de manera relativamente sencilla desde el Catálogo de Aplicaciones Cloud de Arsys, lo que permite centralizar el trabajo de los diferentes equipos de desarrollo en un entorno escalable y seguro.

Construcción de proyecto

NuGet

Es el administrador de paquetes para .NET. Las herramientas del cliente NuGet proporcionan la capacidad de producir y consumir paquetes. La Galería NuGet es el repositorio central de paquetes utilizado por todos los autores y consumidores de paquetes.

Servidor de Integración

Jenkins

Es un servidor autónomo de código abierto que se puede usar para automatizar todo tipo de tareas relacionadas con la construcción, prueba y entrega o implementación de software. Este puede ser considerado como la piedra angular para implementar de manera efectiva IC en una organización

Evaluador de código estático

SonarQube

Es una plataforma de código abierto para realizar revisiones automáticas con análisis estático de código para detectar errores, olores de código y vulnerabilidades de seguridad en más de 20 idiomas de programación, incluidos Java, C #, JavaScript, TypeScript, C / C ++, COBOL y más.

Pruebas unitarias

NUnit

Es un marco de prueba de unidad para todos los lenguajes .NET. Inicialmente portado desde JUnit, es miembro del .NET Foundation, y sus últimas versiones han sido completamente re-escritas con muchas características nuevas y con soporte para una amplia gama de plataformas .NET.

Servidor de despliegue

Internet Information Services

Mejor conocido como IIS según el sitio oficial de Microsoft, es un servidor web flexible, seguro y manejable para hospedar cualquier cosa en la web. Desde la transmisión de medios a las aplicaciones web, la arquitectura abierta y escalable de IIS está lista para manejar las tareas más exigentes.

Proceso de instalación y configuración inicial del Servidor de Integración Jenkins

Prerrequisitos

Hardware

- 2 GB de RAM
- 1 CPU
- 1 Interfaz de red
- Disco duro de 10 GB

Software

- Windows Server 2012 R2
- Usuario con privilegios de administrador.

Nota: Se está utilizando para el sistema operativo el usuario "*jenkinsuser*" como usuario administrador.

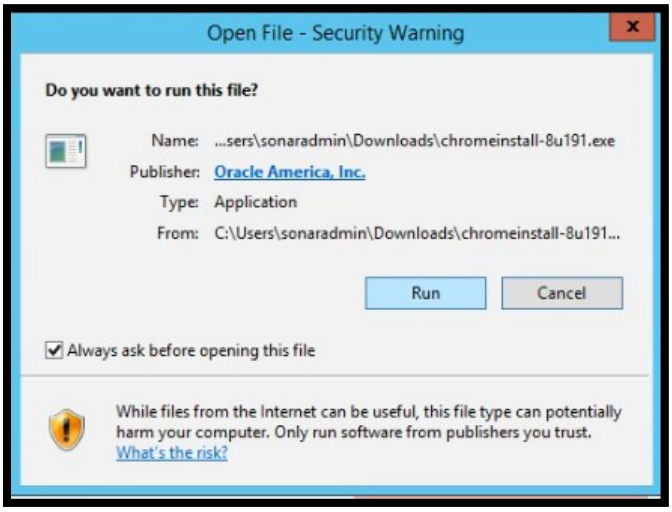
Paso 1. Descarga e Instalación de Java

1. Descargar el Java Runtime Environment (JRE) de Java desde el sitio oficial de Oracle:

`www.java.com.en/download/`

2. Dar clic en el botón de descarga, reconocerá la compatibilidad más adecuada.

3. Ejecutar el instalador dando clic en "*Run*".



4. Dar clic en siguiente o "*Next*" y aceptar los términos de la licencia dando clic en el botón "*Ok*".

5. Esperar a que la instalación termine y dar clic en el botón de cerrar o "*Close*":


Java Setup - Progress
Java
Status: Installing Java
3 Billion
Devices Run Java
Java
#1 Development Platform
ORACLE

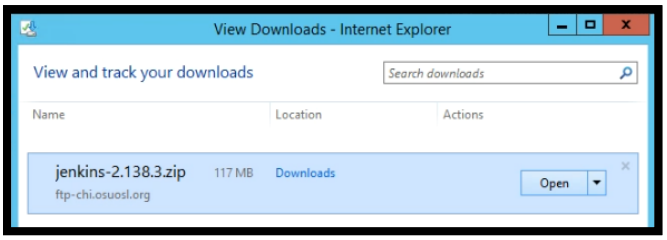

Java Setup - Complete
Java
You have successfully installed Java
You will be prompted when Java updates are available. Always install updates to get the latest performance and security improvements.
More about update settings
Close

Paso 2. Descarga e Instalación de Jenkins

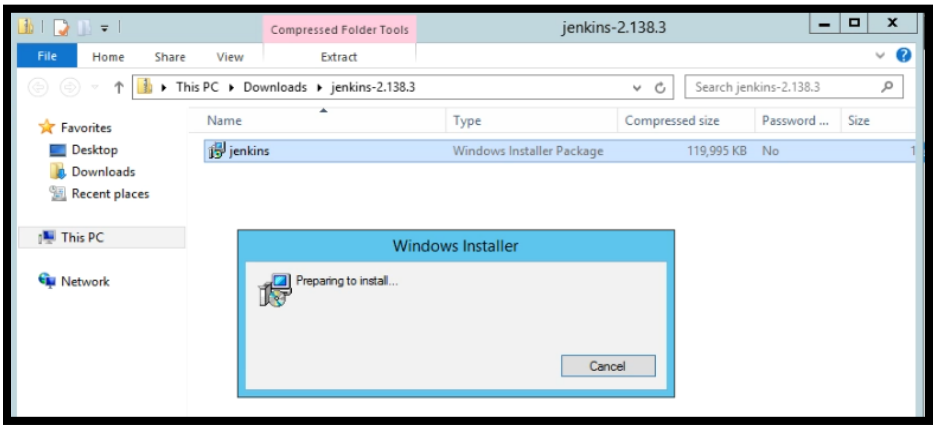
1. Descargar Jenkins desde el sitio oficial para Windows:

`https://jenkins.io/download/`

2. En este escenario se utilizará Jenkins 2.138.3 para Windows.
3. Abrir el archivo Zip descargado.

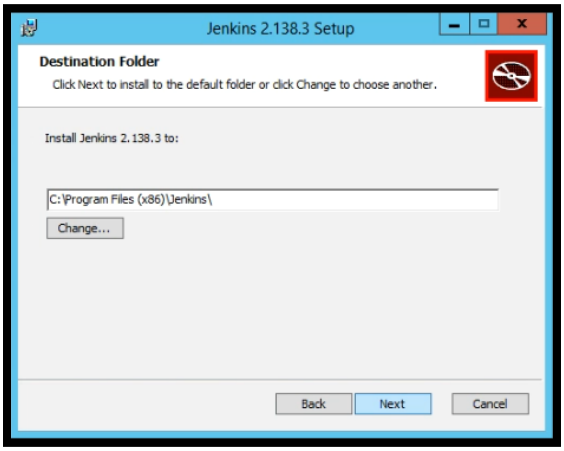


4. Ejecutar el instalador de Jenkins.

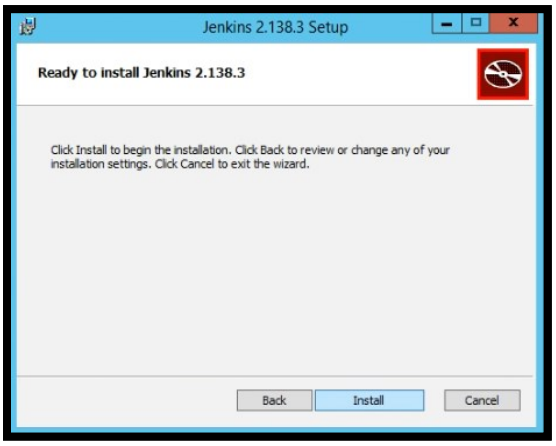


5. Se desplegará un cuadro de diálogo en el cual debemos dar clic en siguiente o "*Next*".

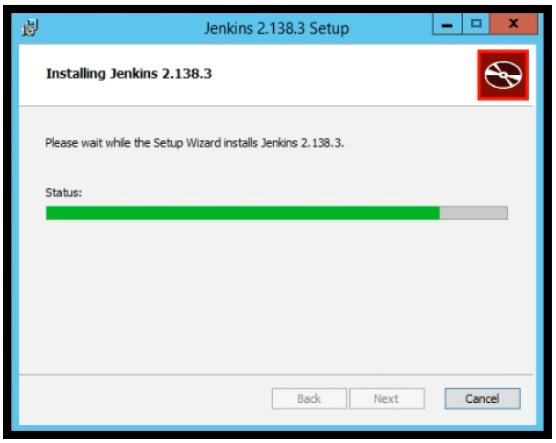
6. Colocar el directorio para la instalación de los archivos pertinentes, y se le da clic en siguiente o "*Next*".

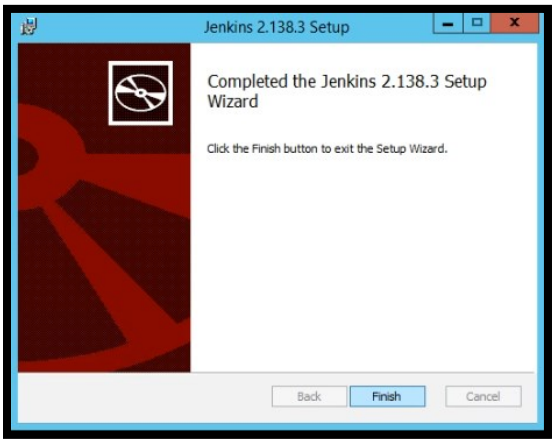


7. Se presenta un cuadro de diálogo mostrando que ya se está listo para empezar la instalación como tal, y se debe dar clic en instalar o "*Install*".



8. Esperar a que la instalación termine y dar clic en el botón de finalizar o "*Finish*".



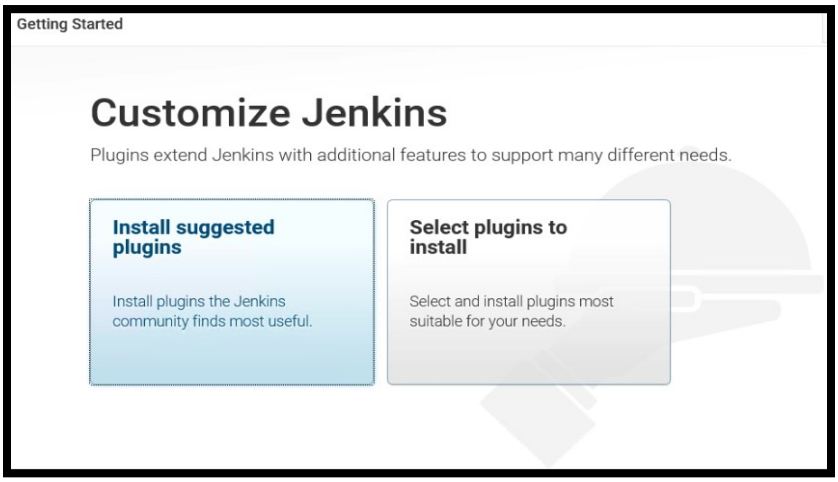


9. En el navegador se abrirá el sitio para las configuraciones iniciales.

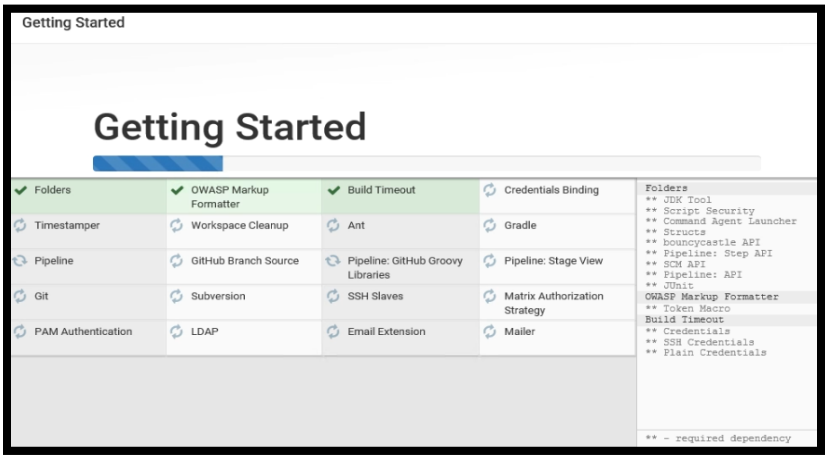
10. Indicará la ruta donde se encuentra en el servidor, el archivo log donde ha generado la contraseña de administrador, por lo que es necesario ir a la ubicación y abrir el archivo.
11. Copiar la contraseña e ingresarla en la dirección que se tiene abierta en el navegador web, para desbloquear el servidor de Jenkins.
12. Dar clic en "*Continue*".

Paso 3. Configuración inicial de Jenkins

1. Instalar los complementos sugeridos.



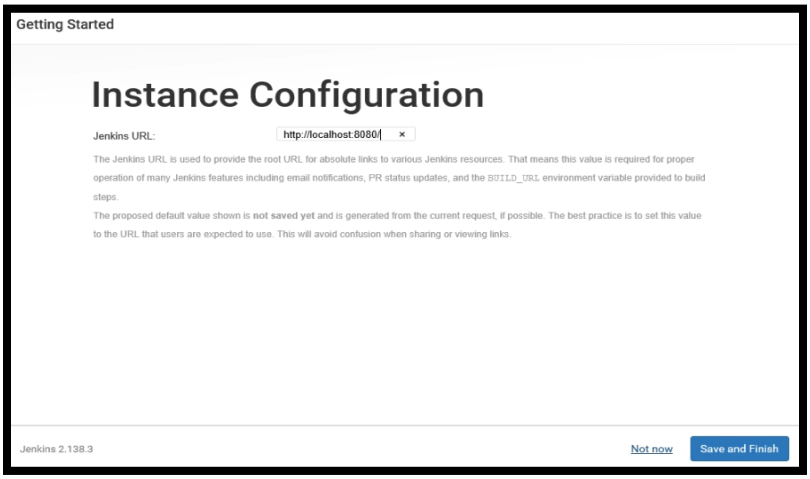
2. Se debe de esperar a que el proceso termine de instalar los complementos.



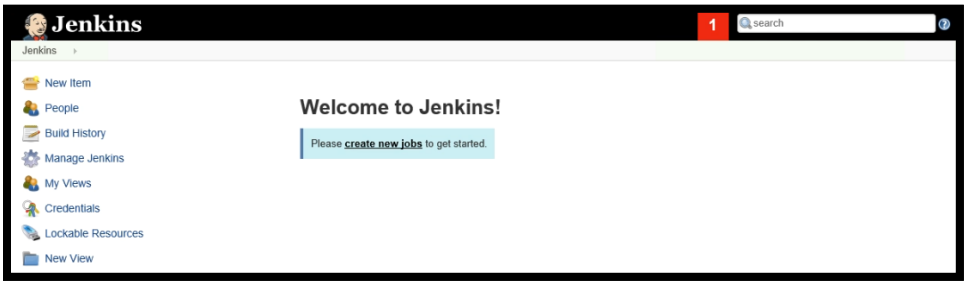
3. Crear usuario administrador.

4. Completar la información necesaria del usuario administrador, dar clic en "*Save and Continue*".

5. Opcionalmente se define la URL de Jenkins, se guardan los cambios y se finaliza proceso de configuración inicial de Jenkins dando clic en "*Save and Finish*".



6. A final debe de mostrarse un mensaje que indique que Jenkins está listo.
7. Dar clic en "*Start using Jenkins*" para iniciar e ingresar al panel de Jenkins.



8. Para poder identificar los servidores, en caso de no tener un servidor de nombres de dominio (DNS), se recomienda añadir la dirección IP del host servidor Jenkins al archivo "*hosts*":

```
C:\Windows\System32\drivers\etc archivo hosts
```

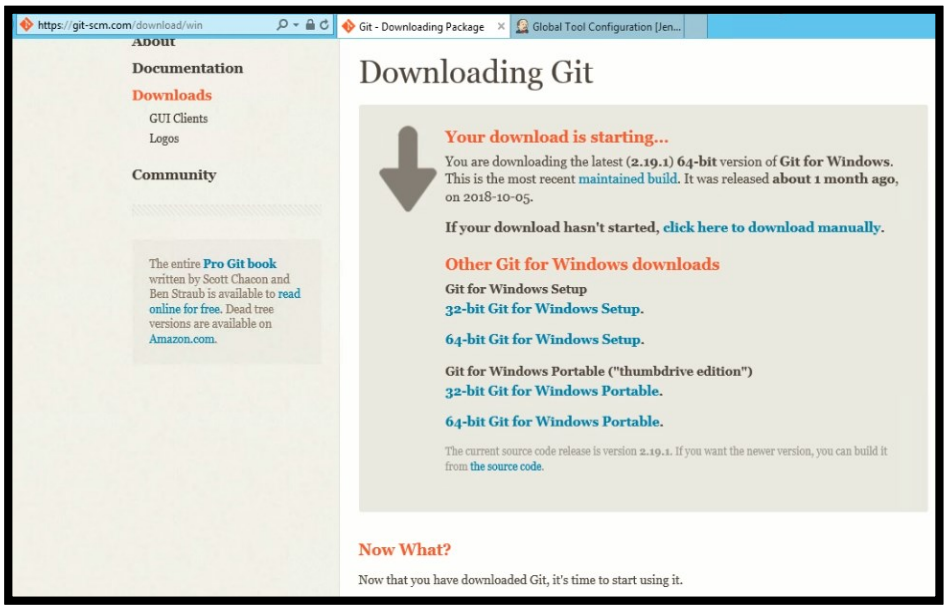
9. Se agrega la línea de comando con la dirección IP y el dominio definidos, para lo cual se deberá cambiar la información según convenga, por ejemplo:

```
192.168.1.XX Jenkins.mas.com
```

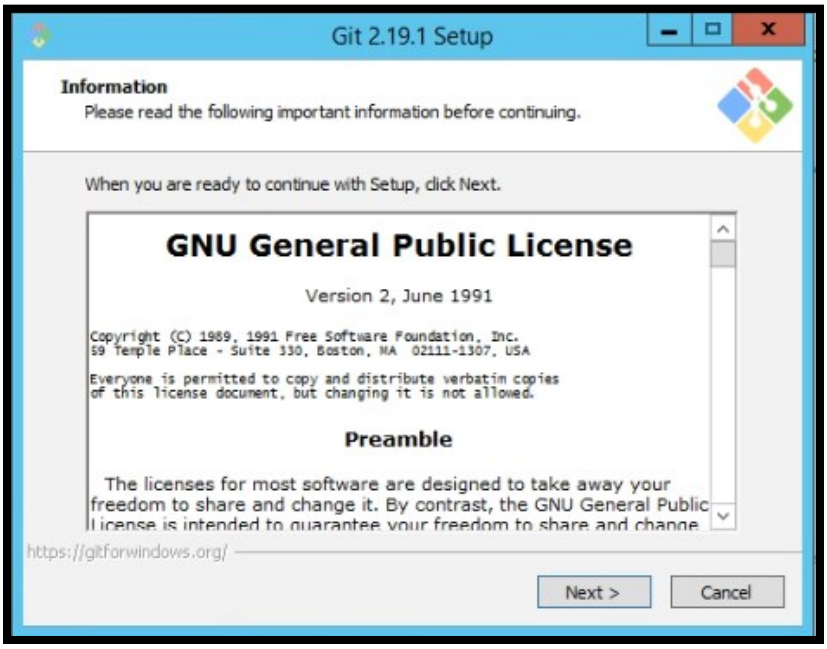
Paso 4. Instalación Git

1. Para poder acceder a los repositorios de código es necesario descargar e instalar Git en el servidor Jenkins. Descargar Git para Windows desde el sitio oficial:

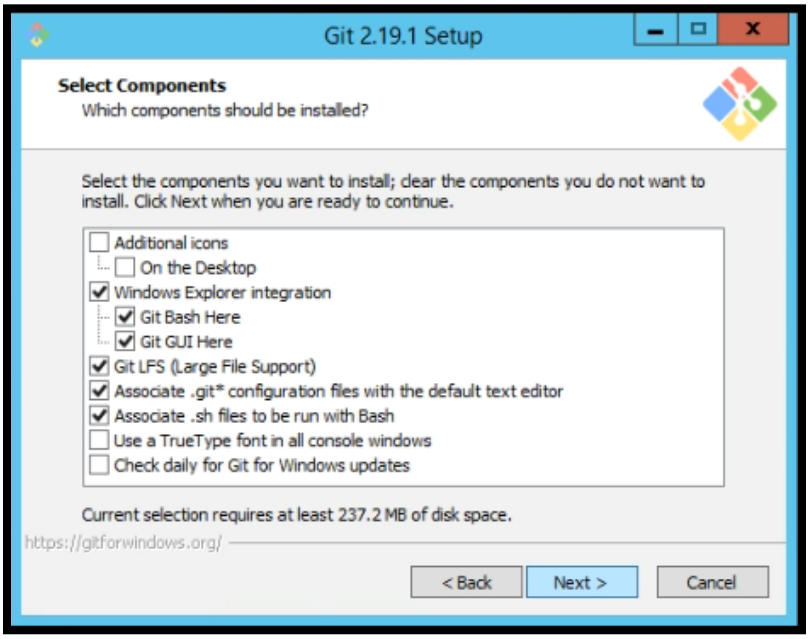
`git-scm.com/downloads/win`



10. Ejecutar el archivo recién descargado y dar clic en siguiente o “Next”.

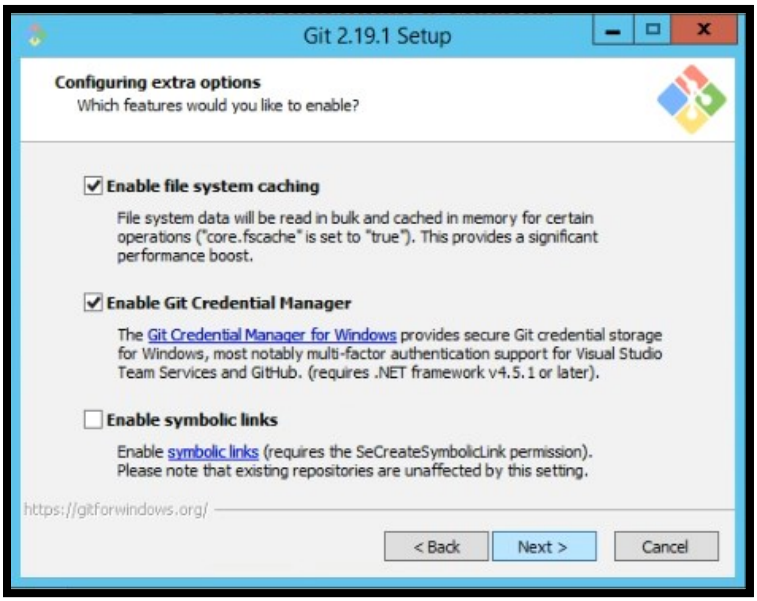


11. Por defecto aparecen seleccionadas los componentes básicos para Git, dar clic en siguiente o "*Next*".

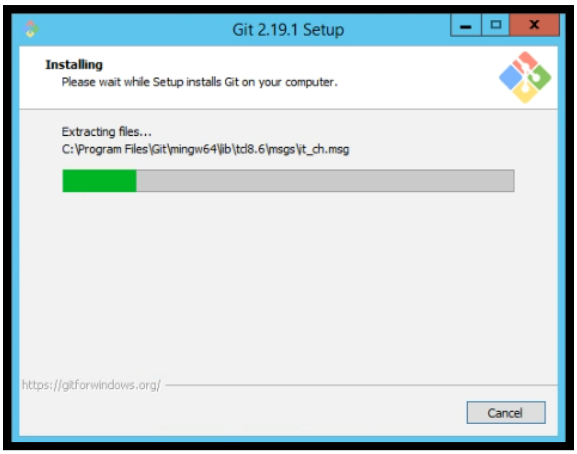


12. Seleccionar la opción de uso de Git desde el "*Windows Command Prompt*" y dar clic en siguiente o "*Next*".

13. Seleccionar la habilitación de la administración de credenciales y la caché del sistema de archivos de Git desde el "*Windows Command Prompt*" y dar clic en siguiente o "N*ext*".



14. Dar clic en "*Install*" y esperar a que la instalación termine para dar clic en el botón de finalizar o "*Finish*".



Proceso de instalación y configuración inicial del Servidor de Repositorios GitLab

Prerrequisitos

Hardware

- 3 GB de RAM
- 1 CPU
- 1 interfaz de red
- Disco duro de 20 GB

Software

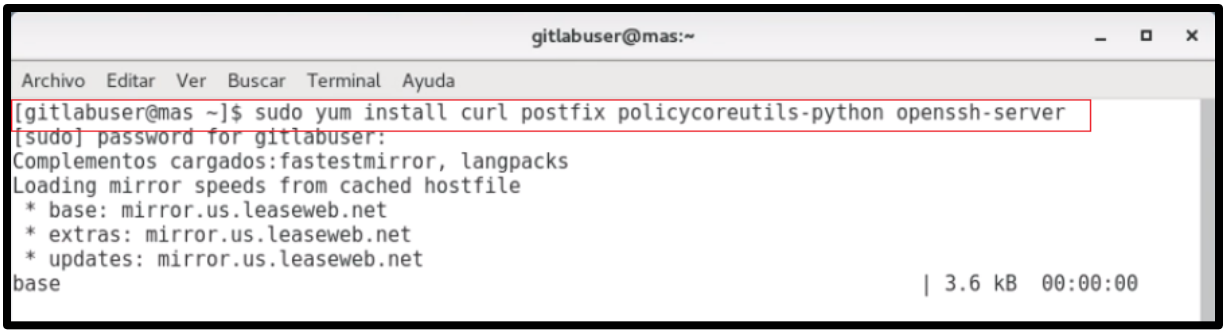
- CentOS 7, 64bit
- GitLab 11.2
- Usuario con privilegios de super administrador.
- Tener instalado el comando wget y firewall-cmd en el sistema operativo.

Nota: Se está utilizando para el sistema operativo el usuario "gitlabuser" como usuario administrador.

Paso 1: Instalación de Paquetes

1. Instalar los paquetes requeridos por GitLab, usando CURL para descargar el instalador del repositorio. Ejecutar el siguiente comando con el usuario administrador:

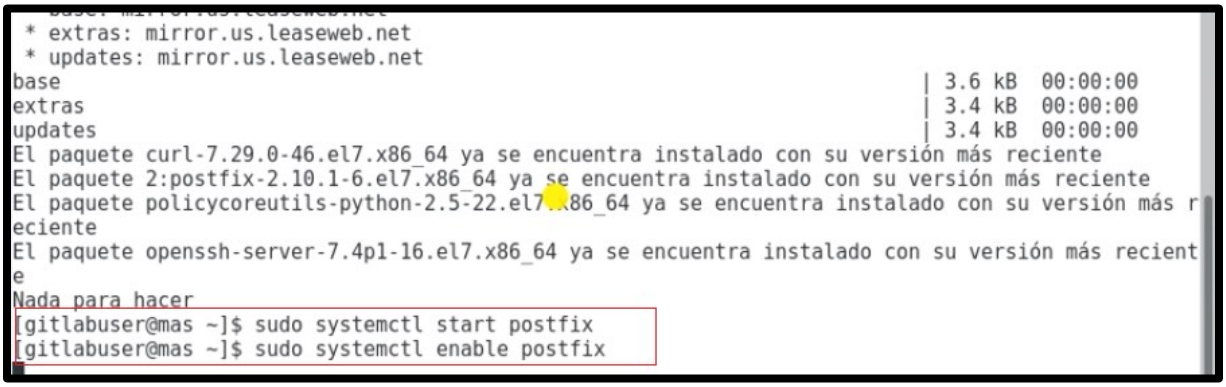
```
sudo yum -y install curl policycoreutils openssh-server openssh-clients postfix
```



2. Se recomienda activar los servicios para que se ejecuten de forma automática cuando inicie el sistema, ejecutando los siguientes comandos:

```
sudo systemctl enable postfix
```

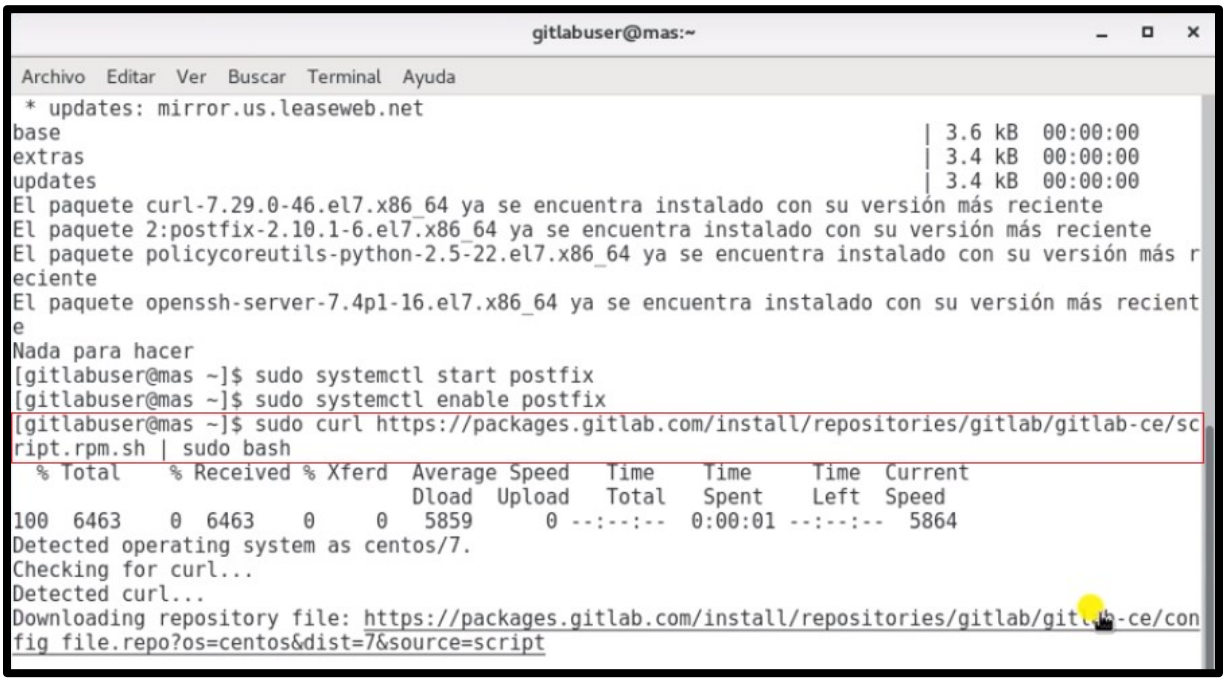
```
sudo systemctl enable postfix
```



Paso 2: Instalar GitLab

1. GitLab provee un instalador para agregar el repositorio oficial de GitLab. Lo único que se debe de hacer es descargar el instalador con CURL y ejecutar el script, tal y como se muestra a continuación:

```
sudo curl https://packages.gitlab.com/install/repositories/gitlab/gitlab-ce/script.rpm.sh | sudo bash
```



2. Para instalar GitLab se debe ejecutar la siguiente línea de comando y esperar a que termine el proceso:

```
sudo yum -y install gitlab-ce
```

3. Cuando la instalación se haya completado, se debe ser capaz de ver como resultado una pantalla similar a la que se muestra a continuación:

```
Thank you for installing GitLab!
GitLab was unable to detect a valid hostname for your instance.
Please configure a URL for your GitLab instance by setting `external_url`
configuration in /etc/gitlab/gitlab.rb file.
Then, you can start your GitLab instance by running the following command:
  sudo gitlab-ctl reconfigure

For a comprehensive list of configuration options please see the Omnibus GitLab readme
https://gitlab.com/gitlab-org/omnibus-gitlab/blob/master/README.md

  Comprobando   : gitlab-ce-11.3.6-ce.0.el7.x86_64                                                                1/1

Instalado:
  gitlab-ce.x86_64 0:11.3.6-ce.0.el7

¡Listo!
```

Paso 3: Configurar un Nombre de Dominio para GitLab

Configurar un dominio que permita identificar el servidor GitLab, en este caso se usará el dominio "*gitlab.mas.com*".

