TEMA 097. MODELOS DE INTEGRACIÓN CONTINUA. HERRAMIENTAS Y SUS APLICACIONES

Actualizado a 23/01/2022

1 INTEGRACIÓN CONTINUA

La integración continua (CI) es un modelo que consiste en hacer integraciones automáticas de un proyecto lo más a menudo posible **para poder detectar fallos cuanto antes**. Tiene como objetivo principal comprobar que cada actualización del código fuente no genera problemas en una aplicación en desarrollo, y de esta manera poder avisar al equipo de desarrollo de forma temprana.

A menudo la integración continua está asociada con las metodologías de desarrollo ágil, por el hecho de que favorece las **entregas parciales**. Sin sistemas de CI, sería difícil y caro entregar software en iteraciones cortas.

Un proyecto de integración continua necesita, al menos, los siguientes componentes:

1. Una **herramienta de gestión y construcción de proyectos** para gestionar la configuración y compilar el código, si fuera necesario. Hay muchas soluciones de construcción diferentes Apache Ant, Maven basado en XML, Gradle basado en Groovy o Kotlin DSL, Rake, MSBuild en .NET, por mencionar algunas.
2. Un **sistema de control de versiones o SCM System Control Management**: es la herramienta que almacena todos los cambios realizados tanto en la estructura de directorios como en el contenido de los ficheros, con el fin de tener un histórico navegable. Ofrece persistencia en un entorno dedicado. Y es imprescindible cuando más de una persona trabaja con los mismos archivos, ofreciendo una trazabilidad del cambio: quién, cuándo, qué, ... Además, permite trabajar con distintas ramas, y generar entregas, etc.

   En este grupo encontramos dos tipos de SCM:
   a. **SCM centralizados**: Subversion (SVN), ClearCase, Perforce, Concurrent Versions System (CVS), Microsoft Visual SourceSafe
   b. **SCM distribuidos** como Git y Mercurial. A diferencia de los anteriores, permiten trabajar de forma descentralizada y ofrecen al usuario su propio sistema de control de versiones. Son más complejos que los anteriores ya que se tiene que interactuar con el sistema de control central y el individual, pero ofrecen más funciones y su uso está altamente extendido. Existen distintos productos Software con funcionalidades de plataformas de software colaborativo o forja basados en Git: Github, Gitlab, BitBucket, Gerrit

3. Un **repositorio de artefactos** donde almacenar archivos binarios o paquetes compilados (.jar, .war, .ear, paquetes npm, …) generados del código construido con una herramienta de Construcción. (1). Ejemplos de Repositorios de Artefactos: Nexus, Artifactory, Apache Archiva.

4. Un **servidor de integración continua**: es el orquestador de CI, permite planificar e implementar multitud de tareas, simplificando los procesos involucrados en el ciclo de vida de desarrollo de un proyecto. Es decir, permite programar, por ejemplo, la construcción del software y la ejecución de las pruebas, mostrando los resultados de forma centralizada y realizando acciones en base a los resultados. Ejemplos: Jenkins, Hudson, Bamboo. Destacar además que la tendencia actual es que las herramientas basadas en Git incorporen módulos con funcionalidad CI como: Github Actions, GitLab CI o Bitbucket Pipelines.
5. Una **herramienta para el análisis estático de la calidad del código**. Ejemplo: SonarQube, SonarCloud, Sonarlint (En local)
6. Herramientas de pruebas
   a. Pruebas unitarias Junit, TestNG,
   b. Pruebas de Regresión Selenium, Cypress, (opcional)
   c. Pruebas de integración de Servicios Web Soapui, Postman (opcional)

