ADR Integración Continua

# Contexto

Se necesita definir una serie de herramientas capaces de proporcionar una forma ágil de integrar los cambios en el proyecto. Asimismo, deben posibilitar que una vez integrados, todo funcione correctamente.

## Procesos

1. Instalación de requisitos previos para la integración y despliegue:
   1. Instalación de herramientas: Git, Docker, Jenkins. (Véanse apartados de “Opciones consideradas” y “Decisión”)
   2. Creación de un pipeline en la herramienta Jenkins. (Véase Proceso 3, “Integración de cambios en el entorno”)
2. Integración de cambios en el repositorio (git push):
   1. Todos los cambios deberán aplicarse a la rama *master* del repositorio, siguiendo la filosofía de integración continua.
   2. Los cambios deberán ir acompañados de un breve comentario que indiqué a grandes rasgos qué cambia.
3. Integración de cambios en el entorno (Pipeline de Jenkins):
   1. Cualquier cambio realizado en la aplicación se actualiza en la rama *master*. No se realizan ramificaciones.
   2. Tests unitarios: definir scripts simples para comprobar el correcto funcionamiento del código implementado. Se debe realizar un test por cada nuevo cambio en el código desarrollado y lanzarlos sobre rama Master.
   3. Si los test unitarios SÍ son satisfactorios:
      1. Se genera un ejecutable
      2. Se genera el correspondiente contenedor con Docker
   4. Si los test unitarios NO son satisfactorios:
      1. revisa los test unitarios paras aplicar las respectivas correcciones.
   5. Tests de integración: comprobar que los cambios realizados no entren en conflicto con partes ya realizadas. Se debe realizar un test por cada nuevo cambio en el código desarrollado.
   6. En caso de que todo funcione correctamente, se procederá al despliegue. (Véase Proceso 4, “Despliegue de la aplicación”)
4. Despliegue de la aplicación (Docker):
   1. TODO: rellenar esto xD

# Opciones consideradas

1. Integración continua

Las herramientas consideradas para la integración continua no sólo se diferencian por sus características, sino que también por sus precios y licencias.

* Jenkins
  + Escrito en Java
  + Se ejecuta en un contenedor EJB
  + Más de 1.000 plugins
  + Asiste también en la entrega y el despliegue continuo
  + Compatible con muchos sistemas de control de versiones
  + Controles mediante GUI (basados en web), API REST o línea de comandos
  + Alojamiento opcional en la nube
  + Gratuita
  + De código abierto (licencia MIT)
* Travis CI

Trabaja en estrecha relación con el popular software de control de versiones. Puede configurarse con un sencillo archivo YAML. GitHub informa a Travis CI de todos los cambios efectuados en el repositorio y mantiene el proyecto actualizado.

* + Programado en Ruby
  + Multiplataforma
  + Funciona con GitHub
  + Se configura con un archivo YAML
  + Gratuita para proyectos de código abierto
  + Precio para proyectos comerciales: entre 69 y 489 dólares/mes
  + De código abierto (licencia MIT)
* Bamboo

Esta herramienta no solo sirve de ayuda en la integración continua, sino también para funciones de despliegue y gestión de lanzamientos. Funciona a través de una interfaz web.

* + Escrito en Java
  + Multiplataforma
  + Fácil integración de otros productos Atlassian
  + Gran cantidad de addons
  + Realización de varias pruebas al mismo tiempo
  + Interfaz web y API REST
  + Gratuita para proyectos de código libre, ONG y centros escolares
  + De lo contrario, pago único de entre 10 y 126 500 dólares, dependiendo del número de servidores utilizados
* GitLab CI

Forma parte del conocido sistema de control de versiones GitLab. Además de integración continua, GitLab ofrece despliegue y entrega continua. Al igual que con Travis CI, la configuración de GitLab CI se lleva a cabo con un archivo YAML. Por lo demás, su utilización es sencilla.