1. Ir al directorio "*etc/gitlab*", ejecutando la siguiente línea de comando en el editor VIM:

```
cd /etc/gitlab/
```

2. Abrir el archivo de configuración "*gitlab.rb*".

```
sudo vi gitlab.rb
```

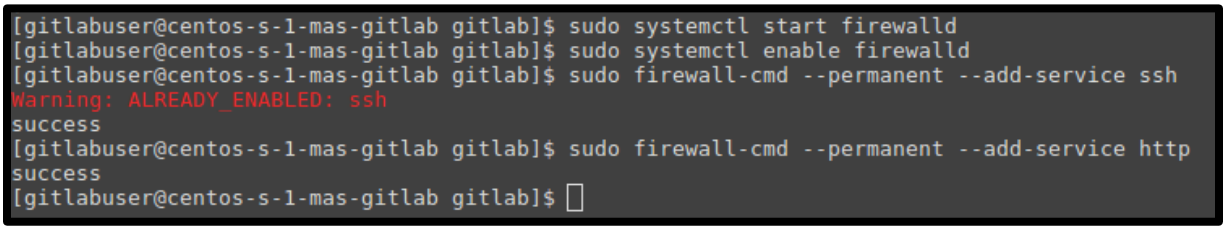
3. Para completar la configuración en el archivo de configuración "*gitlab.rb*", cambiar la línea **`external_url`** por la dirección del dominio.

```
## GitLab configuration settings
##! This file is generated during initial installation and **is not** modified
##! during upgrades.
##! Check out the latest version of this file to know about the different
##! settings that can be configured by this file, which may be found at:
##! https://gitlab.com/gitlab-org/omnibus-gitlab/raw/master/files/gitlab-config-template/gitlab.rb.template


## GitLab URL
##! URL on which GitLab will be reachable.
##! For more details on configuring external_url see:
##! https://docs.gitlab.com/omnibus/settings/configuration.html#configuring-the-external-url-for-gitlab
external_url 'http://gitlab.mas.com'

## Roles for multi-instance GitLab
##! The default is to have no roles enabled, which results in GitLab running as an all-in-one instance.
##! Options:
##!   redis_sentinel_role redis_master_role redis_slave_role geo_primary_role geo_secondary_role
##! For more deatils on each role, see:
##! https://docs.gitlab.com/omnibus/roles/README.html#roles
##!
# roles ['redis_sentinel_role', 'redis_master_role']

## Legend
##! The following notations at the beginning of each line may be used to
##! differentiate between components of this file and to easily select them using
##! a regex.
##! ## Titles, subtitles etc
##! ##! More information - Description, Docs, Links, Issues etc.
##! Configuration settings have a single # followed by a single space at the
##! beginning; Remove them to enable the setting.

##! **Configuration settings below are optional.**
##! **The values currently assigned are only examples and ARE NOT the default
##!   values.**


################################################################################
################################################################################
##                Configuration Settings for GitLab CE and EE                 ##
################################################################################
################################################################################

################################################################################
## gitlab.yml configuration
##! Docs: https://gitlab.com/gitlab-org/omnibus-gitlab/blob/master/doc/settings/gitlab.yml.md
################################################################################
# gitlab_rails['gitlab_ssh_host'] = 'ssh.host_example.com'
# gitlab_rails['time_zone'] = 'UTC'
```

Paso 4: Configuración del Cortafuegos:

1. Se deben de agregar los puertos pertinentes al cortafuegos para acceder al servicio, generalmente el puerto ssh viene agregado por defecto. Para lo anterior ejecutar los siguientes comandos:

```
sudo systemctl start firewalld
sudo systemctl enable firewalld
sudo firewall-cmd --permanent --add-service ssh
sudo firewall-cmd --permanent --add-service http
```

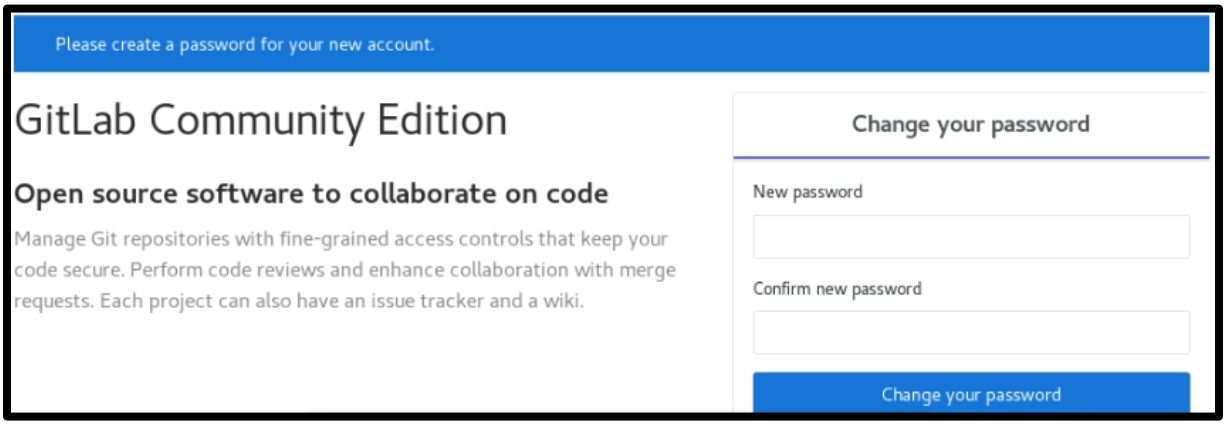


2. Para que se termine la instalación y asegurar que todas las configuraciones queden realizadas correctamente, ejecutar el siguiente comando:

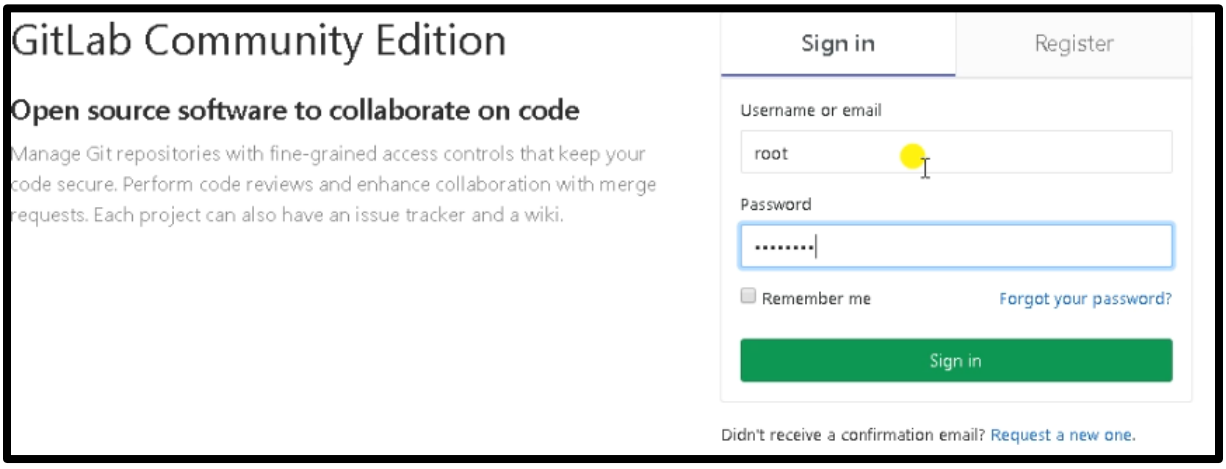
```
sudo gitlab-ctl reconfigure
```

Paso 5: Configuraciones post-instalación

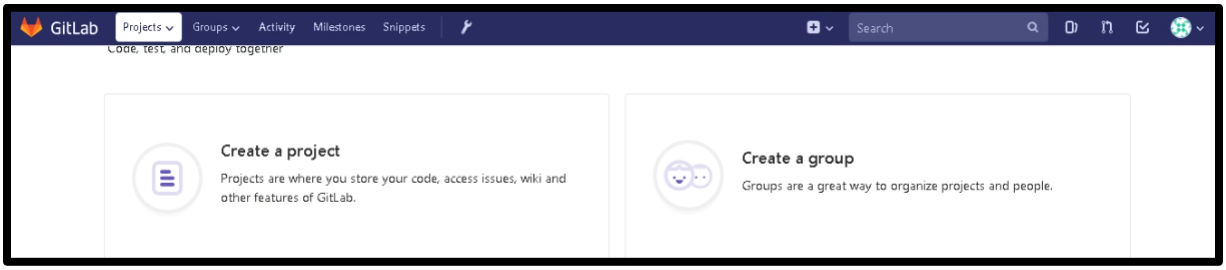
1. Restablecer la contraseña de usuario "*root*" del servidor GitLab, ingresando al servidor GitLab por medio de un navegador y la dirección, en este caso "*gitlab.mas.com*", lo cual abrirá la página web:



2. Ingresar una contraseña segura, que serán las credenciales para ingresar en el inicio de sesión:



3. Por último, se tendrá acceso a la consola administrativa de GitLab, desde la cual se pueden crear grupos, proyectos, configuraciones especiales, entre otras funcionalidades.


GitLab
Projects
Groups
Activity
Milestones
Snippets
Search
Code, test, and deploy together
Create a project
Projects are where you store your code, access issues, wiki and other features of GitLab.
Create a group
Groups are a great way to organize projects and people.

Proceso de instalación y configuración inicial de SonarQube

Prerrequisitos:

Hardware

- 2 GB de RAM
- 1 CPU
- 1 Interfaz de red
- Disco duro de 20 GB

Software

- CentOS 7, 64bit
- SonarQube 6.4
- Comando **unzip** instalado
- Usuario con privilegios de súper administrador.
- Tener instalado el comando **wget** y **firewall-cmd** en el sistema operativo.

Nota: Se está utilizando para el sistema operativo el usuario "*sonaruser*" como usuario administrador.

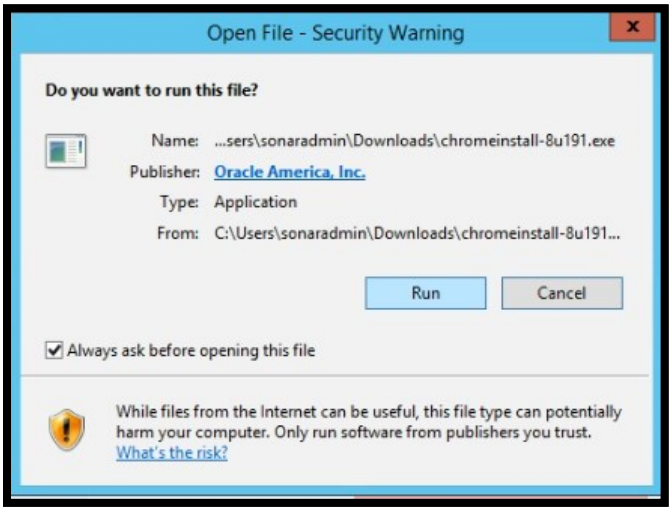
Paso 1. Descarga e Instalación de Java

1. Descargar el "*Java Runtime Environment*" (JRE) de Java desde el sitio oficial de Oracle:

`www.java.com.en/download/`

2. Dar clic en el botón de descarga, reconocerá la compatibilidad más adecuada.

3. Ejecutar el instalador dando clic en "*Run*".



4. Dar clic en siguiente o "*Next*" y aceptar los términos de la licencia dando clic en el botón "*Ok*".

5. Esperar a que la instalación termine y dar clic en el botón de cerrar o "*Close*":

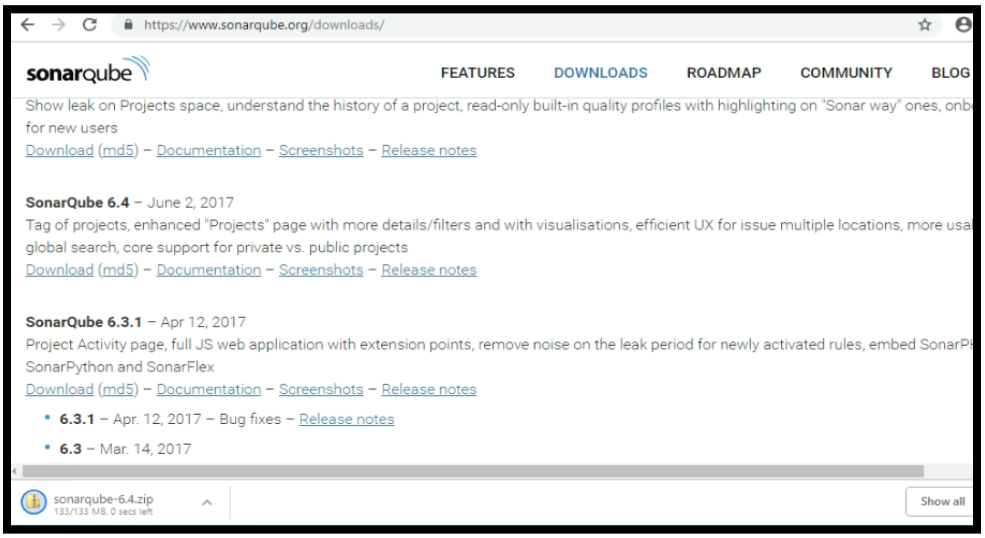

Java Setup - Progress
Java
Status: Installing Java
3 Billion
Devices Run Java
Java
#1 Development Platform
ORACLE


Java Setup - Complete
Java
You have successfully installed Java
You will be prompted when Java updates are available. Always install updates to get the latest performance and security improvements.
More about update settings
Close

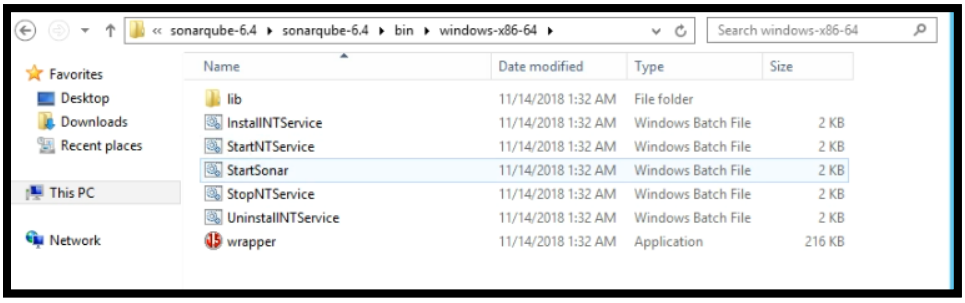
Paso 2. Descarga e Instalación de SonarQube

1. Descargar e instalar SonarQube desde el sitio oficial:

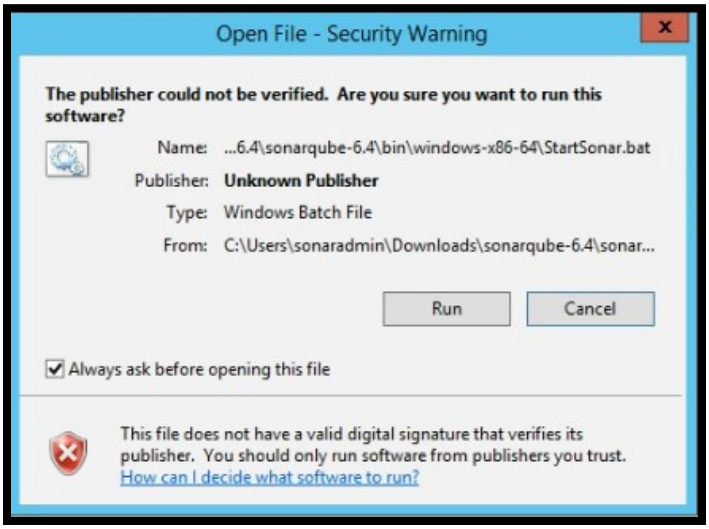
`www.sonarqube.org/downloads/`



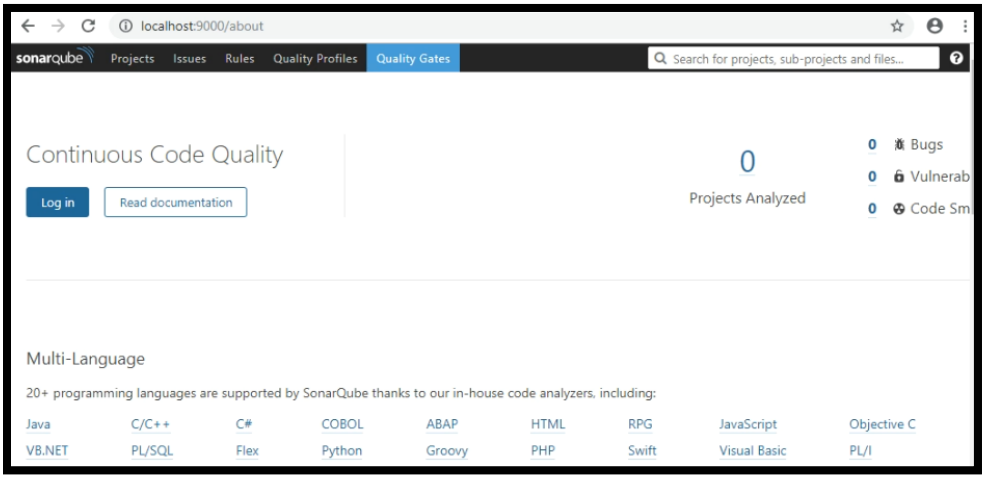
2. Extraer el contenido y dirigirse al directorio el tipo de instalación del sistema operativo, en este caso windows-x86-64.



3. Ejecutar el archivo batch de Windows "*StartSonar*" dando clic en "*Run*".



4. Ingresar a la consola administrativa de SonarQube.



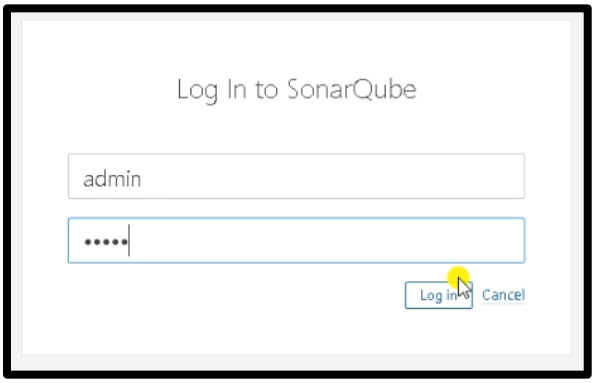
5. SonarQube está instalado correctamente en el servidor, y se accede mediante la dirección siguiente:

http://sonar.yourdomain.com

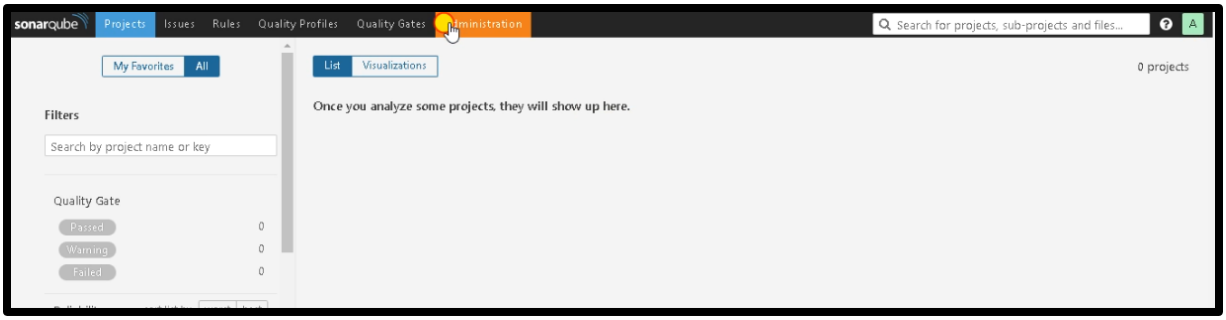
6. Hay que recordar que, si se está trabajando en ambiente local, se debe configurar el dominio en el archivo hosts del sistema operativo.

Paso 3: Crear Usuario

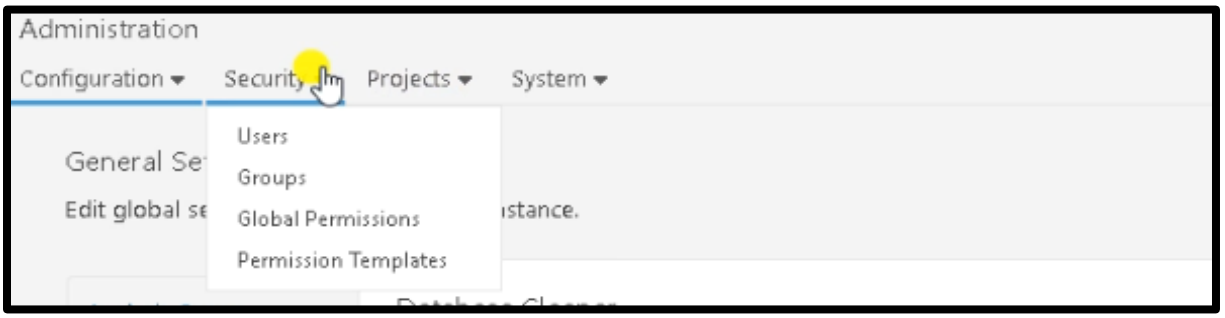
1. Iniciar sesión mediante la cuenta de administrador por defecto, la cual es con el usuario "*admin*" y clave "*admin*".



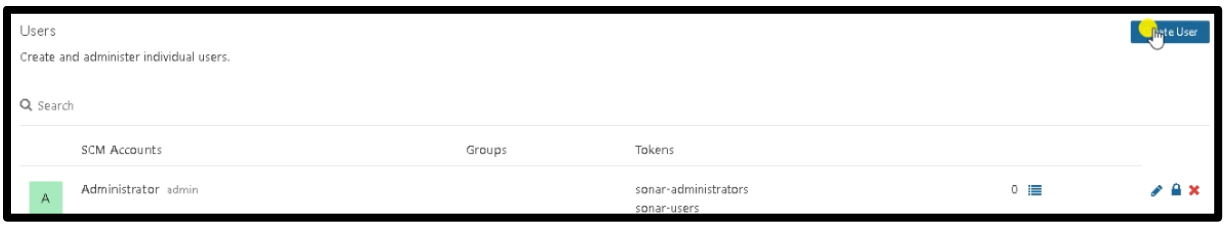
2. Ir a la opción "*Administration*":



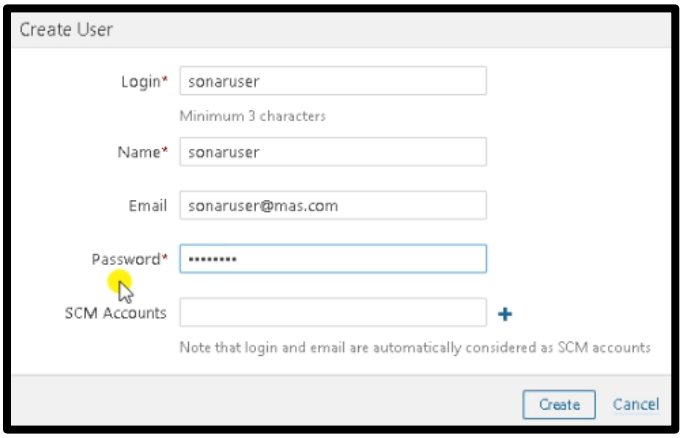
3. Seleccionar Security e ingresar a la opción "*Users*":



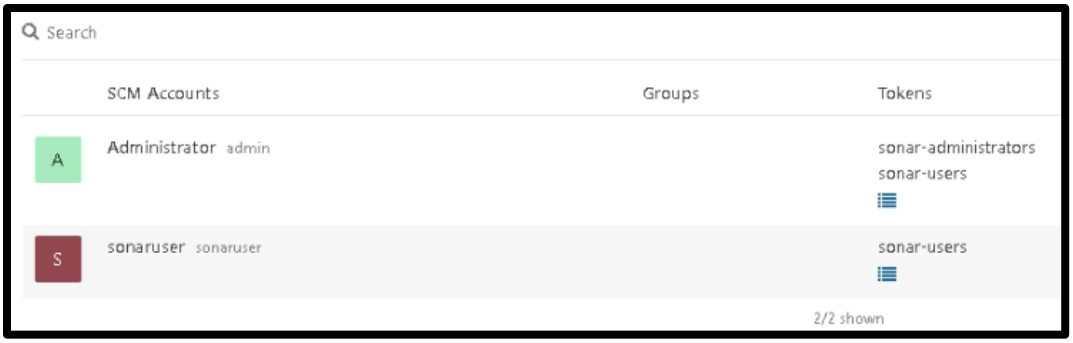
4. Dar clic en la opción "*Create User*":



5. Ingresar los datos correspondientes y hacer clic en "*Create*":



6. Después de eso ya está creado el usuario:



Proceso de instalación y configuración del Repositorio JFrog Artifactory

Prerrequisitos

Hardware

- 2 GB de RAM
- 1 CPU
- 1 interfaz de red
- Disco duro de 10 GB

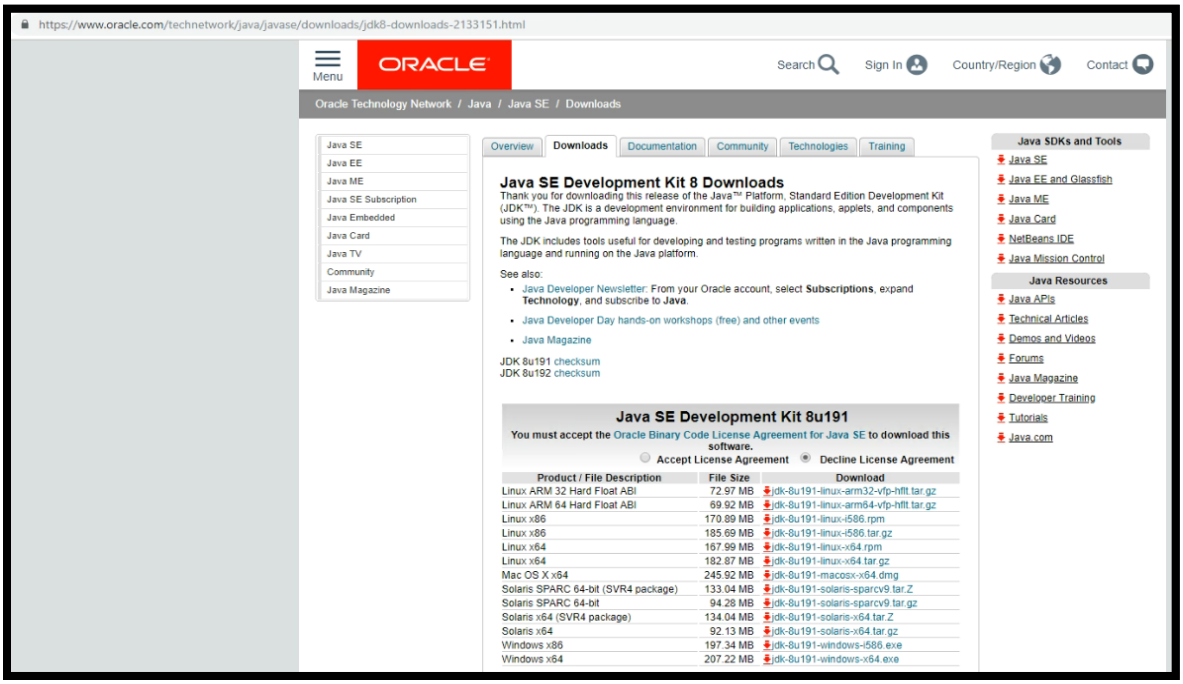
Software

- Windows Server 2012 R2.
- Usuario con privilegios de súper administrador.

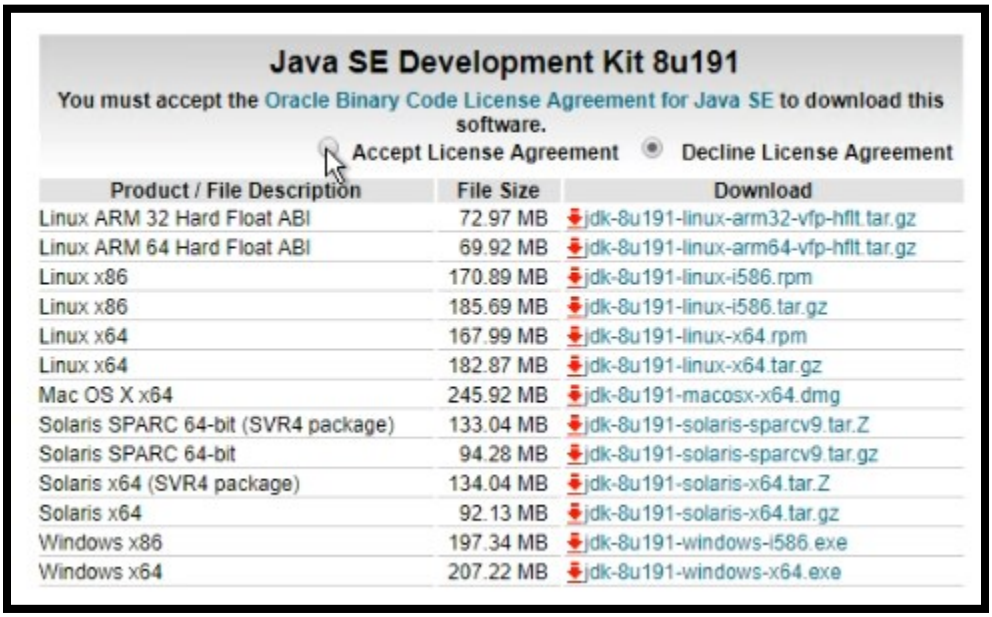
Nota: Se está utilizando para el sistema operativo el usuario "*artifuser*" como usuario administrador.

Paso 1: Instalar Java (JDK)

1. Para la instalación no basta con descargar el JRE, para este se debe de descargar el JDK completo, el cual ya incluye un JRE, desde el sitio oficial de Oracle.



2. Se deben de aceptar los términos de licencia de Java:



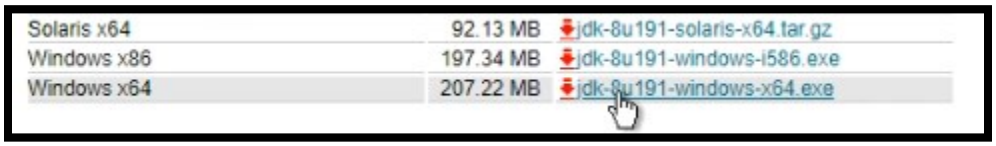
Java SE Development Kit 8u191

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

Accept License Agreement Decline License Agreement

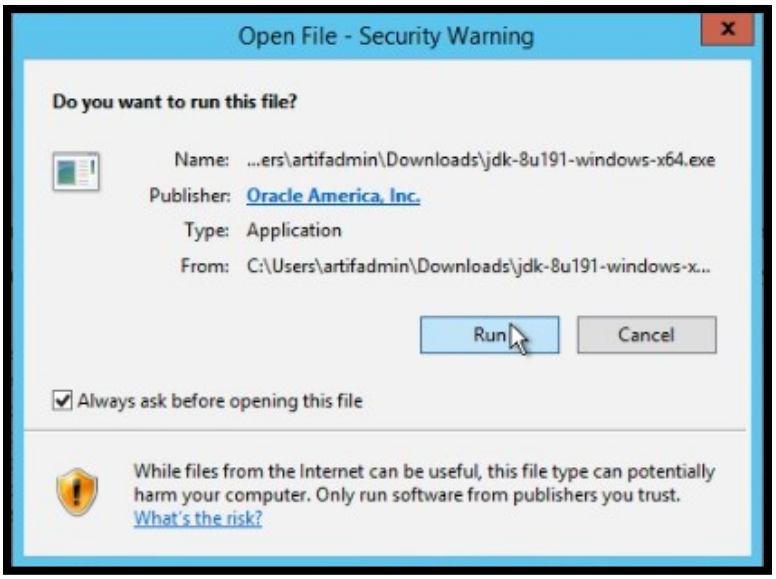
Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	72.97 MB	jdk-8u191-linux-arm32-vfp-hflt.tar.gz
Linux ARM 64 Hard Float ABI	69.92 MB	jdk-8u191-linux-arm64-vfp-hflt.tar.gz
Linux x86	170.89 MB	jdk-8u191-linux-i586.rpm
Linux x86	185.69 MB	jdk-8u191-linux-i586.tar.gz
Linux x64	167.99 MB	jdk-8u191-linux-x64.rpm
Linux x64	182.87 MB	jdk-8u191-linux-x64.tar.gz
Mac OS X x64	245.92 MB	jdk-8u191-macosx-x64.dmg
Solaris SPARC 64-bit (SVR4 package)	133.04 MB	jdk-8u191-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	94.28 MB	jdk-8u191-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	134.04 MB	jdk-8u191-solaris-x64.tar.Z
Solaris x64	92.13 MB	jdk-8u191-solaris-x64.tar.gz
Windows x86	197.34 MB	jdk-8u191-windows-i586.exe
Windows x64	207.22 MB	jdk-8u191-windows-x64.exe

3. Seleccionar la versión del producto a descargar, en este caso es Windows x64:



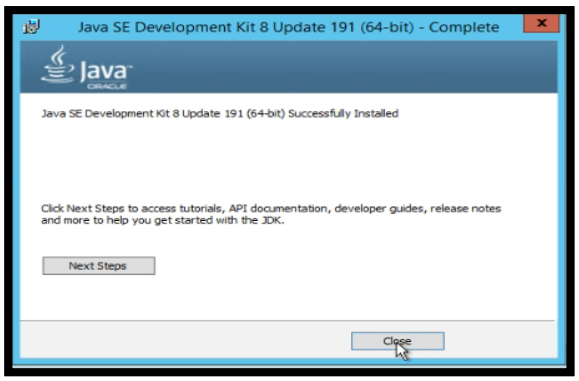
Solaris x64	92.13 MB	jdk-8u191-solaris-x64.tar.gz
Windows x86	197.34 MB	jdk-8u191-windows-i586.exe
Windows x64	207.22 MB	jdk-8u191-windows-x64.exe

4. Ejecutar el archivo recién descargado dando clic en "*Run*".



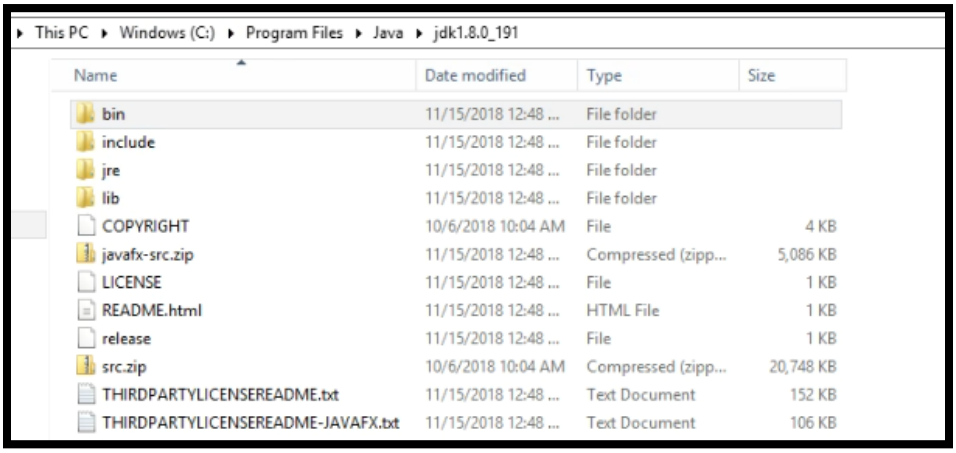
5. Hacer clic en el botón de siguiente o "*Next*":

