**GUÍA PARA DESARROLLAR EL TEXTO GUÍA**

PROGRAMA:

PROCESOS AGILES DE SOFTWARE

(Texto Guía)

DEPARTAMENTO OFERENTE:

INGENIERÍA

AUTOR(A):

JORGE ALEJANDRO AGUIRRE GUTIERREZ

FORMACIÓN:

ESPECIALIZACION DE INGENIERÍA DE SOFTWARE CON ÉNFASIS EN PRUEBAS

Manizales, 2024

**Tabla de Contenidos**

**UNIDAD 2: INTEGRACION Y ENTREGA CONTINUA.**

1. ¿Qué es la integración continua?
2. ¿Qué es la entrega continua?
3. Beneficios del CI/CD
4. El proceso del CI
5. Herramientas del CI
6. Practicas recomendadas del CI
7. El proceso del CD
8. Estrategias en el CD
9. Herramientas del CD
10. Practicas recomendadas del CD
11. ¿Qué es DEVOPS?
12. Principios del DEVOPS
13. Practicas del DEVOPS
14. Herramientas del DevOps
15. Concepto del Build.
16. Herramienta del pipeline.
17. Compilación y configuración de Build para despliegue
18. Configuración para ejecución de pruebas automatizadas

***Antes de iniciar el texto guía, por favor lea con atención estas recomendaciones generales para el escrito:***

*El texto guía es un escrito académico elaborado por el autor del curso y tiene como principal función “presentar de forma didáctica el conocimiento de las distintas disciplinas…Por su diseño, son textos mixtos que reúnen información textual con otra de tipo visual (uso de imágenes, gráficas, tablas, etc.). También contienen muchos tipos de ayudas didácticas, especialmente seleccionadas y diseñadas, con el propósito de favorecer el aprendizaje (Díaz & Hernández, 2010, p. 158), tales como: videos, simuladores, animaciones, cuestionarios de verificación de lectura, ejercicios interactivos (opción múltiple, falso y verdadero, emparejamiento, clasificar conceptos) o juegos interactivos de Educaplay (*[*Ver tipo de actividades*](https://es.educaplay.com/tipos-de-actividades/)*) con recursos multimedia (video, audio, imagen, texto)*

*Siguiendo a Díaz y Hernández (2010), los textos considerados accesibles se caracterizan por:*

* *Tienen un arreglo estructurado y sistemático de las ideas.*
* *Poseen un buen nivel de coherencia de las ideas expuestas*
* *Son concisos y contienen poca información distractora o irrelevante*
* *Toman en cuenta el conocimiento previo del lector*
* *Presentan al lector la estructura del texto de forma clara (con señalizaciones) para beneficiar su comprensión profunda*

*Por favor tenga en cuenta las siguientes recomendaciones:*

* *Utilizar Verdana tamaño 12, interlineado sencillo.*
* *Las fuentes consultadas para desarrollar el contenido no deben ser de más de cinco años de antigüedad. Si se requiere utilizar fuentes de más años, se debe justificar en las referencias al final de la unidad.*
* *Todas las gráficas, ilustraciones o figuras, deben estar enumeradas con la herramienta de Word REFERENCIAS: INSERTAR TÍTULO. Debe registrarse la FUENTE: Autor, fecha, o sitio web. Cuando la gráfica es de autoría del docente, se debe especificar al final del título de la gráfica: FUENTE: ELABORACIÓN PROPIA.*
* *En el desarrollo del contenido, cuando se trate de ideas fundamentadas en otros autores, se debe hacer la cita correspondiente. Del mismo modo diferenciar claramente las opiniones del docente autor del módulo.*
* *Las citas deben referenciarse con Normas VANCOUVER (PARA CIENCIAS DE LA SALUD) – APA (CIENCIAS SOCIALES E INGENIERÍAS)*

**TÍTULO DE LA UNIDAD:**

**INTEGRACION Y ENTREGA CONTINUA**

**INTRODUCCIÓN A LA UNIDAD**

**Indicador de desempeño**

Al finalizar el estudio de esta unidad y desarrollar las actividades propuestas, el estudiante deberá estar en capacidad de:

* Identificar los conceptos de integración, entrega continua y pruebas automatizadas.
* Aplicar la metodología de integración y entrega continua en una plataforma de DevOps, y validar la ejecución de pruebas automatizadas que mejoran la calidad del software.

