

Continuous Integration EGC

AgoraUS - Grupo 1

Manuel Francisco López Ruiz

Evolución y Gestión de la Configuración [EGC]

Grado en Ingeniería Informática - Ingeniería del Software

Universidad de Sevilla (Spain)

23 de noviembre de 2016

Índice

1 Introducción

2 Fases del despliegue

- Fase Make
- Fase Beta
- Fase Stable
- Ejemplos
 - Ejemplo 1
 - Ejemplo 2

3 Implementación de la Integración

- Ejemplo de integración 1
 - Fase Make
 - Fase Beta

Índice

1 Introducción

2 Fases del despliegue

- Fase Make
- Fase Beta
- Fase Stable
- Ejemplos
 - Ejemplo 1
 - Ejemplo 2

3 Implementación de la Integración

- Ejemplo de integración 1
 - Fase Make
 - Fase Beta

¿Que es un sistema de integración continua?

Integración continua

La integración continua [...] consiste en hacer integraciones automáticas de un proyecto lo más a menudo posible para así poder detectar fallos cuanto antes. Entendemos por integración la compilación y ejecución de pruebas de todo un proyecto.

En nuestro caso también desplegará la aplicación para que todos los grupos podamos usarla.

¿Que aporta?

Integración continua - Ventajas

- *Los desarrolladores pueden detectar y solucionar problemas de integración de forma continua, evitando el caos de última hora cuando se acercan las fechas de entrega.*

¿Que aporta?

Integración continua - Ventajas

- *Los desarrolladores pueden detectar y solucionar problemas de integración de forma continua, evitando el caos de última hora cuando se acercan las fechas de entrega.*
- *Disponibilidad constante de una versión para pruebas, demos o lanzamientos anticipados.*

¿Que aporta?

Integración continua - Ventajas

- *Los desarrolladores pueden detectar y solucionar problemas de integración de forma continua, evitando el caos de última hora cuando se acercan las fechas de entrega.*
- *Disponibilidad constante de una versión para pruebas, demos o lanzamientos anticipados.*
- *Ejecución inmediata de las pruebas unitarias.*

¿Que aporta?

Integración continua - Ventajas

- *Los desarrolladores pueden detectar y solucionar problemas de integración de forma continua, evitando el caos de última hora cuando se acercan las fechas de entrega.*
- *Disponibilidad constante de una versión para pruebas, demos o lanzamientos anticipados.*
- *Ejecución inmediata de las pruebas unitarias.*
- *Monitorización continua de las métricas de calidad del proyecto.*

Índice

1 Introducción

2 Fases del despliegue

- Fase Make
- Fase Beta
- Fase Stable
- Ejemplos
 - Ejemplo 1
 - Ejemplo 2

3 Implementación de la Integración

- Ejemplo de integración 1
 - Fase Make
 - Fase Beta

¿Que se realiza en la Fase Make?

- Descargar el código del proyecto tras una modificación en el mismo.
- Ejecutar test para comprobar la integridad.
- Realizar las acciones necesarias sobre el código para usarlo en despliegue.

En resumen...

Ejecutar los test y todas las acciones oportunas antes de desplegar el proyecto.

Posibles subsecciones de la Fase Make

Puede que se necesiten ejecutar ciertas acciones antes de descargar y ejecutar el proyecto. Por ejemplo, desplegar una base de datos.

En esos casos existirán dos fases adicionales dentro de la Fase Make:

- **Pre-Make:** Se prepara el entorno antes de la ejecución del make. Ej: Desplegar la base de datos.
- **Post-Make:** Se deshace la preparación del entorno de ejecución make. Ej: Eliminar la base de datos.

¿Que se realiza en la Fase Beta?

Se ejecuta automáticamente tras la Fase Make.

Fases principales:

- 1 Eliminación restos de despliegue anterior (*contenedores, archivos y carpetas*).
- 2 Preparación del despliegue. *Crear carpetas, mover archivos desde la última compilación a las carpetas deseadas, descargar archivos necesarios...*
- 3 Despliegue.

¿Que se realiza en la Fase Stable?

- Se ejecuta manualmente.
- Las fases principales son las mismas que en la Fase Beta.
- Los archivos usados para el despliegue provienen de la Fase Beta.
- Su fin es la estabilidad.