6. Se debe de esperar a que la instalación termine correctamente:

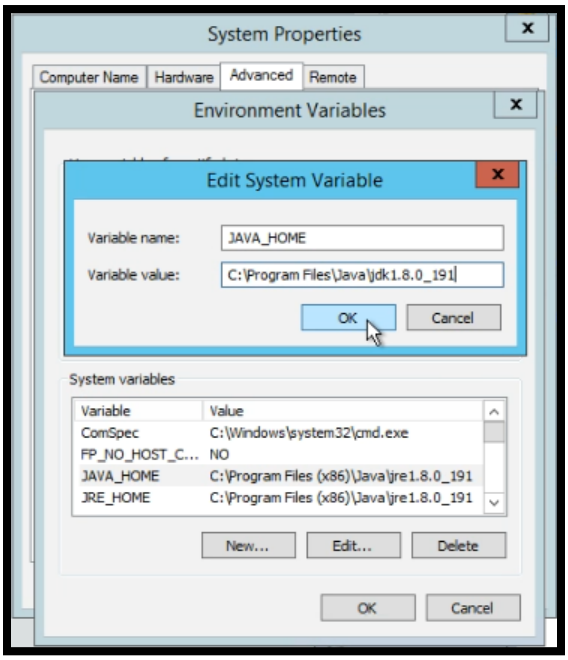


Paso 2: Configurar variables de entorno

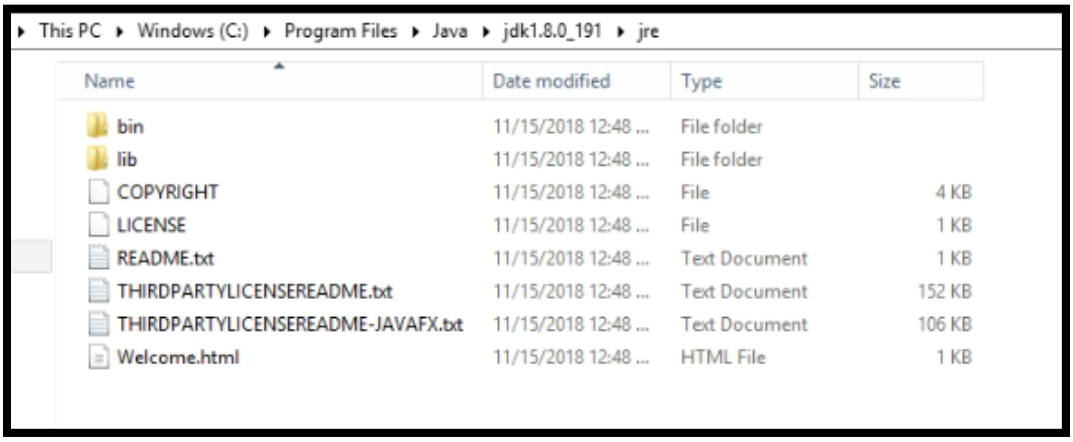
1. Configurar "*JAVA_HOME*" en las variables de entorno según la instalación:



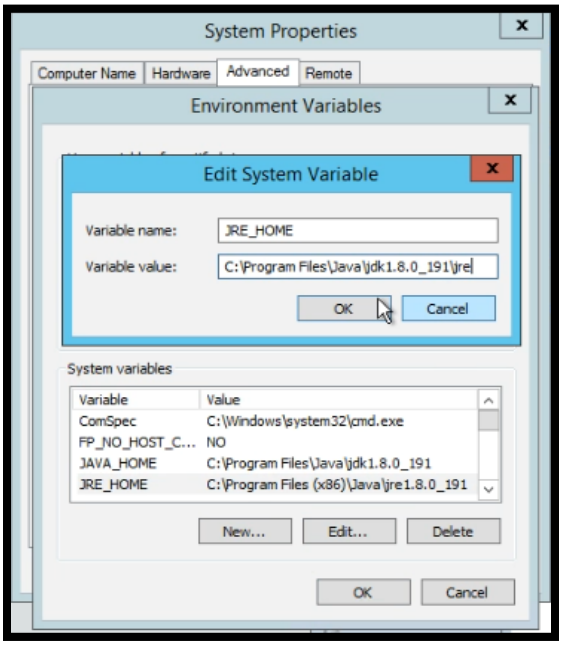
2. En las propiedades del sistema > avanzado > variables de ambiente, se debe crear una nueva variable y colocarle "*JAVA_HOME*", junto con el la ubicación que contiene a la carpeta bin:



3. Configurar "*JAVA_JRE*" en las variables de entorno según la instalación:



4. En las propiedades del sistema > avanzado > variables de ambiente, se debe crear una nueva variable y colocarle "*JRE_HOME*" junto con el la ubicación que contiene al bin:



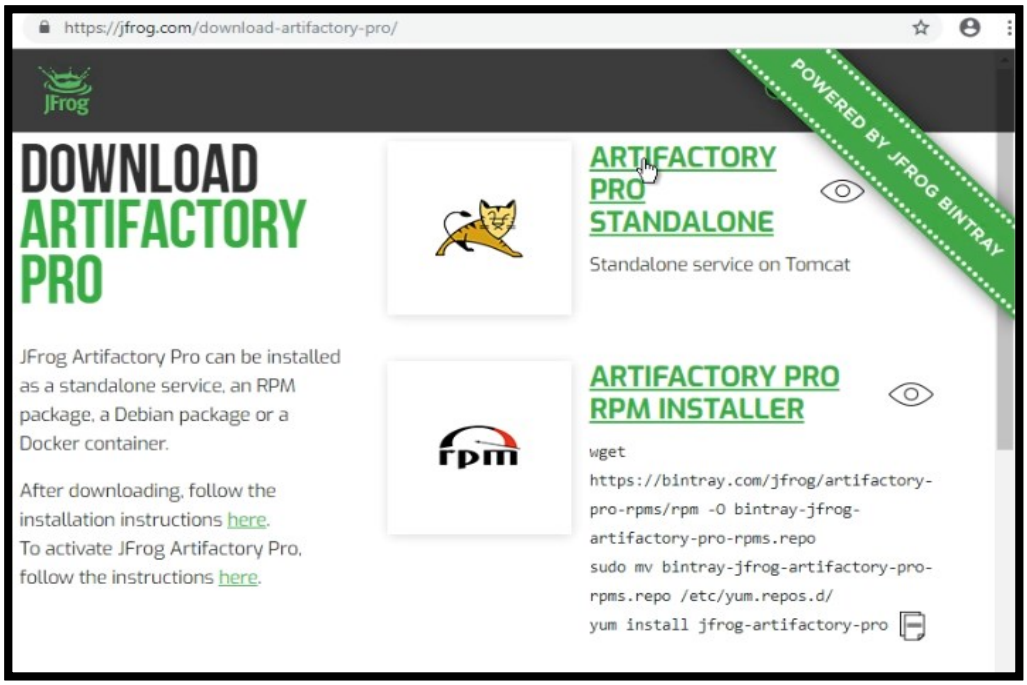
5. Estas variables deben ser adicionadas a la variable "*path*" del ambiente, es decir,

<valor_path>; + %JAVA_HOME%\bin;%JRE_HOME%\bin.

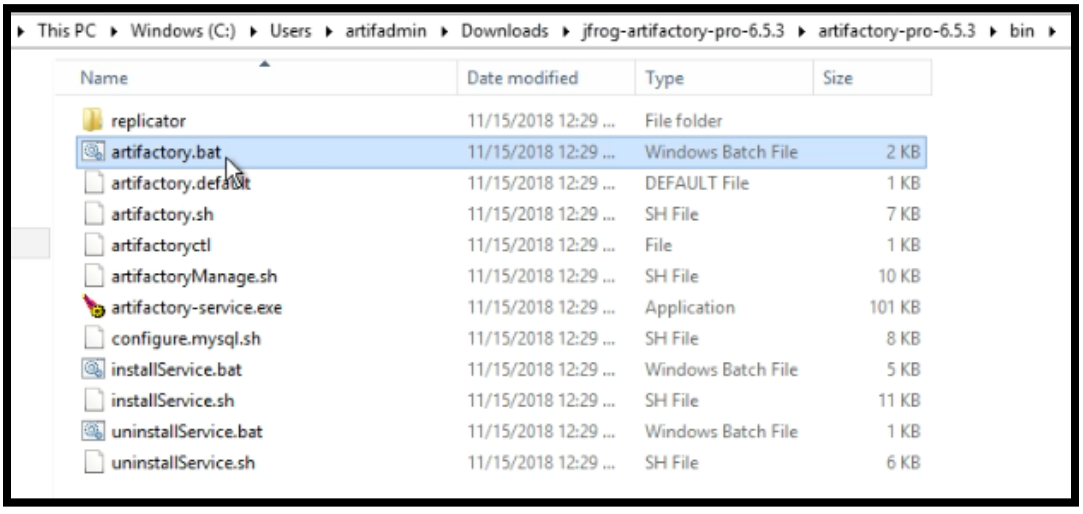
Paso 3: Instalar y Configurar JFrog Artifactory

1. Descargar JFrog Artifactory Pro desde el sitio oficial:

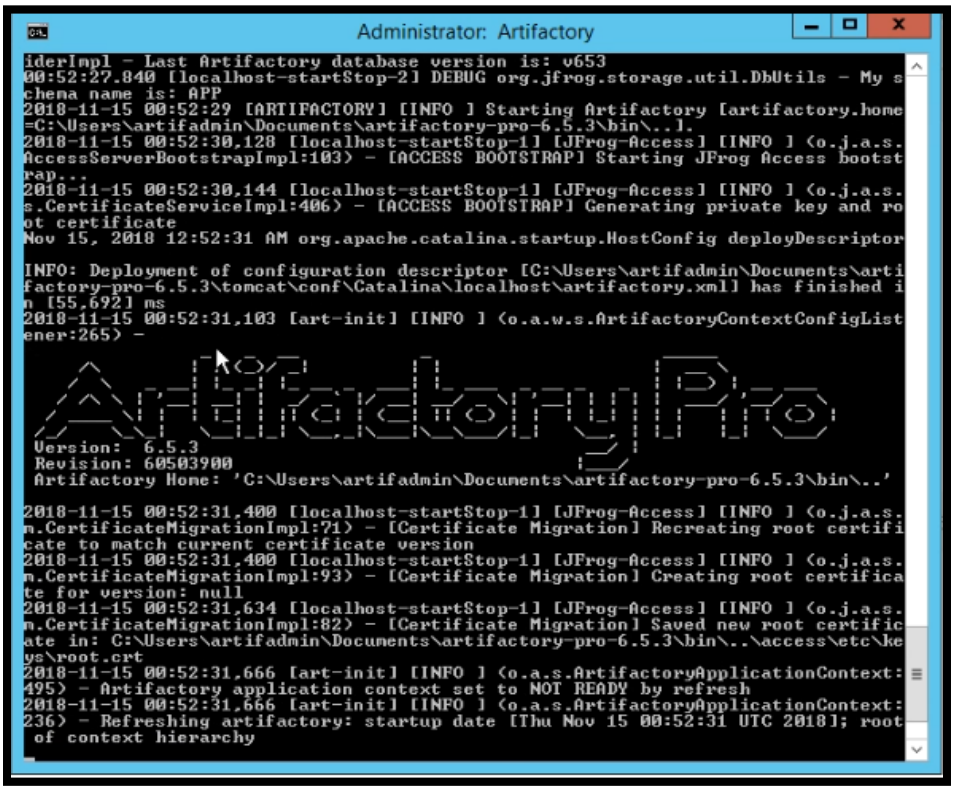
`https://jfrog.com/download-artifactory-pro/`



2. Se identifica y descomprime el archivo recién descargado, y se accede al directorio bin de Artifactory:

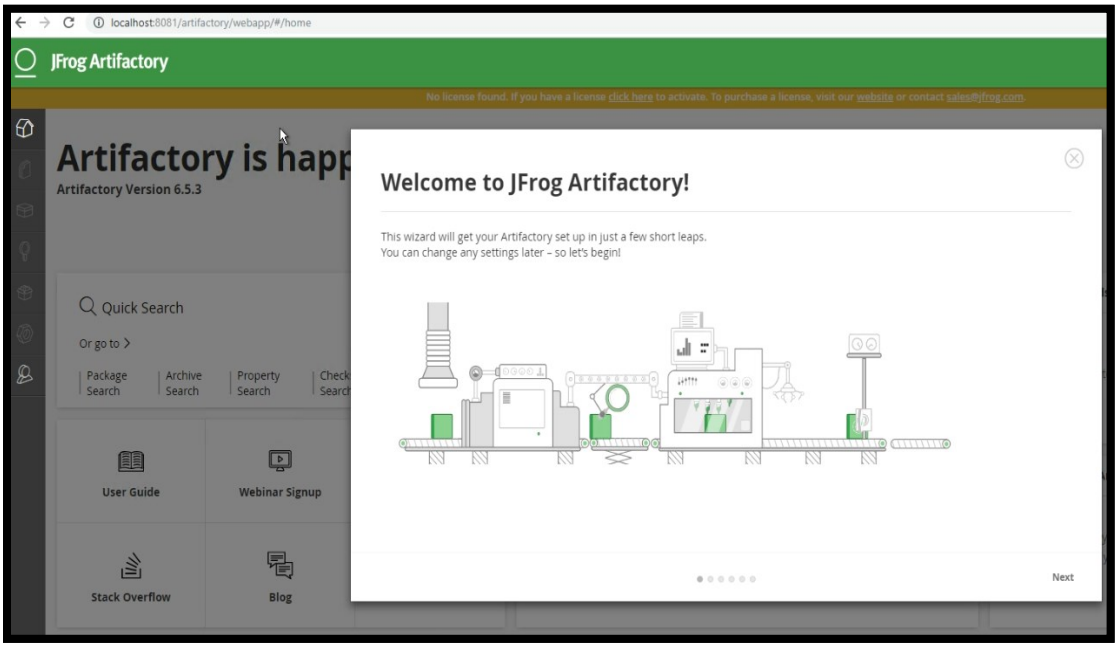


3. Luego desde una ventana de comandos, se debe de dirigir al directorio antes identificado y ejecutar el archivo “artifactory.bat” para iniciar la aplicación.

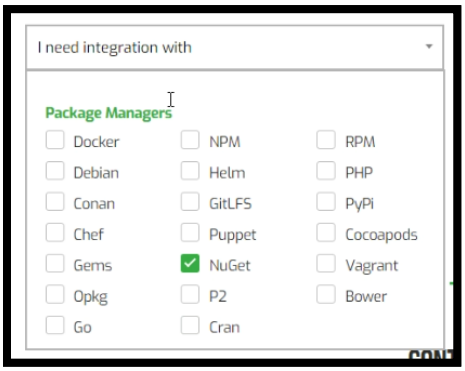


4. La aplicación deberá de indicar el momento que haya terminado de arrancar correctamente:

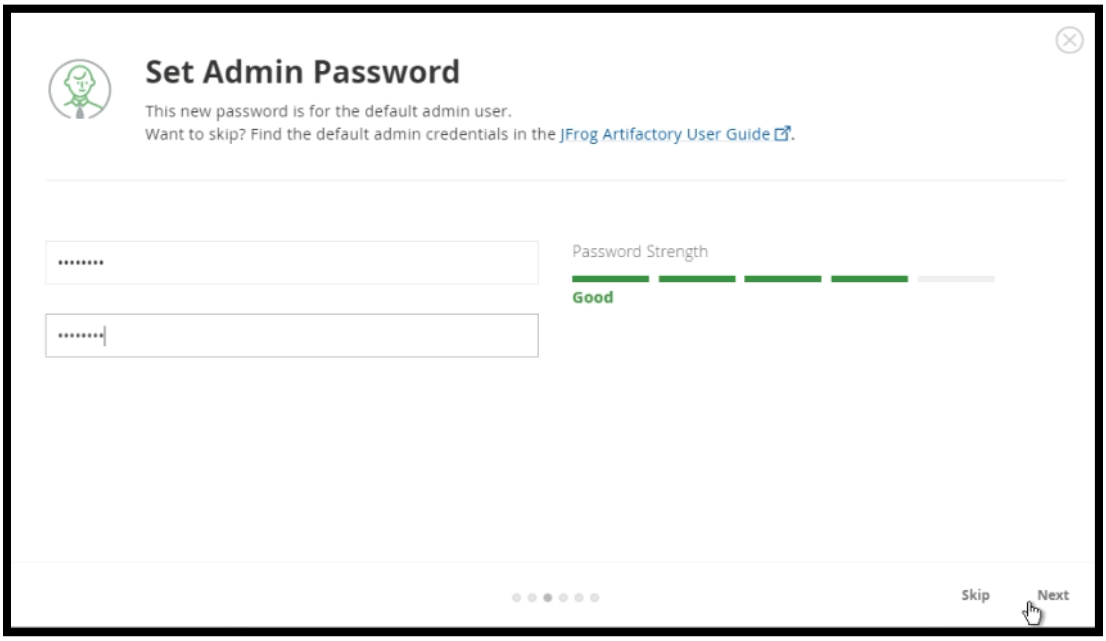
5. Después, se debe de acceder a la ubicación “localhost:8081/artifactory”:



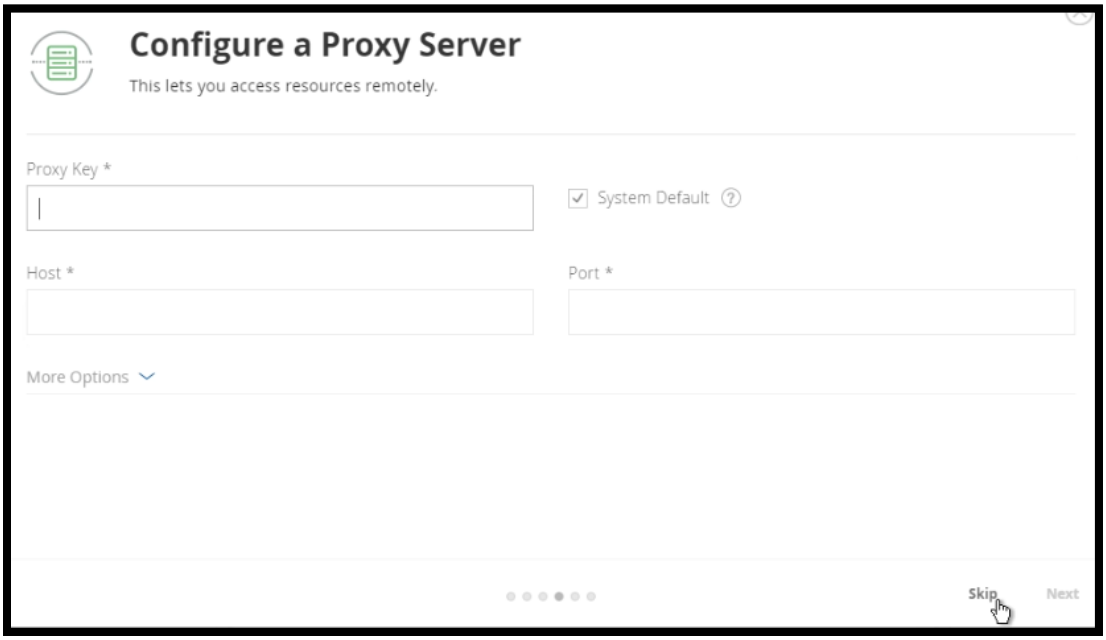
6. Al dar clic en siguiente o "*Next*", mostrará una página en donde se selecciona el tipo de versión "*ON-PREM VERSION*" o "*CLOUD VERSION*". Se seleccionará "*ON-PREM VERSION*".
7. Seleccionar las herramientas con las que se va a integrar, en este caso será con NuGet.



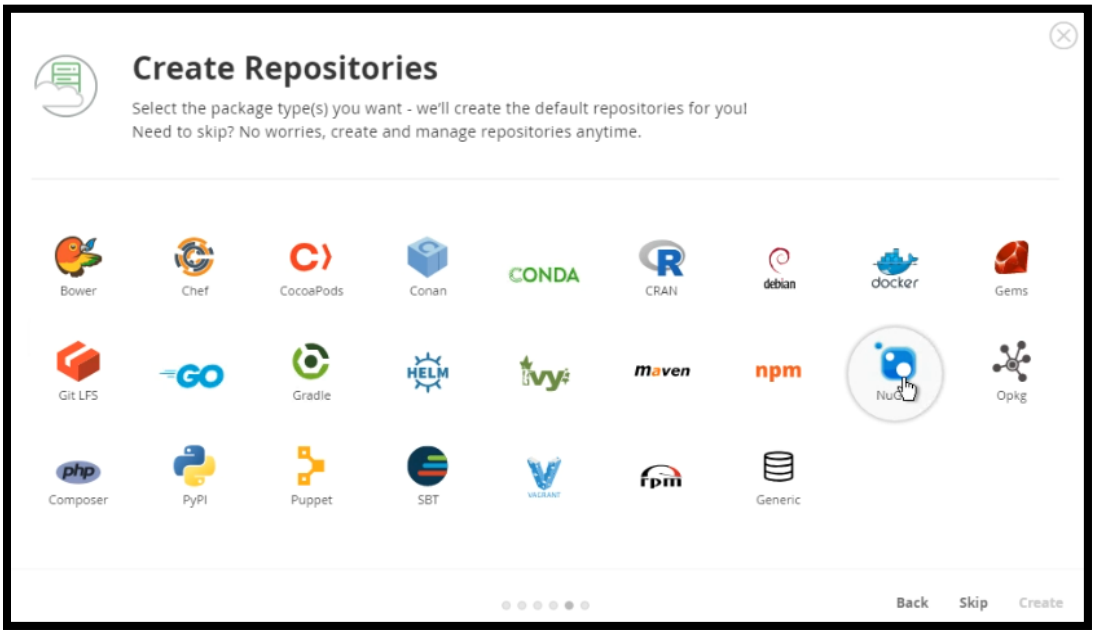
8. Se establece la contraseña de administrador del servidor y se da clic en "*Next*" para continuar.



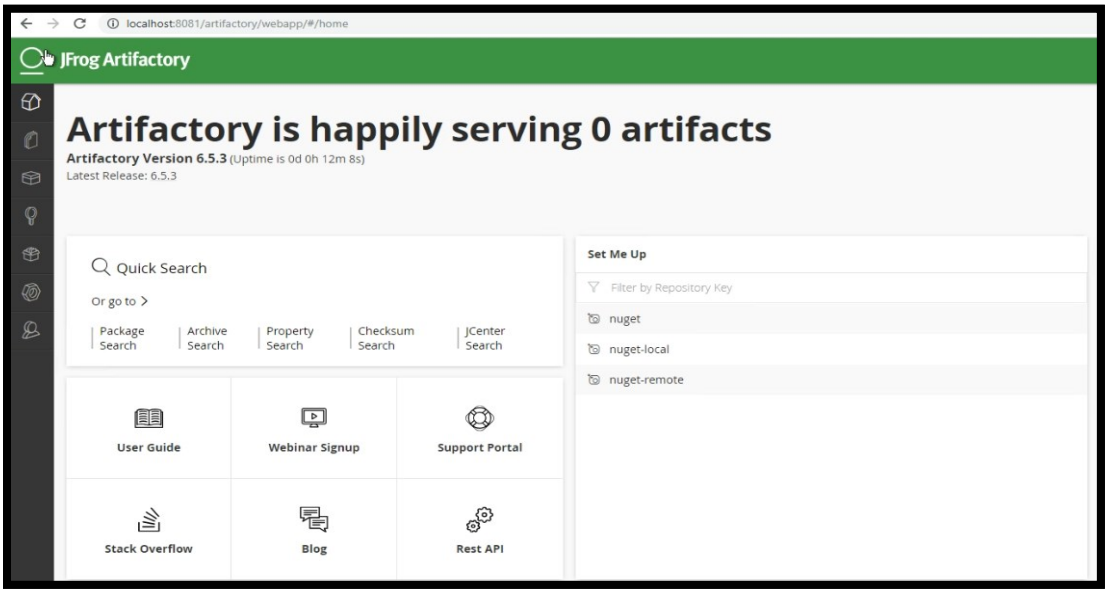
9. Se puede colocar la información del proxy, pero para este caso, será la configuración por defecto, por lo que se da clic en "S*kip*":



10. Crear el repositorio para la herramienta a integrar, seleccionando en este caso NuGet y dando clic en "*Create*":



11. Después de eso la instalación y configuración inicial está completa, por lo que se da clic en "*Finish*".
12. Finalmente se despliega la página de bienvenida en el panel de Artifactory.



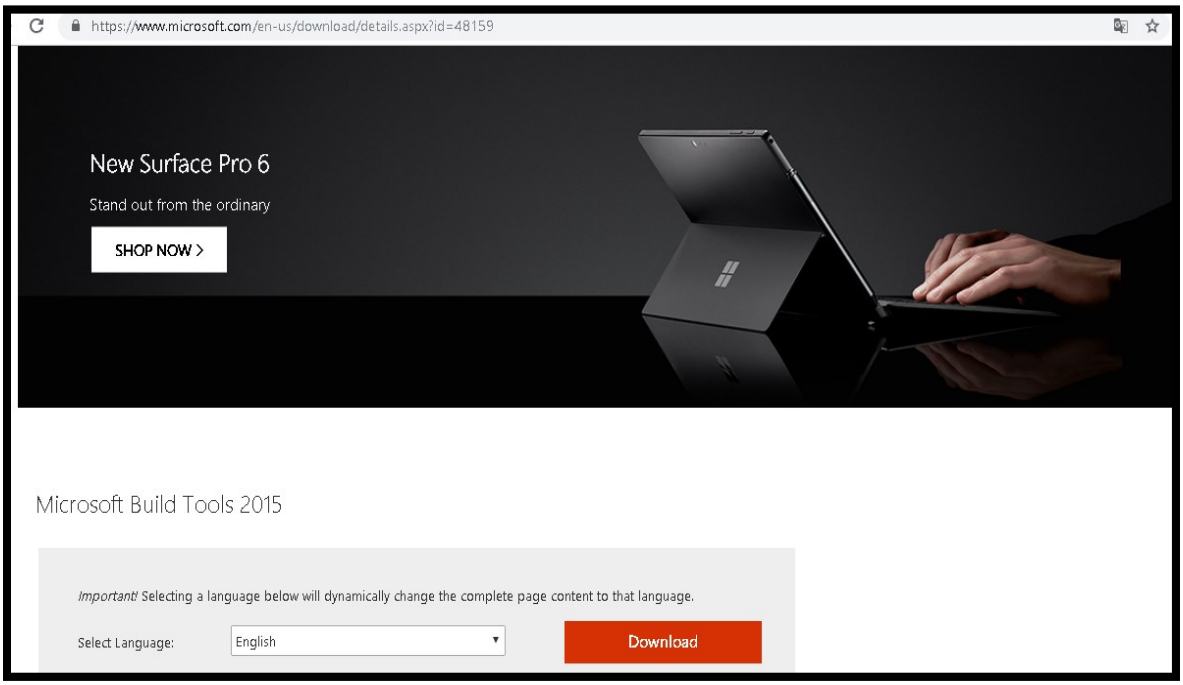
Integración de Herramientas

Antes de comenzar

Para la fase de integración entre las herramientas, primeramente, se debe tener completa certeza de la comunicación a nivel de red y de puertos de los componentes de los servidores.

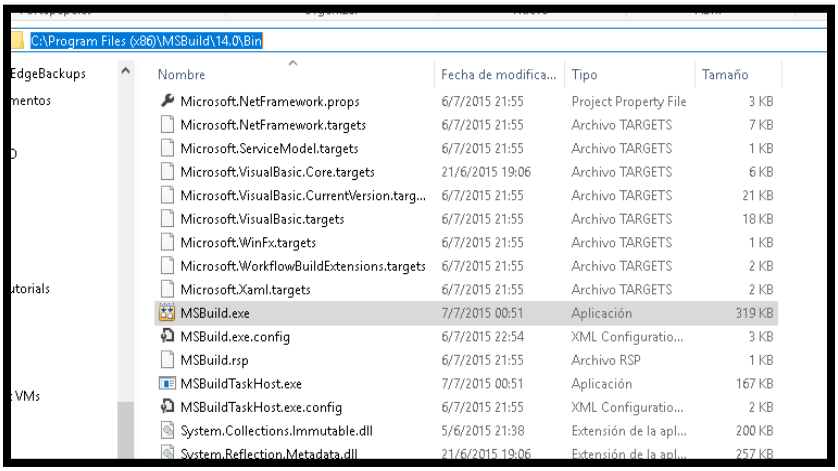
Paso 1. Compilación (MSBuild) y Servidor de Integración (Jenkins)

1. Se debe descargar MSBuild Tools, para este caso será el 2015, desde el sitio oficial de Microsoft:



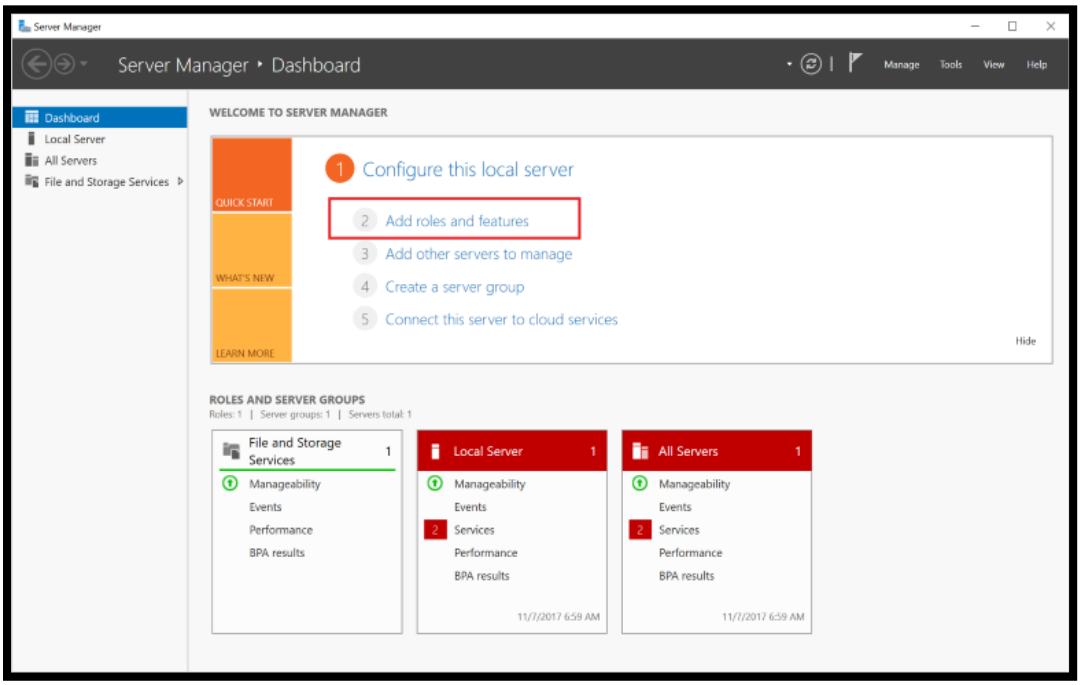
2. Al finalizar se ejecuta el archivo "*.exe*" y se desplegará un cuadro de diálogo de confirmación de la instalación, para lo cual se debe dar clic en "*install*".
3. Cuando finalice la instalación, es necesario dirigirse al directorio del ejecutable MSBuild.exe:

```
C:\Program Files (x86)\MSBuild\14.0\Bin
```

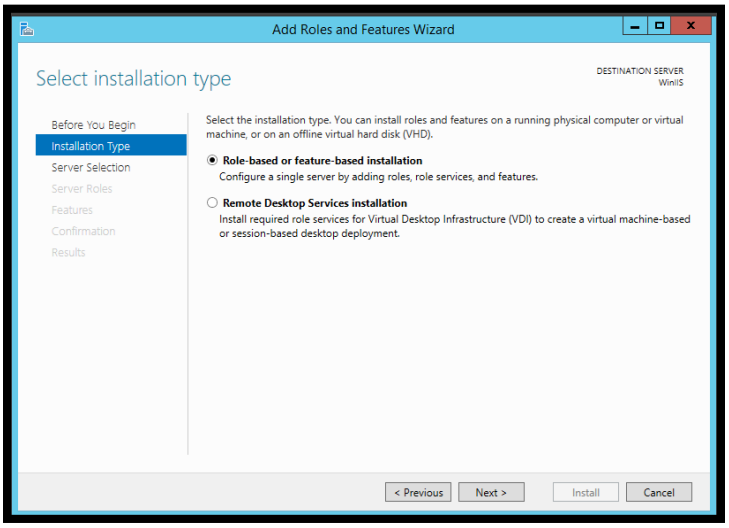


4. Para instalar .NET Framework 3.5 ingresar al "*Server Manager Dashboard*".

5. Dar clic en "*Add roles and features*":



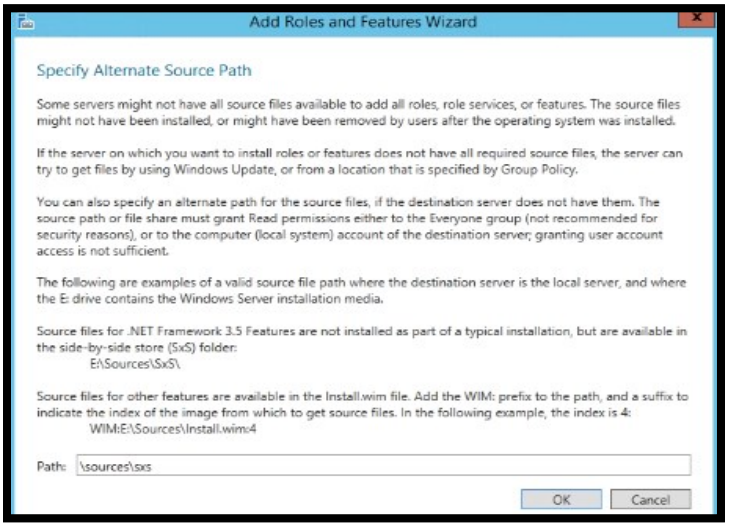
6. Seleccionar "*Role-based or feature based installation*" como el tipo de instalación. Continuar dando clic en "*Next*".



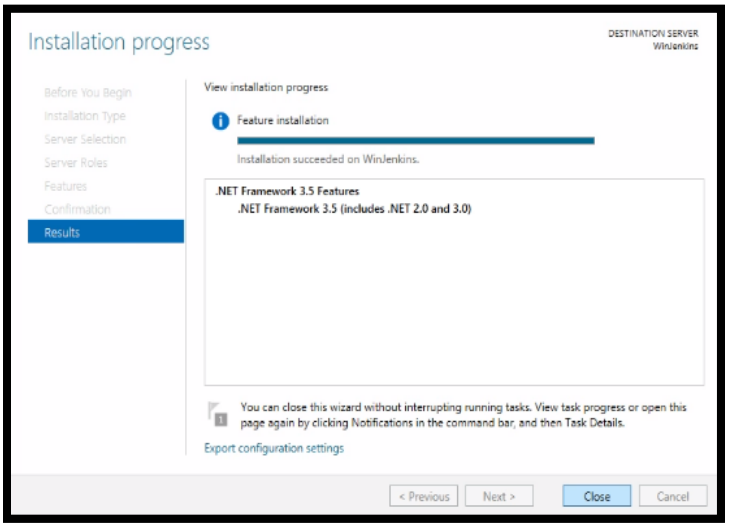
7. En la pestaña "*Server Selection*" seleccionar el servidor a configurar. Continuar dando clic en "*Next*".

