



UNIDAD 4

Software Configuration Management

Gestionando cambios al software



Software Configuration Management

En esta unidad

- Conceptos de SCM
- Identificación
- Control de cambios
- Auditorias
- Build & Release
- Integración Continua
- Despliegue Continuo
- DevOps

Problemas en el desarrollo de Software

“En mi maquina me funciona”

“Actualizamos la API. No sabíamos que todavía había usuarios con v1.0”

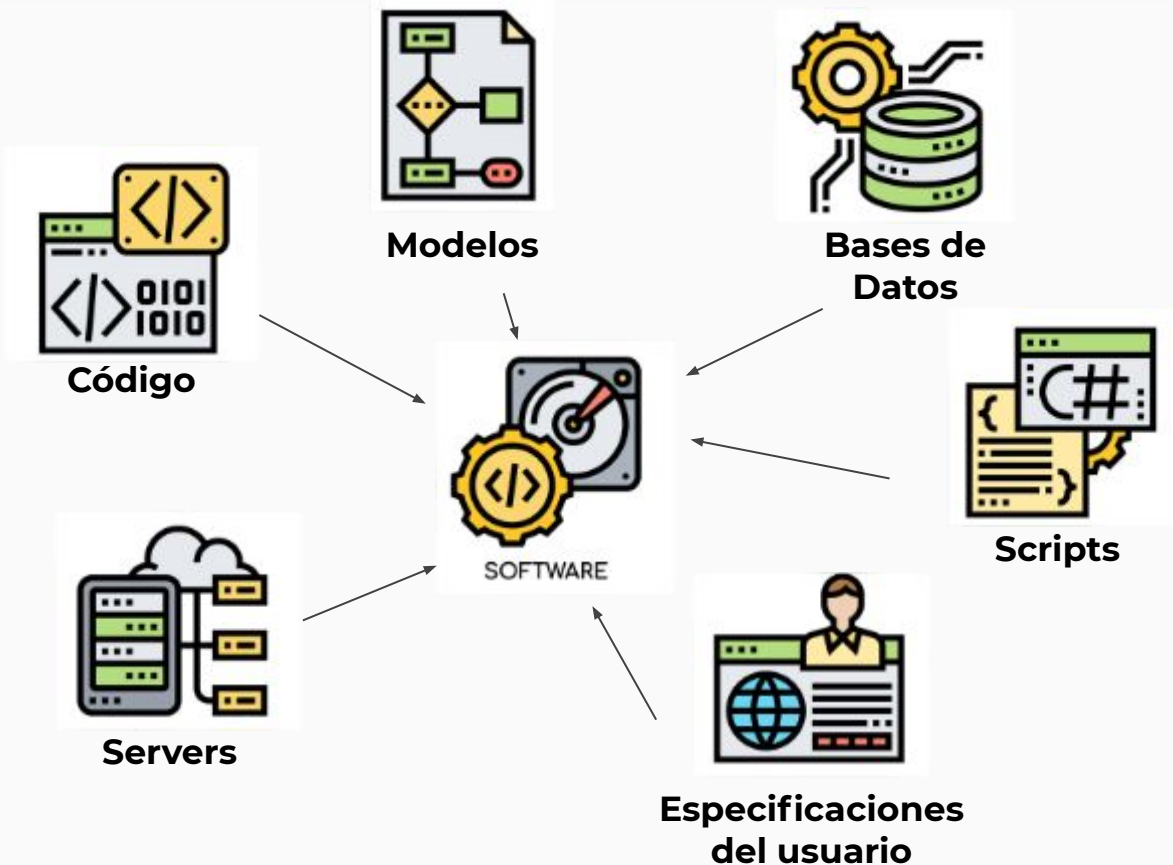
“Armar el entregable toma unos días”

“En qué clientes estaba el bug?”

“Pasa que probaste en el entorno de testing y no en desarrollo...”



¿Pero qué es el Software?



Ítem de Configuración

Es cualquier elemento involucrado en el desarrollo del producto y que está bajo el control de la gestión de configuración

Para cada ítem se conoce información sobre su configuración (nombre, versión, fecha de creación, autor, entre otros)

Configuración

Es el conjunto de todos los componentes fuentes que son compilados en un ejecutable consistente.

Todos los componentes, documentos e información de su estructura que definen una versión determinada del producto a entregar

¿Qué es la gestión de la configuración de software (SCM)?

Es una disciplina orientada a **administrar la evolución** de:

- Productos
- Procesos
- Ambientes

Su **propósito** es **establecer y mantener la integridad** de los productos del proyecto de software a lo largo del ciclo de vida del mismo

Involucra para una configuración:

- **Identificarla** en un momento dado
- **Controlar cambios** sistemáticamente
- **Mantener su integridad** y origen

Acompaña la actividad de cambio con actividades de control



SCM en la vida real

Escenario:

Una compañía los contrató porque todo su equipo de desarrollo renunció y necesitan terminar un proyecto de un cliente.

Solo les dieron el ordenador del líder técnico y un par de documentos.

¿Qué hacemos?

Paso a Paso

Identificar código

- Buscar fuentes
- Chequear repositorios
- Entender la arquitectura



Código

Probablemente lo que intentaremos hacer al principio, y cada uno de diferentes maneras, es tratar de encontrar la CONFIGURACIÓN del software que ayude a replicar la instalación de producción.

Aquí es donde se marca la primera actividad de SCM

1. Identificación de la configuración

Definir qué elementos (ICs) estarán controlados por la gestión de configuración

- No todo necesita estar bajo SCM
- Es necesario contar o definir con guías para qué elementos son parte o no.

Define momentos o condiciones para establecer una línea de base o bien para liberarla (es decir llevarla a otro ambiente, también conocido como “release”)

Identificar ambientes

- Tratar de instalar una versión
- Identificar Bases de datos, servers, configuraciones



Servers

Identificar Documentos

- Buscar especificaciones
- Entender cambios



Especificaciones del usuario

Identificación de la Configuración

Qué actividades realiza

Define qué elementos (ICs) estarán controlados por la gestión de configuración

- No todo necesita estar bajo SCM
- Es necesario contar o definir con guías para qué elementos son parte o no.

Define momentos o condiciones para establecer una línea de base o bien para liberarla (es decir llevarla a otro ambiente, también conocido como “release”)

Ítem de Configuración

Deberá hacerse seguimiento de atributos importantes de un IC:

- Nombre
- Versión
- Autor / Revisores
- Módulos o ítems relacionados
- Última modificación
- Tipo de IC (código, scripts, diagramas, requerimientos, etc.)

Clasificación de ICs de **Proceso** o de **Producto**

Proceso

- Plan de CM
- Propuestas de Cambio
- 10 Riesgos principales
- Plan de desarrollo
- Requerimientos
- Plan de Calidad
- Lista de Control de entrega
- Formulario de aceptación
- Planes de fases
- Registro del proyecto

Producto

- Plan de Integración
- Prototipo de Interface
- Manual de Usuario
- Arquitectura del Software
- Estándares de codificación
- Casos de prueba
- Código fuente
- Gráficos, íconos, ...
- Instructivo de ensamble
- Programa de instalación
- Documento de despliegue

Paso a Paso - Identificamos una configuración -> ¿Y ahora?

Tenemos ahora una configuración identificada.

Para seguir con el proyecto, debemos establecer un orden de para hacer los cambios en el software, ¿Pero como?



Reqs



Bugs



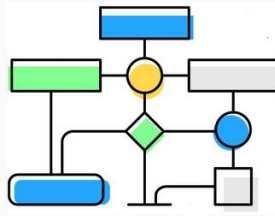
Changes



SOFTWARE



SCCB
Software
Configuration
Control Board



**Change
Management
Flow**



**Aprobado /
Rechazado**

2. Control de la configuración

Asegurar que los ítems de configuración mantienen su integridad ante los cambios a través de:

- La identificación del propósito del cambio
- La evaluación del impacto y aprobación del cambio
- La planificación de la incorporación del cambio
- El control de la implementación del cambio y su verificación

Actividad : Control de la Configuración de SW

Objetivo

Asegurar que los ítems de configuración mantienen su integridad ante los cambios a través de:

- La identificación del propósito del cambio
- La evaluación del impacto y aprobación del cambio
- La planificación de la incorporación del cambio
- El control de la implementación del cambio y su verificación

Actividades

El control de la configuración consiste en establecer un procedimiento de control de cambios y controlar el cambio y la liberación de ICs a lo largo del ciclo de vida.

