CI/CD

Integrando tecnologías, técnicas y negocio para mejorar la eficiencia y calidad de los proyectos

Author/Client
Juan Saavedra

Date
11/06/2019

info@octobot.io — www.octobot.io

Agenda

1. Sobre Octobot
2. Historia
3. ¿Qué es CI?
4. Product Development
5. Escenarios
6. Herramientas

Octobot

Digital Products Experts

Octobot

- Desarrollo de software web a medida mediante la combinación de tecnologías de punta, metodologías ágiles y un gran equipo.
- Nuestro toolkit habitual es:
    - Python/Django
    - React
- Estamos en el *Silicon Bulevar* y construimos productos digitales para organizaciones en USA y Uruguay.
- Trabajamos con nuestros clientes en todo el ciclo de vida de sus productos.
- www.octobot.io

jobs@octobot.io

¿Buscando un desafío?

A short time ago...

In this same planet...

En facultad

Alice - Te parece si cada uno hace su parte en su casa y nos juntamos en la sala de máquinas a armar la entrega?

Bob - Dale demás...

Alice y Bob consiguen trabajo

- Alice y Bob tienen que construir 10 features para dentro de dos meses. Se planifican y dividen el trabajo a medias.
- *Se ponen de acuerdo en no pisarse.* 👹
- *Planifican integrar un par de días antes de salir en vivo* 👹
- *Trabajan en su código cada uno.*👹

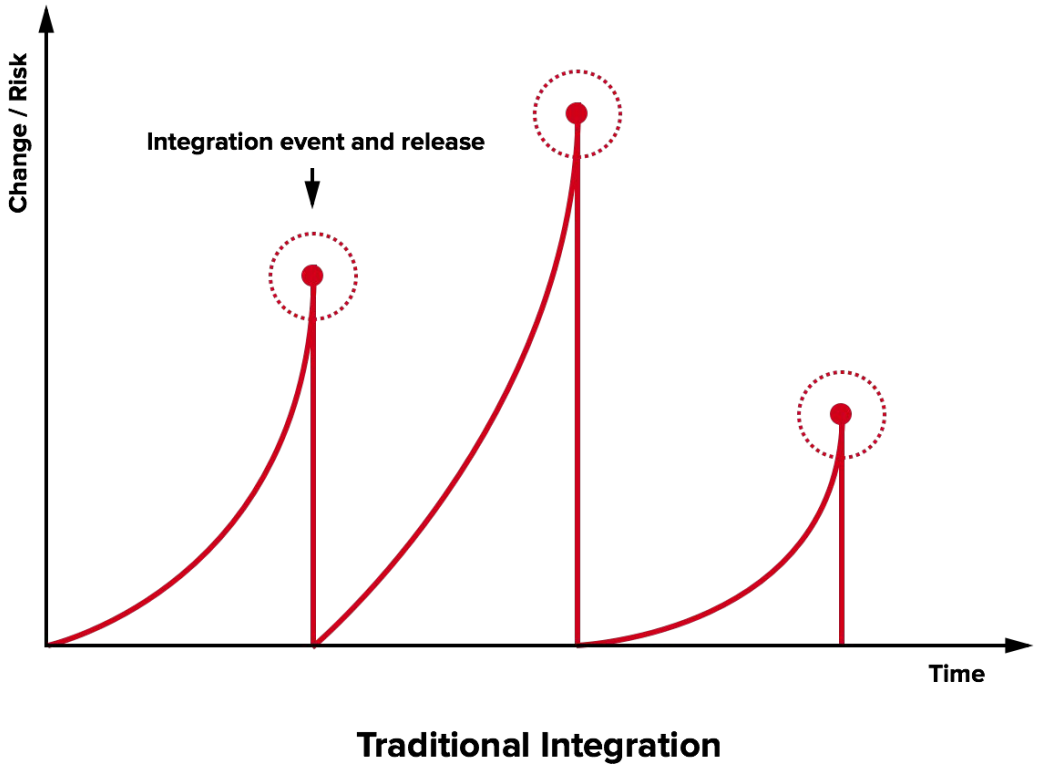
Integration Hell

- *El camino al infierno está lleno de buenas intenciones.*
- Definir zonas de exclusión artificiales genera límites a la productividad (y no funciona).
- Es difícil estimar el trabajo de integrar.
- Multiplicadores de catástrofe
  - Tamaño de la organización
  - El período de sincronización de un dev al repo principal.

Integration Hell

- Peores facetas:
  - Deploys calamitosos - Malo para el equipo y el cliente.
  - Código redundante - Deuda técnica
  - Inestable/buggy - Malo para el cliente
- Peor faceta: **Perdemos tiempo útil por como trabajamos**.
- El problema de fondo, es que Alice y Bob **no comparten su trabajo periódicamente**.