**-----------**

**Desarrollar cada uno de los temas propuestos**

***INDICACIONES:***

*Presentación del contenido utilizando estrategias que cumplan con las siguientes funciones:*

* *Detección de la información principal;*
* *Conceptualización de contenidos;*
* *Delimitación de la organización, estructura interna e interrelaciones entre dichos contenidos;*
* *y mantenimiento de la atención y motivación.*

*A lo largo del escrito haga uso de estrategias tales como: ilustraciones, esquemas, gráficas o resúmenes (al iniciar la unidad, al finalizar la unidad, o al finalizar un tema), preguntas intercaladas, analogías, pistas tipográficas (resaltar ideas centrales, conceptos, mapas conceptuales), ejemplificaciones para aclarar conceptos o mostrar aplicaciones, ejercicios.*

*Utilice recursos multimedia como material complementario: audios, videos, animaciones, simuladores, imágenes, entre otros.*

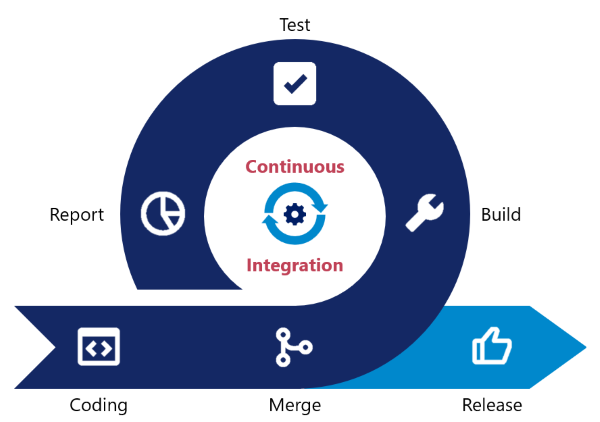
*Recuerde hacer las citas correspondientes con normas VANCOUVER O APA.*

*Recuerde proponer ejercicios interactivos a lo largo del texto para reforzar conocimientos o de verificación de comprensión de la lectura.*

*Recuerde titular y enumerar cada figura, tabla, imagen, gráfica, ilustración y registrar la fuente. Especificar cuando se trate de una gráfica de su propia autoría.*

**TEMA 1: ¿QUÉ ES LA INTEGRACIÓN CONTINUA?**

La Integración Continua (CI) es una práctica de desarrollo de software en la que los desarrolladores integran sus cambios de código en un repositorio central varias veces al día. Cada vez que se integra código, se ejecuta una serie de pruebas automatizadas para detectar errores. Esto ayuda a garantizar que el código siempre esté en un estado que se pueda implementar y que los errores se detecten temprano en el proceso de desarrollo.



**TEMA 2: ¿QUÉ ES LA ENTREGA CONTINUA?**

La Entrega Continua (CD) es una práctica de desarrollo de software en la que el código que pasa las pruebas automatizadas se implementa automáticamente en un entorno de producción. Esto ayuda a garantizar que el software de alta calidad se entregue a los clientes de forma rápida y confiable.

Enlace de profundización sobre la entrega continua: <https://www.youtube.com/watch?v=fiJod8T4P1c>

**TEMA 3: BENEFICIOS DEL CI/CD**

* Mayor velocidad de lanzamiento: CI/CD puede ayudar a las empresas a lanzar software con más frecuencia, lo que les permite responder más rápido a las necesidades del mercado y de los clientes.
* Mayor calidad del software: CI/CD puede ayudar a mejorar la calidad del software al automatizar las pruebas y detectar errores en las primeras etapas del proceso de desarrollo.
* Mayor confiabilidad del software: CI/CD puede ayudar a mejorar la confiabilidad del software al garantizar que todas las pruebas se ejecuten y que el software se implemente de forma coherente en todos los entornos.
* Mayor satisfacción del cliente: CI/CD puede ayudar a mejorar la satisfacción del cliente al proporcionar software de alta calidad de forma rápida y confiable.

