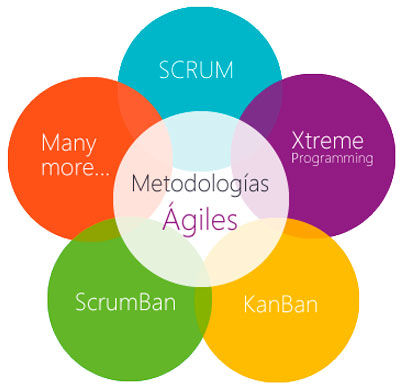
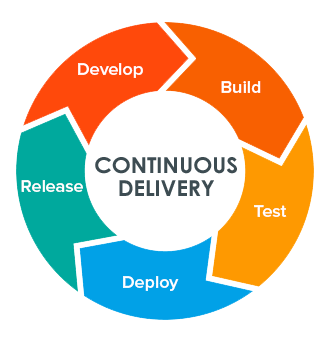
## <https://www.youtube.com/watch?v=s6nfuqMApOk&t=112s>

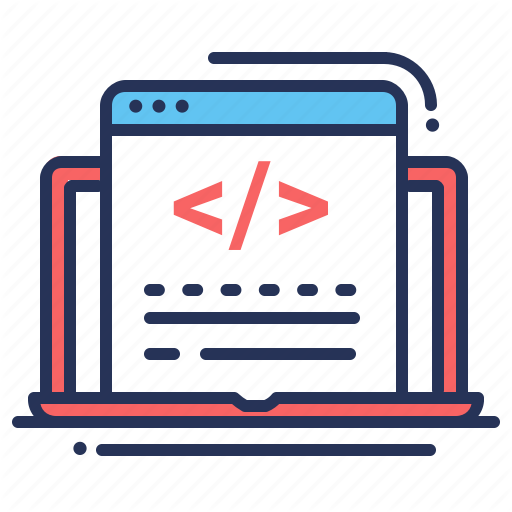
## 1- Introducción al tema (39 seg) 3 escenas

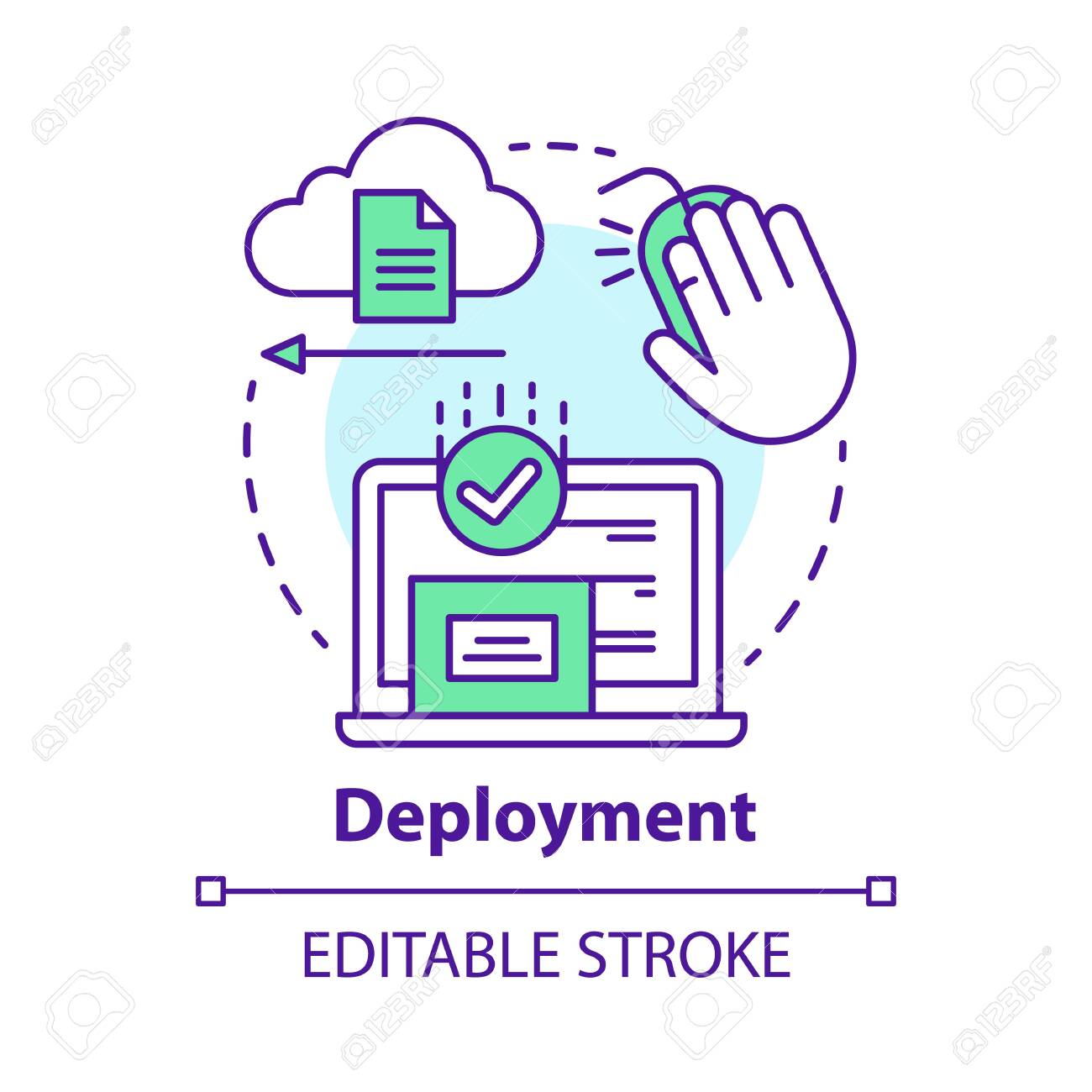
La entrega continua es una práctica de desarrollo de software, que aplica tanto a procesos definidos como a metodologías ágiles. 