Ejemplo 1 - Deliberations

- Se usa Java, Maven y Mysql.
- El código se publica en github y la rama de trabajo principal es *develop*.
- Se debe desplegar en una máquina con Tomcat y Mysql.

Resolución Ejemplo 1 - Deliberations

- **Fase Make:** Se ejecuta al detectar un cambio en la rama *develop*.
 - 1 Se descarga el código de la rama *develop*.
 - 2 Se ejecuta Maven para compilar el proyecto.
 - 3 Se guarda el war generado y el script de población descargado del repositorio.
- **Fase Beta:** se ejecuta automáticamente al finalizar la *fase make*.
 - 1 Se comprueba si es la primera ejecución y, si no lo es, se borran los archivos de la anterior ejecución.
 - 2 Se depositan los archivos en las carpetas oportunas.
 - 3 Se despliega Mysql y se carga el script de población.
 - 4 Se lanza el servidor Tomcat con el war.

Ejemplo 2 - Deliberations Ext.

- Se usa Java, Maven y Mysql.
- El código se publica en github y la rama de trabajo principal es *develop*.
- Deben pasarse los test JUnit con maven y para ello es necesario la conexión con Mysql.
- El script de población para el despliegue se debe crear automáticamente.
- Se debe desplegar en una máquina con Tomcat y Mysql.

Resolución Ejemplo 2 - Deliberations Ext.

- **Fase Make:** Se ejecuta al detectar un cambio en la rama *develop*.
 - 1 **Fase pre-make:** Se despliega Mysql, se crean los usuarios necesarios en la misma y se inicializa la base de datos.
 - 2 Se descarga el código de la rama *develop*.
 - 3 Se invoca Maven para pasar los test y compilar el proyecto.
 - 4 Se realiza un dump de la base de datos Mysql.
 - 5 Se guarda el war generado y el dump de la base de datos.
 - 6 **Fase post-make:** Se elimina el Mysql desplegado.
- **Fase Beta:** se ejecuta automáticamente al finalizar la *fase make*.
 - 1 Se comprueba si es la primera ejecución y, si no lo es, se borran los archivos de la anterior ejecución.
 - 2 Se depositan los archivos en las carpetas oportunas.
 - 3 Se despliega Mysql y se cargan los usuarios necesarios y el dump de la base de datos.
 - 4 Se lanza el servidor Tomcat con el war.

¿Alguna duda sobre como dividir la integración de vuestro proyecto en fases?

Y ahora...

Perfecto, ya se como dividir mi trabajo en las diferentes fases pero...
¿como implemento cada una?

Índice

1 Introducción

2 Fases del despliegue

- Fase Make
- Fase Beta
- Fase Stable
- Ejemplos
 - Ejemplo 1
 - Ejemplo 2

3 Implementación de la Integración

- Ejemplo de integración 1
 - Fase Make
 - Fase Beta

Herramientas utilizadas

- **Jenkins**
- **Docker**
- **Bash**
- **Git/GitHub**

¿Que es Jenkins?



Jenkins proporciona integración continua para el desarrollo de software [...] Soporta herramientas de control de versiones como CVS, Subversion, Git, Mercurial, Perforce y Clearcase y puede ejecutar proyectos basados en Apache Ant y Apache Maven, así como scripts de shell [...]
Liberado bajo licencia MIT, Jenkins es software libre.

jenkins.egc.duckdns.org

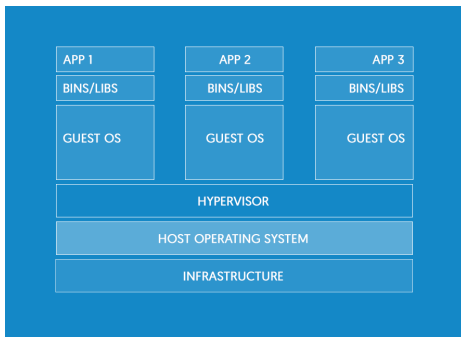
¿Que es Docker?