**TEMA 4: EL PROCESO DEL CI**

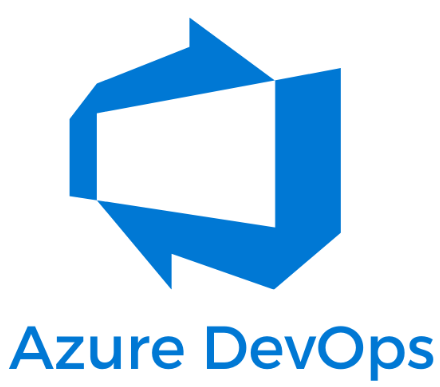
El proceso de CI típico incluye los siguientes pasos:

1. Un desarrollador realiza cambios en el código.
2. El desarrollador integra sus cambios en un repositorio central.
3. Se ejecuta una serie de pruebas automatizadas.
4. Si las pruebas fallan, el desarrollador debe corregir los errores e intentar integrar nuevamente sus cambios.
5. Si las pruebas pasan, el código está listo para implementarse.

**TEMA 5: HERRAMIENTAS DEL CI**

Hay muchas herramientas de CI disponibles, cada una con sus propias características y ventajas. Algunas de las herramientas de CI más populares incluyen:

* Jenkins: <https://www.jenkins.io/> 
* GitLab CI/CD: <https://docs.gitlab.com/ee/ci/> 
* Travis CI: <https://www.travis-ci.com/> 
* CircleCI: <https://circleci.com/> 
* Azure DevOps: <https://azure.microsoft.com/es-es/products/devops>



**TEMA 6: PRACTICAS RECOMENDADAS DEL CI**

* Comience con un proceso de CI simple y vaya aumentando su complejidad a medida que su equipo lo familiarice.
* Utilice una herramienta de CI que se integre con su sistema de control de versiones y sus herramientas de prueba.
* Ejecute pruebas automatizadas en cada integración de código.
* Automatice la implementación del código a un entorno de prueba después de que las pruebas pasen.
* Monitoree el proceso de CI y busque cuellos de botella y errores.

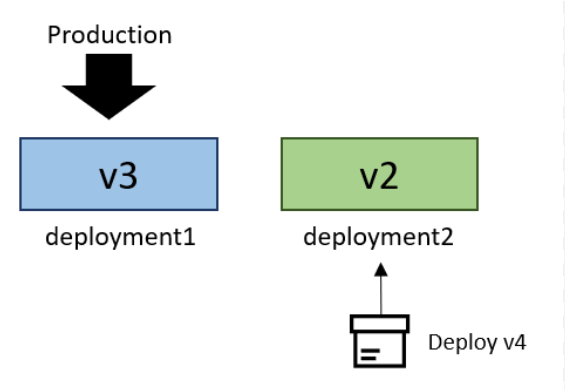
**TEMA 7: PROCESO DEL CD**

El proceso de CD típico incluye los siguientes pasos:

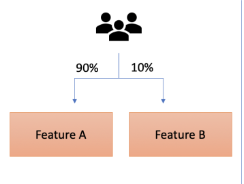
1. El código pasa las pruebas automatizadas.
2. El código se implementa automáticamente en un entorno de producción.
3. Se ejecutan pruebas adicionales en el entorno de producción.
4. Si las pruebas fallan, la implementación se revierte.
5. Si las pruebas pasan, el software está disponible para los clientes.

**TEMA 8: ESTATREGIAS DEL CD**

Implementación azul-verde: En la implementación azul-verde, se crean dos entornos de producción idénticos. El nuevo código se implementa primero en el entorno "verde". Si las pruebas pasan, el tráfico se dirige al entorno "verde" y el entorno "azul" se retira.