Cuando se implementa de manera adecuada, los desarrolladores dispondrán siempre de features listas para producción que se han sometido a un proceso de pruebas estandarizado.



En pocas palabras, en la entrega continua cada pieza de código es construída, testeada y desplegada en un ambiente similar a producción para una evaluación rigurosa, dejando un producto listo y a la espera de ser desplegado manualmente.

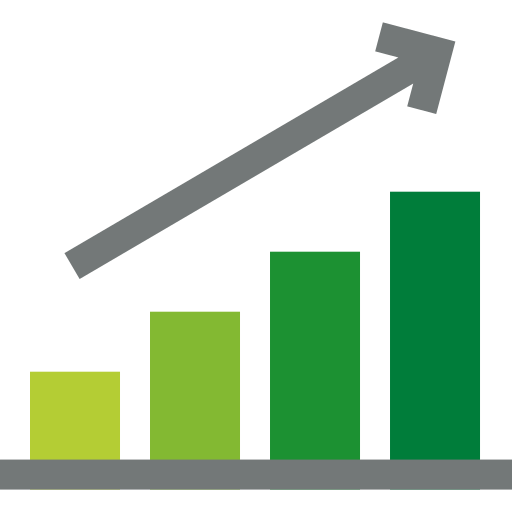


## 

## Desarrollo (29 segundos) (3 escenas)

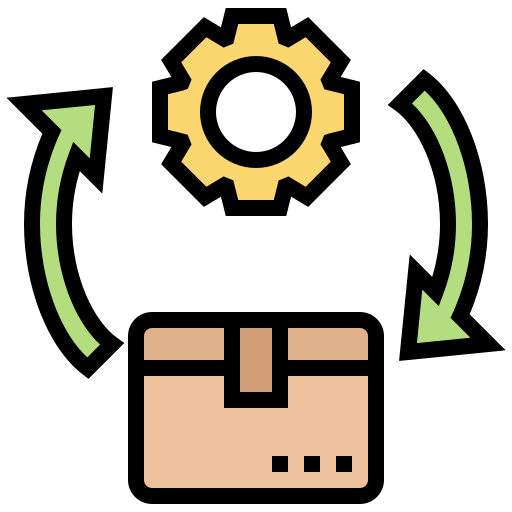
La entrega continua consiste en disponer frecuentemente de releases funcionales y que Evolucionan, mediante un proceso automatizado listos para ser lanzados a producción.

(grafico de titulo)



Con la entrega continua se reduce costo, tiempo, riesgo, e impacto en el ambiente de producción al construir piezas pequeñas de código con mejoras incrementales.

(Gráfico animado de costo tiempo y riesgo reduciéndose)

grafico de piezas de codigo

Por otra parte, aumenta la frecuencia de las entregas, favoreciendo al desarrollo ágil a través de versiones funcionales.