8. En la pestaña "*Features*" seleccionar "*.NET Framework 3.5 Features*". Finalizar dando clic en "*Install*".

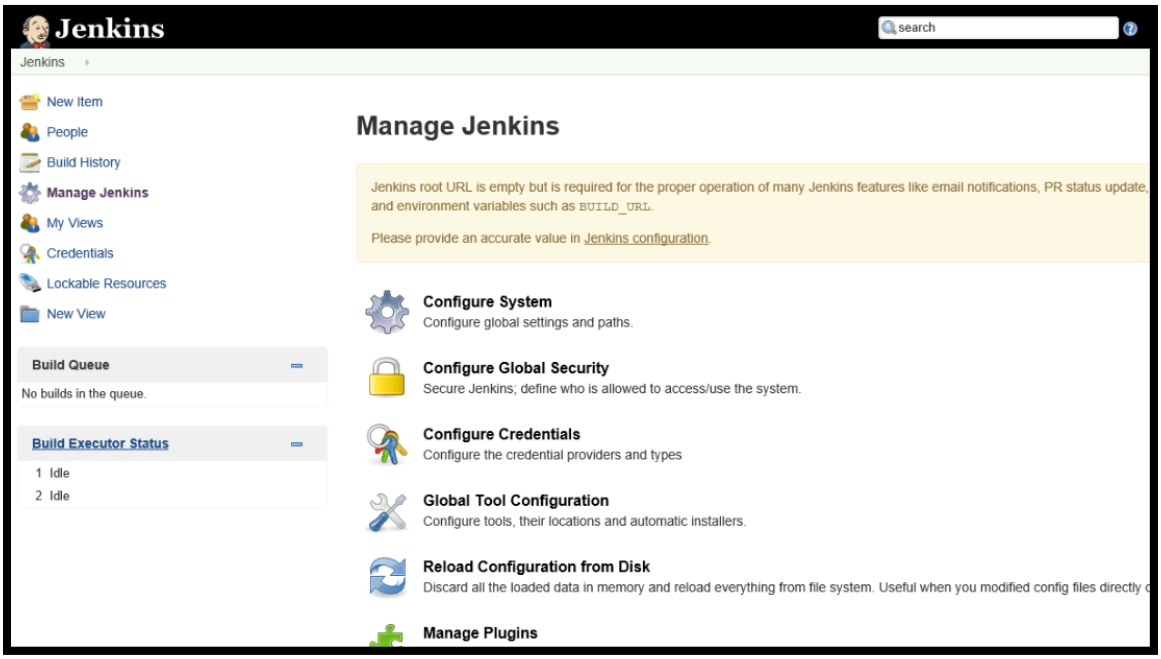
9. Se debe colocar el directorio para "*side-by-side*", se recomienda que se maneje "*\sources\sxs*":



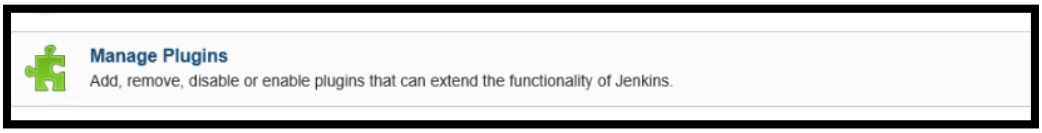
10. Se debe esperar a que la instalación termine y al finalizar dar clic en "*Close*".



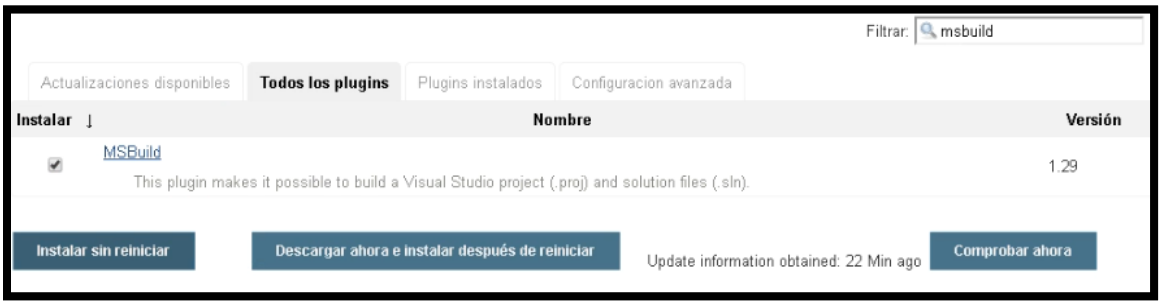
11. Luego de lo anterior, ya se procede a configurar en Jenkins el MSBuild. Dirigirse a la opción "*Manage Jenkins*" o Administrar Jenkins dentro de Jenkins:



12. Seleccionar la opción "*Manage Plugins*" o Administrar Complementos.



13. Buscar el complemento para "*MSBuild*".
14. Dar clic en Descargar ahora e instalar después de reiniciar.



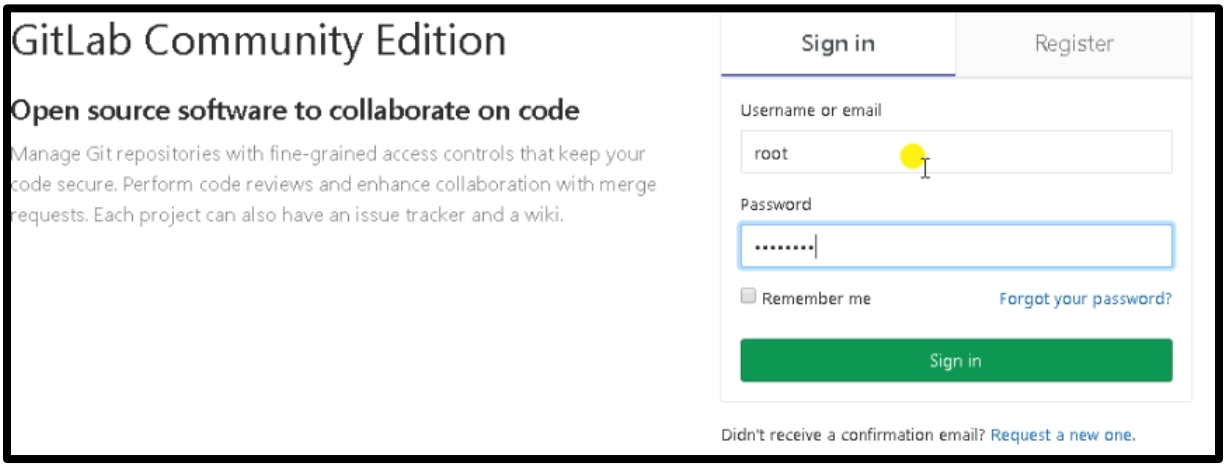
15. Reiniciar el servidor Jenkins, lo recomendable es reiniciarlo de manera segura, colocando /safeRestart en la barra de navegación del explorador.

```
localhost:8080/safeRestart
```

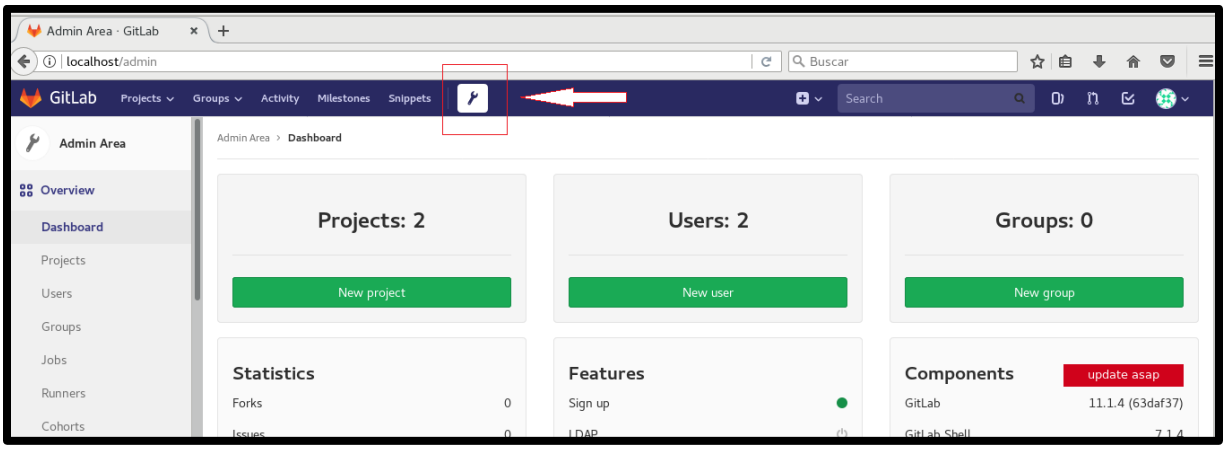
16. En Administrar Jenkins > Configurar el Sistema, se debe configurar la variable de entorno, en "*Path to MSBuild*" colocar la ubicación en el servidor del MSBuild.exe.

Paso 2. Administrador de repositorios (GitLab) y Servidor de integración (Jenkins)

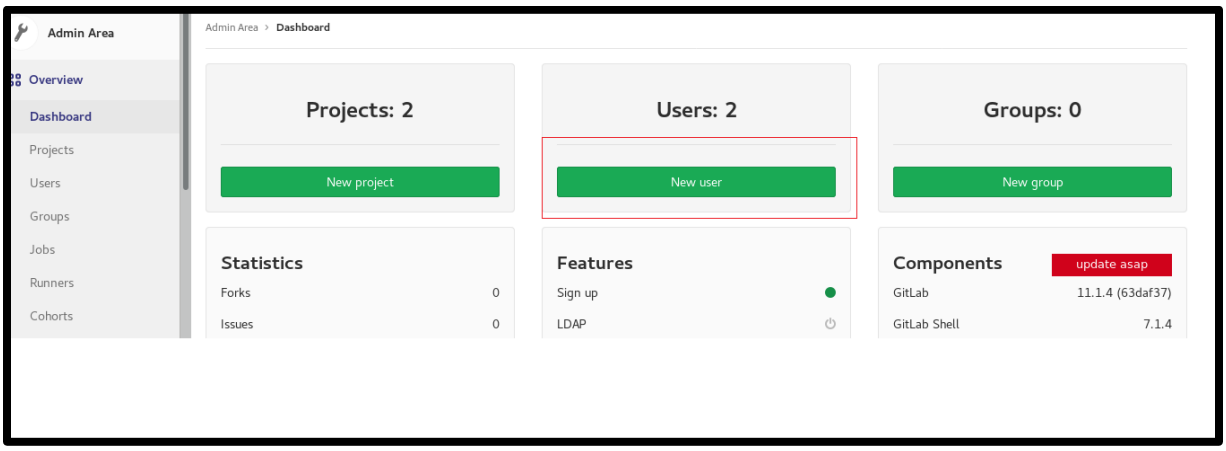
1. Ingresar al servidor GitLab utilizando el usuario "*root*":



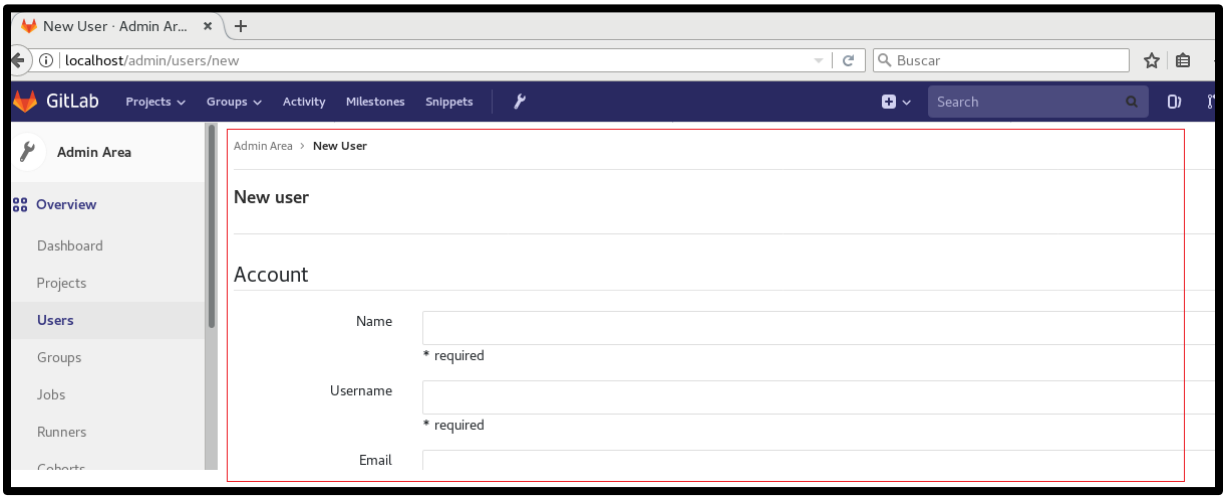
2. Ir a la opción "*Área Administrativa*", identificado con el botón de herramienta, para crear un usuario administrador:



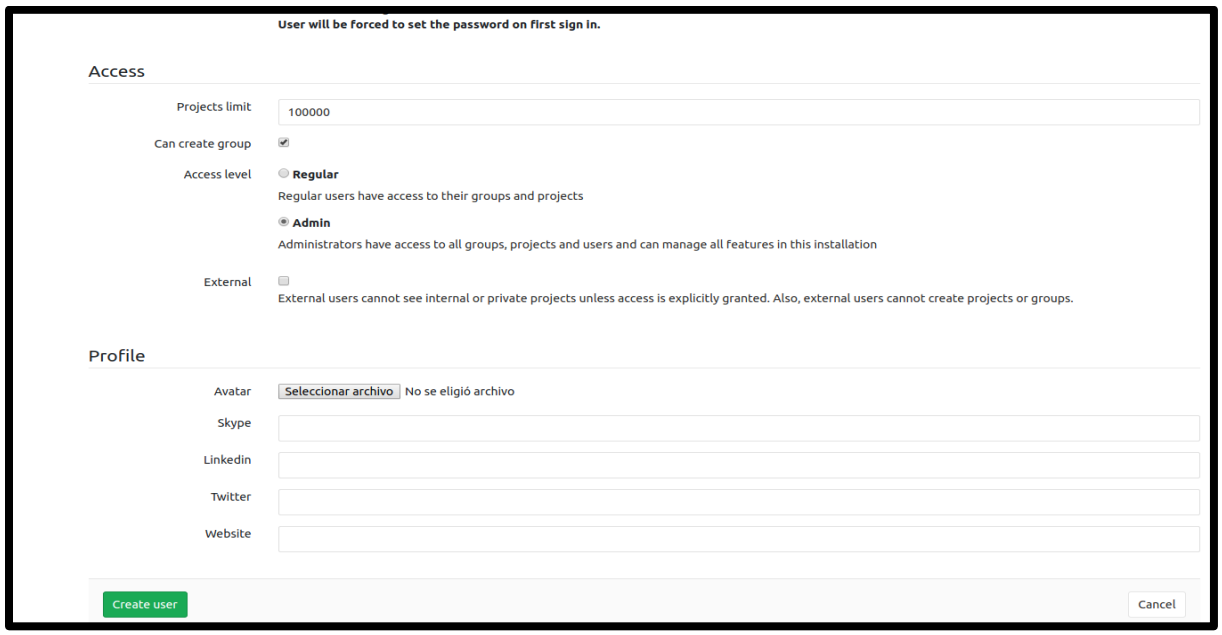
3. Hacer clic en "*New User*":



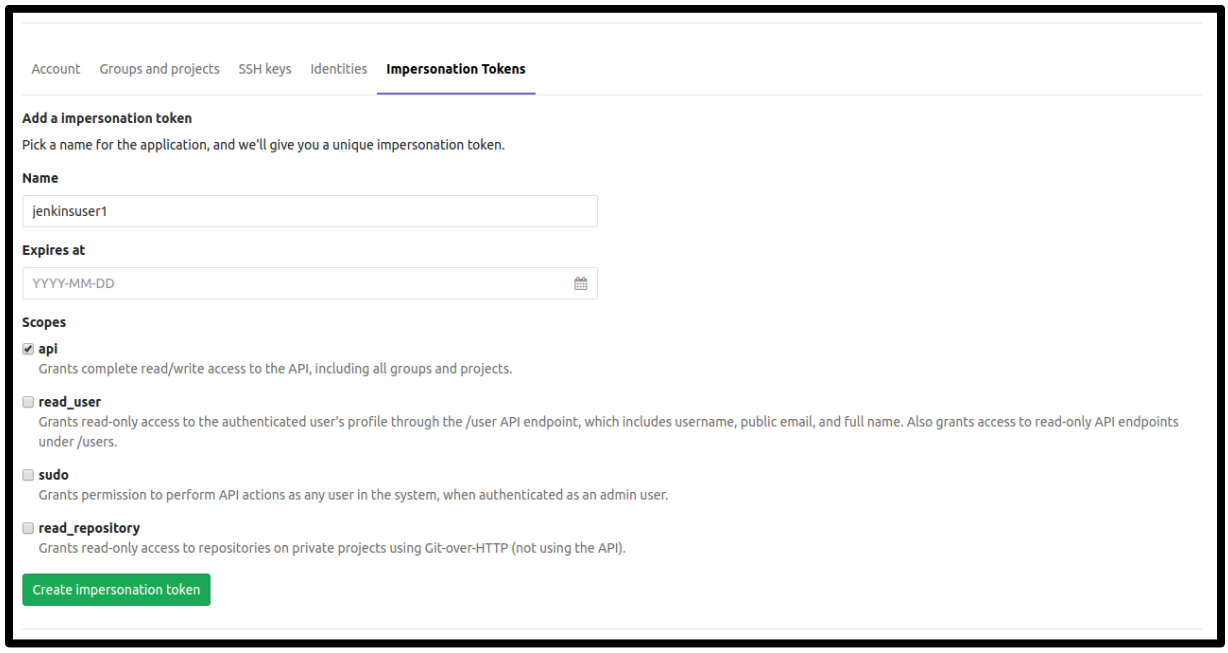
4. Completar el formulario:



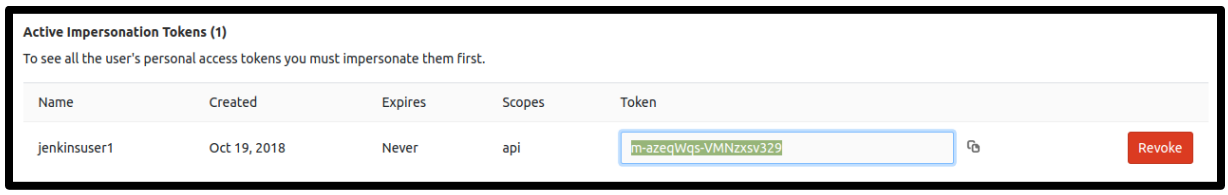
5. Se procede a la creación del usuario con nivel de acceso Admón. y se clic en "*Create user*":



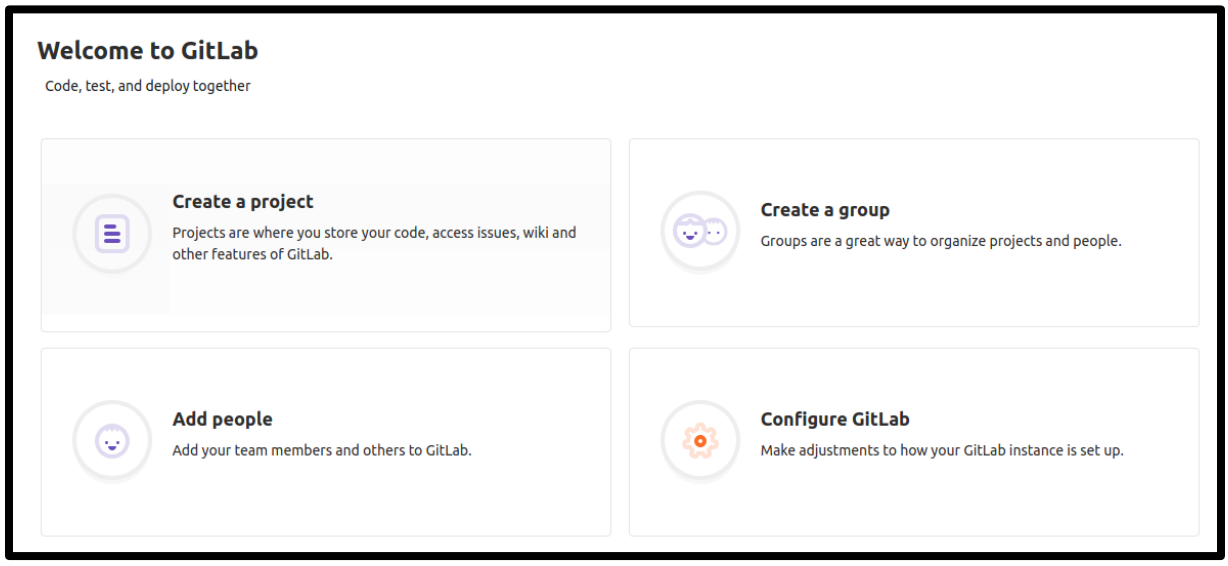
6. Ir a la opción "*Impersonation Tokens*" dentro de la configuración del usuario, para crear un *"Token"* de autenticación con alcance "*api*", permitiendo acceso completo, y fecha de caducidad indefinida:



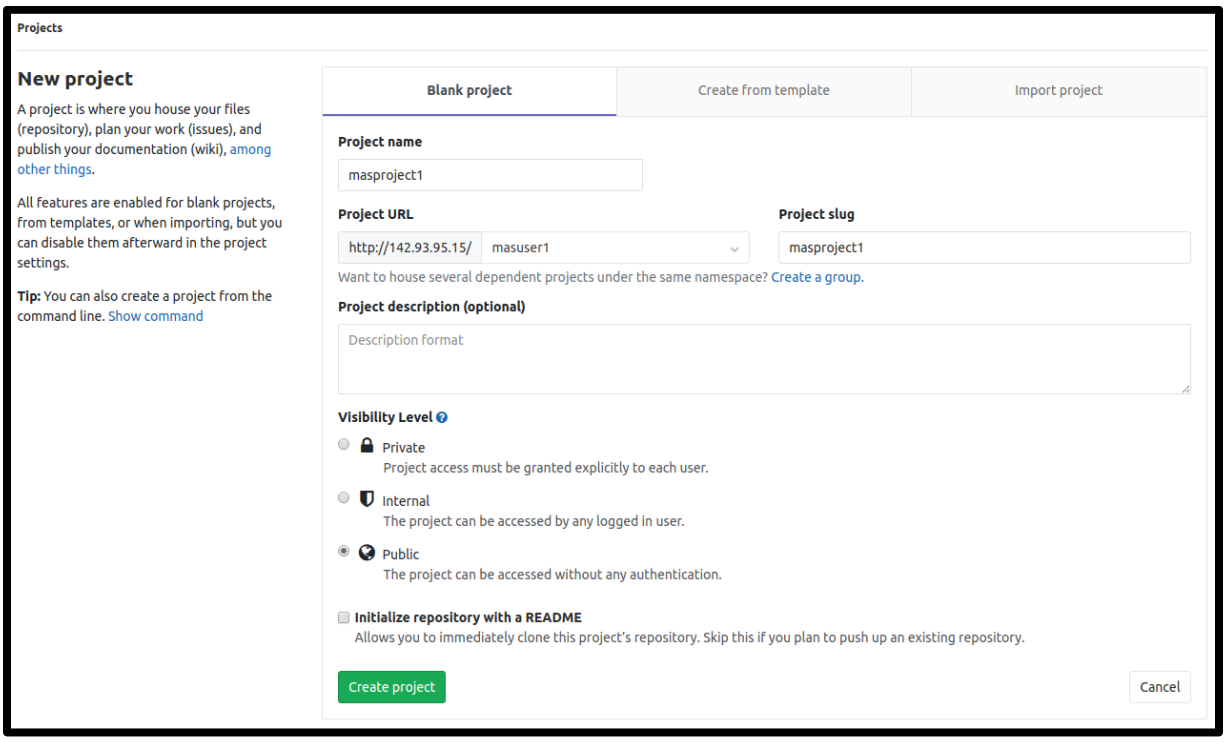
7. Crear, copiar y guardar el *"Token"*, este servirá para la integración con el servidor Jenkins:



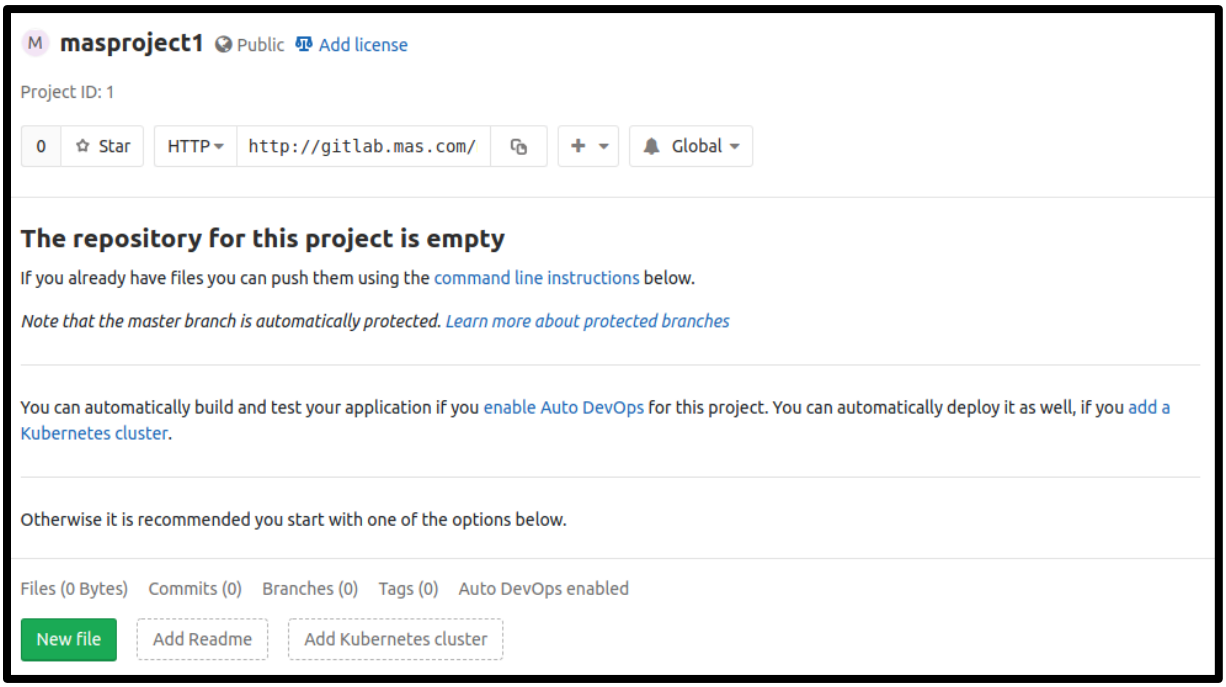
8. Ingresar a GitLab con el usuario administrador y crear un proyecto nuevo:



9. En la creación del proyecto se debe ingresar el nombre del proyecto y seleccionar el nivel de visibilidad, para este caso será "*public*":



10. Al darle clic en "*Create project*" se tendrá la instalación del repositorio Git:



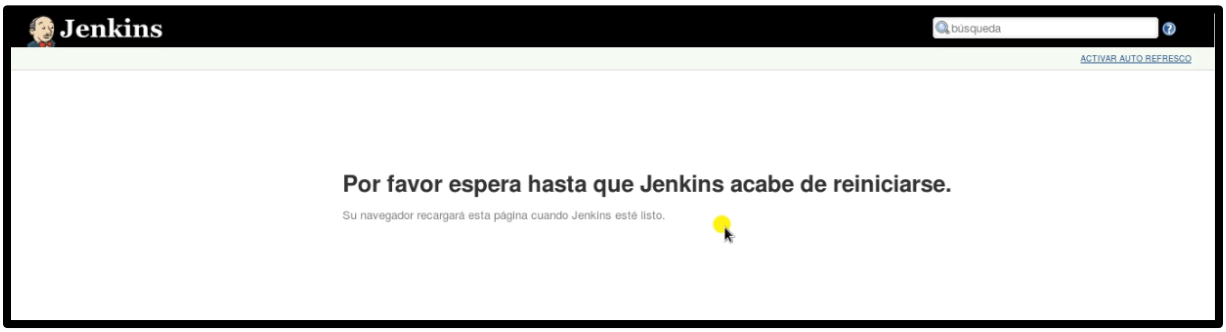
11. Ingresar al servidor Jenkins para colocar los complementos correspondientes para GitLab. Seleccionar la opción "*Manage Plugins*" o Administrar Complementos.

12. Buscar y seleccionar el complemento para GitLab.

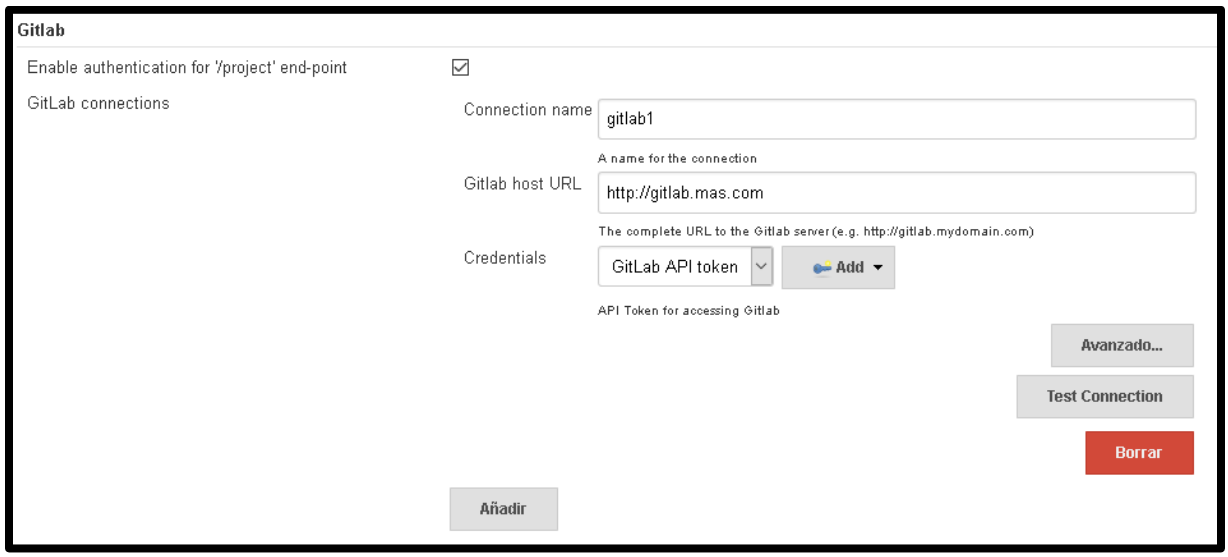
13. Descargar e instalar el complemento.

14. Reiniciar el servidor Jenkins, lo recomendable es reiniciarlo de manera segura, colocando **/safeRestart** en la barra de navegación del explorador.

localhost:8080/safeRestart

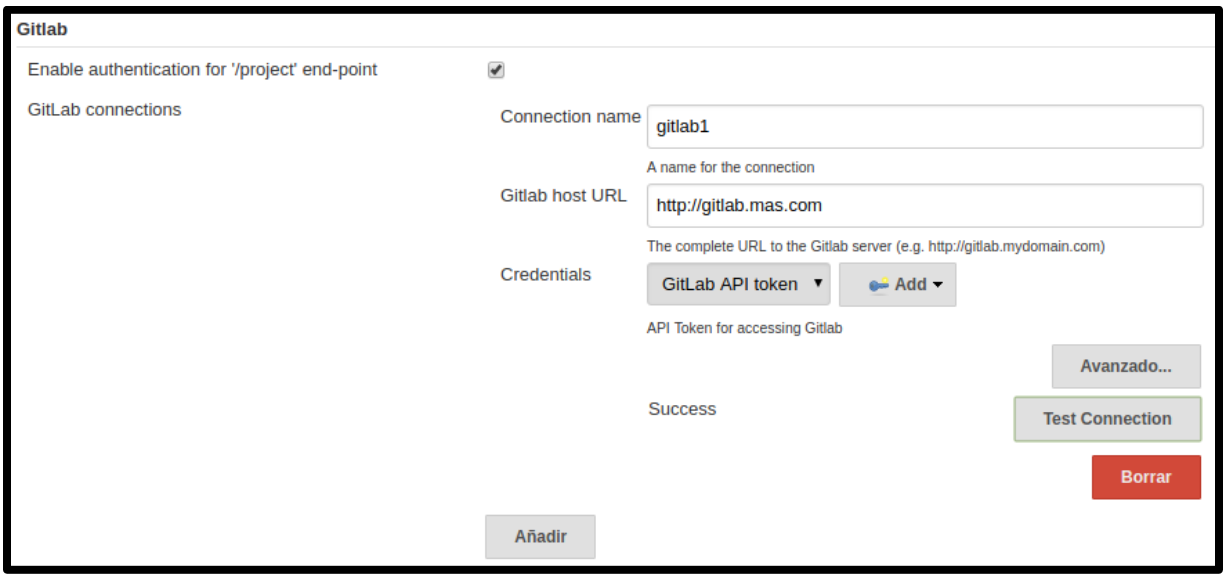


15. En Administrar Jenkins > Configurar el Sistema, se debe añadir la conexión para GitLab, colocando el nombre la URL del servidor y las credenciales correspondientes:

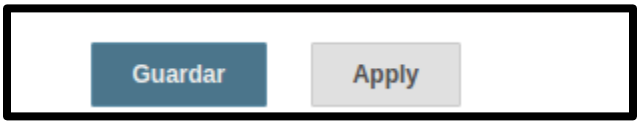


16. Colocar el *"Token"* generado en GitLab:

17. Probar conexión con GitLab:



18. Luego darle clic en "*Apply*" y "*Guardar*" los cambios en la parte inferior:



Paso 3. Crear proyecto en Jenkins

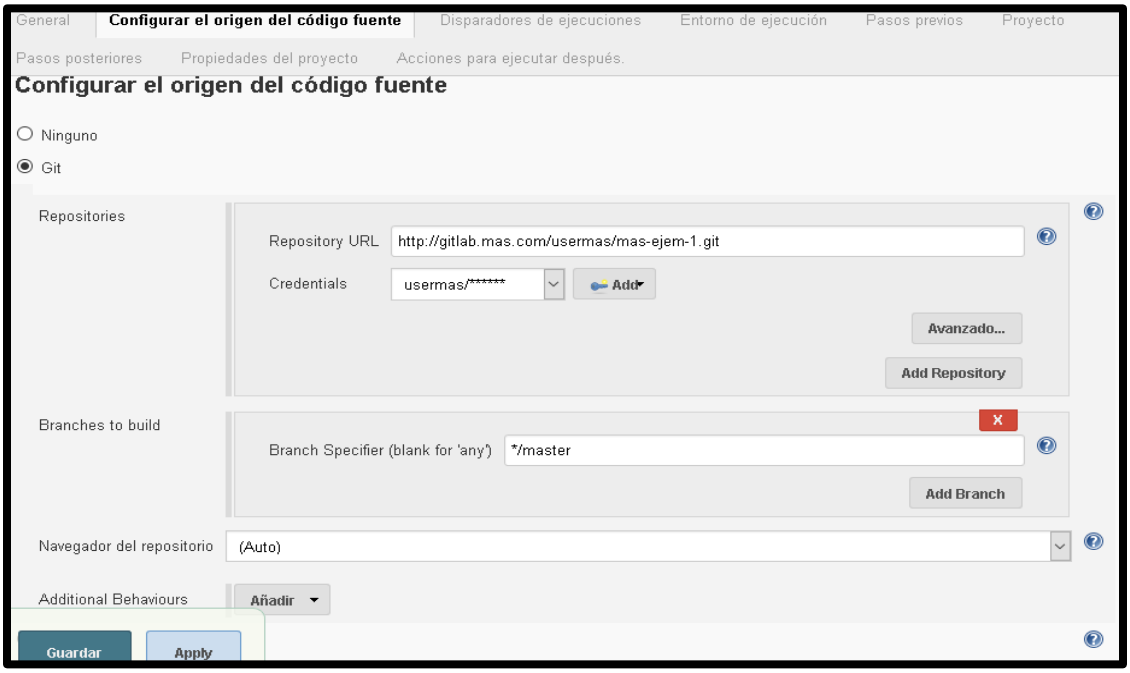
1. Ingresar a Jenkins con usuario administrador.