Implementación Canary: En la implementación de canarias, el nuevo código se implementa gradualmente a un pequeño subconjunto de usuarios. Si la implementación tiene éxito, el código se implementa gradualmente a más usuarios hasta que esté disponible para todos.



**TEMA 9: HERRAMIENTAS DEL CD**

Hay muchas herramientas de CD disponibles, cada una con sus propias características y ventajas. Algunas de las herramientas de CD más populares incluyen:

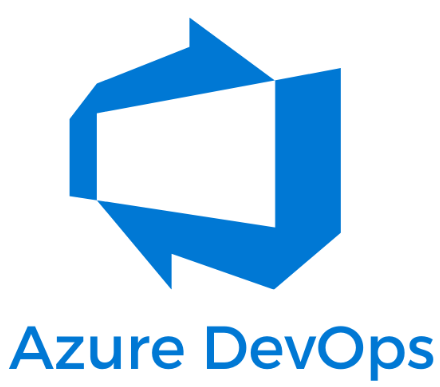
* Spinnaker: <https://spinnaker.io/>



* Ansible: <https://www.ansible.com/> 
* Jenkins: <https://kubernetes.io/>



* Azure DevOps: <https://azure.microsoft.com/es-es/products/devops>

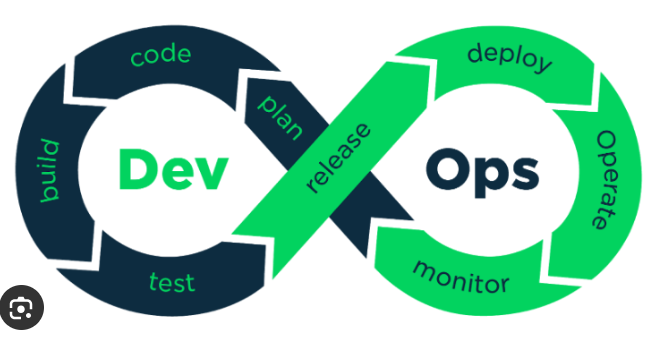


**TEMA 10: PRACTICAS RECOMENDADAS DEL CD**

* Comience con una estrategia de CD simple y vaya aumentando su complejidad a medida que su equipo lo familiarice.
* Automatice la implementación del código a un entorno de prueba después de que las pruebas pasen.
* Ejecute pruebas adicionales en el entorno de producción.
* Monitoree el proceso de CD y busque errores.
* Tenga un plan de reversión en caso de que una implementación falle.

**TEMA 11: ¿QUÉ ES DEVOPS?**

DevOps es una cultura y un conjunto de prácticas que enfatizan la colaboración entre los equipos de desarrollo y operaciones. El objetivo de DevOps es permitir que las empresas lancen software de alta calidad de forma rápida y confiable.



**TEMA 12: PRINCIPIOS DEL DEVOPS**

Los principios clave de DevOps incluyen:

* Cultura de colaboración: Los equipos de desarrollo y operaciones deben trabajar juntos en estrecha colaboración.
* Automatización: Los procesos de desarrollo y operaciones deben automatizarse tanto como sea posible.
* Entrega continua: El código debe implementarse en producción de forma regular.
* Monitoreo y retroalimentación: Los sistemas de producción deben monitorearse y la retroalimentación debe usarse para mejorar los procesos de desarrollo y operaciones.