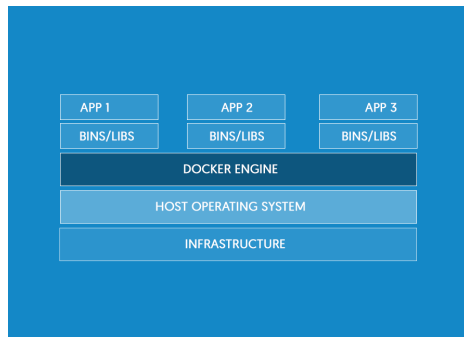
Docker es un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de software, proporcionando una capa adicional de abstracción y automatización de Virtualización [...]. Docker utiliza características de aislamiento de recursos del kernel de Linux [...] para permitir que “contenedores” independientes se ejecuten dentro de una sola instancia de Linux, evitando la sobrecarga de iniciar y mantener máquinas virtuales.

www.docker.com/what-docker
hub.docker.com

¿Que es Docker?

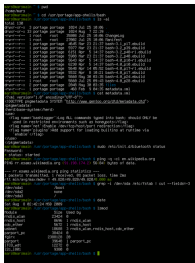


Virtual Machines



Containers

¿Que es Bash?



Bash (Bourne again shell) es un programa informático, cuya función consiste en interpretar órdenes, y un lenguaje de programación de consola. Está basado en la shell de Unix y es compatible con POSIX. [...] es el intérprete de comandos por defecto en la mayoría de las distribuciones de GNU con Linux.

explainshell.com

¿Que es GitHub?

¿Porqué estas tecnologías?

- Todas son **libres** y de **código abierto**.

¿Porqué estas tecnologías?

- Todas son **libres** y de **código abierto**.
- **Jenkins**: el más habitual para sistemas de integración y es muy estable (si, es feo, pero es rápido, ligero y eficaz).

¿Porqué estas tecnologías?

- Todas son **libres** y de **código abierto**.
- **Jenkins**: el más habitual para sistemas de integración y es muy estable (si, es feo, pero es rápido, ligero y eficaz).
- **Docker**: permite correr cada ejecución con su propia configuración independientemente del sistema anfitrión. Además, es posible realizar cada ejecución en un entorno totalmente nuevo (no deberemos reparar lo que la anterior ejecución rompió).

¿Porqué estas tecnologías?

- Todas son **libres** y de **código abierto**.
- **Jenkins**: el más habitual para sistemas de integración y es muy estable (si, es feo, pero es rápido, ligero y eficaz).
- **Docker**: permite correr cada ejecución con su propia configuración independientemente del sistema anfitrión. Además, es posible realizar cada ejecución en un entorno totalmente nuevo (no deberemos reparar lo que la anterior ejecución rompió).
- **Bash**: eficaz y con total control sobre la máquina. Evitamos librerías o usar comandos distintos.

Muy bien, pero...
¿¿COMO LO IMPLEMENTO??

Recordáis...

Fase Make - Jenkins Slave

Jenkins > Configuración

Docker

Name

Docker URL

Docker API Version

Credentials [Add](#)

Connection Timeout

Read Timeout

[Test Connection](#)

Container Cap

Images

Docker Template

Docker Image

Docker Command

LXC Conf Options

Hostname

DNS

Network

Volumes

Volumes From

Environment

Port bindings

Bind all declared ports ☐

Fase Make - Jenkins Slave

Jenkins > Configuración

Bind all declared ports	<input type="checkbox"/>
Memory Limit in MB	<input type="text"/>
Swap Memory Limit in MB	<input type="text"/>
CPU Shares	<input type="text"/>
Run container privileged	<input type="checkbox"/>
Allocate a pseudo-TTY	<input type="checkbox"/>
MAC address	<input type="text"/>
Extra Hosts	<input type="text"/>
Instance Capacity	<input type="text" value="1"/>
Remote Filing System Root	<input type="text" value="/home/jenkins"/>
Labels	<input type="text" value="jenkins-slave-jdk-maven-git"/>
Usar	<input type="text" value="Utilizar este nodo tanto como sea posible"/>
<div>Experimental Options...</div>	
Launch method	<input type="text" value="Docker SSH computer launcher"/>
Credentials	<input type="text" value="jenkins/***** (jenkins/jenkins)"/> <input type="button" value="Add"/>
<div>Avanzado...</div>	
Remote FS Root Mapping	<input type="text" value="/var/jenkins_home"/>
Remove volumes	<input checked="" type="checkbox"/>
Pull strategy	<input type="text" value="Pull once and update latest"/>