2. Seleccionar "*Crear un proyecto de estilo libre*":

3. Hacer clic en "*ok*" en la parte inferior:

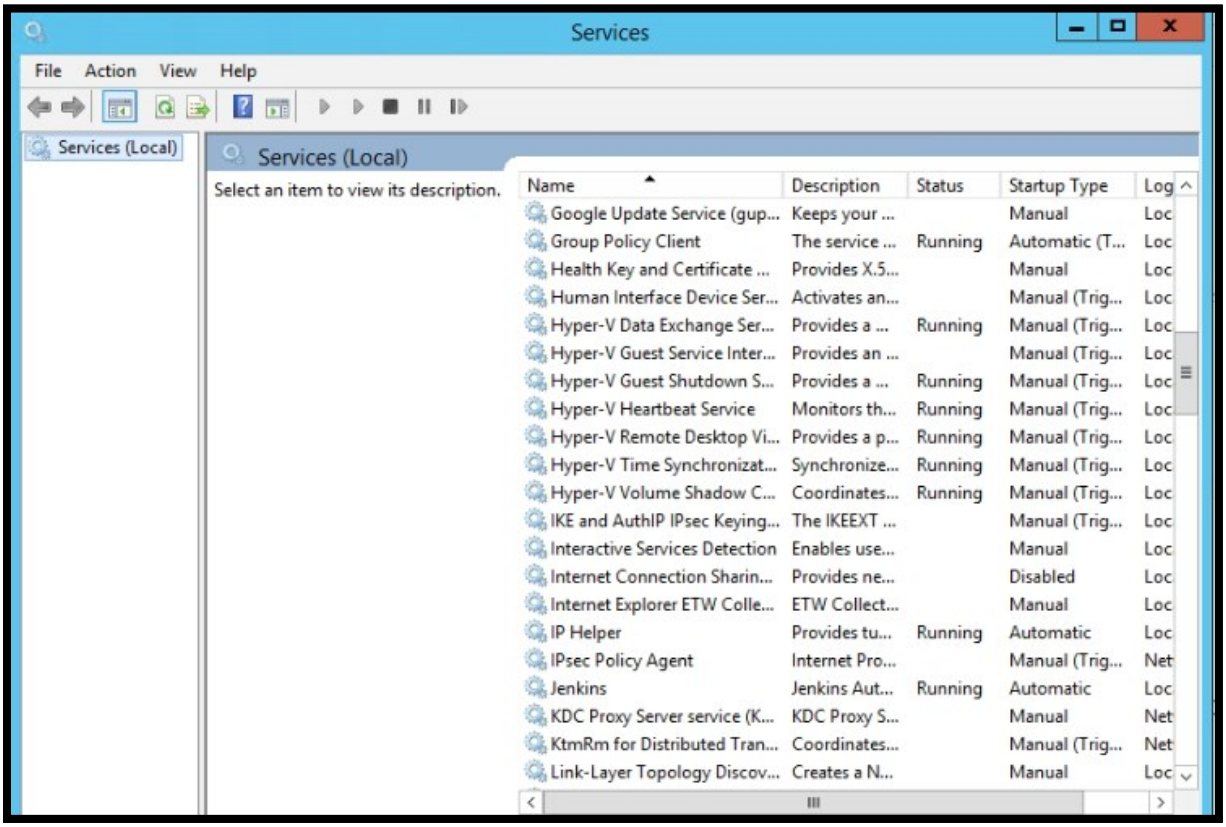
4. En la opción configurar, en la pestaña Propiedades del Proyecto, se debe especificar la conexión de GitLab.

5. En la pestaña de Configurar el origen del código fuente, es necesario colocar la URL del repositorio de Git del proyecto configurado en GitLab, junto con las credenciales correspondientes y la rama a utilizar:

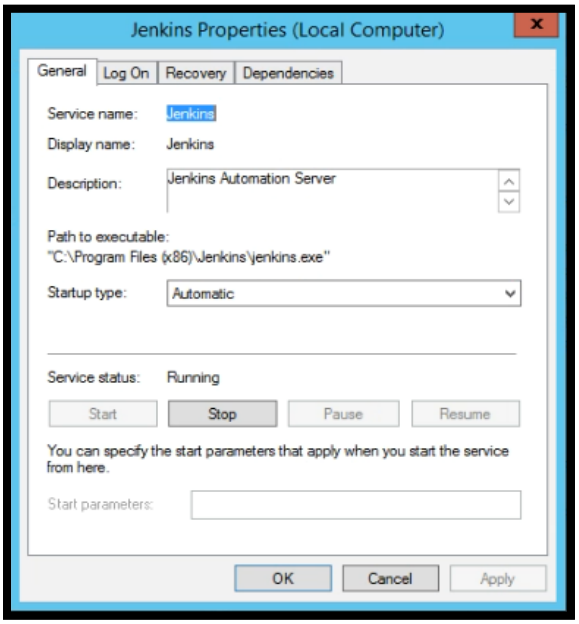


Paso 4. Evaluador de código estático (SonarQube) y Servidor de integración (Jenkins)

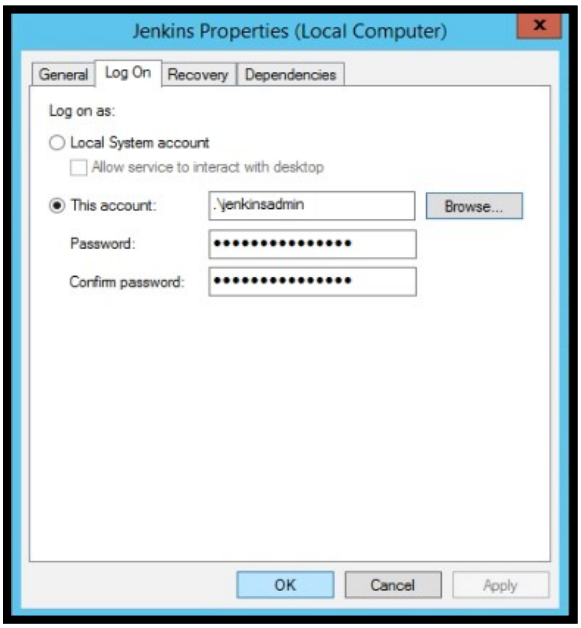
1. En el servidor Jenkins ir a "*Services*" e identificar el servicio correspondiente a éste:



2. Ir a propiedades del servicio:



3. En la pestaño "*Log On*", colocar las credenciales del servidor y dar clic en "Apply".

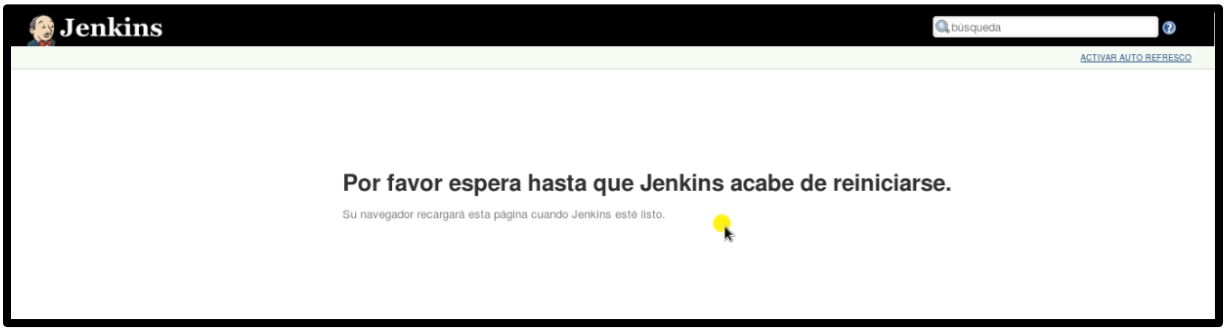


4. Esperar a que los cambios terminen. Al finalizar detener y reiniciar el servicio de Jenkins.

5. Para la instalación del complemento para SonarQube, ir a la consola administrativa de Jenkins, en "*Administrar Jenkins > Administrar Plugins*", y buscar por SonarQube:

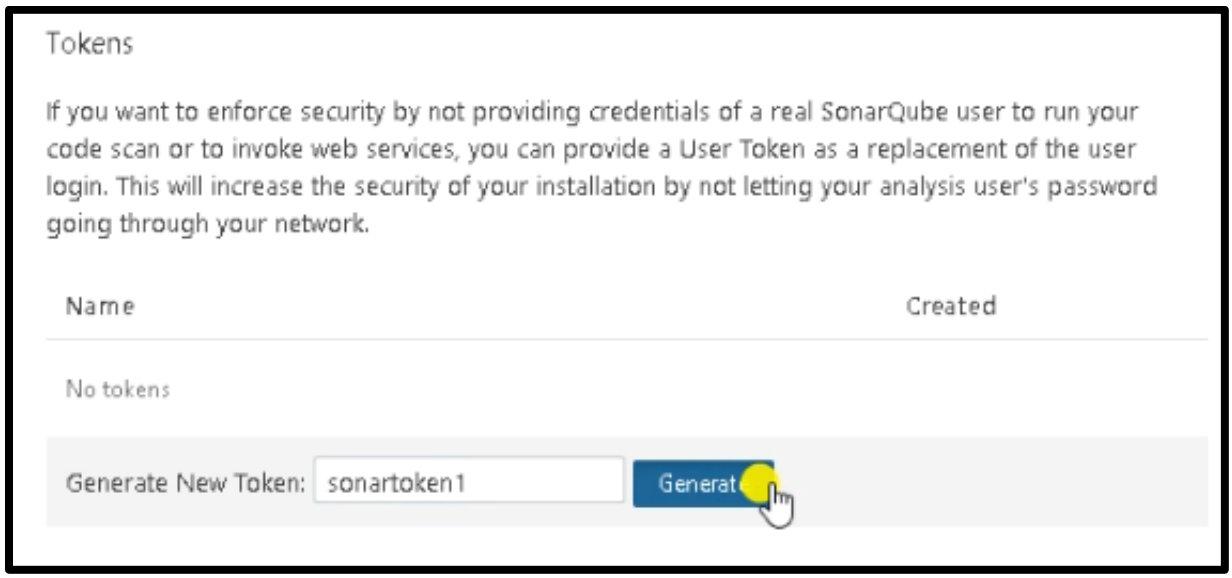
6. Reiniciar el servidor Jenkins, lo recomendable es reiniciarlo de manera segura, colocando **/safeRestart** en la barra de navegación del navegador.

`localhost:8080/safeRestart`

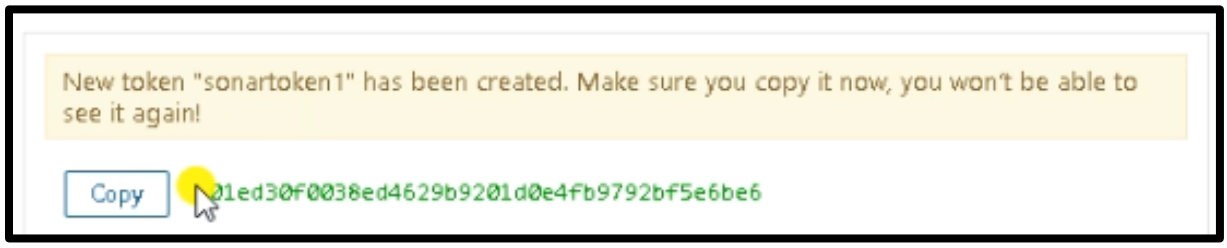


7. Es necesario ingresar al servidor de SonarQube y en el usuario creado luego de la instalación, se debe de agregar un *"Token"* en la opción "*Security > Token*"

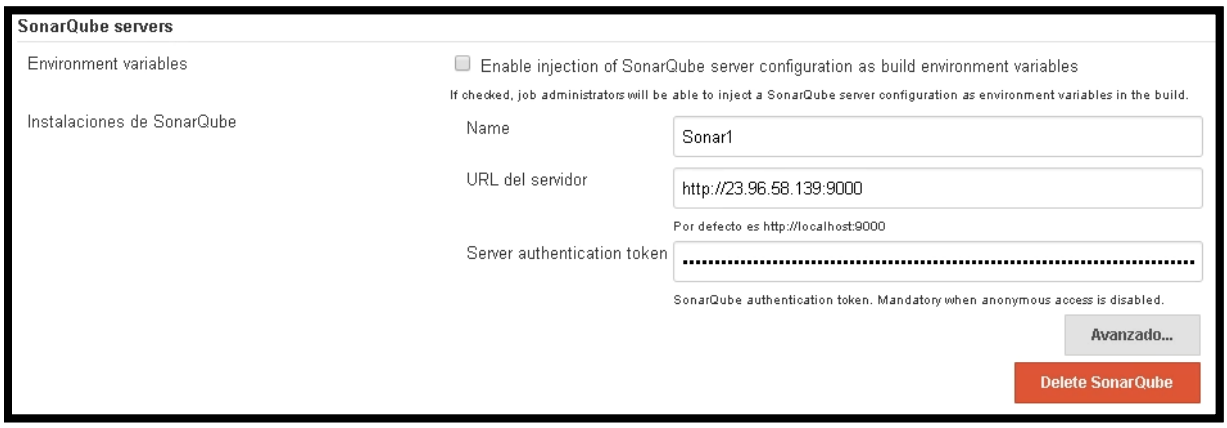
8. Dar clic en "*Generate*":



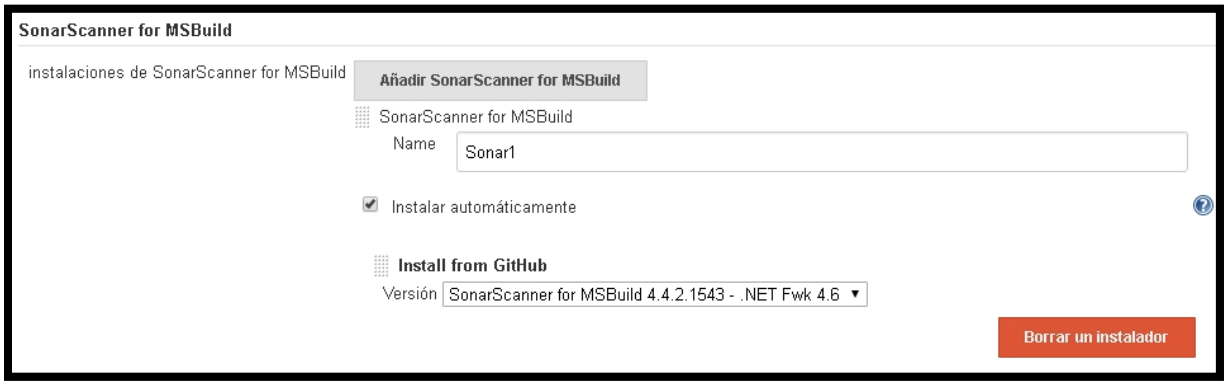
9. Guardar el *"Token"* generado.



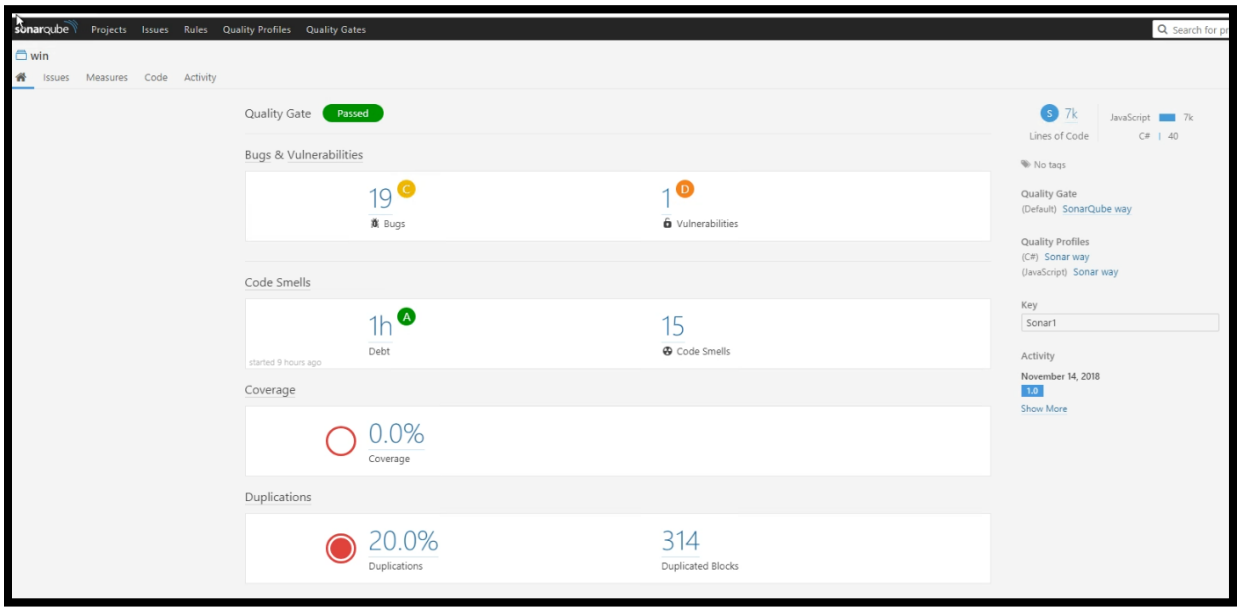
10. Ingresar al servidor en Jenkins, para lo cual se debe de ir a "*Administrar Jenkins* > *Configurar el Sistema*" y en la sección de SonarQube ingresar la siguiente configuración:



11. Configurar el SonarScanner para MSBuild:



12. Luego de esto, se habrá terminado con el enlace entre SonarQube y Jenkins.



Despliegue automático desde Jenkins

Prerrequisitos

Hardware

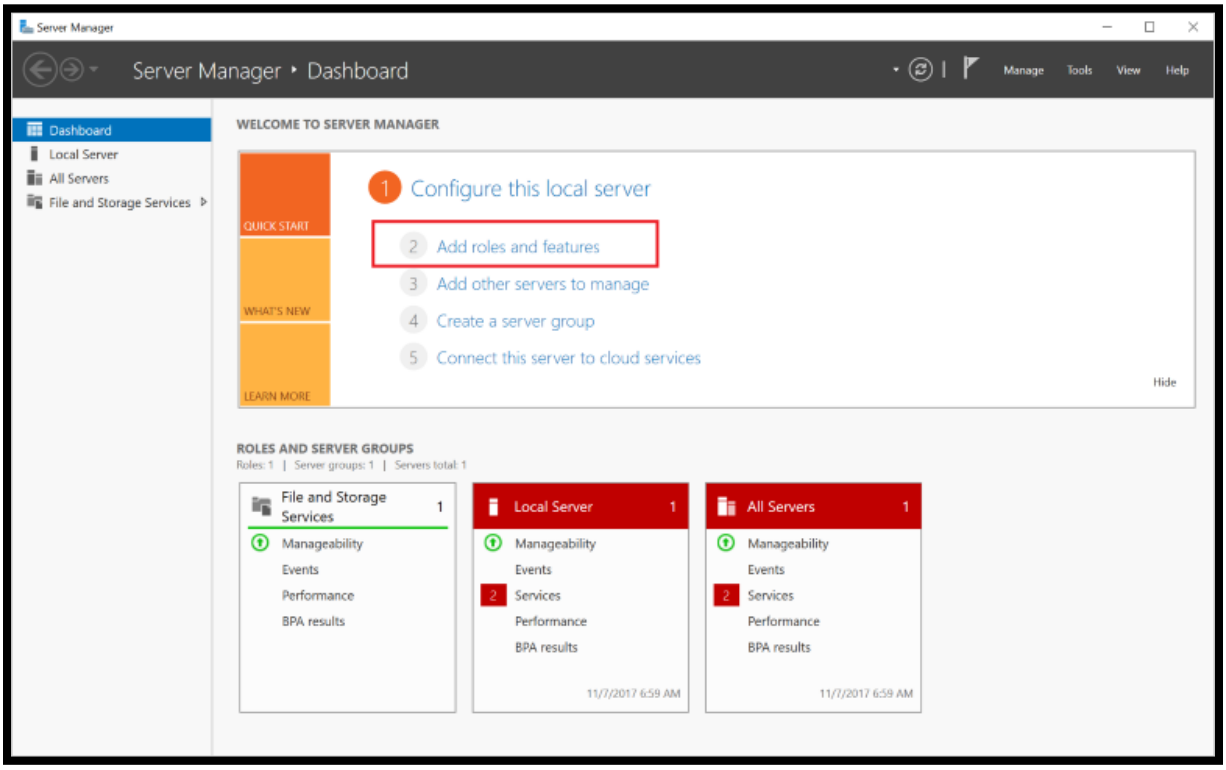
- 2 GB de RAM
- 1 CPU
- 1 interfaz de red
- Disco duro de 10 GB

Software

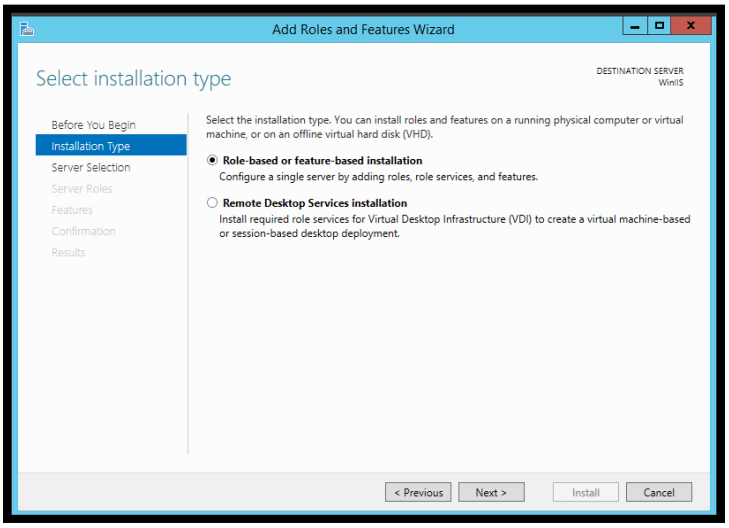
- Windows Server 2012 R2.
- Usuario con privilegios de súper administrador.

Paso 1. Configuración Servidor de aplicaciones (IIS)

1. Ingresar como administrador al servidor de aplicaciones IIS.
2. Ingresar al "*Server Manager Dashboard*".
3. Dar clic en "*Add roles and features*":

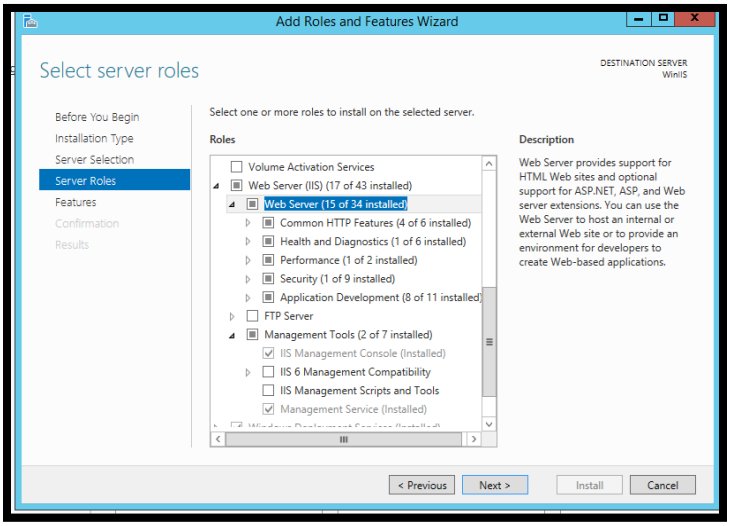


4. Seleccionar "*Role-based or feature based installation*" como el tipo de instalación. Continuar dando clic en "*Next*".

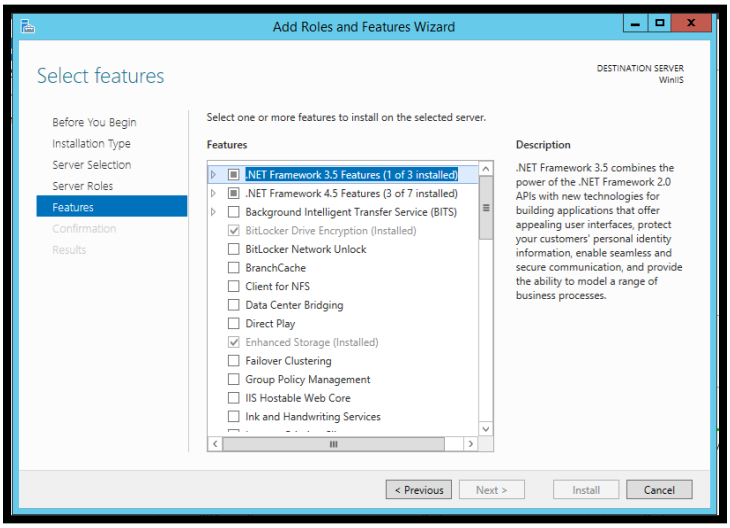


5. En la pestaña "*Server Selection*" seleccionar el servidor a configurar. Continuar dando clic en "*Next*".

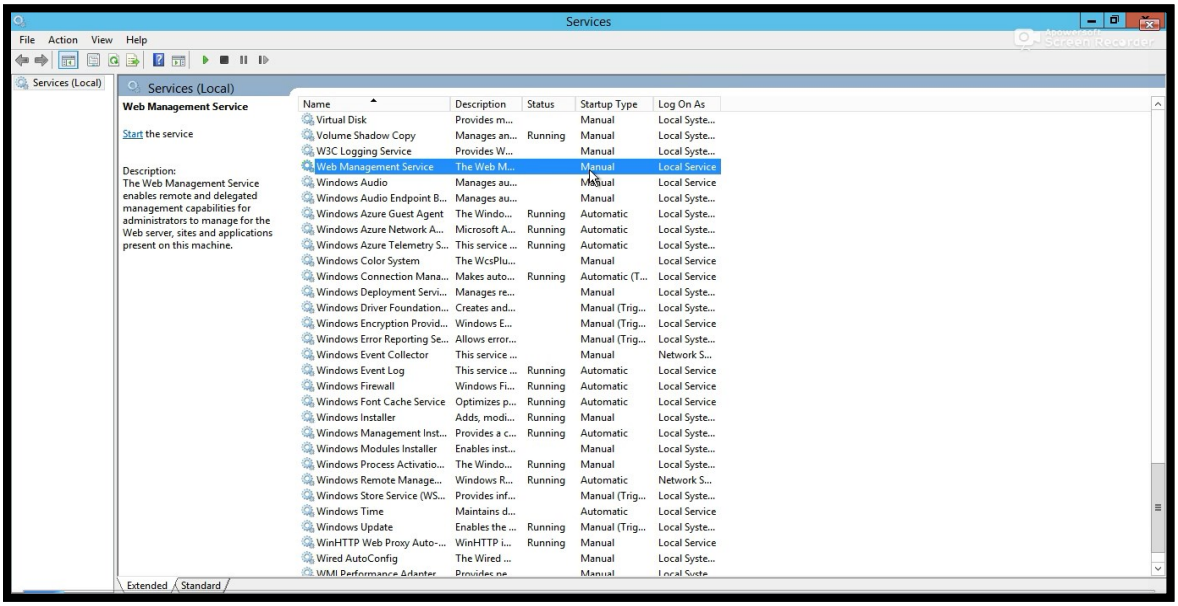
6. En la pestaña "*Server Roles*" seleccionar "*Web Server (IIS)*" con las opciones "*Web Server*" y "*Management Tools*". Continuar dando clic en "*Next*".



7. En la pestaña "*Features*" seleccionar "*.NET Framework 3.5 Features*" y "*.NET Framework 4.5 Features*". Finalizar dando clic en "*Install*".



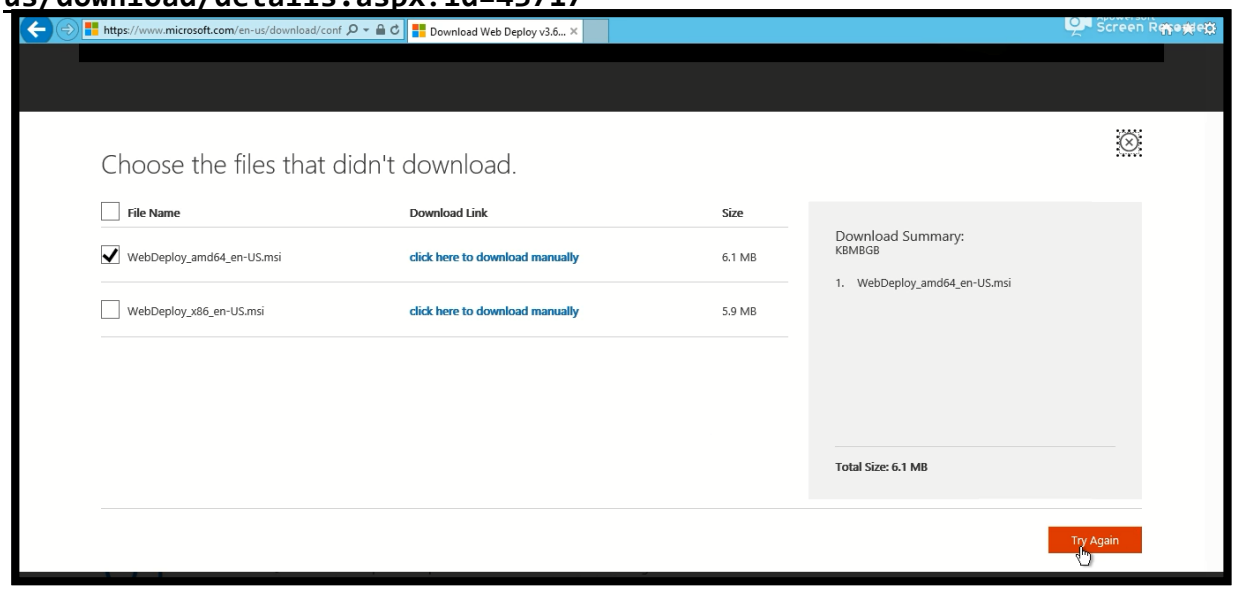
8. Iniciar el Servicio "*Web Management Service*" desde "*Services*".



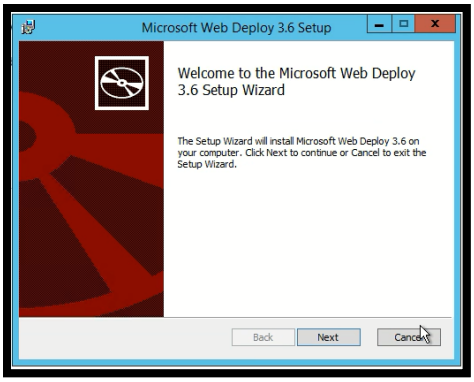
9. Dar doble clic sobre el Servicio "*Web Management Service*" y cambiar "*Startup type*" de "*Manual*" a "*Automatic*" para que inicie automáticamente. Finalizar dando clic en Ok.

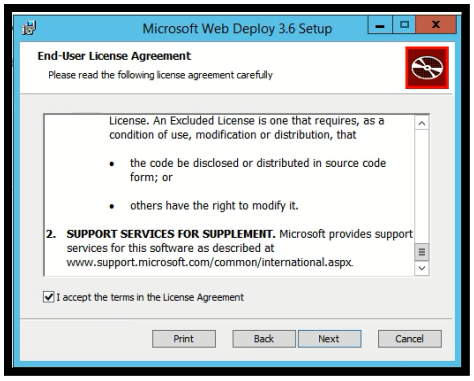
10. Descargar "*Microsoft WebDeploy 3.6*" desde el sitio oficial de Microsoft:

https://www.microsoft.com/en-us/download/details.aspx?id=43717

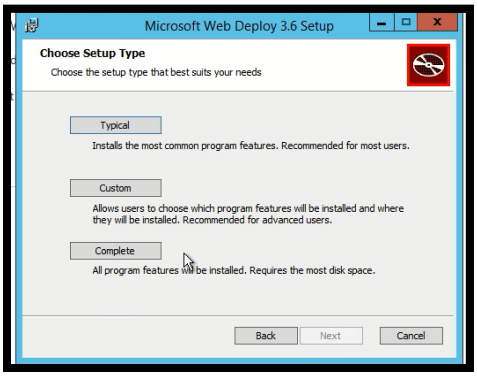


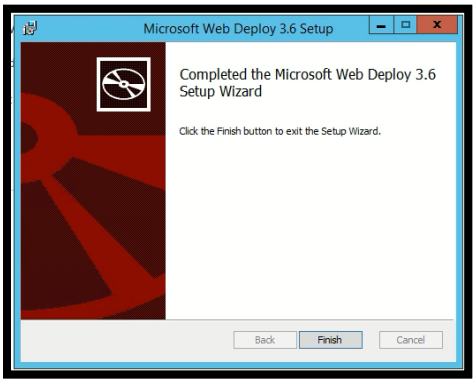
11. Instalar "*Microsoft WebDeploy 3.6*" con el ejecutable descargado. Seleccionar las opciones predeterminadas y continuar dando clic en "*Next*".