**TEMA 13: PRACTICAS DEL DEVOPS**

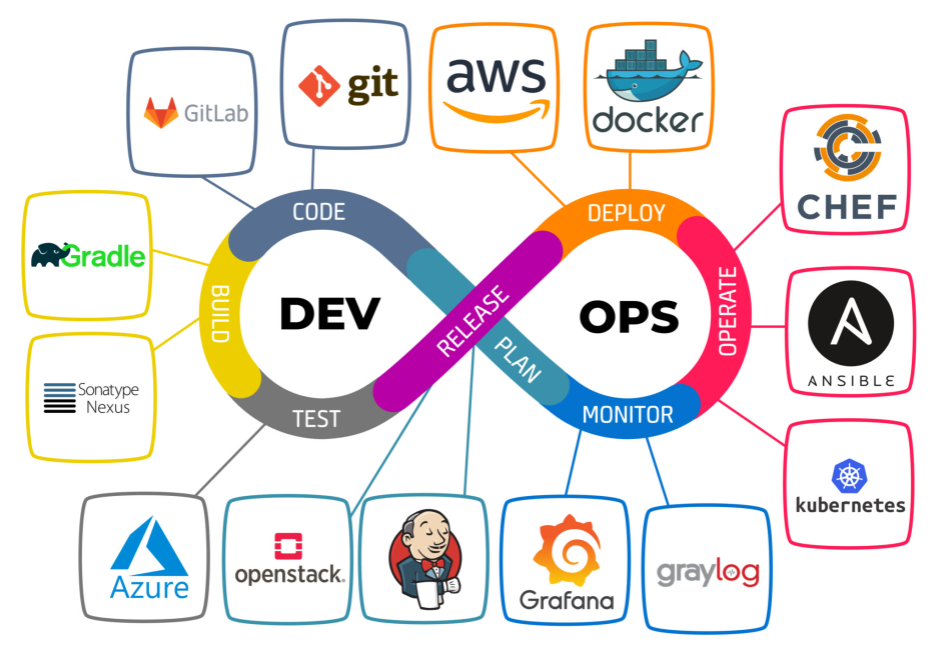
Algunas de las prácticas comunes de DevOps incluyen:

* Integración continua (CI)
* Entrega continua (CD)
* Infraestructura como código (IaC)
* Estándares de seguridad
* Monitoreo y registro
* Cultura de mejora continua

**TEMA 14: HERRAMIENTAS DE DEVOPS**

Hay muchas herramientas DevOps disponibles, cada una con sus propias características y ventajas. Algunas de las herramientas DevOps más populares incluyen:

* Herramientas de CI/CD: Jenkins, GitLab CI/CD, Travis CI, CircleCI, Azure DevOps
* Herramientas de IaC: Ansible, Chef, Puppet, Terraform
* Herramientas de monitoreo: Prometheus, Grafana, Nagios
* Herramientas de registro: ELK Stack, Splunk



**TEMAS 15: CONCEPTO DE BUILD**

Es el proceso de transformar el código fuente de un programa en un formato ejecutable que puede ser utilizado por una computadora. Este proceso implica varias etapas, como:

* Preprocesamiento: El código fuente se analiza y se realizan tareas como la expansión de macros y la inclusión de archivos.
* Compilación: El código fuente se traduce en lenguaje de máquina específico para el procesador y el sistema operativo.
* Ensamblaje: Los archivos de código objeto resultantes de la compilación se unen para crear un ejecutable.
* Enlazado: El ejecutable se vincula con las bibliotecas y recursos necesarios para su ejecución.

**TEMA 16: HERRAMIENTA DEL PIPELINE**

Un pipeline se refiere a una serie automatizada de tareas que se ejecutan para construir, probar y desplegar software.

Los pipelines son un componente clave de las metodologías de desarrollo ágiles y DevOps, ya que permiten automatizar tareas repetitivas y optimizar el flujo de trabajo.

Un pipeline típico puede incluir las siguientes etapas:

* Extracción de código: Obtener el código fuente del software de un repositorio.
* Compilación: Transformar el código fuente en un formato ejecutable.
* Pruebas: Ejecutar pruebas automatizadas para verificar el correcto funcionamiento del software.
* Despliegue: Instalar el software en un entorno de producción o prueba.
* Los pipelines se pueden implementar utilizando herramientas como Jenkins, GitLab CI/CD o Azure DevOps Pipelines.



**TEMA 17: CONFIGURACION DE PIPELINE PARA DESPLIEGUE**

Configurar las etapas y tareas del pipeline:

Un pipeline de despliegue se compone de una o más etapas, cada una con un conjunto de tareas específicas.