“Lo único constante es el cambio”

Heráclito

- Cambios en el negocio
- Cambian los clientes
- Cambian en las condiciones
- Cambian en las tecnologías
- Cambia la organización
- Cambios.... que se terminan reflejando en el proyecto y en el código

Control de cambios en código fuente

- Normalmente es la función más visible e importante de la gestión de configuración ya que los problemas de integridad se presentan principalmente debido a los cambios
- En el desarrollo es común que los elementos deban ser modificados debido a cambios en los requerimientos, defectos o mejoras

Conceptos de Control de Cambios

- Se necesita controlar los cambios a los ítems de configuración estableciendo diferentes niveles de control
 - ◆ Control de versiones
 - ◆ Control de ítems que forman parte de líneas base
- La complejidad del control de cambios surge de las necesidades de modificar archivos que son compartidos por diferentes personas y proyectos
 - ◆ Dos o más personas necesitan modificar el mismo artefacto
 - ◆ Dos o más proyectos trabajan sobre el mismo artefacto
- Lo más importante: **analizar el impacto** del cambio, **autorizarlo** (o no) y realizar el **seguimiento** de los pedidos de cambio **hasta su cierre**

La Solicitud de Cambio


- Debe estar registrada formalmente. Por ello la definición de un Formulario de Solicitud de Cambio es parte de la definición del proceso de CM.
- Esta solicitud tiene por objetivo registrar el cambio propuesto, quien lo propuso, la razón de porque el cambio fue solicitado y la urgencia del mismo (Establecida por el solicitante)
- También se utiliza para registrar la evaluación del cambio, complejidad, el análisis de impacto y las recomendaciones de los equipos.


Ejemplo de Proceso genérico de administración de cambios


- Se solicita el cambio
 - Puede ser solicitado tanto por un usuario como un desarrollador. En general, solo se aceptan cambios de personas autorizadas a pedirlo.
- El cambio es evaluado y comparado contra los objetivos del proyecto
- Un grupo de asesores evalúa el cambio y lo aprueba o rechaza
 - Se suele evaluar el impacto del cambio tanto en el producto como en la calendarización del proyecto
 - Si se rechaza, se puede dejar como parte de una “Segunda Fase” del proyecto, o incorporarlo en una segunda iteración de desarrollo.
- Si es aceptado, se replanifican las tareas a realizar y se le asigna recursos para resolverlo
- El cambio implementado deberá ser revisado en auditorias.

La complejidad de la administración del cambio varía de acuerdo al proyecto. En proyectos pequeños, el pedido de cambio puede ser informal y desarrollarse rápidamente mientras que en proyectos más complejos puede haber comités de varias personas que evalúen el cambio



Paso a Paso - Seguimiento y control de la configuración

 lucass-guru released [Versión de pruebas.](#) at [dieguitux/guru-app-tuguru-ios](#) 4 days ago

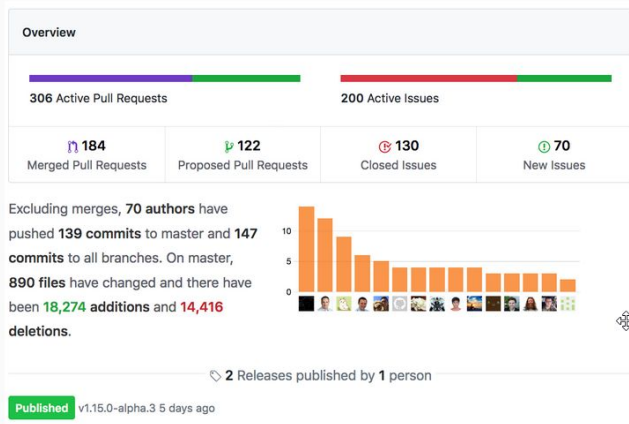
 Source code (zip)







 lucass-guru pushed to [dieguitux/guru-app-tuguru-ios](#) 4 days ago




3 commits to [develop](#)

-  [b26ad57](#) version 1.0.78 (64)
-  [fea4731](#) added banners

[1 more commit »](#)



Default branch	
 master	Updated an hour ago by k8s-ci-robot ✓
Active branches	
 release-1.14	Updated 2 days ago by k8s-ci-robot ✓
 feature-rate-limiting	Updated 3 days ago by k8s-ci-robot ✓
 release-1.13	Updated 3 days ago by k8s-ci-robot ✗
 release-1.12	Updated 4 days ago by k8s-ci-robot ✓
 release-1.11	Updated 12 days ago by Anago GCB ✗

974 Open	✓ 46,618 Closed	Author	Labels	Projects	Milestones	Reviews	Assign
 Fix gollint failures of client-go/tools/auth client-go/tools/portforward + ciol-cic yes kind/strange needs-priority release-note-none sig-auth sig-auth sig-auth #2730 opened 24 minutes ago by k8s-ci-robot							
 kubedrn: prevent PSP blocking of upgrade image prepull + kubedrn sig-auth sig-auth ciol-cic yes ciol-cic yes ciol-cic yes ciol-cic yes ciol-cic yes ciol-cic yes ciol-cic yes ciol-cic yes sig-auth sig-auth sig-auth sig-auth sig-auth sig-auth sig-auth sig-auth #2730 opened 5 hours ago by k8s-ci-robot							
 Fix suggested cmd usage kubectl exec + ciol-cic yes ciol-cic yes kind/strange needs-ci-no-fail needs-priority release-note-none sig-auth sig-auth sig-auth #2730 opened 5 hours ago by k8s-ci-robot							

3. Status & Accounting de la configuración

Registrar y reportar la información necesaria para administrar la configuración de manera efectiva

- Listar los ICs aprobados
- Mostrar el estado de los cambios que fueron aprobados
- Reportar la trazabilidad de todos los cambios efectuados al baseline

Debe poder contestar “¿Qué cambios se realizaron al sistema?”

- Cuándo cambió?
- Quién lo cambió?
- Qué cambió?
- Alcance del cambio
- Quién aprobó el cambio?
- Quién solicitó el cambio?

Se debe informar periódicamente con reportes a todos los grupos afectados

Seguimiento y control de la configuración - Ejemplo: Github Insights

Overview

306 Active Pull Requests

200 Active Issues

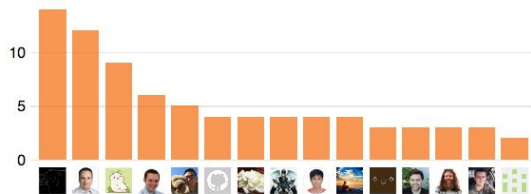
184
Merged Pull Requests

122
Proposed Pull Requests

130
Closed Issues

70
New Issues

Excluding merges, **70 authors** have pushed **139 commits** to master and **147 commits** to all branches. On master, **890 files** have changed and there have been **18,274 additions** and **14,416 deletions**.



2 Releases published by 1 person

Published v1.15.0-alpha.3 5 days ago

Default branch

master Updated an hour ago by k8s-ci-robot

Active branches

release-1.14 Updated 2 days ago by k8s-ci-robot

feature-rate-limiting Updated 3 days ago by k8s-ci-robot

release-1.13 Updated 3 days ago by k8s-ci-robot

release-1.12 Updated 4 days ago by k8s-ci-robot

release-1.11 Updated 12 days ago by Anago GCB

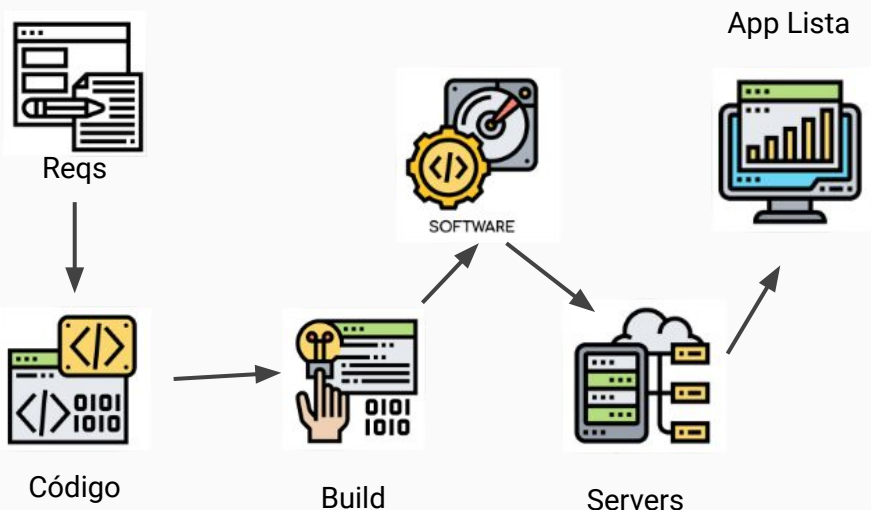
974 Open 46,618 Closed Author Labels Projects Milestones Reviews Ass

Fix golint failures of client-go/tools/auth client-go/tools/portforward • cncf-cla: yes
kind/cleanup needs-priority release-note-none sig/api-machinery sig/auth size/XS
#77793 opened 25 minutes ago by SataQiu

kubeadm: prevent PSP blocking of upgrade image prepull • approved area/kubeadm
cncf-cla: yes do-not-merge/hold kind/bug priority/important-longterm release-note
sig/cluster-lifecycle size/XS
#77792 opened 6 hours ago by neolit123

Fix suggested cmd usage kubect! exec • area/kubect! cncf-cla: yes kind/cleanup
needs-ok-to-test needs-priority release-note-none sig/cli size/XS
#77791 opened 6 hours ago by neolit123

Paso a Paso - Controlamos cambios, Good to Launch



Tenemos el código, hicimos los cambios pedidos.

