PROYECTO DE SOFTWARE
AVANCE DEL PROYECTO:
TEMA: Construcción e Implementación de una guía de buenas prácticas de Integración Continua en proyectos de desarrollo en el área de tecnología en una empresa privada.

INTEGRANTES:
EDISON PAUL MOSQUERA
edison.mosquera.leon@udla.edu.ec
DAVID MARTINEZ
david.martinez.bravo@udla.edu.ec
CARLOS XAVIER RUIZ
carlos.ruiz.agila@udla.edu.ec

Febrero 2021

1 ANTECEDENTES

"La integración continua actualmente pasa por pipelines como nueva forma de trabajo dentro de la metodología DevOps. De esta manera los programadores pueden definir el ciclo de vida de las aplicaciones con ayuda de la codificación.

Cabe destacar que "Pipeline" es un término en inglés que se refiere a una suerte de tubería informática metafórica, la cual tiene como finalidad segmentar datos. De esta forma se puede aumentar el rendimiento de un sistema digital, gracias a un pipeline compuesto por microprocesadores y software." (GL Group, 2019).

En la actualidad son necesarias e importantes las metodologías para trabajo de software, para además de garantizar la competitividad de las empresas producir innovaciones a nivel de las aplicaciones y a nivel corporativo.

Entre los puntos importantes del uso de pipelines está la disminución de errores, desembocando en la entrega de productos en menor tiempo, y con-

tribuyendo a la gestión de recursos. También se genera el beneficio de brindar a los desarrolladores los espacios para realizar tareas necesarias que requieren una ejecución manual.

Los desarrolladores pueden utilizar los pipelines para automatizar procesos útiles en la realización de cambios en los códigos de forma periódica, permitiendo la ejecución de pruebas y revisión de versiones. De esta manera, hacen parte los recursos en línea como los repositorios compartidos a través de Git y demás opciones, y características que permitan reducir los riesgos informáticos posibles.

Otro punto importante, se centra en la reducción de la carga de trabajo por parte de los desarrolladores impidiendo el colapso y posible riesgo de error humano, para cumplir con los protocolos y estándares de uso y rendimiento de los proyectos.

Para finalizar, se argumenta sobre la creación y utilización de pipelines automatizados en pro del dinamismo, aportando en alguna medida a los desarrollos de inteligencias artificiales que permitirán a futuro crear desarrollos de proyectos de manera sencilla, obteniendo retroalimentación y supervisión en tiempo real, consiguiendo resultados satisfactorios en las entregas.

2 ALCANCE

El proyecto realizará la entrega de una guía de buenas prácticas de integración continúa basada en la implementación de pipelines (Scripts Automatizados) para proyectos piloto en el área de desarrollo de tecnología de una empresa privada, aportando a la integración continua y brindando un desarrollo de calidad frente a los posibles errores, ya sean de carácter organizativo, así como de estructura, facilitando un flujo ágil y claro bajo la plataforma de Azure DevOps.

El gobierno de este proyecto estará a cargo de la persona responsable del área de sistemas de la institución o un delegado, quedando de acuerdo en que estará a cargo de brindar todas las facilidades, tanto de procesos como de infraestructura para que el proyecto siga su curso.

Se tienen 3 faces:

1. Investigación y descubrimiento:

   Se trata de un on-boarding del equipo para que se familiarice con las tecnologías y el software desarrollado por la institución con un máximo de 3 proyectos piloto, en este punto no habrá entregables técnicos.

2. Desarrollo de Pipelines (Scripts):

   Con el conocimiento adquirido y familiarizados de la etapa anterior se procederá a generar scripts de automatización empezando por los proyectos

que más trabajo requieran para realizar su compilación teniendo en cuenta que se prevé encontrarse con arquitecturas monolíticas, se puede llamar al entregable de esta etapa un mínimo producto viable.

3. Optimización y generalización de Pipelines (Scripts):

   Cuando el MVP (Mínimo producto viable) esté funcionando correctamente se procederá a aplicar la misma receta para el resto de los proyectos del piloto, haciendo las adecuaciones necesarias para que en el futuro pueda ser aplicado para cualquier proyecto de software que comparta estas tecnologías.