Las etapas comunes incluyen:

* Desarrollo: Compilar y preparar el código para su despliegue.
* Pruebas: Ejecutar pruebas automatizadas para verificar el correcto funcionamiento del software.
* Despliegue: Implementar el software en un entorno de producción o prueba.
* Liberación: Publicar la versión del software para que esté disponible para los usuarios.
* Cada etapa consta de tareas que realizan acciones específicas, como compilar el código, ejecutar pruebas, o actualizar entornos.
* Azure DevOps ofrece una amplia gama de tareas predefinidas y la posibilidad de crear tareas personalizadas.

Configurar desencadenadores y variables:

Defina los desencadenadores que iniciarán la ejecución del pipeline.

Los desencadenadores comunes incluyen:

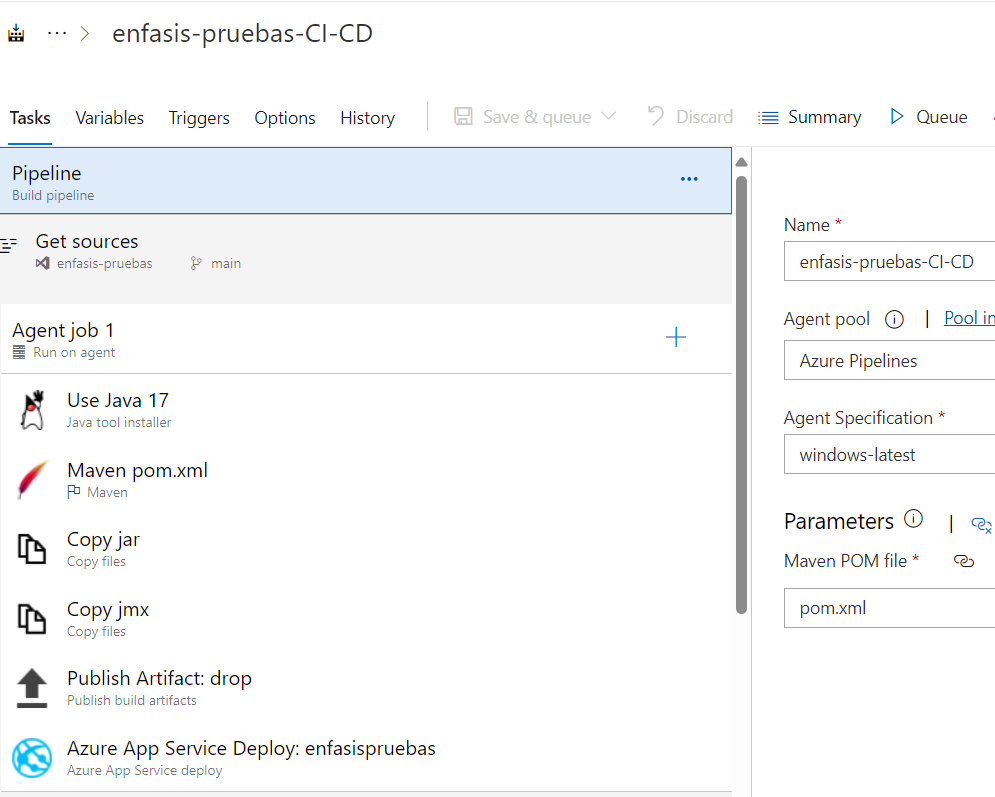
* Push de código al repositorio.
* Creación o actualización de una rama específica.
* Liberación manual por un usuario.
* Establezca variables para almacenar valores que se utilizan en las tareas del pipeline.
* Las variables pueden ser estáticas o dinámicas, dependiendo de la fuente de los datos.

Revisar y ejecutar el pipeline:

* Revise la configuración del pipeline para asegurarse de que esté correcto.
* Ejecute el pipeline manualmente para probar su funcionamiento.
* Monitorice el progreso del pipeline en la interfaz de Azure DevOps.

Implementar el pipeline:

* Una vez que el pipeline esté probado y validado, configure su implementación automática.
* Esto permitirá que el pipeline se ejecute automáticamente cada vez que se cumplan los criterios de desencadenamiento.
* Configure notificaciones para recibir alertas sobre el estado del pipeline.