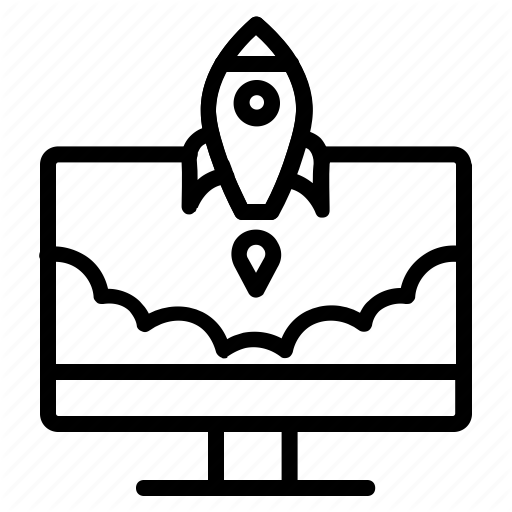
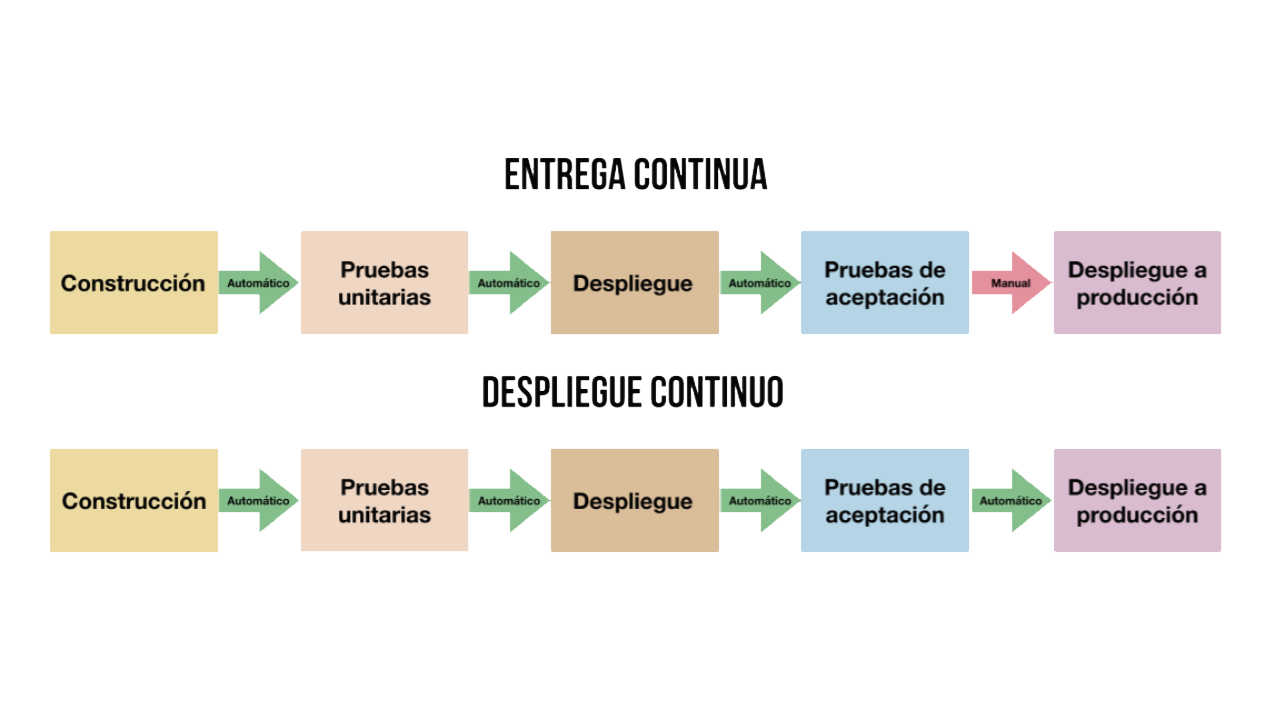
(gráfico aumentando las frecuencias de las entregas) si hago tiempo le meto algo mas

Relación con continuous integration 6

Entrega continua es una extensión de la integración continua para asegurarse de que los nuevos cambios podrán a estar rápidamente a disposición del cliente de una manera sustentable.

(grafico comparativo de los dos)

Esto quiere decir que por encima del testeo automático, también el lanzamiento será automático y se podrá desplegar la aplicación en cualquier momento tan solo con el click de un botón



Relación con continuous deployment 6

Despliegue continuo va un paso más allá. Esto significa que cada cambio que pasa todas las etapas del pipeline de producción se vuelva directamente disponible para los clientes sin intervención humana, por lo que sólo el testing automatizado podrá prevenir las fallas que un nuevo cambio podría introducir.

## 

### Beneficios

* Permite al equipo crear, probar y preparar los cambios a implementar en producción en forma automatizada, lo que implica una entrega de software más eficiente y rápida.
* Ayuda a reducir el número de bugs y errores que se entregan en las implementaciones a los clientes, ya que al automatizar se libera de tareas manuales a los desarrolladores y disminuyen los errores introducidos por falla humana.
* Al testear de manera continua, los errores se descubren y arreglan antes de que se conviertan en problemas graves.
* Los clientes reciben actualizaciones con gran rapidez y frecuencia, ya que siempre se cuenta con features listas para desplegar en producción.

Limitaciones¹

### Si bien el testing automatizado es una ventaja en este proceso, no es suficiente debido a que existen escenarios que necesitan ser testeados manualmente.

### En general, a quienes utilizan el software, no les agrada recibir actualizaciones demasiado frecuentes. Suelen acostumbrarse al uso de un sistema y los cambios pueden significar una amenaza en la experiencia de usuario.

### Al trabajar en distintos entornos de desarrollo, testeo y producción se producen diferencias que pueden provocar incidencias o issues.

### No resulta muy eficiente cuando el contexto demanda un testeo exhaustivo

## 

## Conclusión

Es una práctica ideal para implementar cuando es fundamental brindar rápida respuesta frente a cambios, siendo implementados incrementalmente para que impliquen el menor impacto posible en producción.

Lo interesante es que combina los beneficios de procesos automatizados con la visión crítica del trabajador.

En conclusión debido al cambiante presente en el que vivimos, resulta fundamental aplicar esta técnica si queremos un producto de software de alta calidad.