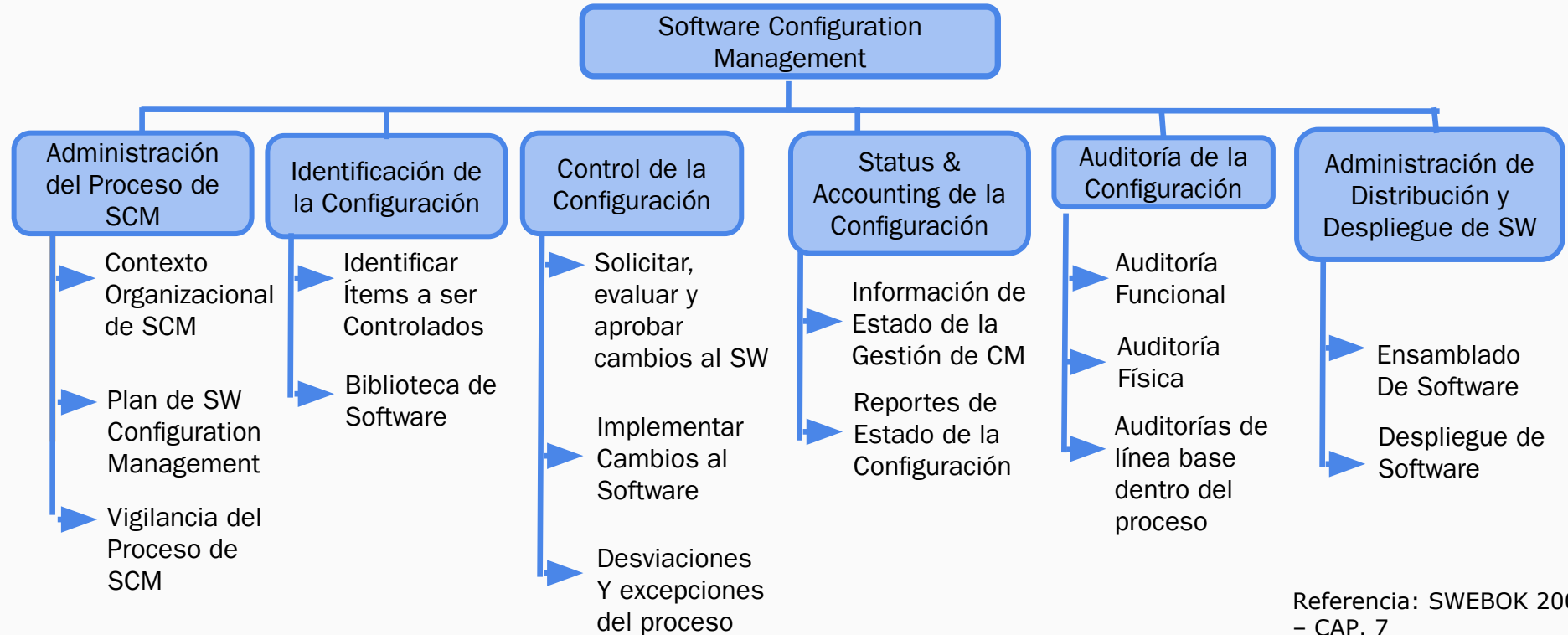
Debemos construir nuestra configuración y habilitarla para que pueda ser utilizada por los usuarios (A.K.A. Producción)

4. Administración de Distribución y Despliegue de SW

- Asegurar la construcción exitosa del paquete de software, basada en los ICs requeridos para la funcionalidad a entregar, para luego liberarlo en forma controlada a otros entornos ya sea de pruebas, producción, usuario final, etc.
- Se divide en 2 partes:
 - Software Building
 - Release Management

Además del ejecutable o paquete de software, comprende la administración, identificación y distribución de un producto.

Software Configuration Management de acuerdo a SWEBOK



Administración del proceso de SCM

¿Qué es?

La etapa previa al arranque del proyecto, donde se debe comprender el contexto organizacional.

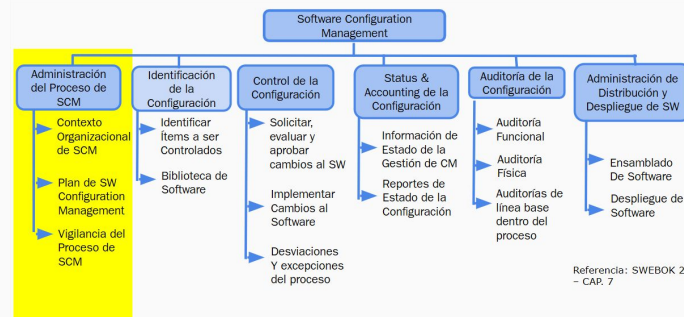
Algunas preguntas a realizar:

- ¿Existe ya algún proceso de SCM en la organización? ¿Qué establece?
- ¿Qué herramientas hay disponibles? Por ejemplo JIRA, GitLab, GitHub, etc.
- ¿Qué cosas deberemos incluir o cambiar en el plan de SCM?

¿Qué actividades incluye?

La realización del Plan de SCM con sus distintas etapas:

- Identificación de la configuración
- Procesos de Control de cambios
- Auditorías de software
- Release management, entre otras
- Políticas de manejo de ramificaciones o branches para que no crezcan



Auditoría de la Configuración

Objetivo

Realizar una verificación del estado de la configuración a fin de determinar si se están cumpliendo los requerimientos especificados

La auditoría puede ser ejecutada con diferentes niveles de formalidad:

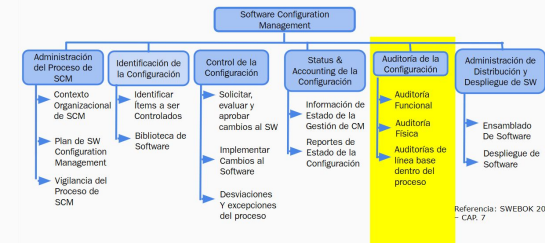
- Revisiones informales basadas en checklists
- Pruebas exhaustivas de la configuración que son planificadas

Tipos de Auditoría

Funcional, que verifica el cumplimiento de requerimientos

Física, que verifica la configuración del producto en cuanto a la estructura especificada

De Proceso, que verifica se haya cumplido el proceso de SCM

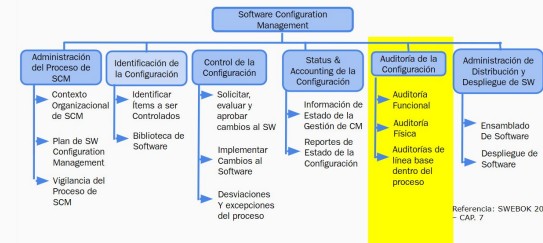


Auditoría Funcional

En las auditorías funcionales, se verifica que una configuración dada cumpla con alguna especificación de requerimientos.

En software puede ser efectuada a través de pruebas funcionales o técnicas (testing)

Se puede verificar los resultados de las actividades de testing para confirmar que son consistentes con los requerimientos especificados para una configuración dada



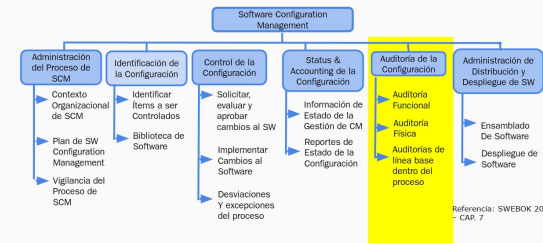
Auditoría Física

En las auditorías físicas se verifica la configuración de cada ítem para determinar si cumple con las especificaciones de configuración establecidas.

En otras palabras, se verifica el ítem para ver si es consistente con la documentación de su configuración

Ejemplo:

- Al identificar una línea base se identifica su composición (qué ítems la conforman)
- Las inconsistencias se reportan y se determina si el defecto corresponde a la configuración misma o a la documentación



A close-up photograph of a person's hand holding a purple marker, drawing on a whiteboard. The background is blurred, showing some office equipment and lights.

El proceso de SCM

Equipo distribuido para el
desarrollo de Software

¿Cómo aplicamos estos
conceptos en la “vida real”?