El entregable final comprende de los siguientes artefactos repartidos durante las 3 anteriores etapas:

1. Pipeline (Script) por cada proyecto que contenga:

   o Revisión automática de calidad y vulnerabilidades código.

   o Compilación automática de código.

   o Ejecución automática de pruebas unitarias (De existir, usualmente aplicativos legados o sin esta práctica no disponen de pruebas unitarias).

   o Versiona miento de archivos compilados.

2. El Pipeline en si genera los siguientes reportes:

   o Informe de vulnerabilidades y calidad de código.

   o Informe de éxito o fracaso del Pipeline.

   o Informe de pruebas unitarias. (De existir)

3. Mediante la herramienta de Azure DevOps:

   o Configuración de bloqueo automático para unir código en caso de que el Pipeline falle.

   o Configuración de revisión por pares obligatoria, nadie puede unir su código a máster si alguien más del equipo no aprueba el cambio.

4. Documentación:

   o Estado inicial de los proyectos piloto (Calidad de Código, Pruebas Unitarias y manejo de secretos).

o Documento de puntos de mejora y posible camino para adoptar metodologías agiles y DevOps dentro de los equipos de desarrollo.

o Documento técnico de funcionamiento de los pipelines generados.

o Documento guía para reusar pipelines en futuros proyectos de software.

Los supuestos por parte de la empresa son los siguientes:

• La empresa dispone de una suscripción activa de Azure DevOps con capacidad suficiente para el equipo de este proyecto (3 personas) y para los equipos de desarrollo de los proyectos piloto.

• La empresa dispone de una de las siguientes opciones:

o Servidor IaaS con un mínimo de 4 CPUs y 8 RAM para que el equipo de proyecto pueda instalar SonarQube en la nube.
o Servicio SaaS de Sonar Cloud con el número de líneas suficientes para los 3 proyectos piloto.

• Acceso al código fuente de los 3 proyectos piloto.

• Acceso privilegiado para al menos una persona del equipo del proyecto dentro de Azure DevOps, en este caso sería para Edison Mosquera.

• Acceso bajo demanda a los desarrolladores de la empresa de los 3 proyectos piloto, esto para que pueda ser compartido su conocimiento del funcionamiento.

3 JUSTIFICACIÓN

Es importante conocer en la actualidad a que se enfrenta los proyectos de codificación desarrollados con sistemas automatizados, evitar esencialmente las amenazas y vulnerabilidades tratando de conocer que puede ayudar a ser óptimos en el momento de su uso o manejo de diferentes entidades o individuos. Obviamente, no toda la aplicación es invencible, pero se pretende que la aplicación no sea vulnerable a errores y se logren corregir.

La presente investigación y desarrollo se enfocará en la temática sobre la integración continua, utilizando pipelines automatizados y cómo este puede ayudar a desarrollar mejores aplicaciones y proyectos de codificación, lo cual podrá ser muy eficiente según los estándares y protocolos que se tienen en cuenta dentro del campo informático, tanto en la construcción como en la verificación por parte de los desarrolladores.

Se cree que mediante una implementación basada en el código y generación de varios archivos ejecutables pueden servir para mejorar la base de las propuestas de proyectos. Los estudios acerca de las aplicaciones y programas llevarán al éxito permitiendo desarrollar a futuro mejores programas en otros ámbitos y beneficiará a más de uno si se sabe indagar acerca de las necesidades de cada producto o servicio que se crea u obtiene en los desarrollos. La importancia del desarrollo es mejorar, siempre intentar avanzar y evolucionar a la siguiente generación.

4 OBJETIVO GENERAL

Implementar una guía de buenas prácticas de Integración Continua en el área de desarrollo de tecnología de una empresa privada mediante el uso de pipelines (Scripts Automatizados), metodología ágil y DevOps para mejorar los entregables de software (su tiempo y calidad), informes de estado y guías metodológicas.

4.1 "OBJETIVOS ESPECÍFICOS"

• Identificar los procesos de desarrollo y liberación de software de la empresa privada y determinar las necesidades de integración continua en su estado actual.

• Producir automatizaciones dentro del ciclo de vida del desarrollo de software para reducir su tiempo de implementación en los equipos de desarrollo de la empresa para que esta mejore su infrestructura

• Elaborar una guía de buenas prácticas al finalizar el proceso de implementación de pipelines que reúna informes de estado y guías metodológicas que refieran los procesos de integración continua como producto final de la presente investigación.

