DevOps Bootcamp

sábado, 28 de enero de 2023 14:28

28/01/2023

DevOps: Desarrollo + Operaciones
Establecer una comunicación clara y constante entre los dos lados.

Que es DevOps?
Es una cultura de trabajo la cual aboga por las buenas prácticas para la buena calidad de software en el proceso de desarrollo.

Dia a día del DevOps:
- Orquestar los proyectos del proceso para que vayan encaminados con la meta del proyecto.
- Aplicar buenas prácticas en el proyecto.

SCRUM y DevOps?
SCRUM es una metodología o marco de trabajo para el desarrollo de proyectos ágil.
DevOps es una cultura, acoge las metodologías agiles.

Relación DevOps con arquitectura de Software:
La arquitectura de software existe para implementar los requisitos no funcionales de un sistema.
DevOps nos pide estar pendientes de los mismos.

DevOps es el observador que trata de encontrar los requisitos no funcionales y exigirle a la arquitectura que los implemente.

4/02/2023
Siempre que se esté peleando con algo muy en específico lo más probable es que ya hay un patrón de solución.

Una conexión saludable / buena debe tener una latencia de 200ms como maximo.

El desarrollo de software es un problema complejo no complicado

Complicado: Se resuelve con recursos

Complejo: Cada persona que le agrego a ese problema lo vuelve más complejo

Desarrollo de software: Algo que hay que construir poco a poco

La vida o el ciclo del desarrollo de software:
- Analisis
- Diseño
- Implementacion
- Verficacion
- Mantenimiento

Las personas dividen la comunicacion en silos de comunicacion segun
las diferentes etapas del proceso (ley de cockwell).

Se llegan a metodologias agiles.

El desarrollo de software se hace en base a la comunicacion y el feedback continuo.

El proceso no es lo importante lo importante es la comunicacion y el trabajo en equipo

Entregar software funcional en lugar de documentacion exhaustiva.

DevOps: Entrega de software continua garantizando la calidad y la estabilidad del mismo.

Beneficios:
- Aceleracion del proceso de desarrollo (tiempo)
- Mejora en la comunicacion y observabilidad
- Desarrollo de entornos mas estables.

DevOps:
1. Valores: Entrega continua de buena calidad, cumplir con las expectativas del cliente.
2. Practicas: Integración Continua
Herramientas: Azure DevOps / Jenkins

11/02/2023

CI: Construir y probar.
CD: Automatizacion de los despliegues; produccion pasa a ser mas manual en la mayoria de casos.

Pipeline: Automatizacion de un proceso.

Saber reaccionar ante errores y que la aplicacion no explote.

DevOps es una cultura de soporte para el ciclo del desarrollo de software, la calidad, la mejora y la eficiencia del mismo.

Software como producto y como valor mas grande, no como medio para un fin dado.

Un producto de software dura en media de 5 a 7 años.

Uno no resuelve la complejidad una la mueve a otra parte.

Siempre empezar un sistema como un monolito y al descrubrir los dominios del negocio transformar a microservicios.

CI: Proceso de unir todo el codigo, validar que fuciona, liberarlo (code base).

- Checkout (descarga de codigo):
Descargar el codigo de un servidor a local.
- Build (ensamblaje):
Tomar el codigo, armarlo, resolver dependencias (mi proyecto depende de otros proyectos) y transformarlo para la ejecucion.
- Definir las dependencias
- Existencia de un servidor donde se guarde las librerias con las cuales estoy trabajando

Para leer:
- Teoría de Janet Gregory sobre pruebas holísticas.

18/02/2023

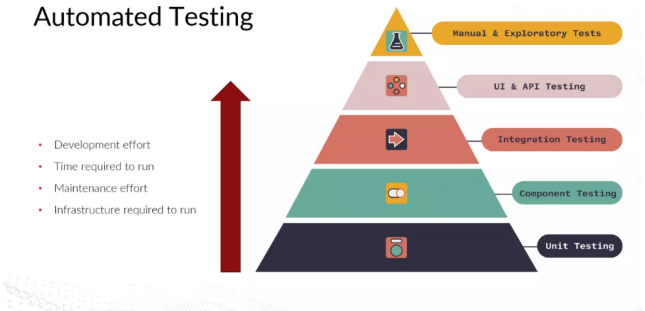
Shift left: Entre más rápido se encuentre el error menos caro es.

La CI se basa en Shift Left como buena práctica para el código de calidad.

Pruebas automáticas: Software para probar otro software; dejarle las tareas repetitivas a las maquinas.

Repetibles: Siempre se siguen los mismos pasos.

Equipo de pruebas ya que al desarrollar hay un sesgo según lo que se entendió del requerimiento.
Los equipos de QA tiene personas que hacen pruebas automáticas y otros que hacen pruebas manuales.



Unit test: Yo como desarrollador estoy comprobando que el software está funcionando de la manera como creo que debería de funcionar.

Pruebas de componentes o integración: Prueban la interacción entre 2 o más componentes del sistema y requieren recursos de infraestructura (correr en el servidor o entorno parecido a produccion).

A medida de que se necesita más infraestructura parecida a producción pasa de componentes a integración.

Pruebas funcionales: Probar una funcionalidad de mi aplicación. La información recorre todo un camino de ida y vuelta para su ejecución.

Pruebas de estrés: Llevar el sistema a su máxima capacidad. Hacer pruebas funcionales de componentes un numero 'n' de veces.

Cobertura de las pruebas unitarias: Que % de la aplicación esta siendo probado con pruebas unitarias (mas que 80%)

Librerias para medir la cobertura de codigo.

Siempre depender de librerias altamente usadas en la industria.

Análisis estático de código: Analizar posibles problemas (malas prácticas, code smells, código que retorna múltiples valores). Hacer uno de estos analisis antes de hacer el merge con la rama principal.

Empaquetado (último paso): Generar el artefacto, paquete que se entrega al servidor para desplegar la app. Generar imágenes de Docker.