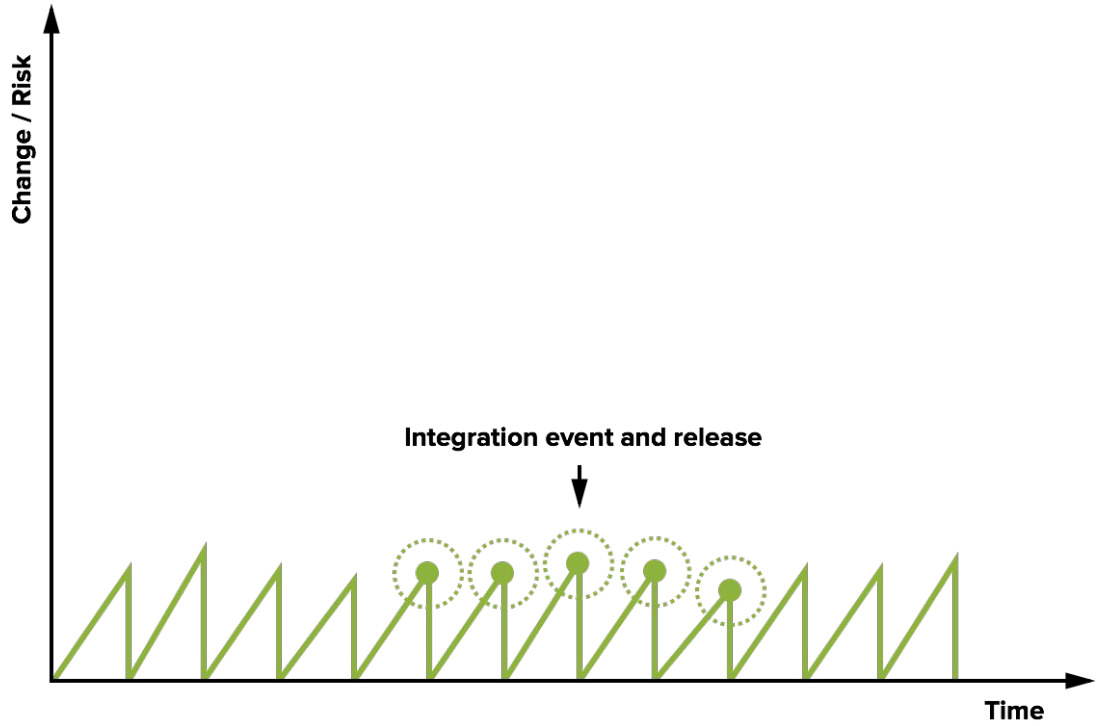
Long Integration Cycle



Traditional Integration

https://semaphoreci.com/community/tutorials/continuous-integration

Short Integration Cycle



¿Qué es CI?

Continuous Integration

¿Qué es CI?

- Conceptualmente, es una práctica de desarrollo que busca que todos los que trabajan en un mismo proyecto integren su trabajo regularmente.
- Y ya que estamos…
  - ¿Y si cada vez que integramos hacemos un build para ver que funcione?
  - ¿Y si cada vez que integramos corremos los tests para ver que no se rompe?
  - ¿Y si hacemos que esto sea automático?

¿Que precisamos tener?

- Repositorio único de proyecto.
- Build automático.
- El build incluye la ejecución de tests
- Un servidor de integración que en base a un commit, ejecute tests y builds.
- Se prueba en un ambiente de pruebas clon de producción.
- Es fácil acceder y ejecutar el proyecto y sus releases.
- Todos tienen visibilidad de lo que está sucediendo en el código.

¿Qué tenemos que hacer?

- Trabajar en el mismo repositorio.
- Subir/integrar el código frecuentemente.
- No subir código que rompe el build.
- No subir código que no pasa los tests.
- No subir código sin tests asociados.
- No subir/integrar nuevos cambios si está roto.
- No irte hasta que lo último que subiste haya hecho build.

¿Qué tareas implica?

1. El desarrollador obtiene el código del repositorio.
2. Hace sus cambios localmente.
3. Corre tests y hace el build.
4. Sube todo al repo
5. El **servidor de CI** monitorea el repo y lanza tests y builds.
6. Si falla, el servidor avisa al equipo.
    - El equipo trata de corregirlo lo antes posible.
7. Si no falla, deja el build disponible.

Ventajas

- Chau Integration Hell 👋
- *Continuous Integration doesn’t get rid of bugs, but it does make them dramatically easier to find and remove* - Martin Fowler
- Chau “En mi máquina anda” 👋
- **Fácil cambiar código existente**
- Mejor capacidad de sacar features más estables.

Continuous Deployment

- Deploy continuo
- ¿Y si agregamos un paso automático más que sea la publicación del último build?
- Lleva un trabajo adicional no menor de:
  - Resiliencia
  - Estrategia de rollback automático
  - Restringe las estrategias de release
- No todos las organizaciones están interesadas en esto.

¿Ya está?