5 METODOLOGÍA

Metodología de Investigación:

Para el presente proyecto, se utilizará la metodología lógica con el tipo de investigación descriptiva y el método inductivo. En primer lugar, utilizando la investigación descriptiva se pretende dar una visión general del proyecto describiendo los componentes, y generando manifiestos que de manera clara estipulen las características y comportamientos de cada fase.

El objetivo de la investigación descriptiva consiste en situarse en saber las situaciones, costumbres y actitudes predominantes a través de la representación exacta de las actividades, objetos, procesos y personas. Su fin no se limita a la

recopilación de datos, sino a la predicción e identificación de las relaciones que existen entre dos o más variables.

En una de las etapas primero se debe examinar las características del problema, luego se define y sé formula la hipótesis de los enunciados, que son supuestos en base a la hipótesis y luego se adoptan, en cuanto se elige esto, los temas y la fuente apropiada se debe seleccionar y se elaboran las técnicas para la recolección de datos ahí se establecen a fin de clasificar los datos, las categorías precisas para adecuar al propósito del estudio, se verifica la validez de las técnicas y a partir de ahí se realizan las observaciones necesarias que deben ser objetivas y exactas, al final se describen, analizan e interpretan los datos obtenidos en términos claros y precisos.

Por otra parte, el uso del método inductivo es requerido para este proyecto, pues conociendo de antemano que se abarca la formulación de esquematizaciones sobre un tema poco tratado en el campo del desarrollo de aplicaciones y otros proyectos afines por estudiantes y aprendices, se busca encontrar soluciones a los distintos casos de uso que serán necesarios y útiles, proporcionando demostraciones y generando el conocimiento sobre los elementos.

Los pasos recomendados propuestos para este proyecto son los siguientes:

- Observación y registro de los hechos.
- Análisis y clasificación de los hechos.
- Derivación inductiva de una generalización a partir de los hechos.

Figura 01. Modelo del método inductivo

Metodología de Desarrollo:

Para el presente proyecto, se utilizará la metodología LEAN mediante la aplicación de DEVOPS.

Los equipos usualmente trabajan de forma aislada de todo el ciclo de vida del desarrollo e implementación de software, las pruebas unitarias o funcionales y los despliegues o puestas en producción son realizadas por diferentes equipos, sumando a esto la problemática que todo el proceso es de manera manual lo que puede ocasionar fallos en la implementación.

En la actualidad se ve que las instituciones que tienen un departamento de DevOps o de TI de alto rendimiento, entrenado para este fin, tienen una mayor probabilidad de exceder sus objetivos de eficiencia y calidad. Por lo que se puede decir que las empresas que apuestan por DevOps obtienen una ventaja mucho mayor en comparación con departamentos de tecnología que utilizan esquemas como ITIL3.

Existen algunas claves en esta metodología que nos ayudan a que se implemente de manera exitosa:

• Gestión de la configuración mediante archivos versionados. • Gestión del despliegue en ambientes previos al productivo. • Integración continua. • Despliegue automático sin intervenciones manuales. • Automatización de pruebas unitarias y funcionales. • Monitoreo de la calidad de código.

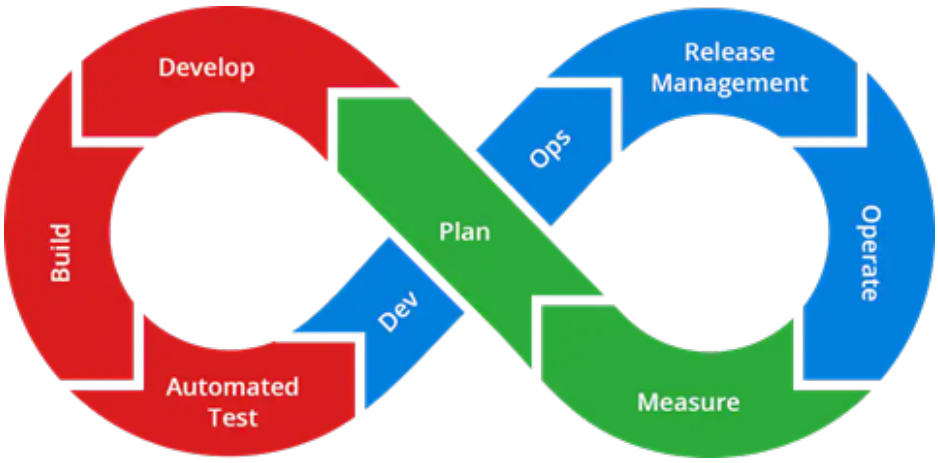

Figura 02. Modelo metodología LEAN

También hay que tomar en cuenta que hay 3 ideas principales dentro de esta metodología:

• DevOps es una metodología ágil para el desarrollo de Software. • La colaboración directa entre los equipos es fundamental. • Nos ayuda a generar mejor software mantenible en el tiempo.

Hay que recordar que en una metodología ágil el centro de los procesos son las personas y no el proceso en sí mismo como ocurre en la metodología cascada por lo cual cada institución adapta DevOps a su cultura organizacional.

6 CRONOGRAMA

ACTIVIDAD	FEBRERO	MARZO	ABRIL	May	June	July	August
Obtener Accesos	15 Dias						
OnBoarding Codigo	15 Dias						
Definicion Estrategia		15 Dias					
Creacion Pipeline Piloto		15 Dias					
Generalizacion Pipeline			15 Dias				
Correcion de errores Pipelines			15 Dias				

Figura 03. Cronograma de trabajo

7 ESTADO DEL ARTE

El estado del arte que se abarca en este proyecto de titulación se encarga de investigar las diferentes implementaciones sobre la aplicación de pipelines que existe tanto a nivel nacional como internacional para las diferentes instituciones educativas superiores. En este trabajo se pretende hacer referencia, investigar y aprender de estos procesos o documentaciones para desarrollar el objetivo principal con la institución educativa y esta sea tan eficiente como las demás. Es importante conocer los conceptos básicos que giran respecto al tema y que sean beneficiosos para los usuarios o socios en los que se encarga la implementación.

Aquí un breve resumen de diferentes aplicaciones o temas relacionados con el proyecto:

7.1 "DevOps y Seguridad Cloud"

Según los autores Caparrós, Guijarro y Cubero (2019), Primero se debe conocer que son las definiciones de DevOps, su terminología, que significa para los equipos, esto permite que los roles que antes estaban aislados estén coordinados para que colaboren en producir varios productos mejores y muy confiables. Son varias prácticas y herramientas que nos da DevOps, los equipos adquieren la capacidad de responder mejor a las necesidades de los usuarios o clientes, ya que lo más importante es su confianza cuando se las crea y logra alcanzar objetivos empresariales en poco tiempo. Se observa la infraestructura de DevOps y se muestra la modalidad del cloud que comprende como un concepto básico de los servicios informáticos. Uno de los pilares de DevOps es la integración continua, la que permite agregar los minúsculos cambios que se realizan por los desarrolladores a la par del software desarrollado de manera que pueda probarse y que se despliega en entorno lo antes posible. [7]

7.2 "DevOps y Seguridad Cloud: Jenkins en detalle"

Jenkins es un software de integración continua open source escrito en Java, está establecido en el proyecto Hudson y proporciona integración continua para el desarrollo de software. Es un método que corre en un servidor que es un contenedor de servlets, como Apache Tomcat. Puede soportar herramientas de control de versiones, como CVS, Subversion, Git o Mercurial, y puede producir proyectos basados en Apache Ant y Apache Maven, asimismo como scripts de shell, etc. Una de las características de Jenkins es tremendamente maleable, con más de 350 plugins disponibles. Es viable poseer varios proyectos configurados en el mismo servidor que utilicen diferentes versiones de JAVA en el instante de la compilación. Jenkins es fácilmente clusterizable, lo que implica que se puede haber varios esclavos configurados en el mismo servidor experto. La ventaja más obvia es que se puede oscilar la carga de los diferentes proyectos en varios servidores; otra ventaja es que los esclavos pueden fundamentarse en diferentes arquitecturas, con lo cual se pueden agrupar aplicaciones basadas en Linux y otras en Windows bajo el propio servidor maestro. Otra característica de Jenkins son sus integraciones con Git, JMeter, SOAP-UI, JUnit, Sonar, Nexus, etc. Esto permite elaborar, por ejemplo, una batería de prueba precedentemente de provenir a la compilación del código con el objetivo de afirmar un cierto grado de atributo, llegando incluso a malograr la compilación si los resultados de los tests no son satisfactorios. Jenkins está diseñado para existir en constante información con los desarrolladores, y para ello consta de un poderoso sistema de notificaciones para continuar el estado de los diferentes proyectos. Jenkins es práctico en enviar correos, tuitear, mover señales luminosas o inclusive sonoras, y mucho más gracias a su amplia API.