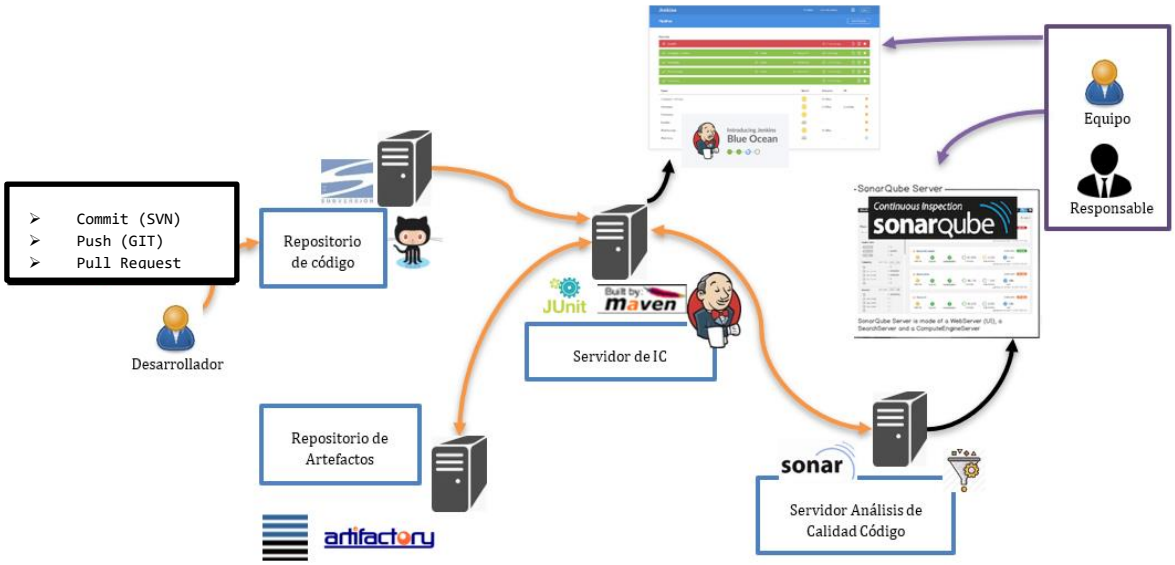

Imagen 1 Interacción entre herramientas IC y el equipo de desarrollo

2 CONCEPTOS IC

JOBS

Un Job en Jenkins es el elemento que se utiliza para configurar tareas. La ejecución de un Job se denomina build o construcción. No confundir con la fase build de construcción de nuestro proyecto. Los Jobs del servidor de CI, se pueden ejecutar manualmente, periódicamente y ejecutar automáticamente a través de disparadores o triggers, por ejemplo: COMMIT: al subir un desarrollador código a GIT.

PIPELINE

Un Pipeline permite definir los pasos del ciclo de desarrollo a ejecutar. Este término también es usado en DevOps haciendo referencia a los pasos de un proceso de desarrollo y de operación. Los pipelines tienen una serie de stages o estadios y estos permiten una serie de steps o pasos. Cada Step permiten incorporar scripts y programar un conjunto de acciones diferenciadas.

QUALITY GATES

El concepto de Qualtiy Gate está asociado a la herramienta de análisis estático de código SonarQube. A través de ella, se puede definir una serie de controles de calidad, en base a los resultados obtenidos del análisis de código.

3 CI/CD - INTEGRACIÓN CONTINUA / ENTREGA CONTINUA / DESPLIEGUE CONTINUO

Podríamos decir que la implantación de CI tiene diferentes niveles de madurez, son también consideradas prácticas DevOps:

- **Integración continua (*Continuous Delivery*):**
  para asegurar que el código subido al control de versiones no rompe nada.

- **Entrega Continua (*Continuous Delivery*):**
  se automatiza el despliegue en un entorno como preproducción y se automatizan las pruebas de aceptación.
- **Despliegue Continuo *(Continuous Deployment):*** Automatiza todos los pasos. Si se pasan todas las pruebas automatizadas en el anterior entorno, entonces se despliega automáticamente en Producción.

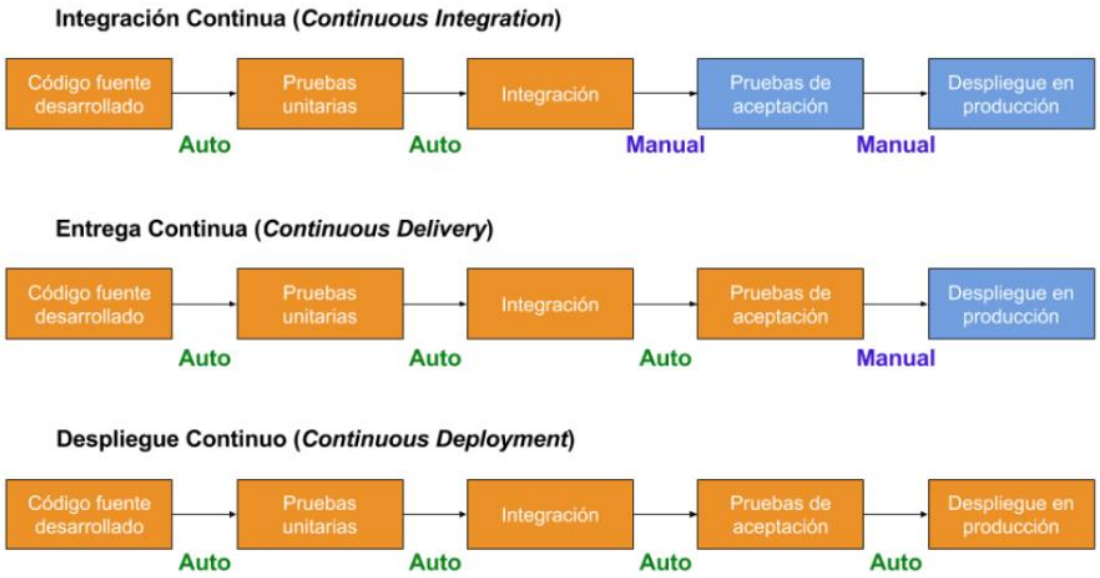

Imagen 2 Comparativa CI/CD

CaC Configuration as Code

El paradigma *AS CODE* se basa en ser capaz de reproducir y restaurar un entorno a través de instrucciones ya establecidas a través de instrucciones y automatización, gestionadas como código.