- ¿Qué pasa si no puedo lograr algunos de los items necesarios?
  - Para algunas plataformas existentes, **no es sencillo** lograr un repo central e integraciones frecuentes.
- ¿Tiene sentido tener CI en todos los proyectos?
- ¿Todos los tests tienen que ser automáticos?
- ¿Que tipo de tests tenemos que tener?
- **¿No hay que hacer más nada?**

Product Development

Un enfoque basado en iteraciones cortas

Iteraciones Cortas

- La introducción de CI se retroalimenta de un desarrollo de productos basado en la metodología Lean
- Principios
  - Eliminate waste
  - **Amplify learning**
  - **Decide as late as possible**
  - **Deliver as fast as possible**
  - Empower the team
  - Build integrity in
  - Optimize the whole

Metodología de proyecto

- La adopción de estas prácticas de desarrollo **debe** estar acompañada por ajustes en la metodología si la misma es incompatible.
- CI es algo simbiótico con metodologías ágiles como SCRUM.
- Podemos alinear ceremonias con eventos de CI.
- Nos provee de agilismo para adaptarnos con bajo costo a cambios de necesidades.

Desarrollo de producto

- Hay elecciones a nivel de equipo de desarrollo.
- Hay elecciones a nivel de negocio y desarrollo de producto.
- Las técnicas y procedimientos a utilizar, dependen del objetivo perseguido y los problemas actuales.

Escenarios

#1 - Obligatorio de Facultad

- Obligatorio de Arquitectura de Computadoras
- 4 Personas - 3 Semanas
- Alta concurrencia en los objetos a trabajar
- Objetivo de negocio: salvar.
- Ponemos CI?

#1 - Obligatorio de Facultad

- Tiempo que se dedica a integrar < tiempo que se dedica a armar ci
- Herramientas poco amigables para testear
- Mantenibilidad?
- Aunque se puede hacer algo de **TDD**

#2 - Taller de programación

- 4 (?) Personas - 3 Meses
- Repositorio central 😐
- Modulos definidos
- Alta concurrencia en los objetos a trabajar
- Gran cantidad de casos
- Herramienta moderna

#2 - Taller de programación

- Parece una buena idea!
- Mayor agilidad al hacer cosas nuevas sin miedo de romper lo anterior.
- Bien modularizado, una aplicación es más fácil de testear, menos esfuerzo de introducir
- Mejora la comunicación del equipo.
- Menos bugs.
- Capaz no es necesario un batallón de tests!

#3 - Startup

- Dos amigos tienen una idea de un servicio online y juntan ahorros para el software.
- Les da para 3 meses de 2 developers.
- Trabajan en descubrir su *Minimum Viable Product (MVP)* y es apenas suficiente para terminarlo.
- Repositorio central
- Features definidas aproximadamente
- Alta concurrencia en los objetos a trabajar
- Poca cantidad de casos
- Herramienta moderna

#3 - Startup

- Podemos hacer algo, pero no es prioritario.
- Precisan *ya* generar ingresos y ventas.
- Es importante ser conscientes de la deuda técnica.

#4 - Proyecto grande y nuevo

- Un banco se propone lanzar una plataforma de préstamos online.
- 5 developers - 6 meses de trabajo.
- Gran inversión - Tiempo de retorno.
- Herramienta moderna
- Repositorio central
- Baja concurrencia de edición pero muchos objetos.

#4 - Proyecto grande y nuevo

- Seguro que si.
- Costo marginal con alto retorno en el largo plazo.
- Aumenta mantenibilidad.
- Facilita evolución.
- Más las ventajas de siempre.
- **Agrega integridad.**

#5 - Proyecto grande y existente

- Un ministerio quiere actualizar su sistema de gestión.
- No tiene CI.
- 3 meses 2 personas.
- Muchas funcionalidades existentes
- Plataforma algo moderna
- Repositorio central
- Alta concurrencia de edición y muchos objetos.

#5 - Proyecto grande y existente

- Hacemos algo híbrido.
- Foco principal del CI: no romper lo existente!
- No podemos integrar todo el proyecto a un CI para hacer los cambios pedidos.
- Súper favorable hacer el build y verificar que funciona.
- Pero si agregamos algunos tests que verifiquen que lo que “está cerca” no se rompe, ganamos.
- Además, agregamos tests a lo nuevo.

6 - Tecnologías Legacy

- Los conceptos de CI se pueden aplicar en tecnologías legacy.
- Aunque no tenemos todas las herramientas, podemos aplicar técnicas.
  - Build fácil e independiente de ambiente.
  - Repositorio centralizado
  - Ambiente de pruebas réplica
- Se pueden agregar *linters* y analizadores estáticos de código.
- Idealmente, agregar casos de prueba.

Tomar en cuenta las necesidades de negocio y maximizar utilidad.

Key Take-Aways

- Hay un objetivo global, pero no hay una receta única.
- Hay un trade-off.
- Atacar problemas en orden de gravedad.
- Es posible migrar.
- A veces trae atado cambios organizacionales.

Herramientas de CI

Version Control System

- GIT
- SVN 💩
- Notar que git **no implica un modelo de trabajo.**
- Alternativas:
  - Github flow
  - git-flow

Servidores CI

- Travis CI
- Jenkins
- TeamCity
- CircleCI
- Gitlab CI
- Go CD
- ...

Features Servidores CI

- Pricing
- Integración con el toolset organizacional
- Dificultad de Setup
- **Definición de pipeline en el código**

Octobot Pick

- Flexible
- Pipelines as code
- Independiente de tecnología
- Integrado con git
- Gran integración con docker
- **Runners escalables**
- Setup fácil

Preguntas

Gracias