El problema SISOP (o de cualquier compañía con + de 1 dev)



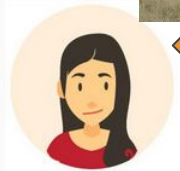
TEAM
PLANNING



Jean-Luc



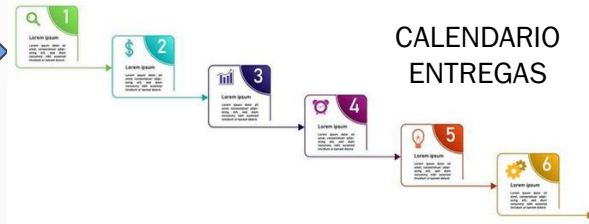
Rey



Leia



Luke



CALENDARIO
ENTREGAS



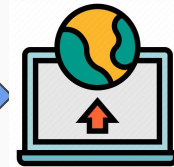
SOFTWARE



SOFTWARE RELEASE



Software en
PRODUCCIÓN



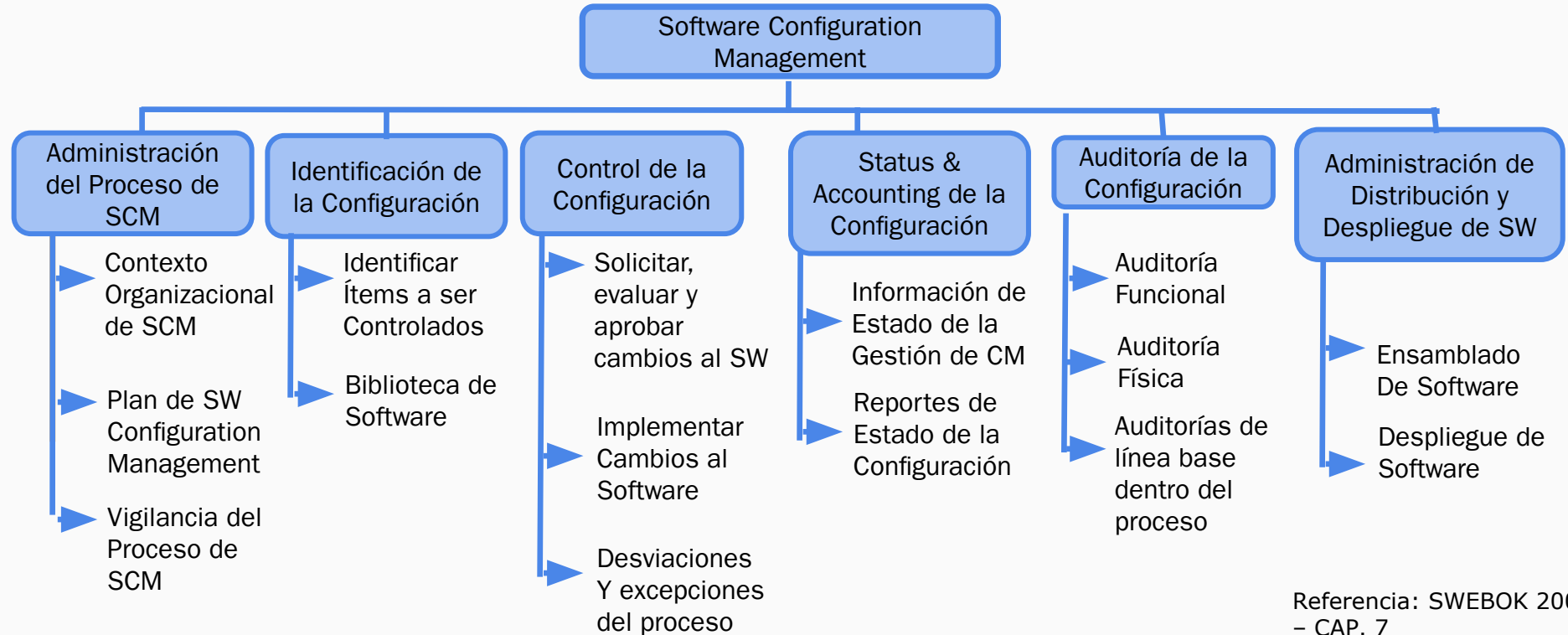
IT'S



ONE DOES NOT SIMPLY

DEPLOY TO PRODUCTION

Recordatorio : Actividades de SCM Según el SWEBOK



Paso a Paso : Planificar el proceso de SCM

Un plan de SCM define principalmente:



Lista ítems de configuración (IC), y en qué momento ingresan al sistema de CM



Define estándares de nombres, jerarquías de directorios, estándares de versionamiento



Define políticas de branching y de merging



Define los procedimientos para crear “builds” y “releases”



Define reglas de uso de la herramienta de CM y el rol del administrador de la configuración



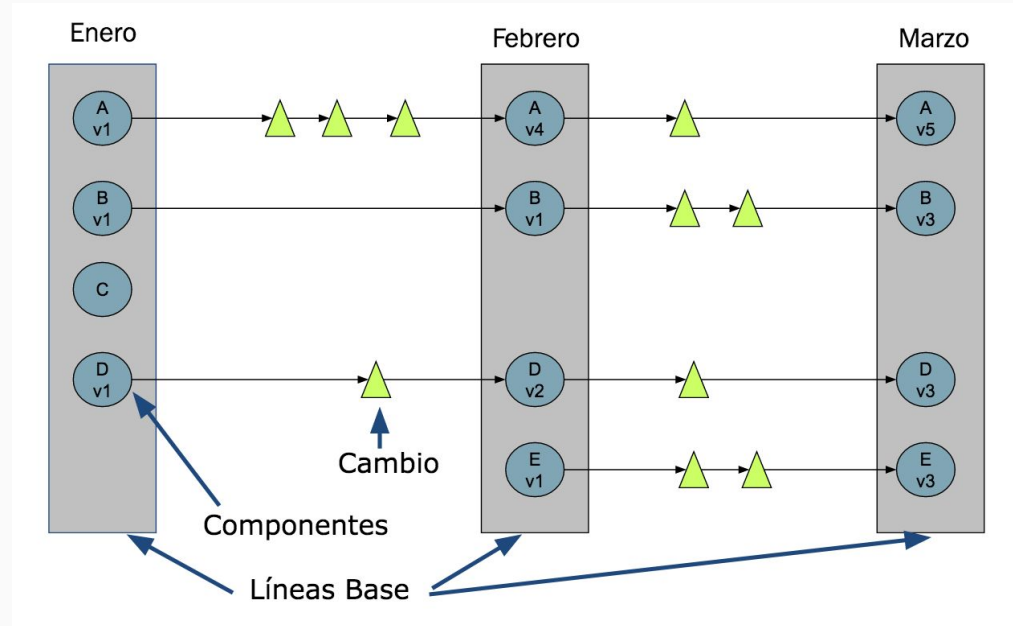
Define el contenido de los reportes de auditoría y los momentos en que estas se ejecutan

Puede ser definido por proyecto o a nivel organizacional (y luego realizar una adaptación al proyecto)

Línea de Base (Baseline)

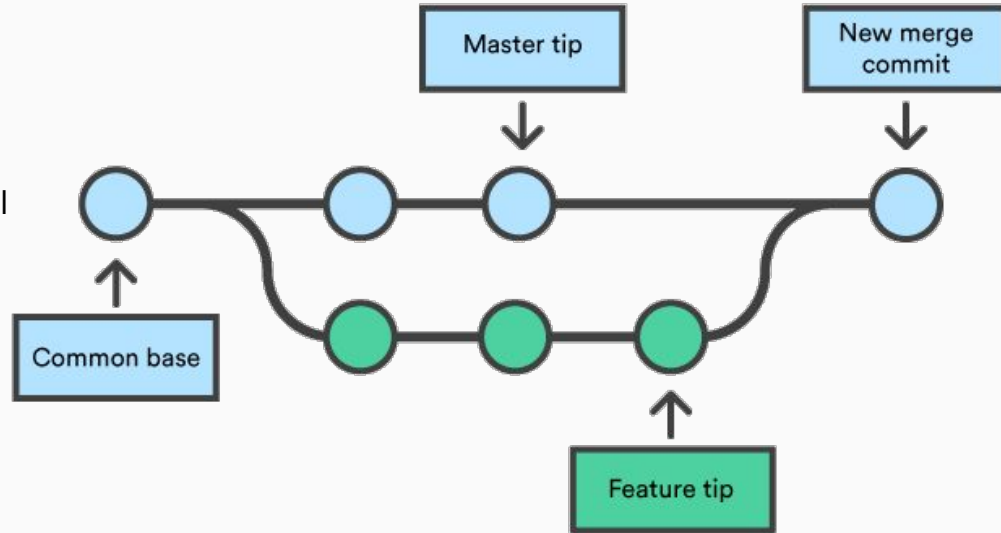
Representa un estado de la configuración de un conjunto de ítems en el ciclo de desarrollo, que puede tomarse como punto de referencia para una siguiente etapa del ciclo.