Fase Make - Jenkins Job Configure

Imagen en DockerHub

Imagen en GitHub

Fase Make - Jenkins Job Configure

Jenkins > AgoraUS-G1-Deliberations_make >

General Configurar el origen del código fuente Disparadores de ejecuciones Entorno de ejecución Pasos previos Proyecto Pasos posteriores Propiedades del proyecto

Acciones para ejecutar después.

Maven project nombre

Descripción

[Plain text] [Visualizar](#)

☒ Desechar ejecuciones antiguas ?

Strategy

Número de días para mantener ejecuciones de proyectos
si no está vacío, sólo se mantendrán las ejecuciones con una edad inferior a este número de días

Número máximo de ejecuciones para guardar
si no está vacío, sólo se guardarán un número de ejecuciones inferior a este valor

[Avanzado...](#)

☐ Docker Container

☐ Esta ejecución debe parametrizarse ?

☒ GitHub project ?

Project url ?

[Avanzado...](#)

Rebuild options: ☒ Rebuild Without Asking For Parameters ?
☐ Disable Rebuilding for this job

☐ Throttle builds ?

[Guardar](#) [Apply](#)

Fase Make - Jenkins Job Configure

Jenkins > AgoraUS-G1-Deliberations_make >

General | Configurar el origen del código fuente | Disparadores de ejecuciones | Entorno de ejecución | Pasos previos | Proyecto | Pasos posteriores | Propiedades del proyecto

Acciones para ejecutar después.

- ☐ Throttle builds
- ☐ Prepare an environment for the run
- ☐ Desactivar la ejecución
- ☐ Lanzar ejecuciones concurrentes en caso de ser necesario
- ☒ Restringir dónde se puede ejecutar este proyecto.
Expresión:
Label is serviced by no nodes and 1 cloud

[Avanzado...](#)

Configurar el origen del código fuente

☐ Ninguno
☒ Git

Repositories

Repository URL: [?](#)

Credentials: [Add](#)

[Avanzado...](#)
[Add Repository](#)

Branches to build

Branch Specifier (blank for 'any'): [?](#)

[Add Branch](#)

Navegador del repositorio: [?](#)

Additional Behaviours: [Añadir](#)

☐ Subversion [?](#)

[Guardar](#) [Apply](#)

Ejecuciones








Fase Make - Jenkins Job Configure

jenkins > AgoraUS-G1-Deliberations_make >



General Configurar el origen del código fuente **Disparadores de ejecuciones** Entorno de ejecución Pasos previos Proyecto Pasos posteriores Propiedades del proyecto

Acciones para ejecutar después.

Disparadores de ejecuciones

☒ Ejecutar siempre que cualquier 'SNAPSHOT' de los que dependa sea creado 
☐ Schedule build when some upstream has no successful builds 
☐ Lanzar ejecuciones remotas (ejem: desde 'scripts') 
☐ Build after other projects are built 
☐ Ejecutar periódicamente 
☒ Build when a change is pushed to GitHub 
☐ Consultar repositorio (SCM) 

Entorno de ejecución


☐ Delete workspace before build starts
☐ Abortar la ejecución si se atasca
☒ Add timestamps to the Console Output
☐ Inject environment variables to the build process 
☐ Inject passwords to the build as environment variables
☐ Use secret text(s) or file(s) 


Pasos previos

Añadir un paso previo ▾

Proyecto

Versión de maven

Fichero POM raíz 

Goles y opciones 

Manuel Francisco López Ruiz (EGC)

Continuous Integration EGC

23 de noviembre de 2016

32 / 39

Fase Make - Jenkins Job Configure

Jenkins » AgoraUS-G1-Deliberations_make »

General Configurar el origen del código fuente Disparadores de ejecuciones Entorno de ejecución Pasos previos Proyecto **Pasos posteriores** Propiedades del proyecto

Acciones para ejecutar después.

