Trabajo conceptual 1

# **Continuous delivery**

# Guión

## Introducción

Nuestro grupo de trabajo está compuesto por:

* Axel Wismer
* Benjamín Leonardi
* Rodrigo Toledo
* Sebastián Demasi
* Sabrina Pescetti
* Nahuel Piguillem

## Introducción al tema

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.

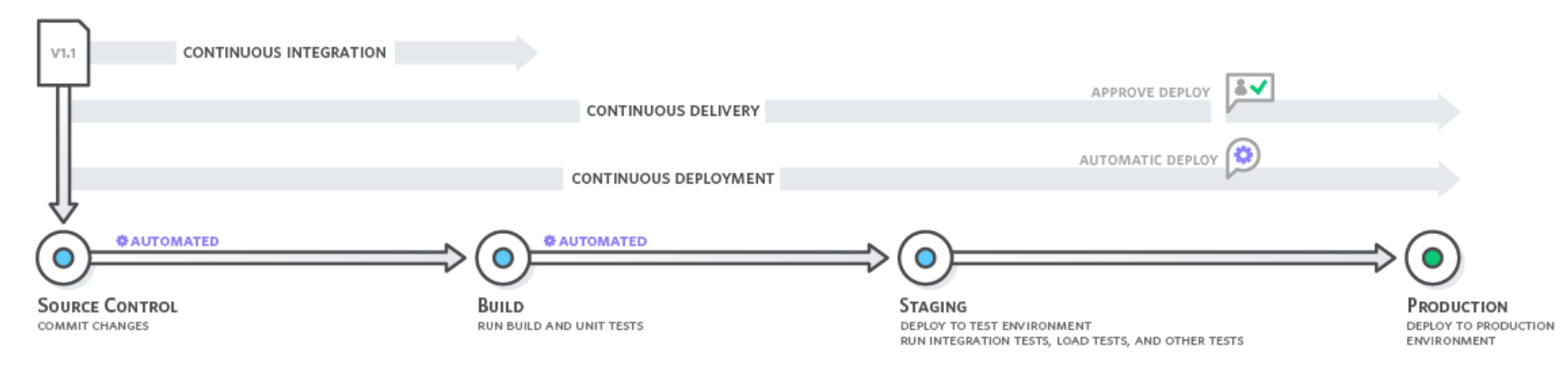
## Desarrollo (N segundos)

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

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.

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

# Entrega continua



*La entrega continua automatiza todo el proceso de publicación de software. Cada revisión efectuada activa un proceso automatizado que crea, prueba y almacena la actualización. La decisión definitiva de implementarla en un entorno de producción en vivo la toma el desarrollador.*

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. 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

### 

### Bibliografia 1:<https://www.sciencedirect.com/science/article/pii/S0164121217300353?via%3Dihub> No aplica a nada

### 2:<https://aws.amazon.com/devops/continuous-delivery/>

3:<https://www.atlassian.com/continuous-delivery>

4:<https://blog.calendly.com/continuous-delivery-gives-your-development-process-competitive-edge/>  
5:<https://www.bmc.com/blogs/continuous-delivery-continuous-deployment-continuous-integration-whats-difference/>

6 <https://www.atlassian.com/continuous-delivery/principles/continuous-integration-vs-delivery-vs-deployment>

## 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.

## 

## Estructura para el video:

### Introducción:

**Introducción del grupo:(10 seg)** (Somos el grupo tanto de la materia ing de software y vamos a exponer sobre el tema Continuous delivery

* Aca se muestra la “carátula” del video

**Introducción Posta:** (40 seg)

### Desarrollo:

**Qué es? explicación:**

(Introduzca subitems)

**Que se hace (acciones):**

* Alguna animación re copada donde represento las acciones que se hacen

**Donde se ubica:** Explicar su relación con las otras cosas

* Grafico de continuous integration,continuous delivery, continuous deployment

**Diferencia con continuous deployment:** Explicar que uno es manual y el otro automático

* Gráfico donde muestro las acciones de los dos mientras se explica

**Beneficios:**

* Gráfico de ventajas

**Limitaciones:**

* Gráfico de limitaciones

### Conclusión

**Conclusión:**

* Algun grafico super genérico que de alguna forma se refiera a la idea de la conclusión

**Integrantes:**

* Nombre de cada uno de los integrantes y por ahí una caricatura de cada uno, encontré unas aplicaciones que las hacen igual si no hay ganas ni,

**Bibliografía y agradecimientos:**

* A lo sumo le meto las portadas de los libros que usamos y los logos de las páginas (investigar el formato de las bibliografías)

TP3

1. Definir una estructura de carpetas para un repositorio que sirva para brindar soporte a la cursada de la materia en este cuatrimestre

2. Seleccionar e identificar todos los ítems de configuración que mantendrán bajo control de versiones indicando: nombre del ítem, Telga de nombrado con extensión del archivo y etiquetas en el caso de ser necesarias, y finalmente la ruta completa en donde serán. Almacenados dentro de su estructura de repositorio

3. Definir el criterio para la creación de una línea base (o varias, según lo que definan)

Ejemplo: Minuta de Reunión

DE\_Minuta\_deReunion<aaaammdd>.txt

/proyecto/minutas/

4. Escriban un glosario al final del documento

5. Implementar el repositorio planteado en una herramienta de acceso público

TP4 Usar mucho y bien el repo

<embed width="800" height="450" src="https://speed.animaker.com/appswf/assets/ANIMAKER\_PLAYER\_V03.swf" type="application/octet-stream" flashvars="projecturl=https://projectxml.s3.amazonaws.com/QzZ4p31kkwoN.xml?versionId=pXvKnkrLLATMy1iVBkBaS8XcgzrC.eRw&amp;projectType=horizontal&amp;isVideoOutro=1&amp;isWatermark=1" allowscriptaccess="always" allowfullscreen="true" wmode="transparent">

<embed width="800" height="450" src="https://speed.animaker.com/appswf/assets/ANIMR\_PRESENTATION\_V16.swf" type="application/octet-stream" flashvars="userid=Axel&amp;sessionid=&amp;isVideoOutro=1&amp;isWatermark=1&amp;projectid=QzZ4p31kkwoN&amp;projecturl=https://projectxml.s3.amazonaws.com/QzZ4p31kkwoN.xml?versionId=GcfM2er9Vag3JIyfLbPOATgp3h\_1f77o" bgcolor="#CCC" allowscriptaccess="always" allowfullscreen="true" wmode="transparent">