Se establece porque se verifica que esta configuración del ítem o conjunto de ítems satisface (n) algunos requerimientos funcionales o técnicos.

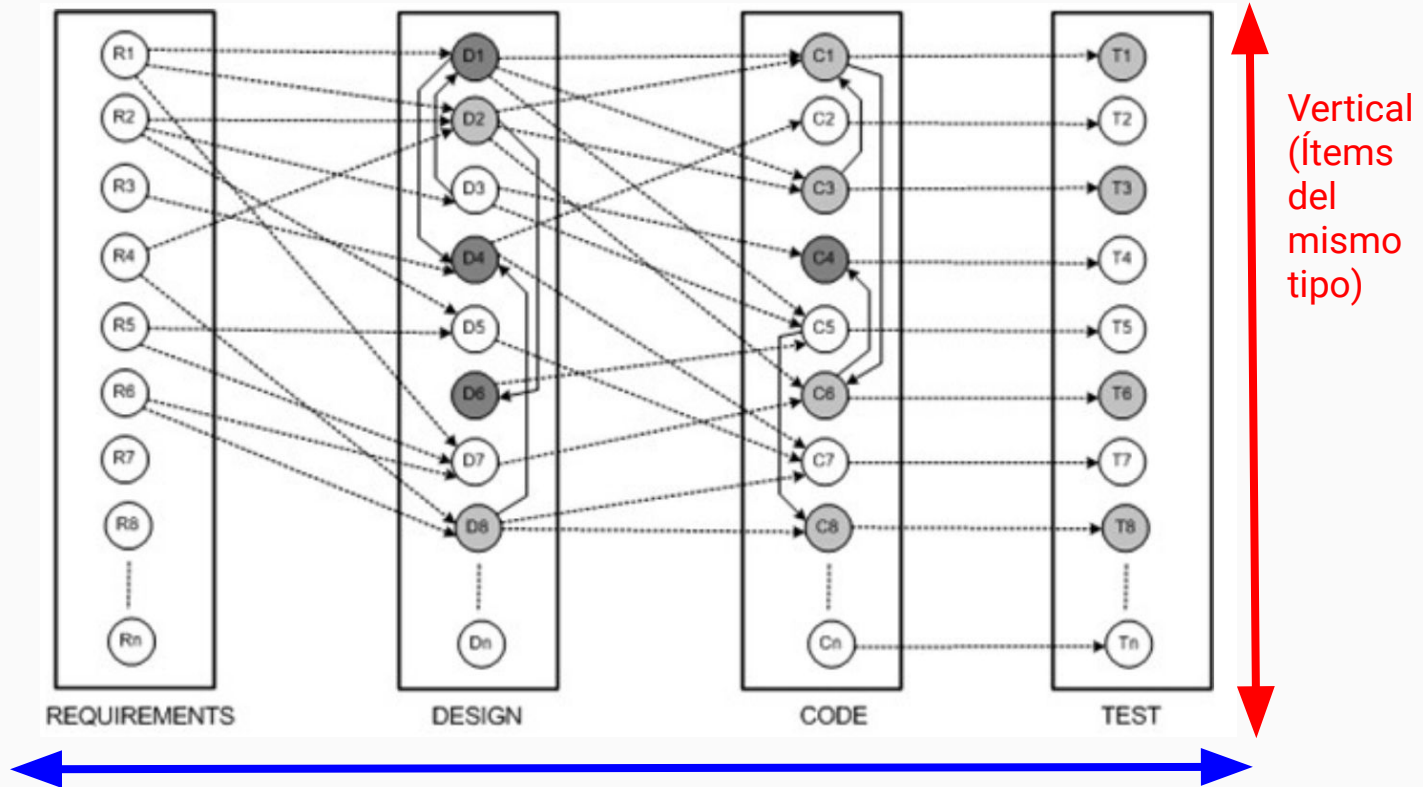


Rama (Branch)

- Es la acción de crear líneas de desarrollo separadas.
- Estas líneas utilizan las líneas base de un repositorio existente como punto de partida
- Permite a los miembros del equipo trabajar en múltiples versiones de un producto, utilizando el mismo set de Ítems de Configuración



SCM : Trazabilidad entre ítems de configuración



Paso a paso : Proceso de SCM definido

Podemos decir que tenemos un proceso definido cuando (Por ejemplo):

Está definido cómo se registran cambios de software

Se establecieron los roles que aprueban o rechazan el cambio

Están identificados los repositorios de los artefactos

Hay establecida una política de trabajo sobre esos repositorios

Están identificados los momentos en los que se realizan las líneas base

Los cambios son gestionados completamente, desde el ingreso hasta su puesta en producción



Jean-Luc
analiza y decide si se hace o no junto con el PO



como repositorio de código



como repositorio de paquetes



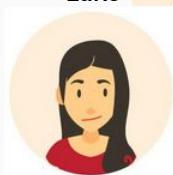
como gestor de paquetes



Luke

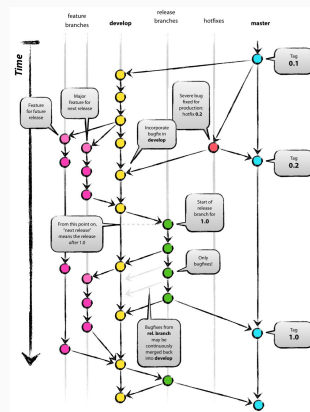


Rey



Leia

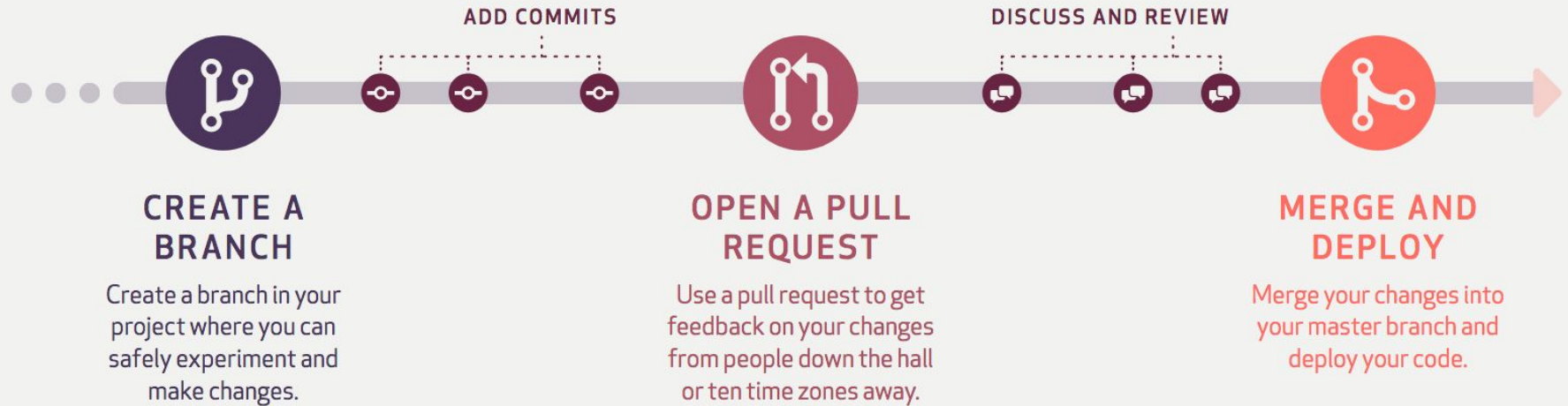
El equipo conoce las reglas de desarrollo