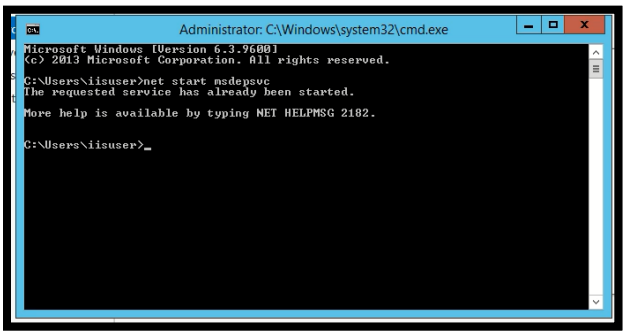
12. Es importante que se seleccione "*Complete*" como el "*Setup Type*", para que instale el agente que servirá para el despliegue.





13. Una vez instalado, se procede a iniciar el servicio "*Web Deploy*". Se puede realizar desde "*Services*" o desde línea de comando:

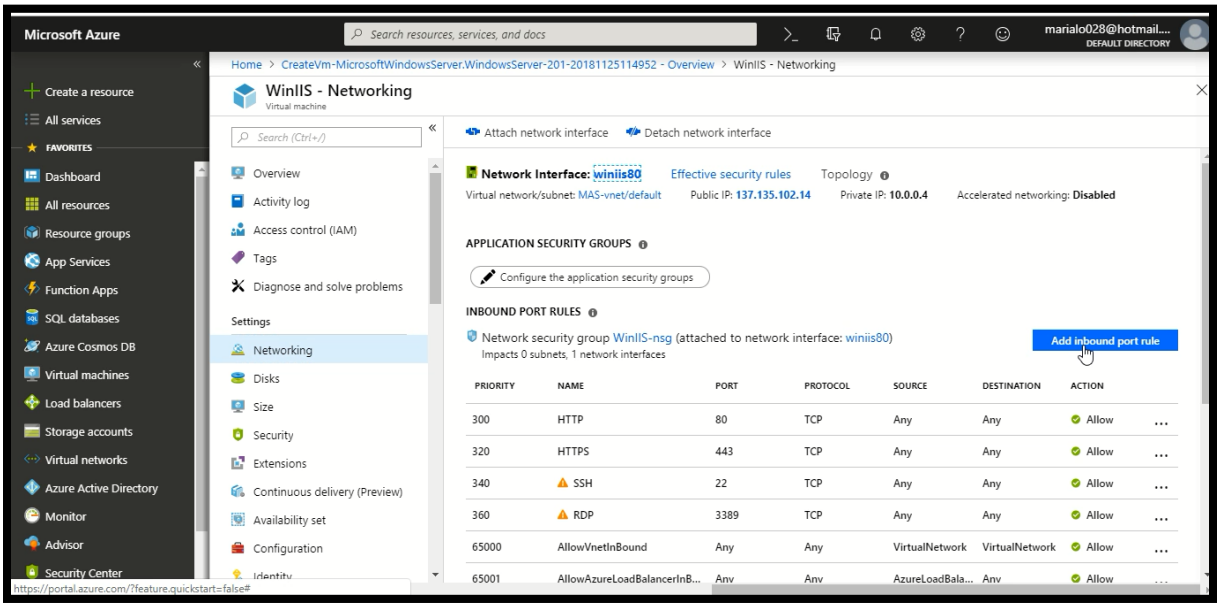
```
net start
```



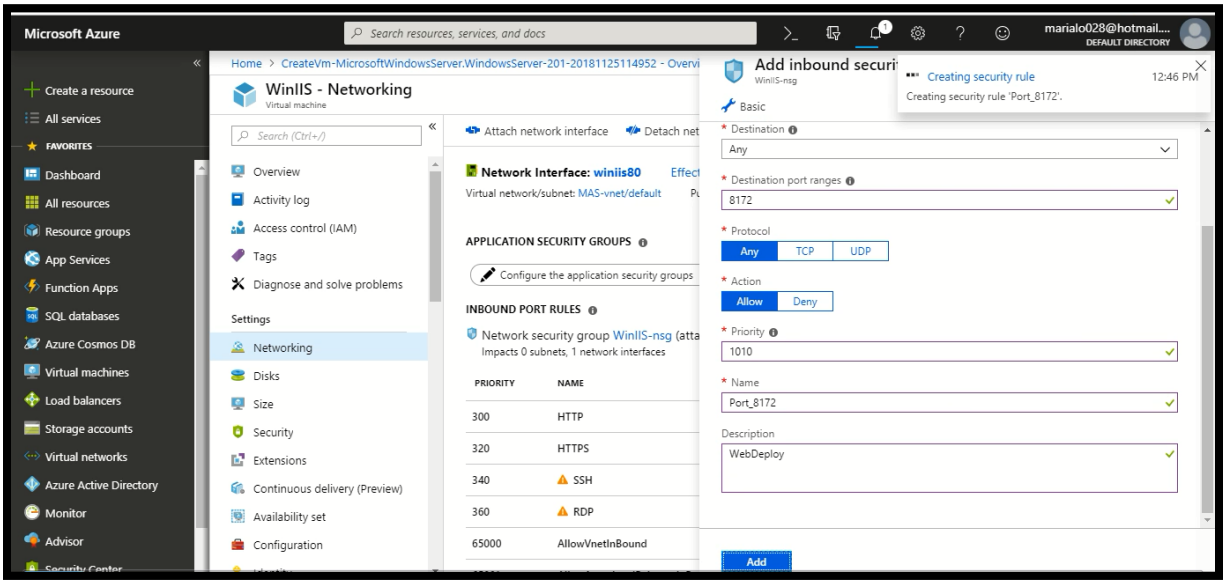
14. Dentro de "*Services*" dar doble clic sobre el Servicio "*Web Management Service*" y cambiar "*Startup type*" de "*Manual*" a "*Automatic*" para que inicie automáticamente. Finalizar dando clic en "*Ok*".

15. El servicio "*Web Deploy*" utiliza el puerto 8172, por lo que hay que habilitarlo. Ir al Portal de Azure y dirigirse a la opción "*Networking*".

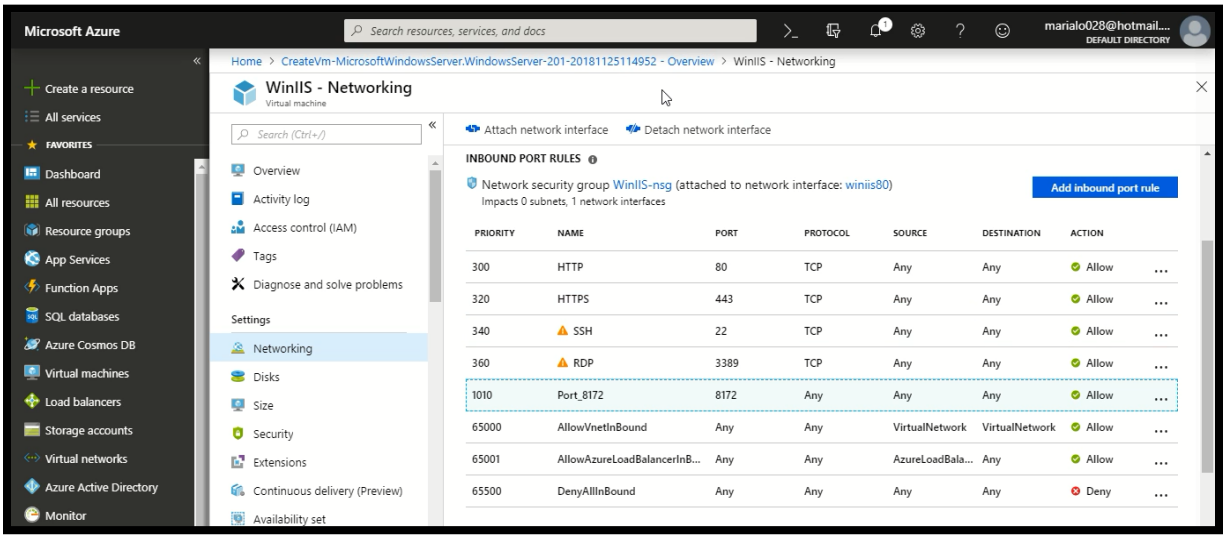
16. Dar clic en "*Add inbound port rule*":



17. Ingresar la información correspondiente al puerto 8172 para habilitar:



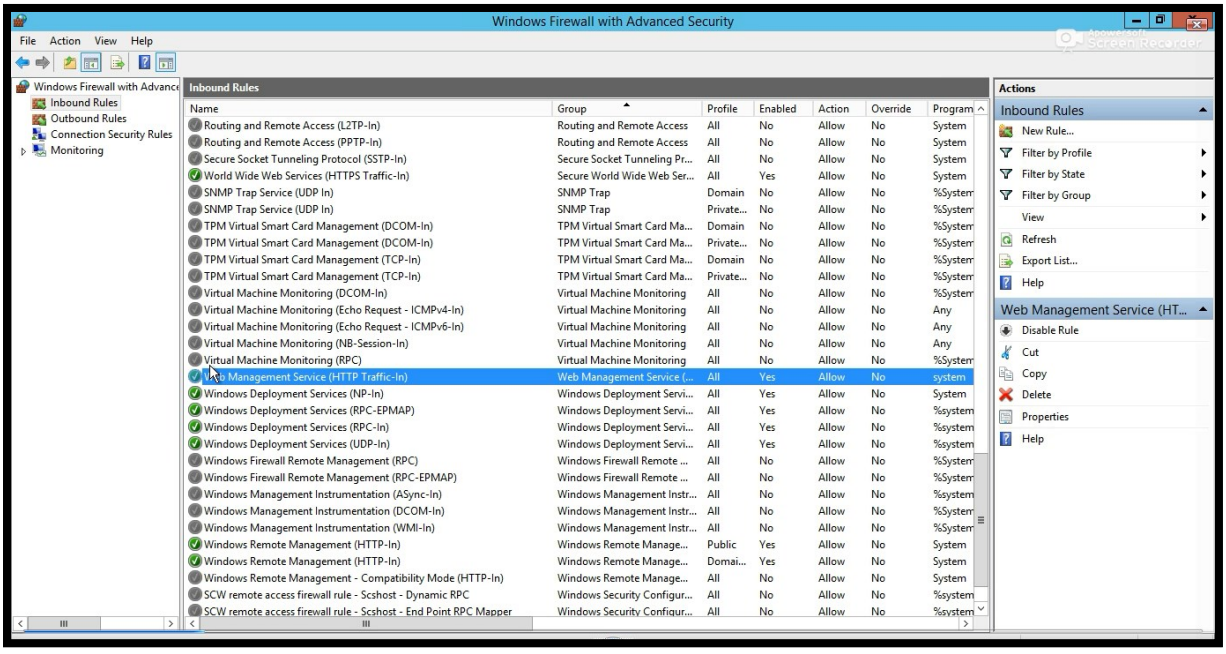
18. Verificar que se encuentre habilitada la regla del puerto de entrada:



19. Ingresar al servidor IIS con usuario administrador.

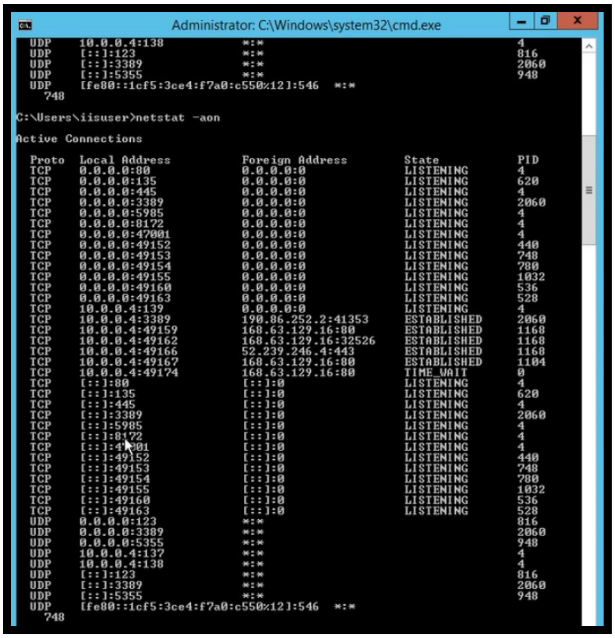
20. Ingresar a “Windows Firewall”.

21. Verificar dentro la opción “*Inbound Rules*” que se encuentre habilitado el puerto 8172 para el servicio “*Web Management Service*”:



22. En el cmd revisar que el puerto 8172 se encuentre activo con el comando:

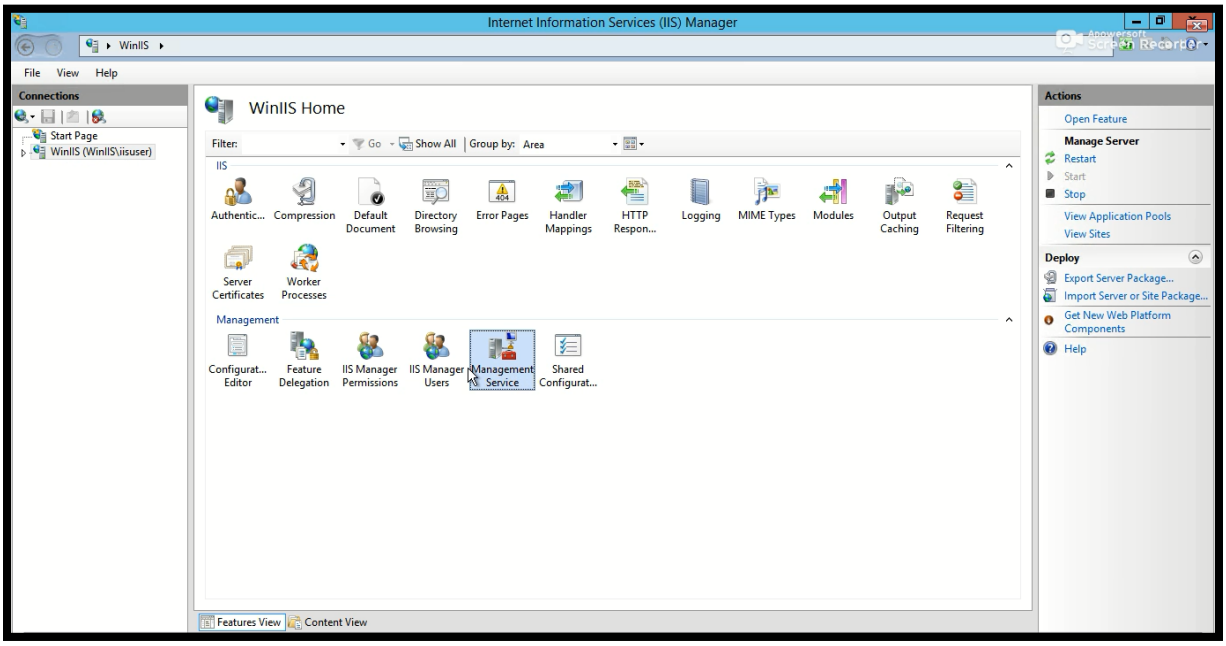
`netstat -aon`



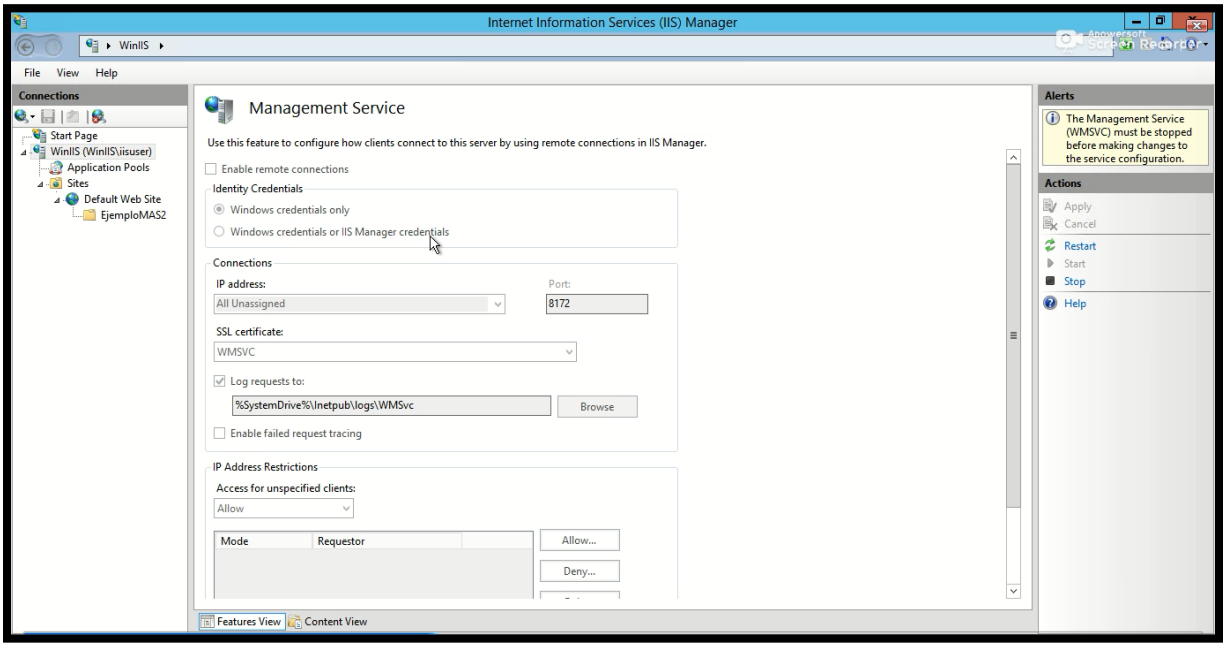
23. Ingresar a "*Internet Information Services (IIS) Manager*"

24. Dar clic sobre el servidor de aplicaciones.

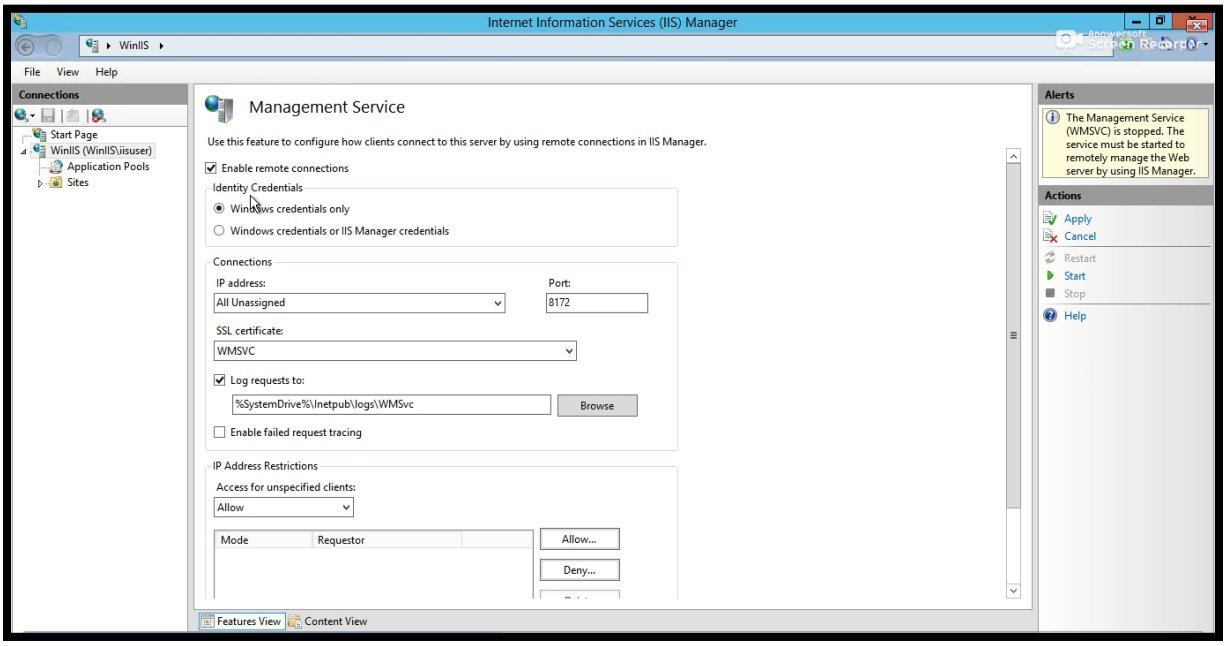
25. Dar clic en la opción "*Management Service*" para configurarlo.



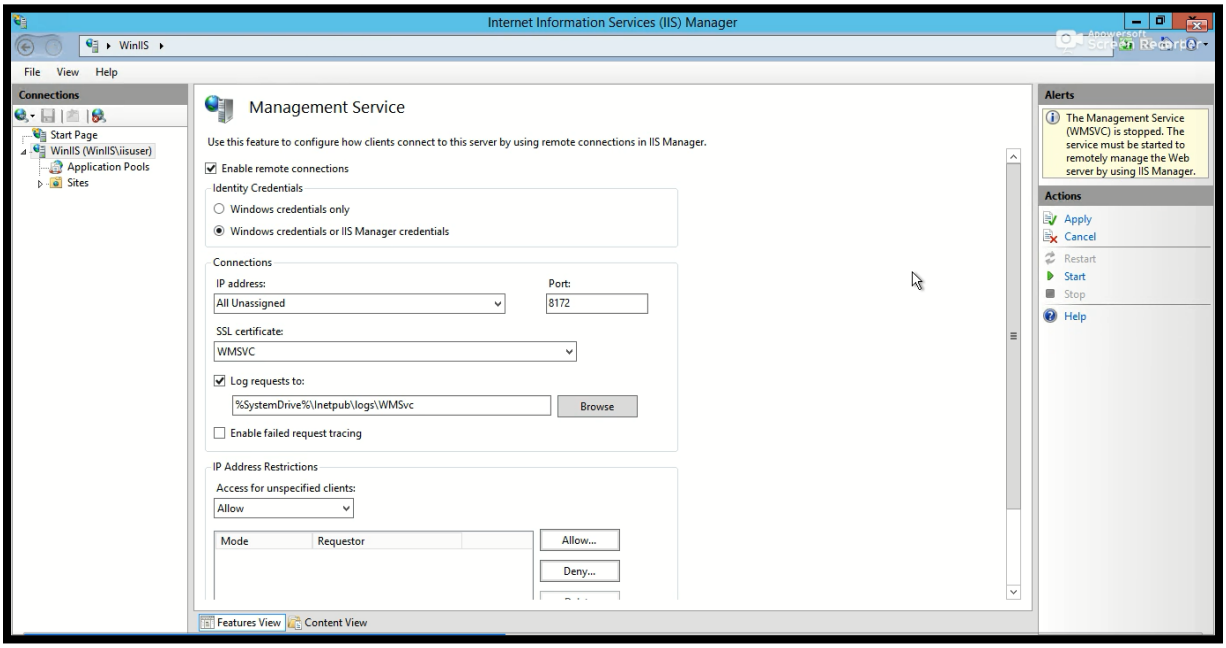
26. Dar clic en "*Stop*" para detener el servicio y poder configurarlo.



27. Habilitar conexiones remotas, seleccionando la opción "*Enable remote connections*". Esto permitirá el despliegue desde el servidor de integración.

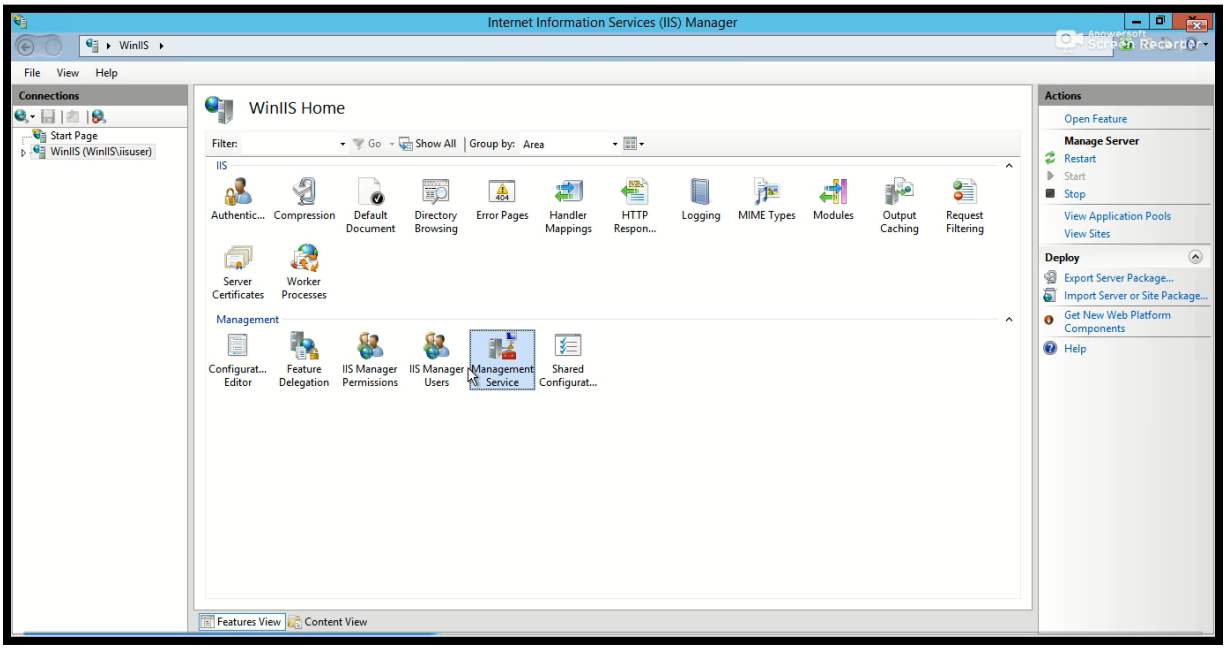


28. En "*Identity Credentials*" seleccionar "*Windows credentials or IIS Manager credentials*".

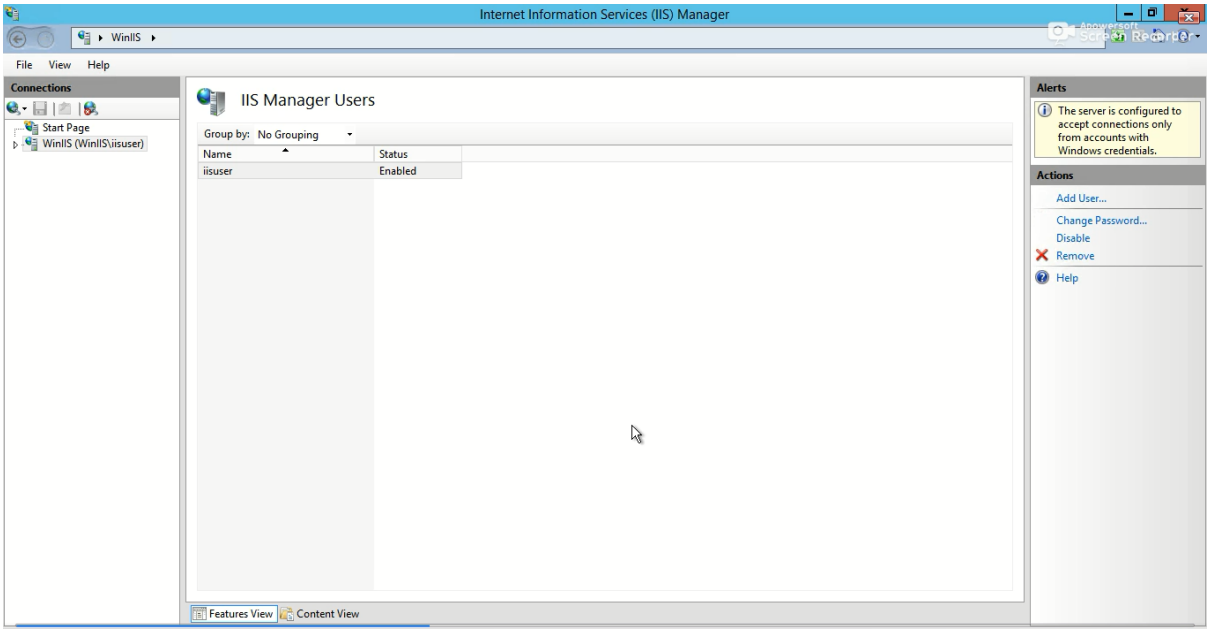


29. Dar clic en aplicar e iniciar el servicio.

30. Dar clic en la opción "*IIS Manager Users*":

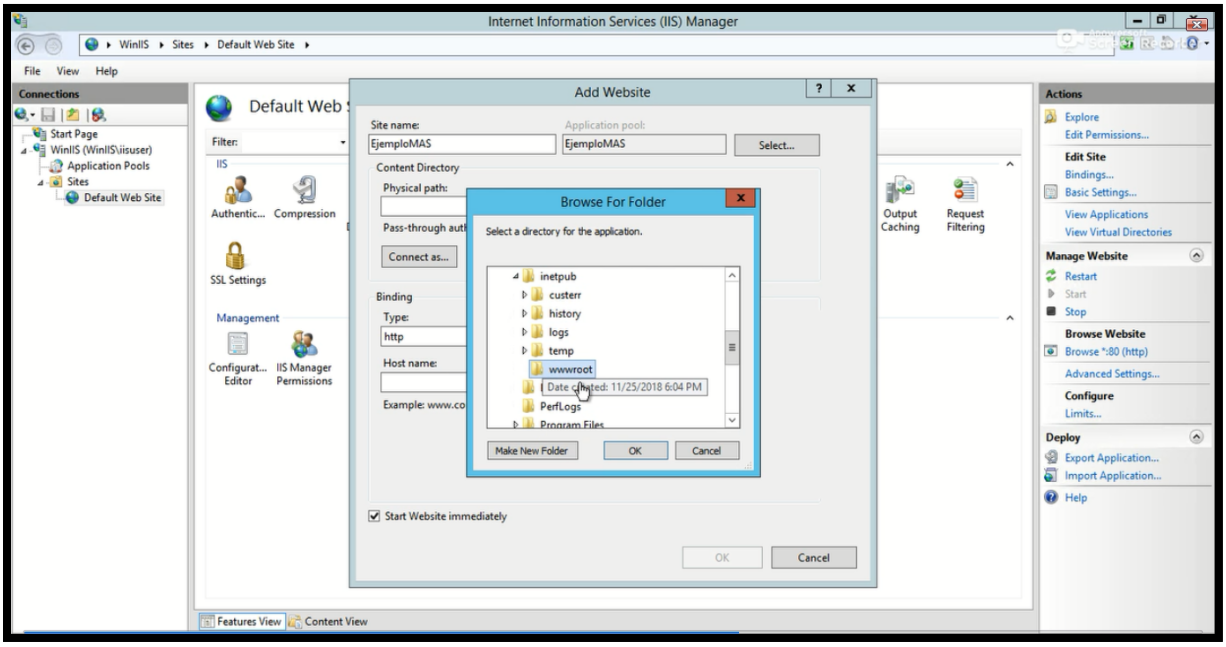


31. Dar clic en "*Add User*" para agregar y habilitar como "*IIS Manager User*" al usuario con el que se realizará el despliegue.



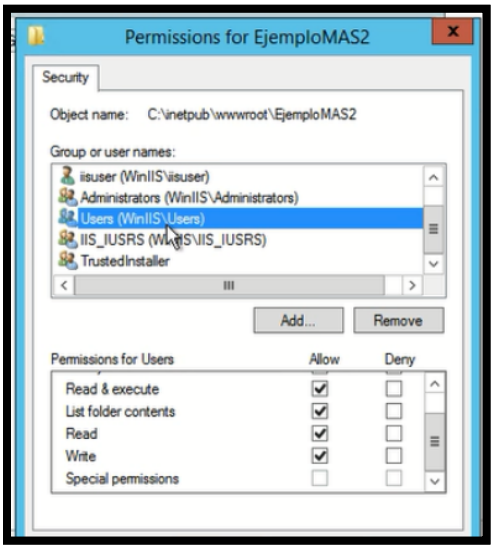
Paso 2. Crear Sitio Web en el Administrador IIS

1. Dar clic derecho sobre "*Sites*" seguido de "*Add WebSite*".
2. Ingresar el nombre del Sitio.
3. Buscar la dirección física donde se encontrará el publicado.