Pasos posteriores

☐ Ejecutar sólo en caso de éxito ☐ Ejecutar sólo cuando la ejecución fué buena o inestable. ☒ Ejecutar siempre (sea cual sea el resultado)

Crterios de ejecución

Añadir un paso posterior ▾

Propiedades del proyecto

☐ Notificación por E-mail

Acciones para ejecutar después.

Guardard los archivos generados

Ficheros para guardar

Avanzado...

Ejecutar otros proyectos

Proyectos a ejecutar

☐ Trigger only if build is stable
☒ Lanzar incluso si el resultado de la ejecución fué inestable.
☐ Lanzar incluso si la ejecución acabó con errores.

Añadir una acción ▾

Guardar Apply

Fase Make - Jenkins Job Configure

Es aconsejable copiar el archivo de configuración del trabajo generado por jenkins al repositorio con el fin de facilitar la restauración de la configuración.

Fase Beta - Jenkins Job Configure

jenkins > AgoraUS-G1-Deliberations_beta >

General Configurar el origen del código fuente Disparadores de ejecuciones Entorno de ejecución Ejecutar Acciones para ejecutar después.

Proyecto nombreAgoraUS-G1-Deliberations_beta

Descripción

[Plain text] [Visualizar](#)

☐ Desechar ejecuciones antiguas

☐ Docker Container

☐ Esta ejecución debe parametrizarse

☐ GitHub project

Rebuild options:

☐ Rebuild Without Asking For Parameters

☐ Disable Rebuilding for this job

☐ Throttle builds

☐ Prepare an environment for the run

☐ Desactivar la ejecución

☐ Lanzar ejecuciones concurrentes en caso de ser necesario

☐ Restringir dónde se puede ejecutar este proyecto.

Avanzado...

Configurar el origen del código fuente

☒ Ninguno

☐ Git

☐ Subversion

Disparadores de ejecuciones

GuardarApply

Scripts (ejem: desde 'scripts')

Fase Beta - Jenkins Job Configure

jenkins » AgoraUS-G1-Deliberations_beta »

General Configurar el origen del código fuente **Disparadores de ejecuciones** Entorno de ejecución Ejecutar Acciones para ejecutar después.

Disparadores de ejecuciones

- ☐ Lanzar ejecuciones remotas (ejem: desde 'scripts')
- ☐ Build after other projects are built
- ☐ Ejecutar periódicamente
- ☐ Build when a change is pushed to GitHub
- ☐ Consultar repositorio (SCM)

Entorno de ejecución

- ☐ Delete workspace before build starts
- ☐ Abortar la ejecución si se atasca
- ☐ Add timestamps to the Console Output
- ☐ Inject environment variables to the build process
- ☐ Inject passwords to the build as environment variables
- ☐ Use secret text(s) or file(s)

Ejecutar

Ejecutar línea de comandos (shell)

Comando `bash $JENKINS_HOME/continuous-delivery-playground/AgoraUS/G1-Deliberations/beta.sh`

[Visualizar la lista de variables de entorno disponibles](#)

Añadir un nuevo paso ▾

Acciones para ejecutar después.

Añadir una acción ▾

Guardar **Apply**

Fase Beta - Jenkins Job Configure

- AgoraUS/G1-Deliberations/beta.sh
- Imagen MySQL
- Imagen Tomcat

Referencias I



Wikipedia.

Bash — wikipedia, la enciclopedia libre.

<https://es.wikipedia.org/wiki/Bash>, 2016.

[Internet; descargado 23-noviembre-2016].



Wikipedia.

Docker (software) — wikipedia, la enciclopedia libre.

[https://es.wikipedia.org/wiki/Docker_\(software\)](https://es.wikipedia.org/wiki/Docker_(software)), 2016.

[Internet; descargado 23-noviembre-2016].



Wikipedia.

Integración continua — wikipedia, la enciclopedia libre.

https://es.wikipedia.org/wiki/Integracion_continua, 2016.

[Internet; descargado 23-noviembre-2016].

Referencias II



Wikipedia.

Jenkins — wikipedia, la enciclopedia libre.

<https://es.wikipedia.org/wiki/Jenkins>, 2016.

[Internet; descargado 23-noviembre-2016].