gitflow como flujo de trabajo



GitHub-Flow: Ejemplo de gestión de cambios en el código



<https://guides.github.com/introduction/flow/>

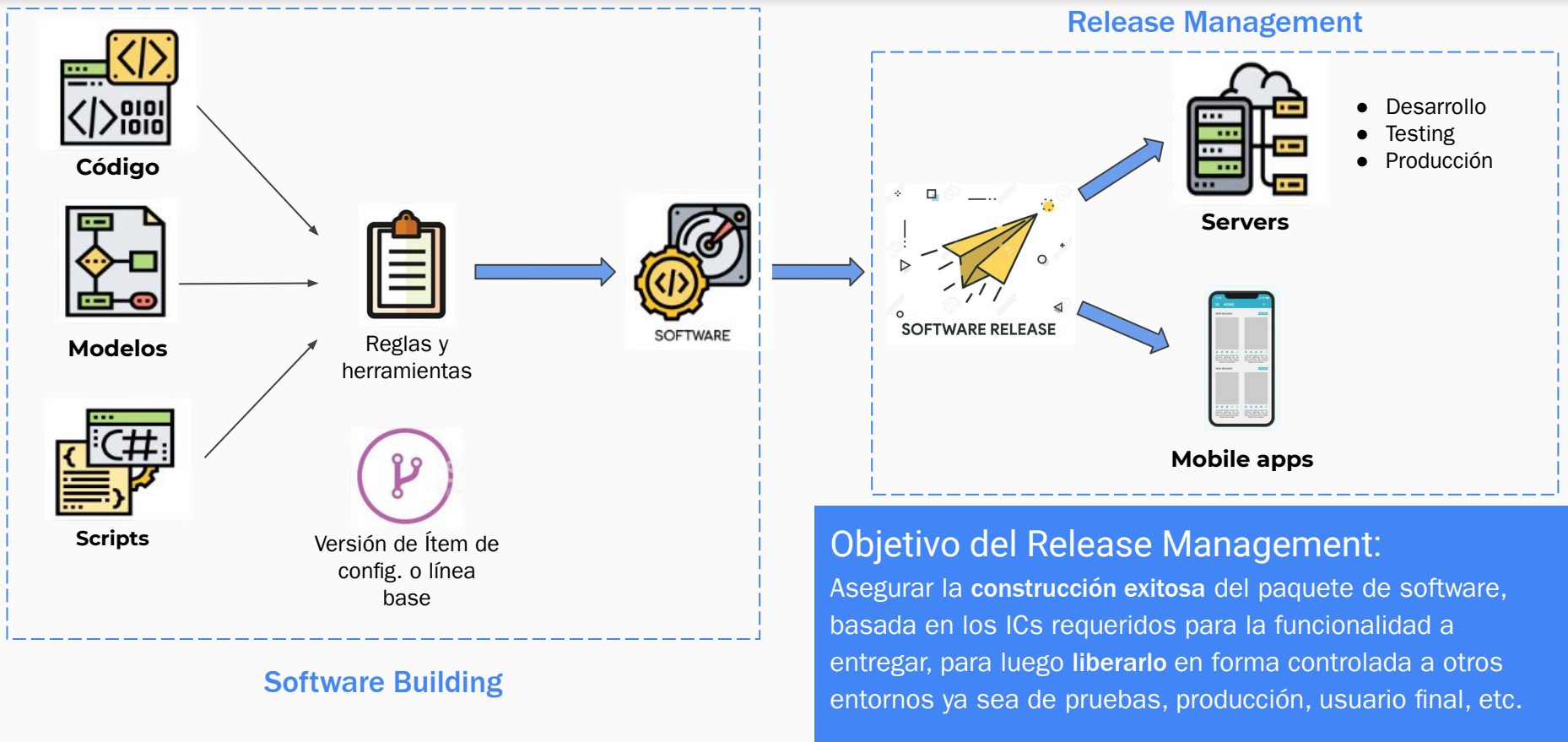
Es una simplificación del post original: [A Successful Git Branching Model](#)

Distribución y Despliegue (Release & Deploy) del Software

AND
NOW



Distribución y Despliegue (Release Management) del Software



Conceptos de Software Building

Tipos de Builds

- **Local** builds: El developer lo realiza localmente en su entorno de desarrollo, corre Pruebas Unitarias (UT)
- **Integration** builds: su objetivo es generar el entorno completo para pruebas de integración
- **Nightly** builds: su objetivo es ejecutar la construcción en forma diaria y generar reportes con información sobre estabilidad, tiempo de build, etc.
- **Release** builds: Se disparan cuando bien un administrador decide crear una nueva versión a ser liberada, o por el mismo sistema de integración si se utiliza el modo de deployment continuo

Acerca de los builds :

- Deben ser automáticos
- Deben permitir la generación de reportes
 - Cada “build” genera un reporte del estado del mismo
- Beneficios para el negocio
 - *Reducen la cantidad de defectos*
 - *Mejora la reproducción de problemas y trazabilidad*
 - *Mejora la performance del equipo de desarrollo*

Conceptos de despliegue (Deploy) del Software

Una vez generada la versión o release debemos buscar el mecanismo más efectivo para hacer despliegue -deploy- controlado a los distintos usuarios.

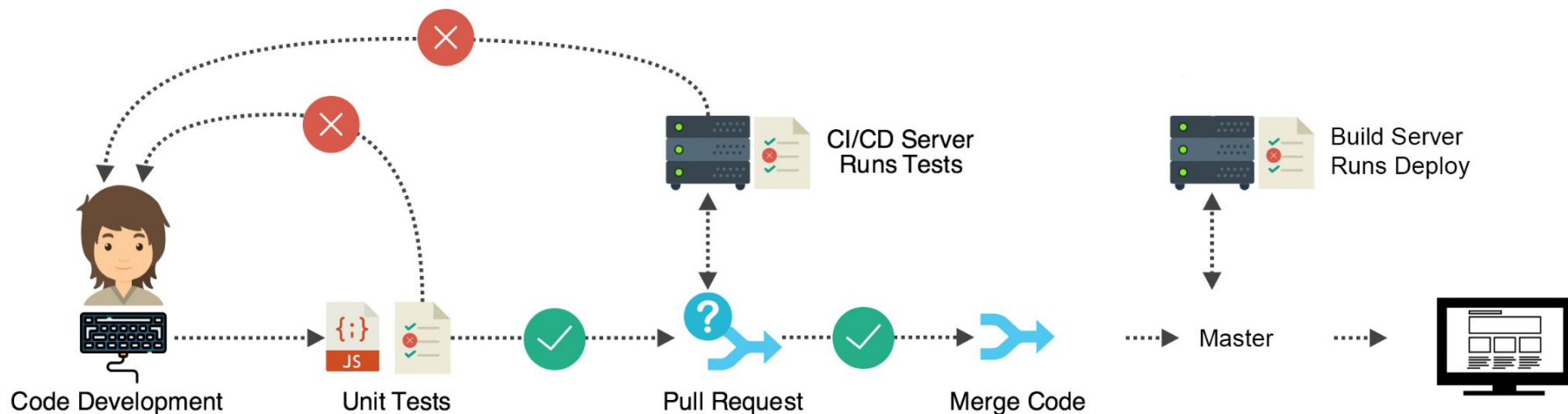
Se deberá evaluar:

- Qué usuarios deberán recibir los cambios (geográfico, premium, por segmento)
- En qué forma se deberá hacer el despliegue
 - Entornos por los que deberá pasar (testing, staging, producción, otros)
 - Procesos de Roll Forward
 - Procesos de Rollback
 - Validación de despliegue correcto / incorrecto
- Riesgos del despliegue y cómo se minimizan
- Aprobación por el negocio y/o área de QA
- Quienes realizarán el deployment de la versión definida

Build & Deployment Pipelines

Es la manifestación automatizada del proceso para llevar el software desde la aprobación del cambio de SCM hasta que llega a manos de los usuarios.

Esto incluye el camino complejo desde que se compila y construye el software, seguido del progreso a través de varias etapas de testing y deployment. Requiere colaboración entre individuos y equipos.



Distribución y Despliegue (Release & Deploy) del Software

Release Management: Objetivo

Asegurar la **construcción exitosa** del paquete de software, basada en los ICs requeridos para la funcionalidad a entregar, para luego **liberarlo** en forma controlada a otros entornos ya sea de pruebas, producción, usuario final, etc.

Se divide en 2 partes:

- Software Building
- Release Management

Release Management

Además del ejecutable o paquete de software, comprende la administración, identificación y distribución de un producto. Esto incluye, por ejemplo, el ejecutable, la documentación, notas de la versión, etc.

Los “Releases” deben incorporar cambios al sistema, producto de errores en el mismo o para incorporar nuevas funcionalidades en el sistema.

Estos cambios deben haber sido aprobados por control de Cambios.

Conceptos de Release Management

- **Versión** Es una instancia de un sistema que es funcionalmente distinta en algún aspecto de otras instancias.
- **Variante** Una instancia de un sistema que es funcionalmente igual a otras instancias, pero difiere a niveles no funcionales (Ejemplo: Mismo producto para Version Linux / Version Windows)
- **Release** Según la IEEE, el termino “Release” utilizado en este contexto, hace referencia a la distribución del Software fuera del entorno de desarrollo. En algunas organizaciones, se utilizan el concepto de un “tag” como equivalente de release.

Build Pipelines: Gated Commits

Gated Commits, también conocido como “pre-tested” check-in es una buena práctica de construcción que resuelve el problema puntual de mantener *master* en estado íntegro tras hacer un cambio.