* + Forma parte de GitLab
  + Programado en Ruby y Go
  + Configuración con un archivo YAML
  + Asiste también en la entrega y el despliegue continuo
  + Open Core
  + Alojamiento propio o en la nube
  + Versión gratuita con pocas funciones
  + Precio para otras versiones, entre 4 y 99 dólares/mes por usuario.
* Resumen de características orientada a la toma de decisión:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Jenkins** | **Travis CI** | **Bamboo** | **GitLab** |
| Entrega continua | ✓ | ✗ | ✓ | ✓ |
| Alojamiento en la nube | ✓ | ✓ | ✓ | ✓ |
| Licencia | MIT | MIT | De propietario | MIT/EE |
| Precio versión comercial | - | 69-489 $/mes | 10 - 126 500 $ (pago único) | 4-99 $/mes |
| Versión gratuita | ✓ | ✓ | ✓ | ✓ |
| Particularidades | Numerosos plugins | Conexión directa con GitHub | Conexión directa con otros productos Atlassian |  |

1. Despliegue de la aplicación

* Docker

Sistema que nos permite construir, transferir, desplegar y ejecutar los contenedores con nuestras aplicaciones dentro de una manera muy sencilla y confiable, garantizando un despliegue escalable de forma eficiente sin importar el sistema operativo anfitrión.

* Kubernetes

Sistema que se encarga de gestionar todo el cluster de servidores, distribuye los contenedores a través del sistema según los recursos disponibles en el cluster, ademas de crear, ejecutar, vigilar, medir, destruir y relanzar los contenedores, debe mantener y controlar en todo momento cada aspecto relevante de los contenedores y su estado.

|  |  |  |
| --- | --- | --- |
|  | **Docker** | **Kubernetes** |
| Escalado | Sin escalado automático | Autoescalado |
| Configuración de cluster | Desafiante y complicado | Simple (con dos comandos) |
| Instalación | Fácil y rápido | Complicado y lento |
| Volumen de datos | Comparte entre varios contenedores el mismo Pod | Comparte con cualquier otro contenedor |
| Balance de carga | Autobalanceo de carga | Configuración manualmente |
| Relación de tolerancia | Alta tolerancia a fallos | Baja tolerancia a fallos |
| Límite de contenedores | 95.000 | 300.000 |
| Multiplataforma | ✓ | ✓ |
| Tiempo de creación | Pocos segundos | Minutos |
| Tiempo de arranque | Milésimas de segundo o unos pocos segundos | Minutos |

# Decisión

1. Para la **integración continua**, se ha decidido usar **Jenkins** porque es una herramienta gratuita (en comparación a las otras tres aplicaciones), es compatible con muchos sistemas de control de versiones (en nuestro caso usaremos GitHub) y dispone de numerosos plugins. Además, existe la disposición de *Pipelines* para los tests de integración.

Se podría haber decidido usar **Travis CI** por la conexión directa con GitHub, pero la condición para ser una herramienta gratuita es que el proyecto sea de código abierto y el nuestro no lo es.

1. Para el despliegue de la aplicación se ha decido usar **Docker** porque es una herramienta que todo el equipo conoce y la instalación del mismo es fácil y rápida.
2. Se decide que no es necesario realizar tests de rendimiento (comprobaciones de que el sistema mantiene el servicio en momentos de gran afluencia de usuarios o peticiones).

* Se consideran actualmente omisibles debido a que el sistema no se encuentra en situaciones de tal envergadura, en principio, luego el tiempo dedicado a diseñar los tests y probarlos con cada cambio significativo en el servicio, no parece razonable. Esta decisión podría cambiar si se dispusiera de tiempo suficiente al final del proyecto para realizar un estudio rigurosa de la escalabilidad de nuestro sistema, aunque por ahora, como se ha dicho, no es lo más relevante.

# Estado

(Tachar las que no apliquen)

1. Aceptado
2. ~~Rechazado~~
3. ~~Reemplazado~~

# Implicaciones

La utilización de las herramientas elegidas supone que el equipo de desarrollo deba aprender a manejarlas, lo cual conllevará un tiempo considerable a tener en cuenta en la gestión inicial del proyecto.

### Fecha

27 de marzo de 2020