[7]

Dentro del siguiente apartado se recopila información referente al proyecto de titulación orientada al desarrollo de entregas continuas de software, para lo cual, se obtuvo puntos relevantes de diferentes proyectos que se relacionan con su desarrollo y se muestra a continuación:

7.3 "Arquitectura tecnológica para la entrega continua de software con despliegue en contenedores"

Según Luis Alberto Iñiguez Sánchez (junio 2017), con la implementación de un pipeline de entrega continua de software los desarrolladores obtienen varios beneficios: i) disponibilidad inmediata de entornos similares a los de producción; ii) automatización inmediata de tareas repetitivas; iii) menor tiempo de aprobación para la conformidad de las tareas realizadas; iv) retroalimentación más rápida de los usuarios para mejorar el software.

En un modelo de entrega continúa basado en tecnología que provee un soporte completo al stack tecnológico definido por la unidad de desarrollo de software de la Dirección de Tecnologías de la Información de la Universidad de Cuenca con herramientas específicas para la ejecución de pruebas de cada uno de los componentes de la arquitectura de la solución, implementándose una prueba de concepto para validar la arquitectura tecnológica planteada.
[4]

7.4 "The BOSS Index: Tracking the Explosive Growth of Open-Source Software."

En la investigación realizada por Shahin, Babar, Zhu, (2017) en la que se aborda la entrega continua de software, los autores realizan una revisión sistemática de varios artículos en los cuales de 449 artículos se escogió a 5 que determinaron que existen elementos de índole procedimental, organizacional y operacional que facilitan el uso y la implementación de prácticas de entrega continua de la siguiente manera: reduciendo el tiempo de construcción y pruebas en la entrega continua durante la etapa de integración continua. Aumentando la visibilidad y conciencia de los resultados de la etapa de integración continua. Dando soporte(semi)automatizado de pruebas continuas. Detectando violaciones, defectos y fallas durante la integración continua. Abordando los problemas de seguridad y escalabilidad en el proceso de implementación. Mejorando la confiabilidad y fiabilidad del proceso de implementación.
[6]

7.5 "Modelado de las diferencias de prácticas de integración continua en el desarrollo de software de la industria"

Esta sección presenta la propuesta del modelo, el estudio de caso ilustrativo, claramente de la revisión de la literatura conducida que actualmente no hay consenso en la integración continua como práctica única y homogénea. Una comparación significativa de los proyectos de desarrollo de software, simplemente afirmando que utilizan la integración continua es una información insuficiente, ya que en su lugar es necesario preguntarse qué tipo de se utiliza la integración continua. También algunos que, teniendo en cuenta las diferencias dramáticas de los efectos de integración continúa experimentados (StåhlandBosch, 2013), se necesita preguntar que tiene variantes de continuo la integración de cualquier beneficio propuesto (o desventaja) es un efecto de este propósito, basado en los hallazgos del estudio, se ha propuesto un modelo descriptivo para una mejor documentación de las variantes de integración continua.
[8]

En la siguiente sección de este documento, se recopila información referente a los entornos y conceptos básicos con los cuales se trabajará en el desarrollo del proyecto de titulación:

A. Cloud: Para Caparrós, Guijarro y Cubero (2019), la entrega de servicios informáticos a clientes o usuarios por medio de una red. Este nuevo modelo de prestación de servicios permite añadir una capa de abstracción frente a los clientes que no saben dónde estos están ubicados (normalmente alojados en varios proveedores y repartidos por todo el mundo) ni la gestión de recursos que usan. Los servicios en la nube atienden las peticiones recibidas y aportan una flexibilidad y adaptabilidad de recursos frente a la demanda de forma totalmente transparente.
[7]