Síntomas

El build se suele romper por problemas triviales, no relacionados a features nuevas

Problema

Los desarrolladores no saben en forma anticipada el efecto de su cambio en el repositorio central. Solo pueden enterarse **luego** de haber hecho su merge

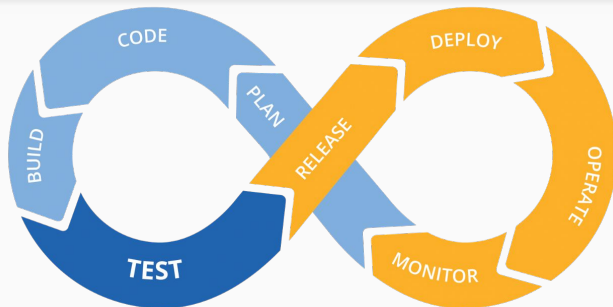
Solución

El **gated commit** es un proceso que debe ser soportado por el servidor o plataforma de build y el sistema de SCM, que **valida el cambio antes de hacer merge**

El proceso funciona de la siguiente forma:

1. El desarrollador pide un gated commit - o bien el servidor de CI lo tiene obligatorio.
2. La herramienta hace un *merge* con la rama de desarrollo principal o *master* **en un espacio temporal** aplicando los cambios del desarrollador/a.
3. El servidor de CI luego ejecuta la construcción del artefacto en el pedido de cambio con el nuevo código. Esto puede incluir compilación, ejecutar tests, e incluso despliegue etc.
4. Si el paso 3 de construcción y validación son exitosos, la herramienta permite el merge o lo realiza automáticamente, según como esté configurado. En caso de error se notifica al autor/a del cambio.

Integración Continua y Despliegue continuo (CI/CD)



Integración Continua

Realizar integración de código al menos una vez por día o más, para minimizar problemas de código.

Prácticas de la Integración Continua

- Repositorio de código único.
- Automatizar el proceso de build
- Hacer el build testeable automáticamente.
- Todo commit debe construirse por una herramienta de integración - no por el dev
- El build debe ser rápido
- entre otras

Continuous Delivery

Asociado Integración Continua (CI), *Continuous Delivery* es un conjunto de prácticas que permiten asegurar que el software puede desplegarse en producción rápidamente y en cualquier momento

Esto lo logra haciendo que cada cambio llegue a un entorno de *staging* o semi productivo, donde se corren exhaustivas pruebas de sistema. Luego se puede pasar a producción **manualmente** cuando alguien apruebe el cambio con solo apretar un botón.

Integración Continua (CI)

Origen y Definición

Acuñado por Kent Beck en XP, pero extendido por Martin Fowler en 2006.

Realizar integración de código al menos una vez por día o más, para minimizar problemas de código.

O a la inversa, cuanto más tiempo pasemos en *branches* separados o sin merge a la rama principal, más probable que tengamos que pasar más tiempo corrigiendo errores de integración, que son difíciles de encontrar.

Prácticas de Integración Continua

- Repositorio de código único.
- Automatizar el proceso de build
- Hacer el build testeable automáticamente.
- Todo commit debe construirse por una herramienta de integración - no por el dev
- El build debe ser rápido
- Se prueba en un clon de producción
- Cualquiera debe obtener fácilmente la última versión del ejecutable
- Todos deben poder ver qué pasa con el proceso de integración (transparencia)
- Automatizar el proceso de despliegue (deployment)

Integración Continua (CI)

Responsabilidades del equipo

- Hacer check-in de código frecuentemente
- No subir código roto
- No subir código no testeado
- No subir código cuando el build no pasa
- No irse a casa hasta que el build central compile.

Ventajas

- ✓ No hay integraciones de días de trabajo
- ✓ Mejora la visibilidad y la comunicación
- ✓ Atrapar errores de integración complejos en forma temprana
- ✓ Mayor rapidez para lanzar software al reducir problemas de integración
- ✓ Fin del “en mi máquina funciona”

Herramientas de CI



Jenkins



Bamboo



circleci



Travis CI



GitLab-CI



TFS



SEMAPHORE

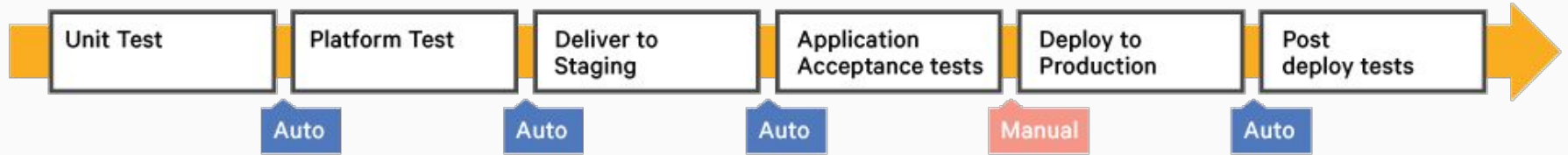


CODESHIP

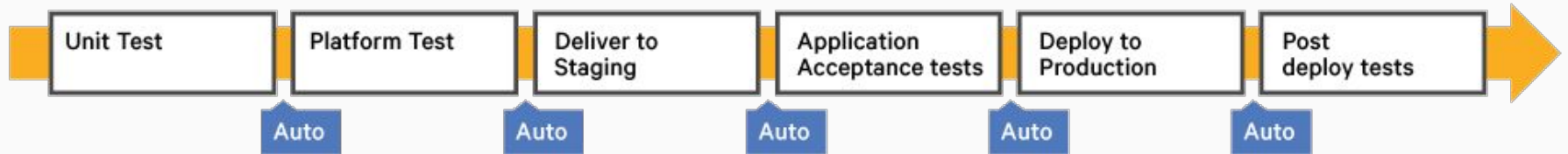
by CloudBees

No confundir CD con CD :)

Continuous Delivery

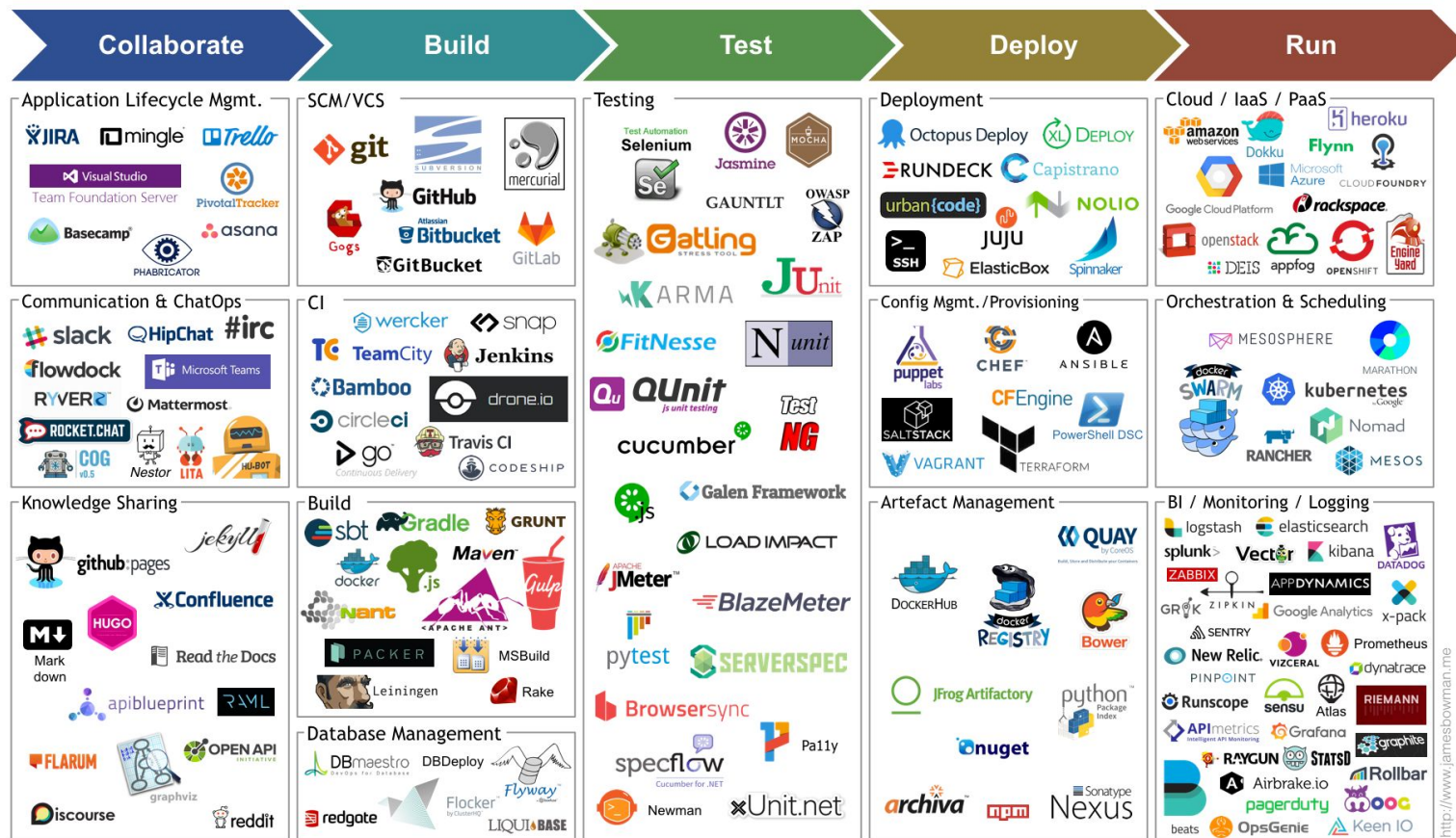


Continuous Deployment




- El continuous deployment es el paso siguiente al continuous Delivery, en donde también el despliegue a producción se realiza en forma automática por un proceso y no por personas.