CONFIGURACION DE PIPELINE PARA DESARROLLO DE PROYECTOS

|  |  |
| --- | --- |
| 1. Inicialmente se nos muestra una pantalla de configuracion del pipeline en donde tenemos que definir que maquina utilizar, para Azure DevOps las maquinas son Agentes, para nuestro caso seleccionamos el agente de Windows. |  |
| 1. En este paso nos pide seleccionar el repositorio con el cual se hará el despliegue de la aplicación. |  |
| 1. Un pipeline trabaja mediante etapas llamadas Jobs y nos pide si requerimos trabajar con otro agente en la etapa para nuestro caso no seleccionamos agente, se trabajará con el primero que se selecciono | ue |
| 1. La primera tarea dentro de la maquina será la instalación de un JDK de Java |  |
| 1. La segunda tarea es la compilación del pom de nuestro proyecto |  |
| 1. Posterior a la compilación es necesario copiar el archivo .jar generado, para su posterior publicación en el pipeline. |  |
| 1. Publicación del artefacto en Azure Pipelines para posterior despliegue |  |
| 1. Tarea para el despliegue del artefacto java en un app service de Azure. |  |

**TEMA 18: CONFIGURACION PARA EJECUCION DE PRUEBAS AUTOMATIZADAS**

En el editor de pipelines, agregue una nueva etapa.

Asigne un nombre a la etapa, como "Ejecutar JMeter".

En la lista de tareas disponibles, busque y seleccione la tarea "Instalar JMeter".

Configurar la tarea JMeter Load Test:

Proporcione los detalles de su prueba de carga JMeter, incluyendo:

* Nombre de la prueba: Un nombre descriptivo para su prueba de carga.
* Archivo de prueba: La ruta al archivo.jmx que contiene su script JMeter.
* Plan de ejecución: Agregue los comandos necesarios para ejecutar JMeter y las pruebas.

Ubicación del Load Testing:

* Seleccione la ubicación de Load Testing donde desea ejecutar la prueba.
* Agente de prueba: Elija el tipo de agente de prueba que desea utilizar.
* Parámetros adicionales: Proporcione cualquier parámetro adicional que requiera su prueba de carga.

Agregar etapas adicionales:

Puede agregar etapas adicionales antes o después de la etapa de ejecución de JMeter, como:

* Etapa de compilación: Para compilar su código antes de ejecutar las pruebas.
* Etapa de publicación de resultados: Para publicar los resultados de las pruebas hacer respectivo análisis.

|  |  |
| --- | --- |
| 1. Para la ejecución de las pruebas automatizadas se crea un nuevo Job. |  |
| 1. La primera tarea es la copa del archivo .jmx de JMeter. |  |
| 1. Luego se hace la instalación de JMeter en la máquina de Azure. |  |
| 1. Se crea una tarea de línea de comandos en donde se escribe el comando según la imagen para la ejecución de JMeter |  |
| 1. Se crea la tarea para la publicación del reporte de JMeter en donde se podrá hacer la validación de las pruebas. |  |

**CIERRE DE LA UNIDAD**

La Integración y Entrega Continua (CI/CD) es una metodología de desarrollo de software que automatiza los procesos de integración, pruebas y despliegue de código. Su objetivo principal es acortar el ciclo de desarrollo y entregar software de alta calidad a los usuarios de manera más rápida y confiable.

En resumen, CI/CD es una práctica esencial para el desarrollo de software moderno que permite a las empresas entregar software de manera más rápida, confiable y segura.

***INDICACIONES:***

*Una vez terminado el desarrollo del contenido, procure formar una visión sintética, integradora e incluso crítica de la unidad, a través de estrategias como: resumen, mapa conceptual, preguntas o conclusiones generales de los temas presentados en el escrito.*

*Proponga material o lecturas complementarias si lo considera necesario.*

**REFERENCIAS BIBLIOGRÁFICAS**

***INDICACIONES:***

*Registre la documentación (NORMAS APA O VANCOUVER) consultada para desarrollar el contenido.*

*Utilice bibliografía con una antigüedad no superior a los cinco años.*