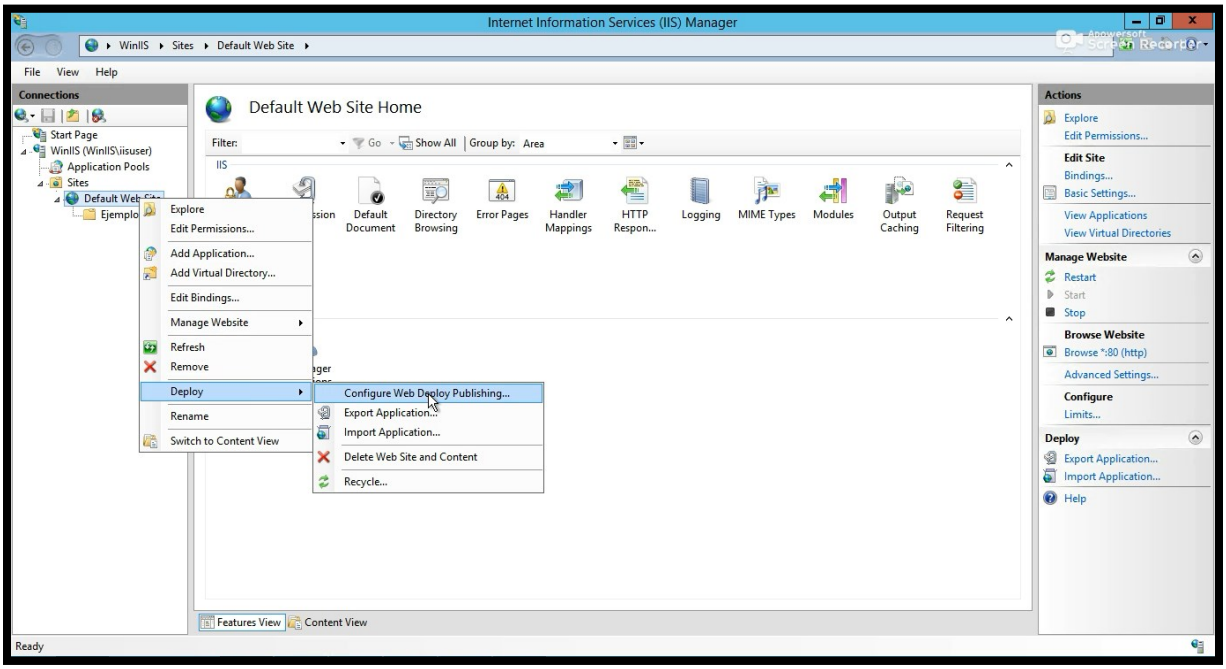


4. Ingresar información del "*Binding*" y finalizar dando clic en Ok.

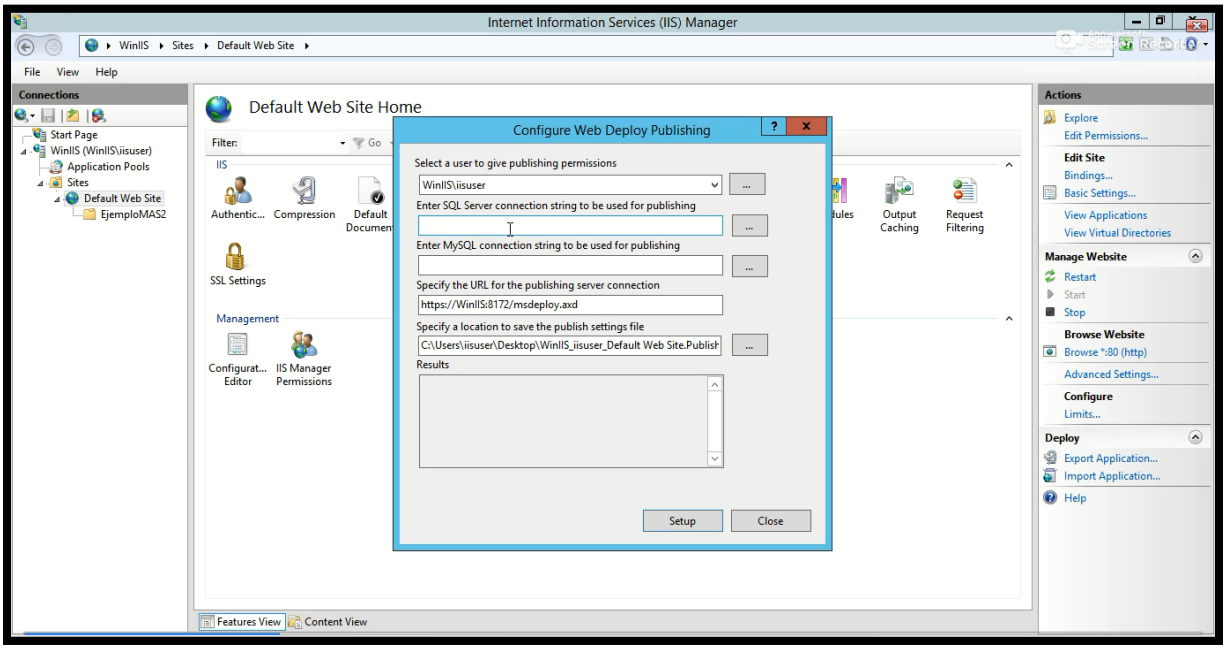
5. Dar clic en "*IIS Manager Permissions*" dentro del Sitio recién creado para brindar acceso al usuario a la carpeta donde se encuentra la aplicación Web.



6. Dar clic derecho sobre el Sitio recién creado seguido por seleccionar la opción "*Deploy*" y "*Configure Web Deploy Publishing…*" para así asignar los permisos al usuario para desplegar la aplicación Web.



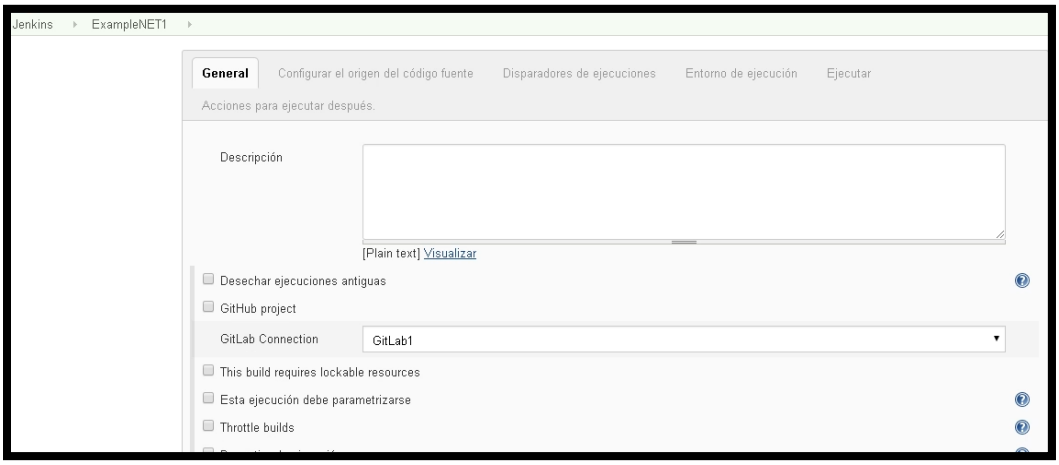
7. Dar clic en "*Setup*" para finalizar.


Internet Information Services (IIS) Manager
WinIIS ▸ Sites ▸ Default Web Site ▸
File View Help
Connections
Start Page
WinIIS (WinIIS\iisuser)
Application Pools
Sites
Default Web Site
EjemploMAS2
Default Web Site Home
Filter:
Go
IIS
Authentic...
Compression
Default Documen
SSL Settings
Management
Configurat... Editor
IIS Manager Permissions
Output Caching
Request Filtering
Features View
Content View
Configure Web Deploy Publishing
Select a user to give publishing permissions
WinIIS\iisuser
Enter SQL Server connection string to be used for publishing
Enter MySQL connection string to be used for publishing
Specify the URL for the publishing server connection
https://WinIIS:8172/msdeploy.axd
Specify a location to save the publish settings file
C:\Users\iisuser\Desktop\WinIIS_iisuser_Default Web Site.Publish
Results
Setup
Close
Actions
Explore
Edit Permissions...
Edit Site
Bindings...
Basic Settings...
View Applications
View Virtual Directories
Manage Website
Restart
Start
Stop
Browse Website
Browse *:80 (http)
Advanced Settings...
Configure
Limits...
Deploy
Export Application...
Import Application...
Help

Proceso de Integración dentro de tarea en el servidor Jenkins:

Paso 1. Configurar proyecto

1. Crear proyecto en Jenkins.



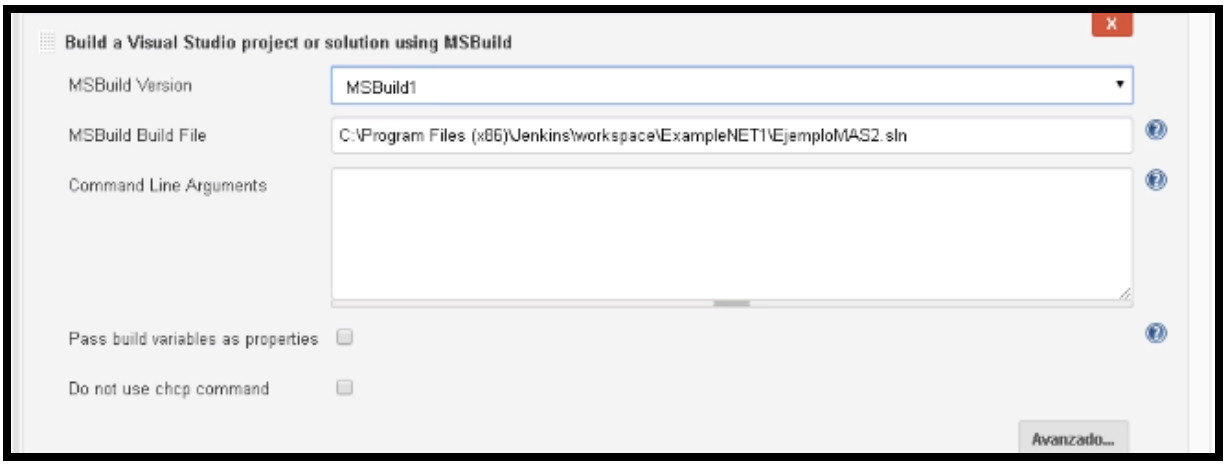
2. Seleccionar la pestaña Entorno de ejecución:

3. Para NuGet, colocar el siguiente comando:

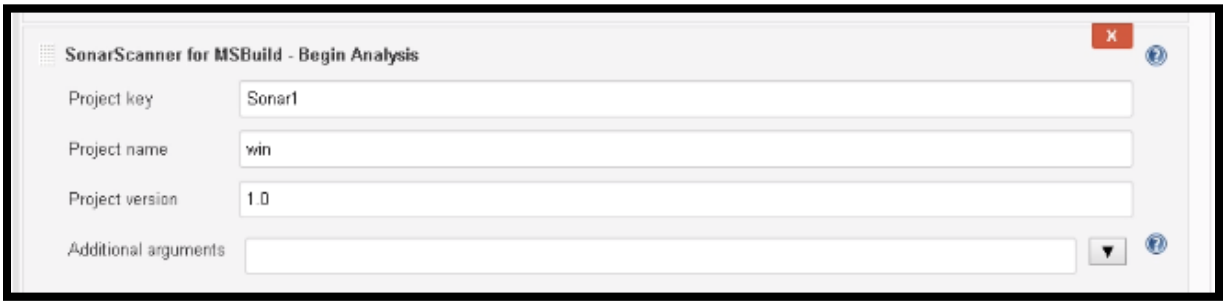
<directorio_nuget\nuget.exe>

"<workspace_jenkins\solucion .net"

4. Se indica la ubicación del proyecto o solución a construir mediante el MSBuild:

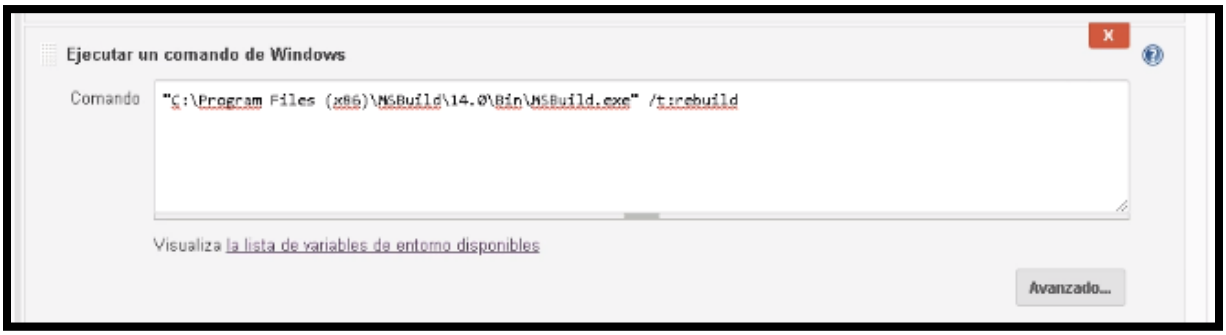


5. Después se le debe indicar a SonarScanner el punto de inicio del análisis:



6. Se coloca el siguiente comando para forzar a limpiar y reconstruir el proyecto:

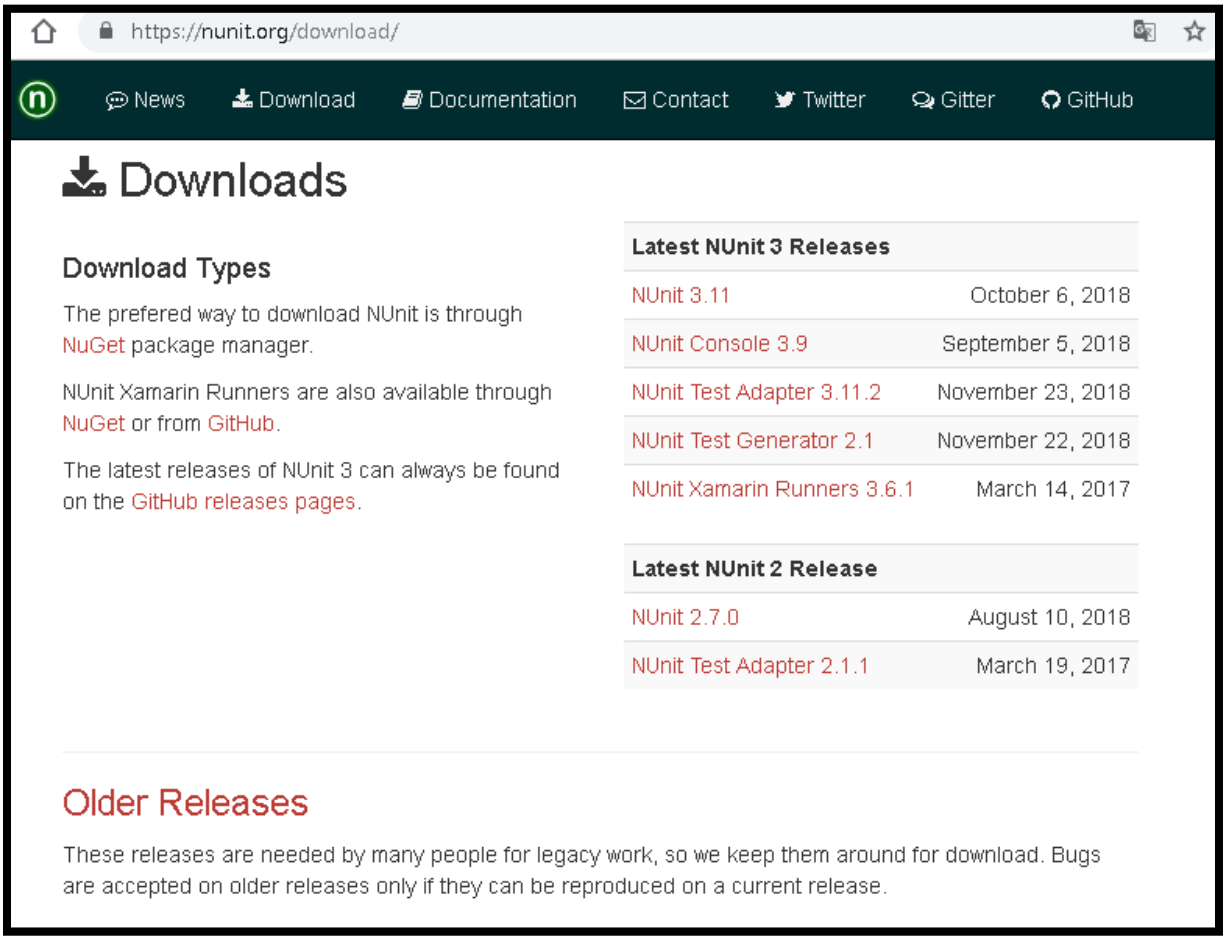
“<directorio_msbuild\MSBuild.exe>” /t:rebuild



7. Finalmente se indica al MSBuild que puede finalizar el análisis:

8. Para configurar el siguiente paso, se debe descargar la herramienta de NUnit nunit-console.exe para la ejecución de sus comandos, la cual se debe descargar directamente desde el sitio oficial:

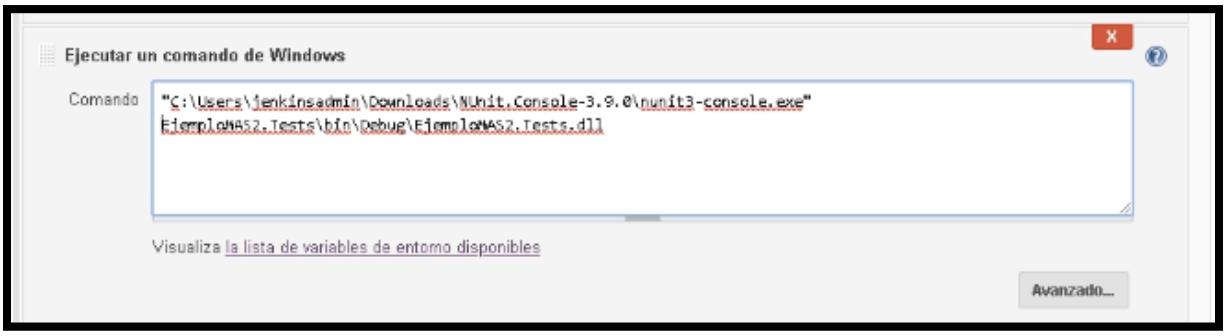
https://nunit.org/download/



Para evitar conflictos se recomienda descargar el archivo "*.Zip*".

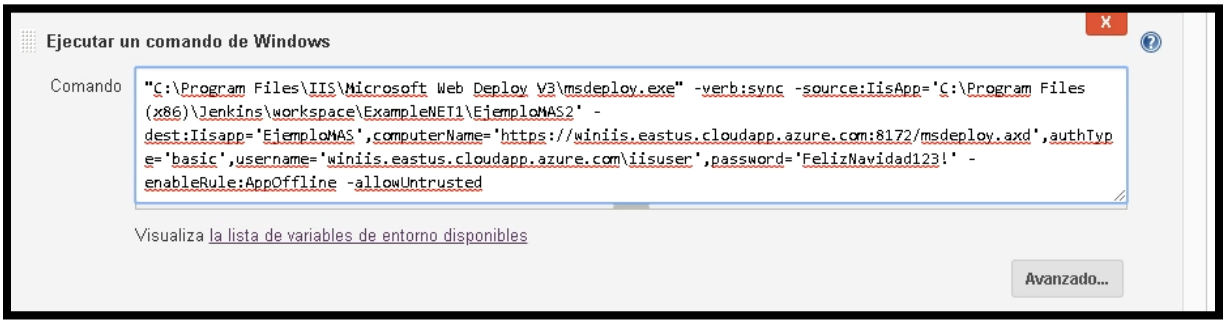
9. Luego de eso, es necesario ejecutar las pruebas unitarias de NUnit, lo cual se realiza mediante el comando siguiente:

"<directorio_nunit_console\nunit3-console.exe>" <directorio_test.dll\test.dll>



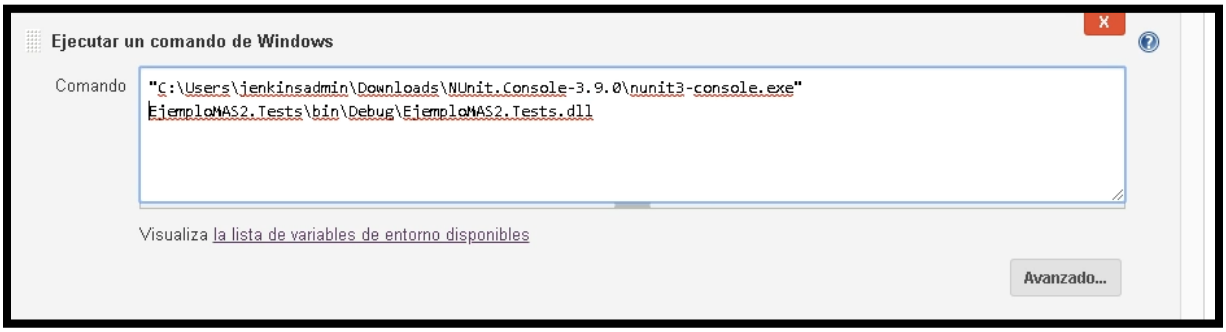
10. Es necesario desplegar la aplicación en un ambiente de pruebas para que Selenium realice su función:

```
"<directorio_msdeploy\msdeploy.exe>" -verb:sync -source:IisApp='<workspace_jenkins\solucion_.net\proyecto_.net>' -dest:Iisapp='<nombre_sitio_TEST>',computerName='<sitio_web/msdeploy.axd>',authType='basic',username='<dominio\nombre_usuario_iis>',password='<clave_usuario_iis>' -enableRule:AppOffline -allowUntrusted
```



11. Se ejecutan las pruebas de Selenium, con la aplicación recién desplegada contra lo definido en los archivos de pruebas:

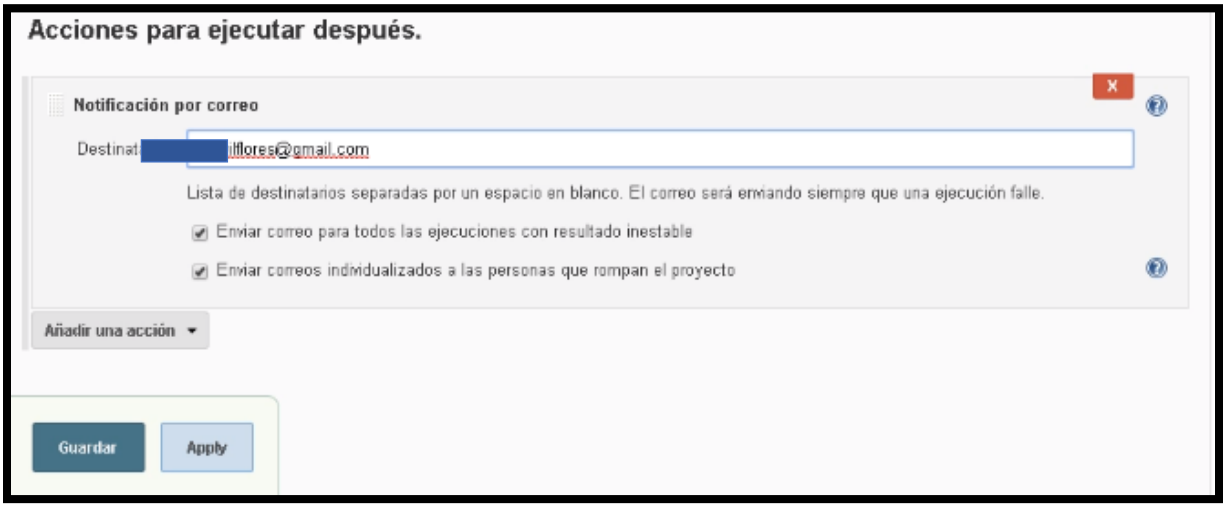
```
“<directorio_nunit_console\nunit3-console.exe>” <directorio_test_sel.dll\test_sel.dll>
```



12. Si todo resulta satisfactorio, se indica que la aplicación puede ser desplegada en un servidor final o de producción:

```
"<directorio_msdeploy\msdeploy.exe>" -verb:sync -source:IisApp='<workspace_jenkins\solucion_.net\proyecto_.net>' -dest:Iisapp='<nombre_sitio_PROD>',computerName='<sitio_web/msdeploy.axd>',authType='basic',username='<dominio\nombre_usuario_iis>',password='<clave_usuario_iis>' -enableRule:AppOffline -allowUntrusted
```

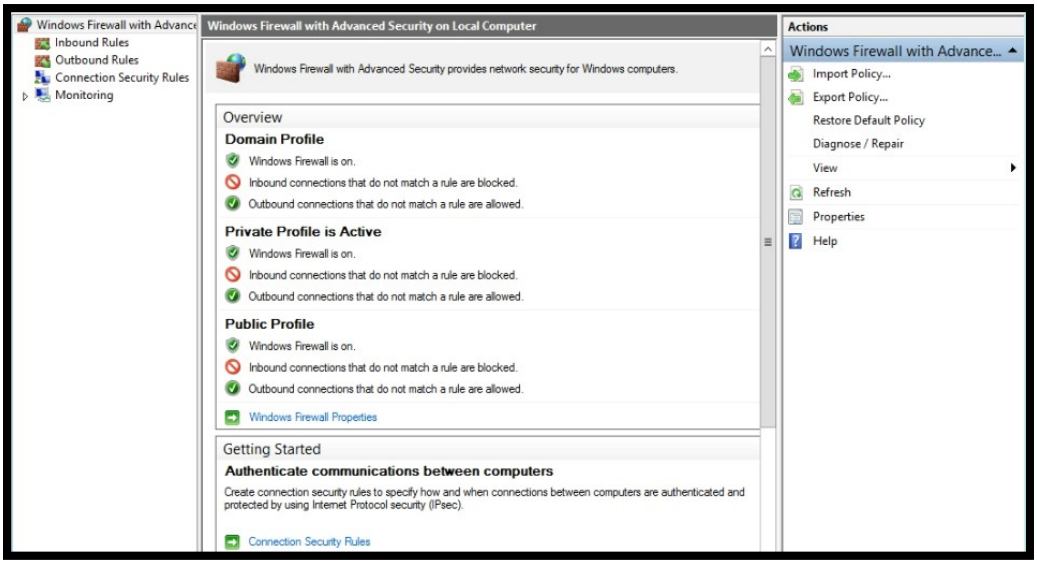
13. Finalizando la configuración, se puede definir una dirección de un responsable al cual notificarle en el caso de resultados inestables o con errores:



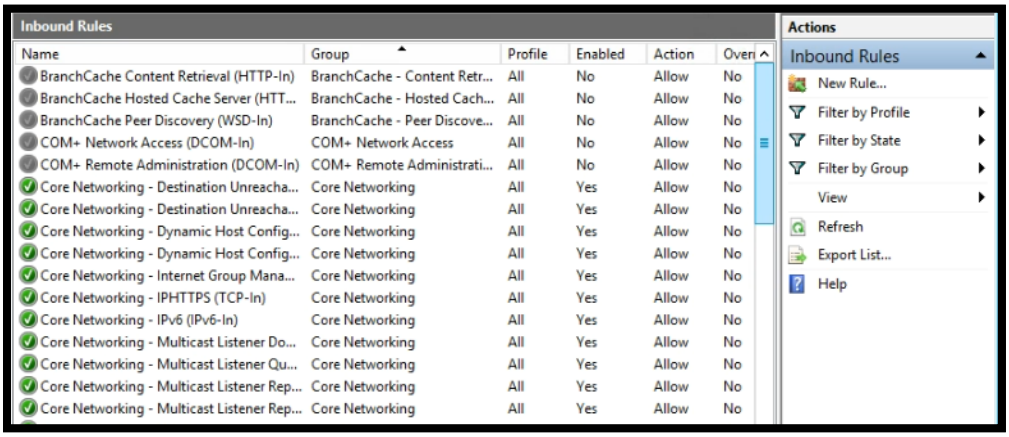
Configuración de firewall

Lo primero a realizar a la hora de configurar el corta-fuego es crear una regla de entrada que abra los puertos desde otros equipos. De manera general, para configurar este tipo de regla, se debe de realizar lo siguiente:

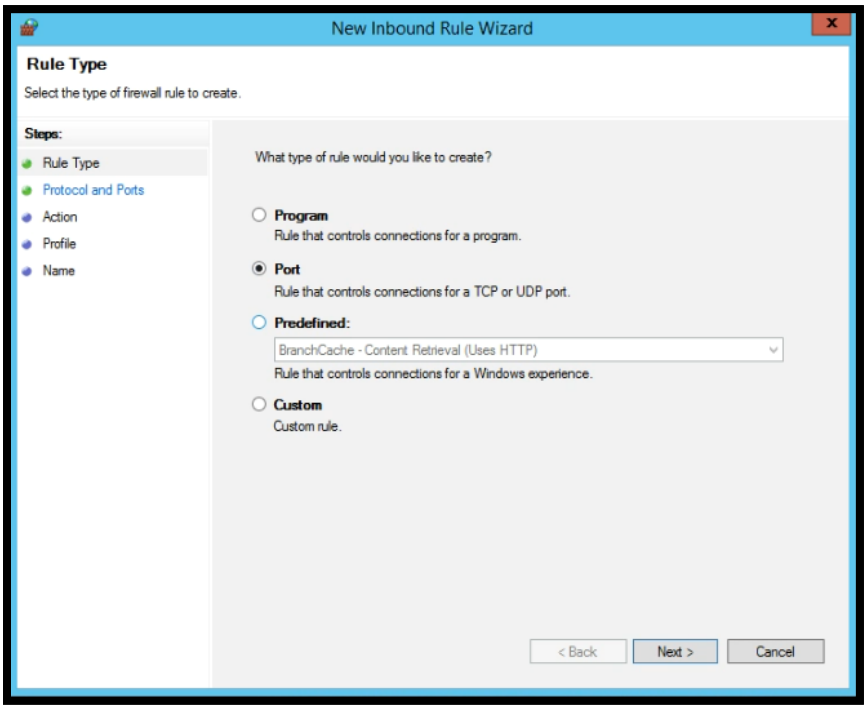
1. Se busca y ejecuta la consola de administración del corta-fuego o firewall de Windows.
2. Ir a la opción de reglas de entrada en la parte izquierda.



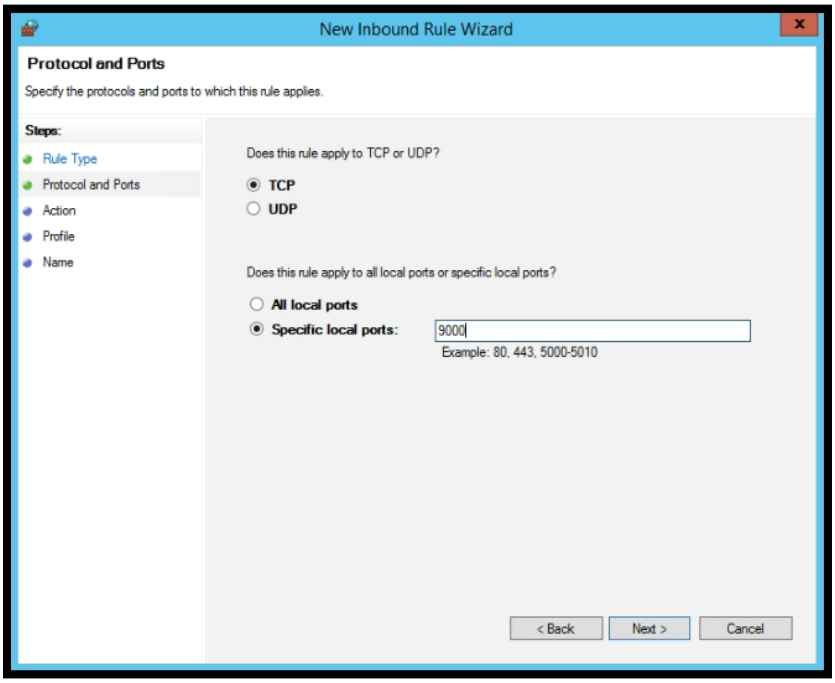
3. Seleccionar regla nueva o "*new rule*":



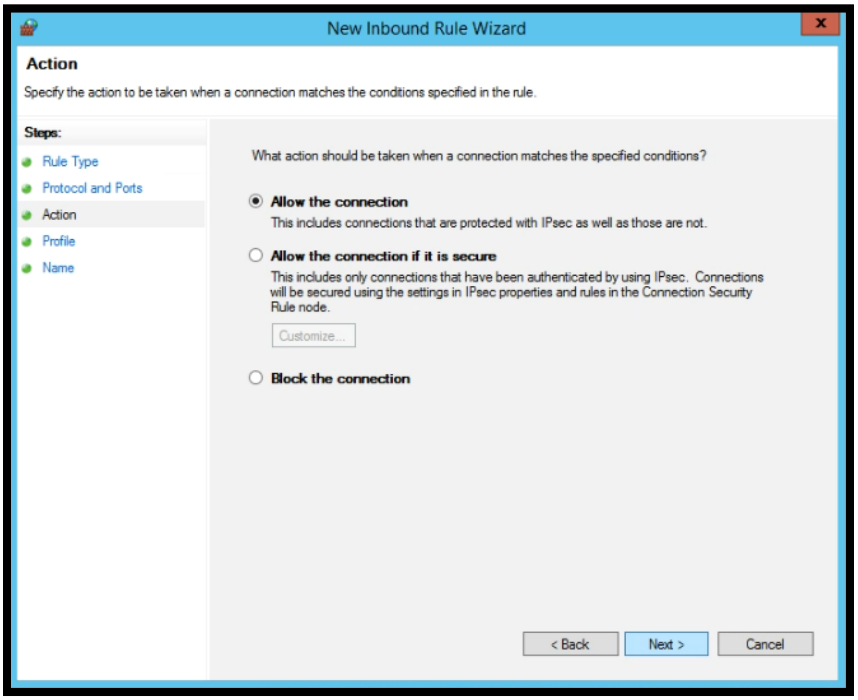
4. Para lo cual se desplegará un cuadro de diálogo en el cual se debe colocar el tipo de regla a aplicar, para este documento, será mediante puerto:



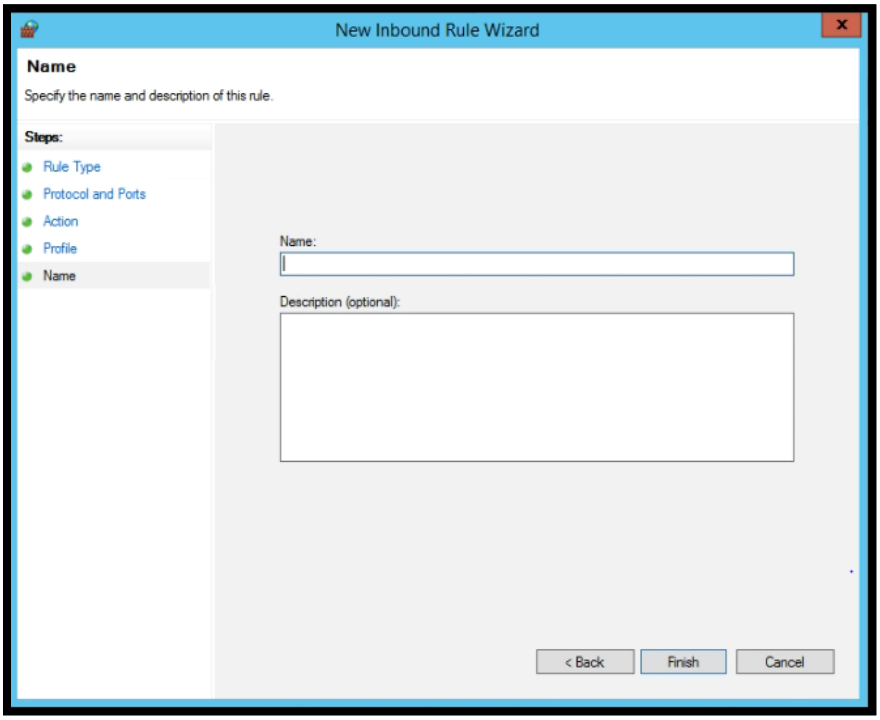
5. Luego se coloca el protocolo, para lo cual se coloca TCP. En la parte baja, se especifica el puerto sobre el cual será aplicada la regla, por ejemplo el puerto 9000 o el 8080:



