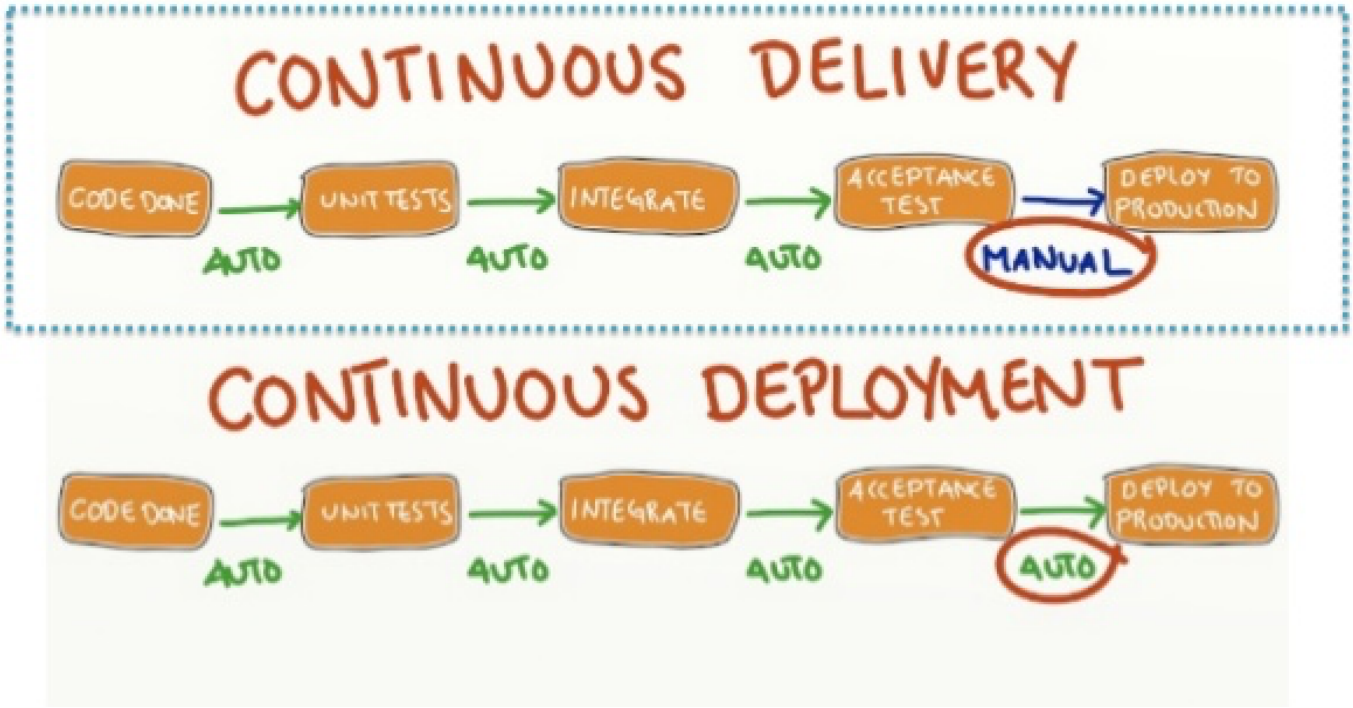
La Entrega Continua es una metodología para el despliegue de aplicaciones en producción que surge de las mejores prácticas de gestión de configuración, pruebas automatizadas, integración continua, gestión de datos y gestión de versiones. El objetivo es automatizar al máximo el despliegue de un sistema en un ambiente productivo, minimizando los riesgos asociados a dicho proceso, y facilitando la detección de regresiones.

Tenemos que distinguir entre Entrega Continua (*Continuous Delivery*) y Despliegue Continuo (*Continuous Deployment*). La entrega continua se refiere a la práctica de tener automatizado todos los procesos de validación del software y su despliegue en producción, pero el proceso de actualizar el entorno de producción lo inicia el personal de operaciones. Por contra, nos referimos a despliegue continuo cuando el proceso de actualizar el entorno de producción es también automático y cada *commit* en nuestro repositorio de código va directamente a producción después de pasar todo el proceso de integración continua. La diferencia se muestra en la siguiente figura:



Desde un punto de vista tecnológico, ambas prácticas son equivalentes. La decisión de tomar un enfoque de entrega continua o despliegue continuo suele responder a decisiones tácticas de la empresa. Lo normal es que empresas de pequeño y mediano tamaño opten por una metodología de entrega continua para controlar cuando lanzan nuevas funcionalidades, mientras que grandes empresas como Amazon o Google suelen optar por el despliegue continuo. En esta lección vamos a centrarnos en cómo hacer entrega continua con Docker.

La entrega continua es la evolución natural de tener unos procesos de integración continua robustos y completos. La práctica de la entrega continua está condenada al fracaso si no disponemos de una buena cobertura en nuestras pruebas unitarias, de integración y de aceptación. También hay que destacar que todo proceso de entrega continua tiene que tener estas propiedades:

- Versionado de todos los componentes de nuestra aplicación. Todo nuestro código y nuestras configuraciones deben existir en un repositorio (por ejemplo tipo *git*) y cada *release* tiene que ser *tageada* para facilitar los procesos de identificación de errores y *rollback*.

- Nuestros *build* y *tests* deben mantenerse en tiempos de ejecución lo más bajos posibles. En caso contrario, un *hotfix* urgente a producción tardará demasiado en pasar por todo el proceso automatizado de integración y entrega continua y podemos tener la tentación de saltarnos los procesos automatizados de integración y entrega continua para adelantar la publicación del hotfix, con el riesgo de posibles regresiones. Además, el tener builds y tests rápidos acelerará todo el ciclo de desarrollo.

- Notificaciones: los desarrolladores y el personal de operaciones deben de recibir notificaciones cuando un build o una prueba haya fallado o haya tenido éxito. De igual manera, deben notificarse los despliegues a producción. Cuando una prueba o un *build* falla, la prioridad del equipo de desarrollo debe ser arreglarlo.