Configuration as Code trata de establecer la configuración de una determinada herramienta o entorno, través de instrucciones fácilmente interpretables por un humano y que puede ser ejecutadas de forma automática.

4 DEVOPS

DevOps (acrónimo inglés de development -desarrollo- y operations -operaciones-) es un conjunto de prácticas que agrupan el desarrollo de software (Dev) y las operaciones de TI (Ops). Su objetivo es hacer más rápido el ciclo de vida del desarrollo de software y proporcionar una entrega continua de alta calidad.

IaC Infrastructure as Code

Asociado a DevOps, se encuentra el concepto **Infraestructura como código (IaC):** hace referencia al conjunto de herramientas de desarrollo de operaciones utilizadas para configurar y actualizar los componentes de la infraestructura para garantizar un entorno controlado. Trata las infraestructuras como un software y esta configuración se versiona a través de las herramientas de SCM.

Servidores Inmutables

El concepto de servidor inmutable significa que, una vez montados los servidores, estos no se cambian sobre el entorno. Si tenemos que cambiar de servidor lo que haremos será montar ese nuevo servidor en un nuevo entorno. Gracias a IaC realizar esta operación será relativamente rápido y seguro.

HERRAMIENTAS IAC

Terraform, Chef, Puppet, Ansible, AWS CloudFormation, Pulumi, Vagrant

5 DEVSECOPS

Es un aumento de DevOps, incluyendo prácticas de seguridad a lo largo de todo el ciclo.

Imagen 3 Tareas Sec en DevOps

Asociado a DevSecOps, existe el concepto Seguridad como código (SaC): cuando se incorporan la seguridad al desarrollo y a la operación, de herramientas informáticas o aplicaciones.

DEVSECOPS – HERRAMIENTAS

Existen muchas herramientas que dan soporte a las funcionalidades DevSecOps, se indican en la imagen y en la tabla siguiente, las más relevantes:

PLAN	CODE	BUILD	REPOS	CI	TEST	DEPLOY & RELEASE	RUNTIME	OPERATE	MONITOR
JIRA SLACK TAIGA REDMINE	SVN GITHUB GITLAB BITBUCKET	GRADLE MAVEN ANT MSBUILD	NEXUS ARTIFACTORY	JENKINS BAMBOO CIRCLECI GITLAB ACTIONS GITHUB ACTIONS AMIGO GO CD NEVERCODE CODESHIP	JUNIT SONARQUBE SOAPUI JMETER SELENIUM CYPRESS	CLOUDBEES JENKINS RUNDECK KUBERNETES ANSIBLE PUPPET CHEF CAPISTRANO OPENSTACK OPENNEBULA CLOUDSTACK	OPENSHIFT DOCKER	ANSIBLE PUPPET CHEF KUBERNETES	APPDYNAMICS SPLUNK BLUEPILL MCOLLECTIVE GRAHITE GRAFANA GRAYLOG CFENGINE ZABHIS NAGIOS
				SEC					
JIRA	GITROD CHECKMARX		SONATYPE NESSUS TANIUM	PLUGINS DEPENDENCY CHECK – JENKINS SPLUNK	VULNERABILITIES (SONARQUBE) OWASP RULES (SONARQUBE) FINDBUGS	INSPECT BEAKER		OWASP ZAP	FORTYFY SPLUNK FIREEYE METASPOILT EVIDENT.IO

Tabla 1 herramientas DEVSECOPS

6 SRE

SRE (Site Reliability Engineering), creado por Ben Treynor en Google, es un sistema en el que el desarrollo controla las operaciones, DevOps tiene como objetivo cerrar la brecha entre el desarrollo y las operaciones al alinear culturalmente sus tareas, objetivos e iniciativas; SRE coloca al equipo de desarrollo a la cabeza de toda la iniciativa. DevOps y SRE comparten los mismos objetivos finales

7 ESTRATEGIAS DE DESPLIEGUE

BLUE/GREEN Deployment:

En esta estrategia se tiene un entorno de producción oculto, en el que se despliega la nueva versión del código. Tras las pruebas en el entorno oculto, se procede a mover todo el tráfico de usuario mediante el balanceador de carga. El beneficio es un Zero un despliegue sin pérdida de servicio (*Zero Downtime)*

Dog Fooding:

Técnicamente similar al anterior, pero se deja acceder a las nuevas funcionalidades a los empleados internos. Para recoger feedback interno antes de abrir a usuarios finales.

CANARY deployment:

En vez de abrir las nuevas funcionalidades a todos los usuarios finales, se hace de forma progresiva y controlada. Por ejemplo, se abre la nueva funcionalidad a sólo el 5% de usuarios finales y toma feedback antes de seguir abriendo progresivamente al resto de usuarios.