Continuous Deployment

Por Eric Ries

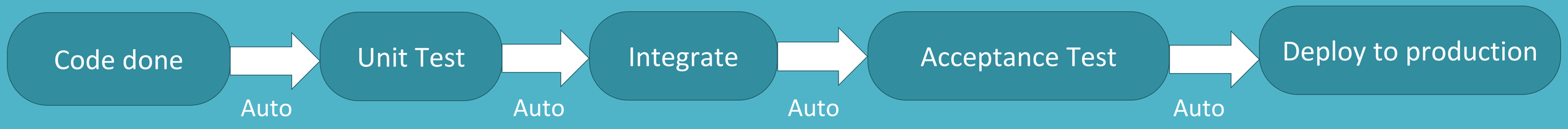
Carranza Alexis, Banda José, Leguizamón Tomassi Nicolás, Pérsico Santiago, Urquiza Mayra.
Universidad Tecnológica Nacional – Facultad Regional Córdoba – Cátedra de Ingeniería de Software.

Introducción

El tema a tratar en el presente póster es un enfoque de la ingeniería de software denominado **Continuous Deployment**. Se analizarán los beneficios que aporta su utilización en un proyecto y los pasos para su implementación.

¿En qué consiste?

Continuous Deployment o despliegue continuo es un proceso mediante el cual cada cambio que pasa las pruebas automatizadas en el ambiente de test se pone en producción sin ninguna intervención manual.



¿Por qué y cómo realizar despliegue continuo?

El despliegue continuo aplicado de manera correcta permite reducir el tiempo del ciclo al desplegar mayor cantidad de código en menor tiempo, mejora la calidad del software, reduce costos y evita regresiones, interrupciones o daños a las métricas clave del negocio aplicando un conjunto riguroso de estándares.
Para lograr una implementación continua a lo largo del tiempo se deben seguir cinco pasos.

Paso 1: Continuous integration server

Invertir en un servidor de integración continua donde todas las pruebas automatizadas se puedan ejecutar y monitorear por cada commit realizado.
Es el **sostén** del despliegue continuo.
Se agregan pruebas al servidor cada vez que se encuentre y solucione un error.

Paso 2: Source Control commit check

Consiste en un script de confirmación que se ejecuta con cada commit realizado antes de que el mismo sea aceptado, de manera que si alguna de las pruebas falla, el script de confirmación prohibe que se agregue un nuevo código al repositorio hasta que el problema sea solucionado.

Paso 3: Simple Deployment script

Se trata de un script que implementa de manera incremental el software máquina por máquina y monitorea el estado del clúster y de la empresa durante todo el proceso de manera que se pueda hacer una reversión rápida si algo sale mal.

Paso 4: Real-time alerting

Consiste en una plataforma de monitoreo que permita informar cuando algo sale mal de manera que una persona pueda participar en la depuración de errores.

Paso 5: Root cause analysis (five whys)

Recibe su nombre debido a que consiste en preguntarse recursivamente por qué y el cinco es el número de iteraciones típicamente requeridas para resolver el problema.
La idea es llegar a la causa raíz de cualquier falla inesperada que pueda ocurrir en el sistema.
Este análisis permite el despliegue continuo si cada vez que se realiza, se hace una inversión proporcional en prevención.

Conclusión

La implementación de este enfoque puede parecer complicada y no muy beneficiosa, ya que puede requerir grandes costos iniciales. Sin embargo puede mejorar en gran medida la calidad del software y reducir considerablemente la duración de los ciclos asegurando que el software liberado es confiable, además, asegura una mínima interrupción maximizando el beneficio a los clientes quienes reciben los productos solicitados en tiempos más acotados. De manera que a lo largo del tiempo pueden verse los beneficios tanto para el equipo de desarrollo como para los clientes y todos los interesados.

Bibliografía

Paper Continuous deployment in 5 easy steps: http://radar.oreilly.com/2009/03/continuous-deployment-5-eas.html
https://engineering.imvu.com/2010/04/09/imvus-approach-to-integrating-quality-assurance-with-continuous-deployment/
https://electric-cloud.com/resources/continuous-delivery-101/continuous-deployment/
http://www.javiergarzas.com/2016/01/devops-continuous-delivery-continuous-deployment-integracion-continua-aclarando-terminos.html