B. Pipelines: Según D. Óliver L. Sanz San José (2018) un pipeline es un conjunto de elementos de procesos, conectados en serie, de modo que la salida de uno de estos es utilizado como entrada del siguiente El uso de pipelines facilita la automatización y la reutilización de las herramientas de procesos, permitiéndonos usar los mismos elementos en distintas aplicaciones. [10]

C. Scripts: Un script se trata o se desarrolla como un código de programación informática, usualmente fácil y sencillo en su uso, que contiene comandos o instrucciones que se van ejecutando de manera secuencial, normalmente se utilizan para manejar el comportamiento de una aplicación o para interactuar con el sistema operativo.

D. Automatización: La automatización de tareas es, en informática, el conjunto de métodos que sirven para hacer tareas repetitivas en un computador. Algunos métodos para la automatización de tareas son la programación simple, los macros, los intérpretes y las bombas lógicas.

E. Integración continua: La integración continua es una práctica de desarrollo de software mediante la cual los desarrolladores combinan los cambios en el código en un repositorio central de forma periódica, tras lo cual se ejecutan versiones y pruebas automáticas. La integración continua se refiere en su mayoría a la fase de creación o integración del proceso de publicación de software y conlleva un componente de automatización (p. ej., CI o servicio de versiones) y un componente cultural (p. ej., aprender a integrar con frecuencia). Los objetivos clave de la integración continua consisten en encontrar y arreglar errores con mayor rapidez, mejorar la calidad del software y reducir el tiempo que se tarda en validar y publicar nuevas actualizaciones de software.
[9]

8 CONCLUSIÓN

Una de las grandes metas de la ingeniería es poder facilitar el trabajo continuo a través de los años, la implementación de los pipelines es un beneficio para la institución educativa que quiera implementarla ya que este trabajo tedioso se vuelve fácil de realizar con la debida experiencia y su buen uso. Al conocer las herramientas que se tienen es posible llegar a tener un rendimiento eficiente en todos los datos que maneja la educación en general, esto es un proceso de retroalimentación rápida que los usuarios pueden utilizar en el software.

9 Bibliografia:

1. Soni, Mitesh. (2017). Implementing DevOps with Microsoft Azure. Recuperado el 18 de noviembre del 2020, de https://eds-a-ebscohost-com. bibliotecavirtual.udla.edu.ec/eds/detail/detail? vid=0sid=ee774f02-34b9-4307-ac72-844024ba82a2 l2ZQ

2. Santacroce, Ferdinando. (2017). Git Essentials - Second Edition. Recuperado el 18 de noviembre del 2020, de https://eds-b-ebscohost-com.bibliote cavirtual. udla.edu.ec/eds/detail/detail?vid=0sid=6085e8bb -fbe1-40d4-bcf0-c70c1de6d6a2Mmc2l0ZT1lZHMtbGl2ZQ

3. GL Group. (2019). ¿Qué es el pipeline de la integración y entrega continua?. Recuperado el 18 de noviembre de 2020 de https://www.gylgroup. com/novedades/pipeline-integracion-continua

4. Iñiguez, Luis. (2017). Arquitectura tecnológica para la entrega continua de software con despliegue en contenedores. Recuperado el 1 de febrero del

2021, de http://dspace.ucuenca.edu.ec/bitstream/123456789/28529/1/Trabajo

5. Beck, K. (February de 2001). Agile Manifest Principles. Obtenido de http://agilemanifesto.org/iso/es/principles.html

6. Shahin, M., Babar, A., Zhu, L. (2017). Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices. IEEE Software.

7. Guijarro Olivares, J. Caparrós Ramírez, J. y Cubero Luque, L. (2019). DevOps y seguridad cloud. Editorial UOC. https://elibro.net/es/ereader/udla/128889?page=27

8. Ståhl, D., Bosch, J. (2014). Modeling continuous integration practice differences in industry software development. The Journal of Systems and Software.

9. ¿Qué es la integración continua? Recuperado el 18 de noviembre del 2020, de https://aws.amazon.com/es/devops/continuous-integration/

10. Sanz San José, Óliver Luis. (2018). Biblioteca Python para el apoyo al desarrollo de pipelines de procesamiento de datos con Spark . http://uvadoc.uva.es/handle/10324/33254