6. Después de eso se debe colocar el tipo de acción a realizar. Como caso didáctico, se habilitará permitir la conexión o *"Allow the connection"*:

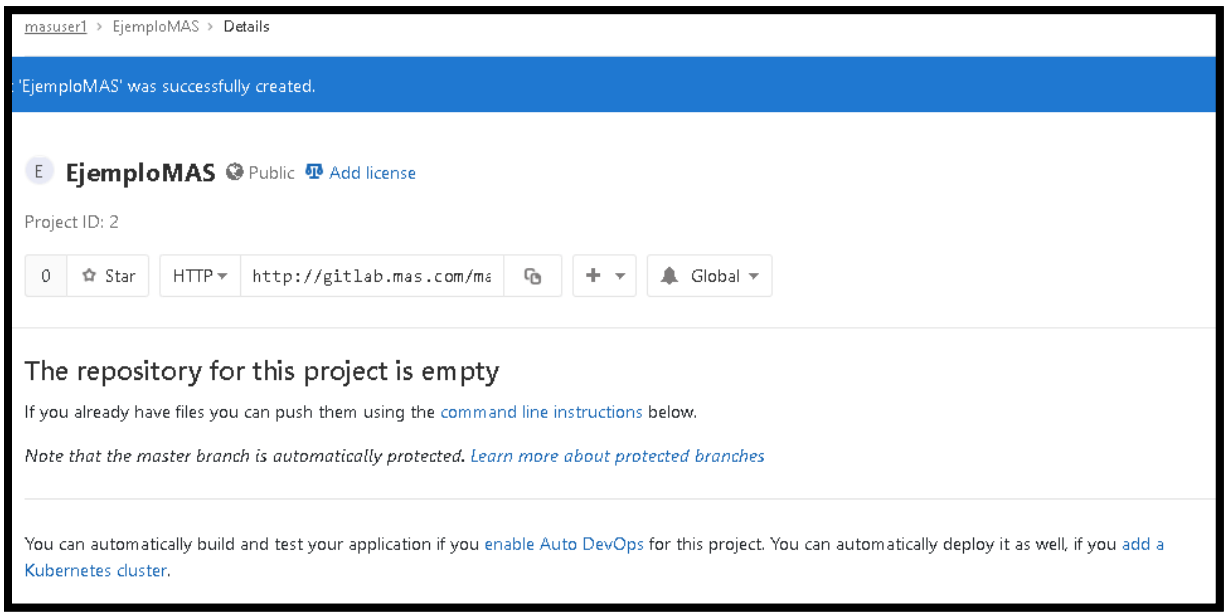


7. Por último, se coloca un nombre distintivo y una descripción para la regla, que puede variar según las políticas de nombre que cada empresa maneja, por ejemplo, **P_9000_ALL**:

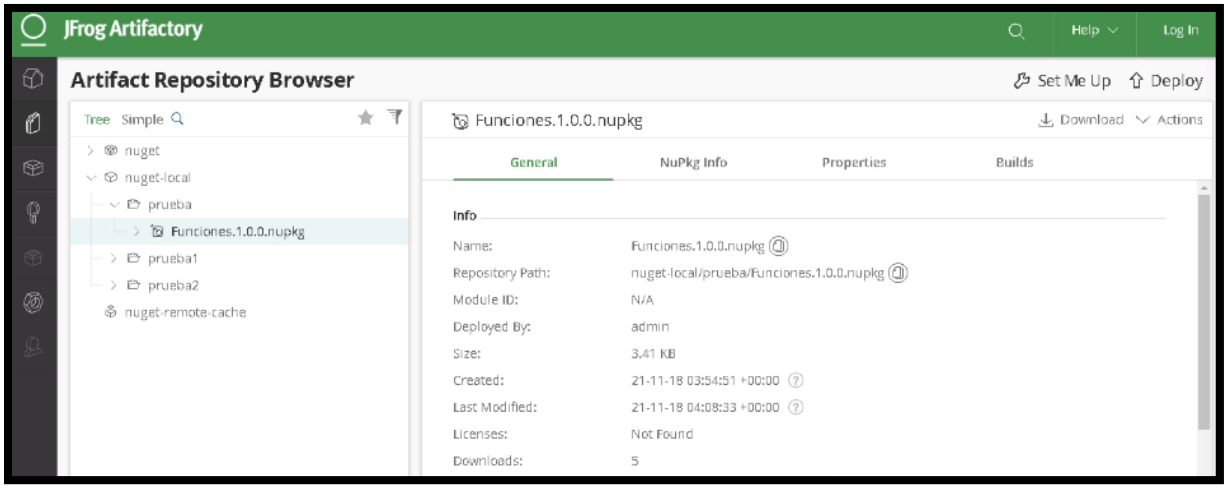


Validación entorno de Integración Continua

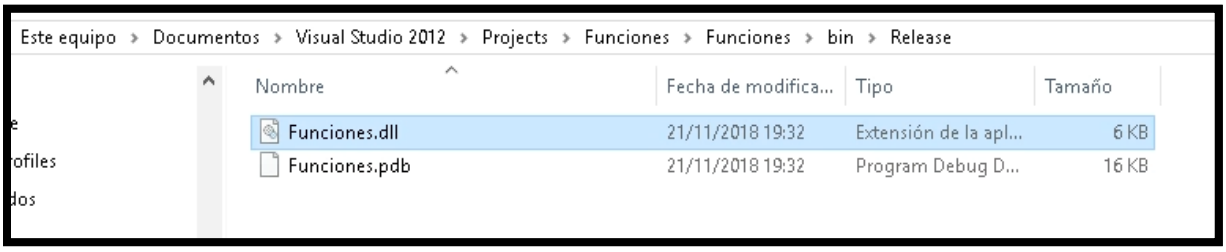
1. Se creó un repositorio Git en GitLab para almacenar el código fuente:



2. Se creó un repositorio en Artifactory, en el cual se cargó una librería creada a manera de ejemplo:



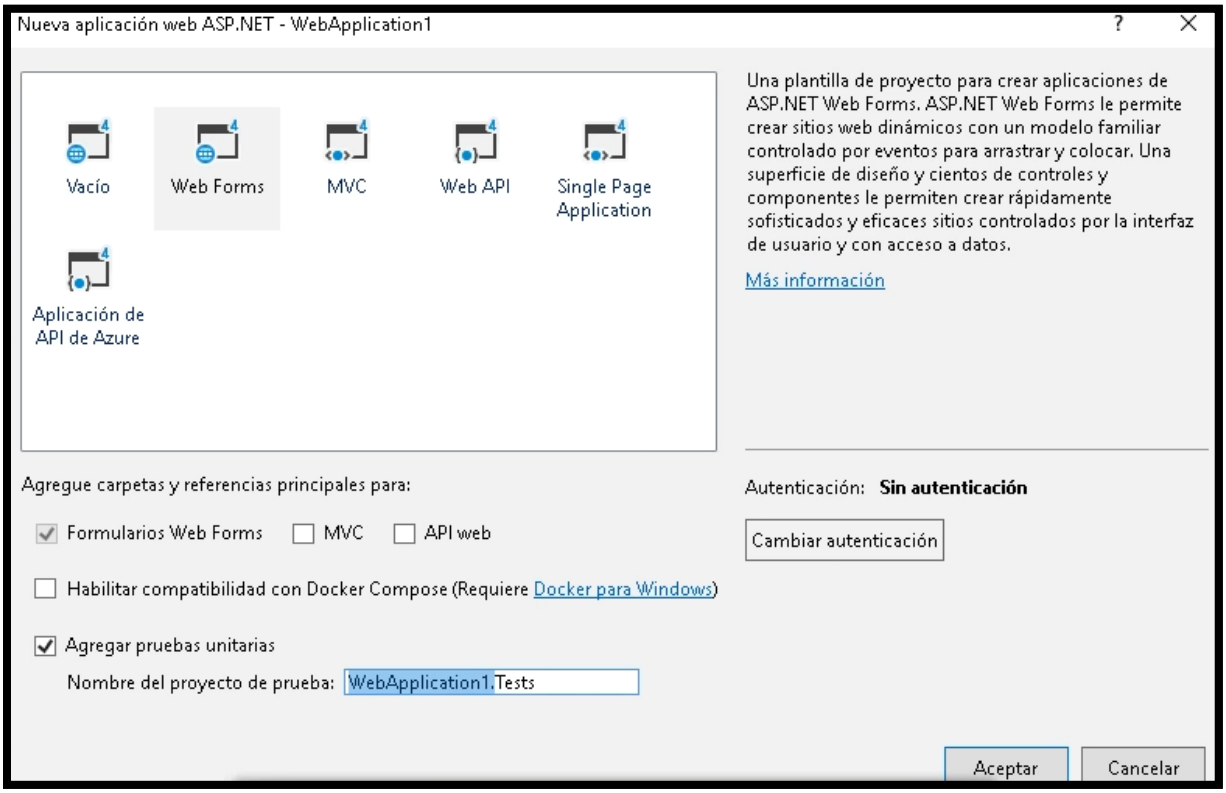
3. Se construye el proyecto librería con NuGet, luego hay que dirigirse al directorio donde se encuentra la librería.



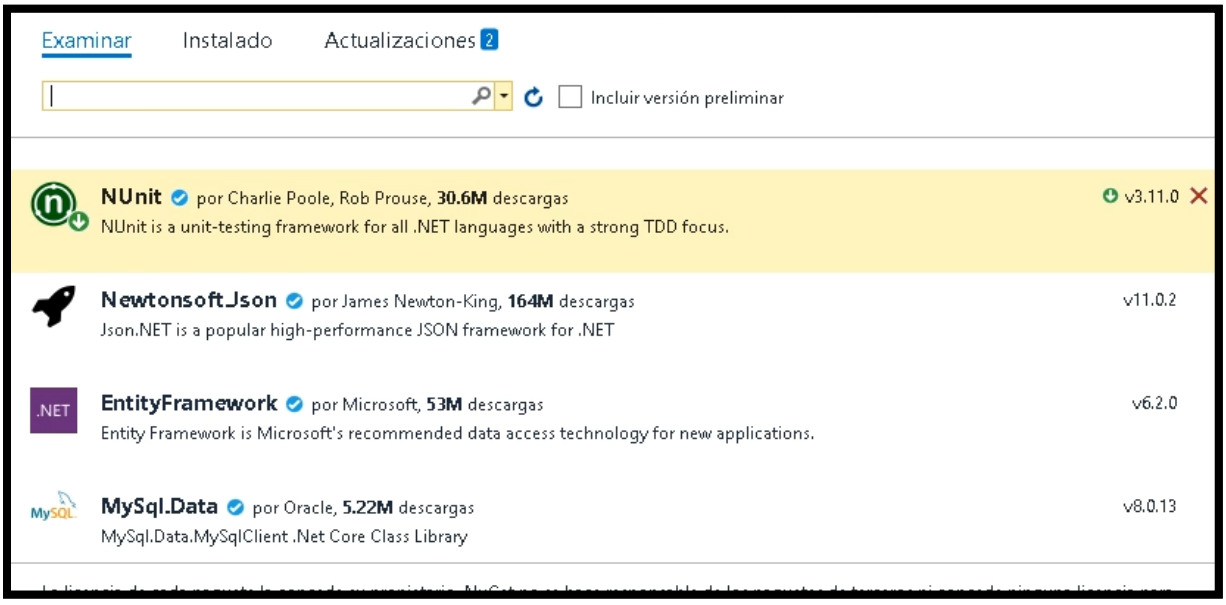
4. Desde una consola se ejecutó el siguiente comando para cargar la librería a Artifactory:

```
C:\Users\W2BFlores\Documents\Visual Studio 2012\Projects\Funciones\Funciones>C:\Users\W2BFlores\Downloads\nuget.exe push "Funciones.1.0.0.nupkg" -Source http://40.87.54.150:8081/artifactory/api/nuget/nuget/prueba1 -ApiKey usermas:User123!
```

5. Se creó un proyecto con .NET Framework 4.5, que tiene como objetivo, dar el resultado de la suma de 2 números. El proyecto incluye el archivo para pruebas:



6. Se le debe adicionar los paquetes referentes a NUnit desde el administrador de paquetes NuGet:



7. Se crean las clases de la aplicación de muestra, a lo cual es oportuno mencionar que se utiliza la librería “Funciones” obtenida desde el repositorio de artefactos:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Funciones;

namespace EjemploMAS2
{
    public partial class _Default : Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
        }
        protected void Button1_Click(object sender, EventArgs e)
        {

            Ejemplo ejemplo = new Ejemplo();
            ejemplo.Num1 = int.Parse(TextBox1.Text);
            ejemplo.Num2 = int.Parse(TextBox2.Text);
            Label4.Text = suma(ejemplo.Num1, ejemplo.Num2).ToString();

        }
        public int suma (int num1, int num2)
        {
            int total;
            total = num1 + num2;

            return total;
        }
    }
}
```

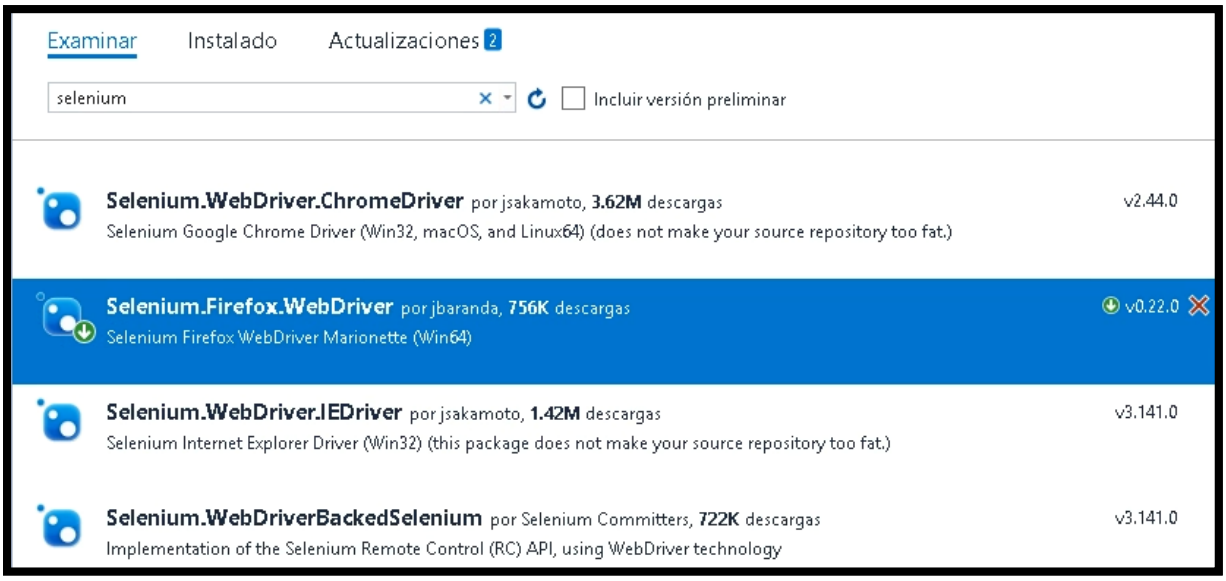
8. Se incorporó las pruebas unitarias con NUnit :

```
using System;
using EjemploMAS2;
using NUnit.Framework;

namespace EjemploMAS2.Tests
{
    [TestFixture]
    public class UnitTest1
    {
        [Test]
        public void SumaPositivos()
        {
            EjemploMAS2._Default mas = new EjemploMAS2._Default();
            Assert.IsTrue(mas.suma(5,7) == 12);

        }

        [Test]
        public void SumaNegativo()
        {
            EjemploMAS2._Default mas = new EjemploMAS2._Default();
            Assert.IsTrue(mas.suma(-5, -7) == -12);

        }

        [Test]
        public void SumaPositivoNegativo()
        {
            EjemploMAS2._Default mas = new EjemploMAS2._Default();
            Assert.IsTrue(mas.suma(5, -7) == -2);

        }
    }
}
```

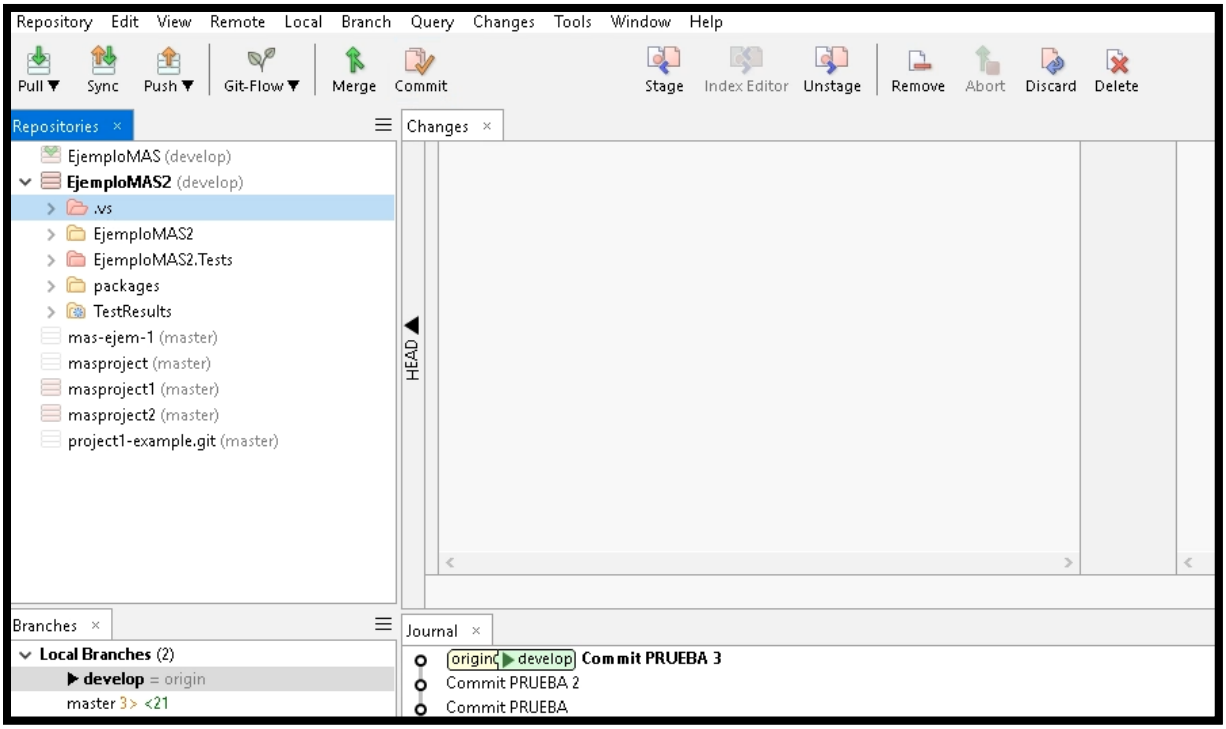
9. Y como último punto relevante, se creó la parte de las pruebas con Selenium:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using NUnit.Framework;
using OpenQA.Selenium;
using OpenQA.Selenium.Firefox;
using OpenQA.Selenium.Remote;

namespace EjemploMAS2.Tests
{
    [TestFixture]
    public class Class1
    {
        [Test]
        public void SeleniumTest1()
        {

            IWebDriver driver = new FirefoxDriver();
            driver.Url = "http://winiis.eastus.cloudapp.azure.com:85/";
                        IWebElement query = driver.FindElement(By.Id("MainContent_Label1"));
            try
            {
                Assert.IsTrue(query.Text == "SUMA DE NUMEROS v3:");
            }
            finally
            {
                driver.Quit();
            }
        }
    }
}
```

10. También, para la carga correcta de Selenium es necesaria la adición del controlador web de Firefox para Selenium:

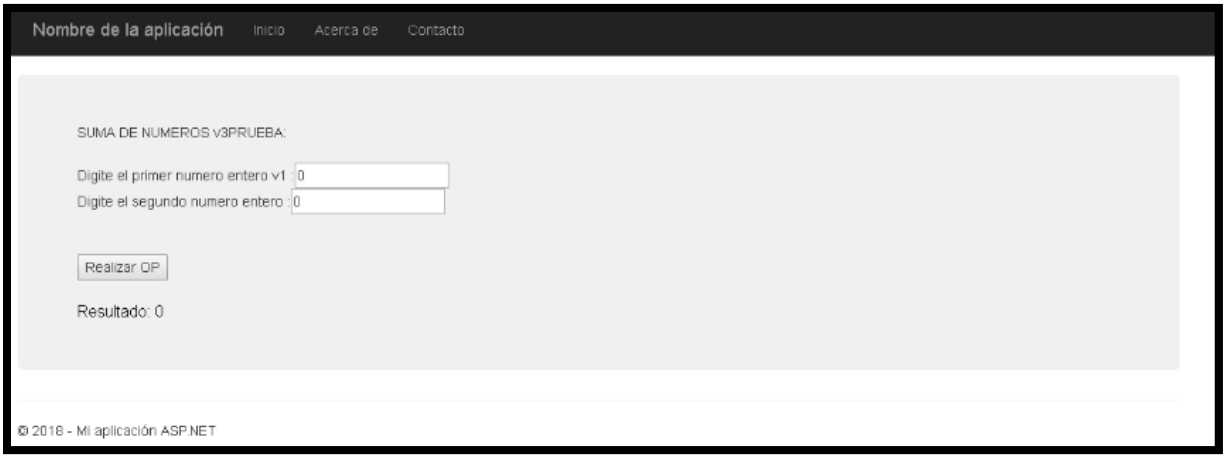


11. Mediante el cliente SmarGit se cargó el código fuente a la rama "*develop*" del repositorio Git:



12. Dentro de Jenkins en Configurar el origen del código fuente, se debe configurar la conexión a GitLab y establecer el repositorio Git:

13. Dentro de la tarea de Jenkins se construye el código mediante el MSBuild.
14. De igual forma se configura la herramienta de análisis de código estático SonarQube, siempre respetando la manera que se ha colocado en la guía.
15. Luego de lo anterior se pasa a la parte de pruebas unitarias de NUnit, tal como se mostró en la parte de integración de Jenkins.
16. Después despliega el archivo en el servidor IIS:



17. Después se realizan las pruebas definidas con Selenium a la aplicación que se encuentra en el ambiente de Pruebas, lo cual se configura de manera evidente, tal como se ha visto antes en la guía.
18. Si al finalizar de todo el ciclo, no se presentaba algún fallo, la aplicación puede ser desplegada en Producción, caso contrario, se puede enviar una notificación a los interesados o responsables.

Referencias

HugeServer. (5 de Noviembre de 2018). *HugeServer Knowledgebase*. Obtenido de How to Install GitLab on CentOS 7: https://www.hugeserver.com/kb/how-install-gitlab-centos7/

Atlassian Confluence 6.12.2. (24 de Diciembre de 2018). *JFrog Become a Professional!* Obtenido de Installing Artifactory on Windows: https://www.jfrog.com/confluence/display/RTF/Installing+on+Windows

Howtoforge. (1 de Ocutbre de 2018). *Howtoforge Linux Tutorials*. Obtenido de How to Install and Configure GitLab CE on CentOS 7: https://www.howtoforge.com/tutorial/how-to-install-and-configure-gitlab-ce-on-centos-7/

Kili, A. (25 de Octubre de 201). *Tecmint: Linux Howtos, Tutorials & Guides*. Obtenido de How to Fix “firewall-cmd: command not found” Error in RHEL/CentOS 7: https://www.tecmint.com/fix-firewall-cmd-command-not-found-error/

Microsoft. (5 de Enero de 2017). *Microsoft Hardware Dev Center*. Obtenido de Enable .NET Framework 3.5 by using the Add Roles and Features Wizard: https://docs.microsoft.com/en-us/windows-hardware/manufacture/desktop/enable-net-framework-35-by-using-the-add-roles-and-features-wizard

Microsoft. (26 de Octubre de 2017). *Windows Support*. Obtenido de Error 0x80070057 when you format a hard disk drive to install Windows 7: https://support.microsoft.com/en-hk/help/2476568/error-0x80070057-when-you-format-a-hard-disk-drive-to-install-windows

Stack Exchange Inc. (27 de Marzo de 2018). *Stackoverflow*. Obtenido de Open firewall port on CentOS 7: https://stackoverflow.com/questions/24729024/open-firewall-port-on-centos-7

Stack Exchange Inc. (4 de Abril de 2018). *Stackoverflow*. Obtenido de Nuget package in Artifactory does not show up in available packages in Visual Studio: https://stackoverflow.com/questions/49639412/nuget-package-in-artifactory-does-not-show-up-in-available-packages-in-visual-st

Bibliografía

Amazon.com, Inc. (28 de Diciembre de 2018). Explicación de la integración continua. Obtenido de Amazon Web Services, Inc.: https://aws.amazon.com/es/devops/continuous-integration/

Arias, F. G. (1999). El Proyecto de Investigación. 3ra edición. Caracas: Episteme. Oral Ediciones.

Aumaille, Benjamín.,(Noviembre 2002), J2EE Desarrollo de aplicaciones Web, Barcelona, España: Ediciones ENI.

Berna Zipa, M. M. (2015). Gestión por Procesos y Mejora Continua, Puntos Clave para la Satisfacción del Cliente. Bogotá: Universidad Militar Nueva Granada.

Bioul, G., Escobar, F., Alvarez, M., Nardin, A., & Ricci Aparicio , E. (2010). Metodologías Ágiles, análisis de su implementación y nuevas propuestas. CACIC 2010 - XVI Congreso Argentino de Ciencias de la Computación, 597-606.

Blanchard, B. (1995) Maintainability: A Key to Effective Serviceability and Maintenance Management, John Wiley & Sons. Inc., New York.

Business Wire. (14 de 10 de 2018). Business Wire. Obtenido de Business Wire: https://www.businesswire.com/news/home/20181014005052/en/Gartner-Announces-Winners-2018-Gartner-Eye-Innovation.

CA Tecnologies. (10 de 2018). CA Tecnologies. Obtenido de CA Tecnologies: https://www.ca.com/us/modern-software-factory/content/how-agile-and-devops-enable-digital-readiness-and-transformation.html#formanchor.

Casal, J., & Mateu, E. (2003). TIPOS DE MUESTREO. Revista Epidem. Med. Prev, 1, 3-7.

Casteleyn, S. et al. (2009). Engineering Web Applications. Springer-Verlag Berlin Heidelberg.

Clareburt, J. (12 de 2017). Github. Obtenido de Github: https://github.com/aspnet/Tooling/blob/AspNetVMs/docs/create-asp-net-vm-with-webdeploy.md.

Código de Comercio (2008). Corte Suprema de Justicia de El Salvador Centro de Documentación Judicial, de www.redhat.com.

Corona, B., Muñoz, M., Miramontes, J., Calvo-Manzan, J. A., & San Feliu, T. (Abril de 2016). Recibe, Revista Electrónica. Obtenido de Estado de arte sobre métodos de evaluación de metodologías ágiles en las pymes: http://recibe.cucei.udg.mx/revista/es/vol5-no1/computacion03.html.

Directorio de Clientes que utilizan IC con CircleCI. circleci.com. Recuperado el 27 de Noviembre de 2016, circleci.com: https://circleci.com/customers/

Duvall, P. M. (2008). Continuous Integration, Improving Software Quality and Reducing Risk. Dorling Kindersley.

Forsgren, N., & Humble, J. (27 de Octubre de 2015). The Role of Continuous Delivery in it and Organizational Performance. Obtenido de In the Proceedings of the Western Decision Sciences Institute (WDSI) 2016, Las Vegas, NV: https://ssrn.com/abstract=2681909.

Fowler, M. (01 mayo de 2006). MartinFowler.com. Recuperado 20 de noviembre del 2016, de martinfowler.com: http://www.martinfowler.com/articles/continuousIntegration.html.

García Orozco, A. M. (2015). La Integración Continua y su Aporte al Aseguramiento de la Calidad en el Ciclo de Vida del Desarrollo de Software. Bogotá: Universidad Distrital Francisco José de Caldas.

Hernandez, R. (5 de 7 de 2015). Testeando Software. Obtenido de Testeando Software: https://testeandosoftware.com/sonarqube-instalacion-basica/.

Honig, R. (2015). Best Practices Integration Testing, Recuperado el 11 de 2016, de http://techbeacon.com/6-best-practices-integration-testing-ci-environment

IDG Network. (02 de Noviembre de 2016). Computerworld from IDG. Obtenido de Las 10 principales predicciones de la industria TIC: http://www.computerworld.es/tendencias/las-10-principales-predicciones-de-la-industria-tic.

IEEE. (1983). IEEE Standard Glossary of Software Engineering Terminology (ANSI/IEEE Std 729-1983). USA: IEEE, 1983. doi: 10.1109/IEEESTD.1983.7435207.

Joshi, A. (02 de 08 de 2012). Microsoft. Obtenido de Microsoft: https://docs.microsoft.com/en-us/iis/publish/troubleshooting-web-deploy/troubleshooting-web-deploy-problems-with-visual-studio

Jummp. (20 de 5 de 2010). Jummp. Obtenido de Jummp: https://jummp.wordpress.com/2010/05/20/reflexiones-sobre-la-adaptacion-de-sonar-a-nuestra-metodologia-de-calidad-para-su-uso-en-mi-organizacion/.

Knowledge Powerhouse (2016). Microservices Interview Questions: Good Collection of Questions Faced in Architect Level Technical. Edición Digital.

Microsoft Corporation. (1 de Septiembre de 2018). An introduction to NuGet. Obtenido de Microsoft Docs: https://docs.microsoft.com/en-us/nuget/what-is-nuget.

Miranda, João. (17 de marzo de 2014). www.infoq.com. Recuperado el 27 de noviembre de 2016, de www.infoq.com: https://www.infoq.com/news/2014/03/etsy-deploy-50-times-a-day.

Martínez Miguélez, M. (2006). La Investigación Cualitativa (Síntesis Conceptual). Revista Invest.en Psicol., 9(1), 123-146.

Martínez-Salgado, C. (2012). El muestreo en investigación cualitativa. Principios básicos y algunas controversias. Ciência & Saúde Coletiva, 17(3), 613-619.

Moquillaza Henríquez, S. D., Vega Huerta, H., & Guerra Grados, L. (Diciembre de 2010). Programación en N capas. Revista de Investigación de Sistemas e Informática, 57-67.

Navarro Cadavid, A., Fernández Martínez, J. D., & Morales Vélez, J. (2013). Revisión de metodologías ágiles para el desarrollo de software. Prospect. Vol. 11, No. 2, 30-39.

Null, C. (12 de 2017). Tech Beacon. Obtenido de Tech Beacon: https://techbeacon.com/10-companies-killing-it-devops.

Oliveros, A., del Valle Rojo, S., Wehbe, R., & Rousselot, J. (2011). Requerimientos para Aplicaciones Web. XIII Workshop de Investigadores en Ciencias de la Computación, 577-582.

Palacio, A. L. (2015). Evaluación del grado de agilismo basado en los objetivos y necesidades de los equipos de trabajo. Valencia: Universidad Politécnica de Valencia.

Pinzón, S., Carlos, J., & Bolaños, G. (junio de 2008). La gestión, los procesos y las metodologías de desarrollo de software. Actualidad Tecnológica, 83-98.

Pressman, R. S. (2010). Ingeniería del Software: Un enfoque práctico (Séptima ed.). New York: McGraw-Hill.

Randell, B. (1996). *The 1968/69 NATO, Software Engineering Reports.* Dagstuhl: Universidad de Newcastle.

Real Academia Española. (2014). Continuo. En Diccionario de la lengua española (23.a ed.). Recuperado de http://dle.rae.es/?id=AVqx1Ac.

Red Hat, Inc. Red Hat, Red Hat Enterprise Linux (2012) IC con JBoss Trading, una aplicación empresarial de referencia. JBOSS Enterprise Middleware, de www.redhat.com.

Sampieri, R., Collado, C., & Lucio, P. (2010). Metodología de la Investigación. México DF: McGRAW-HILL / INTERAMERICANA EDITORES, S.A. DE C.V.

Sayed, A. (13 de 7 de 2018). Tech Beacon. Obtenido de Tech Beacon: https://techbeacon.com/why-agile-devops-are-key-any-digital-transformation.

Setende, Hamilton. (2012). www.academia.edu. Recuperado el 23 de noviembre del 2016, de www.academia.edu: http://www.academia.edu/4865003/Differences_advantages_and_disadvantages_between_in-house_development_IT_systems_and_industry_standard_ERP_system.

Software in the Public Interest. (Diciembre de 2018). Jenkins User Documentation. Obtenido de https://jenkins.io/doc/.

Sommerville, I. F. (2005). Ingeniería del Software. / Ian Sommerville . Madrid, España: PEARSON.

Techopedia. (2016). Build. En Enciclopedia Digital Techopedia. Recuperado de https://www.techopedia.com/definition/3759/build.

TIOBE software BV. (febrero de 2018). TIOBE The Software Quality Company. Obtenido de https://www.tiobe.com/tiobe-index/.

Torrejon, L. (07 de noviembre de 2017). Las predicciones de IDC proporcionan un modelo y las bases para convertirse en una empresa nativa digital. Obtenido de http://www.blog-idcspain.com/predicciones-idc/.

Uchino, M. (12 de 2016). Matthew Yuki Uchino. Obtenido de Matthew Yuki Uchino: http://matthewyukiuchino.com/how-to-use-jenkins-to-deploy-c-web-applications-to-iis/.

Unidad Editorial Información General, S. (22 de 12 de 2015). Unidad Editorial. Obtenido de Unidad Editorial: http://www.expansion.com/economia-digital/companias/2015/12/22/56795576268e3ea2388b471f.html.

Universidad Tecnológica del Suroeste de Guanajuato. (2013). Ciencias de la Ingeniería y Tecnología, Handbook T-I. Guanajuato: ECORFAN.

Vaca Barahonaa, B., Hidalgo Poncea, B., & Arcos Medinaa, G. (2015). La gestión y la producción de software con plataformas tecnológicas colaborativas. Revista Tecnológica ESPOL, 105-120.

Yepes González, J. D., Pardo Calvache, C. J., & Gómez Gómez, O. S. (2015). Revisión sistemática acerca de la implementación de metodologías ágiles y otros modelos en micro, pequeñas y medianas empresas de software. Revista Tecnológica ESPOL – RTE, Vol. 28, N. 5, 464-479.