El Landscape de herramientas



<http://www.jamesbowman.me>



Bonus Track: DevOps

L@s de IT y Operaciones se
vuelven developers?

O desarrollo se vuelve
operaciones?

DevOps: El problema original

En los modelos de desarrollo tradicionales...

- El que desarrolla no escribe los requerimientos
- El que desarrolla no hace testing funcional
- El que desarrolla no pasa a producción
- El que desarrolla no está de guardia

Durante décadas los operadores de IT y sistemas vivieron en plena queja con desarrolladores que no consideraban

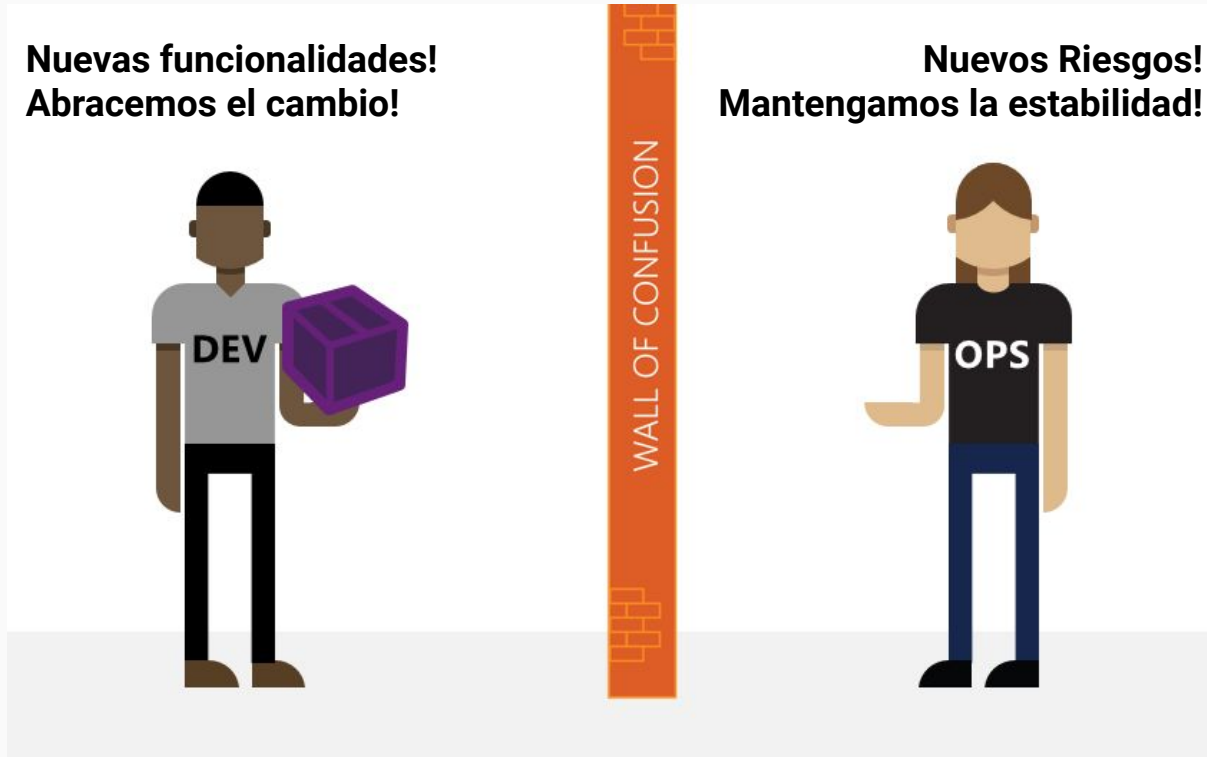
- a) que los entornos pueden tener diferencias
- b) que la puesta en producción no es gratuita
- c) cuestiones como la **escalabilidad, performance, disponibilidad, tolerancia a fallos**, entre otros.
- d) hacer software no es *fire & forget*[™], y esperaban que operaciones maneje automáticamente cualquier problema, sin conocimiento del dominio

GENERANDO FALTA DE COMUNICACIÓN Y COLABORACIÓN



DevOps: El problema original

Cuando el cliente pide un nuevo cambio... se alza la gran muralla



Origen y Definición

Se acuña el término en 2009, en una conferencia que era de infraestructura pero quería atraer desarrolladores.

[DevOps days](#) en Bélgica.

DevOps es un conjunto de prácticas destinadas a reducir el tiempo entre el cambio en un sistema y su pasaje a producción, garantizando la calidad y minimizando el esfuerzo.

Es la combinación de desarrollo y operaciones, normalmente en un equipo

Prácticas de DevOps

- Planificación del Cambio
 - Conversación y colaboración
- Coding & Building
 - Automated Build Pipelines
 - Infrastructure as Code
- Testing
 - Automated Testing
 - Chaos Testing / Fault Injection
- Release & Deployment
 - Continuous Integration
 - Continuous Delivery
- Operation & Monitoring
 - Virtualization & Containers
 - Monitoring & Alerting tools

CALMS, una filosofía de trabajo

Cultura: Ser dueños del cambio para mejorar la colaboración y comunicación.

Automatización: Eliminar el trabajo manual y repetitivo lleva a procesos repetibles y sistemas confiables, reduce error humano.

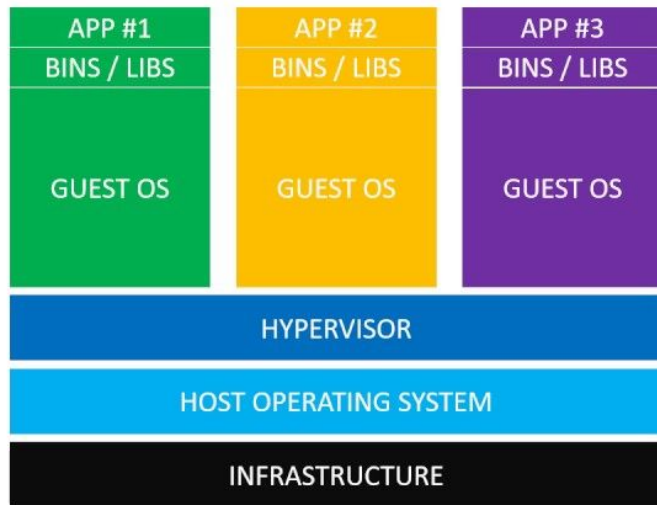
Lean: Remover la burocracia para tener ciclos más cortos y menos desperdicio

Métricas: Medir todo, usar datos para refinar los ciclos.

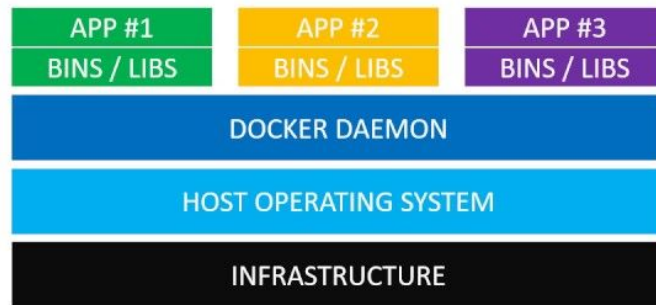
Sharing: Compartir experiencias de éxito y falla para que otros puedan aprender.



DevOps: Contenedores



Virtual Machines



Containers

DevOps: Automatizar el armado de entornos

Contenedores



Hardware: Independiente del hardware – solo respeta arquitectura. Memoria/Cpu flexible.

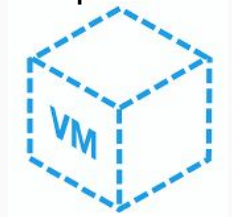
Aislamiento: Filosofía de *1 container por servicio*, aislando los servicios entre sí.

Cambios: Tienen archivos de configuración, son versionables, hay repositorios de containers.

Performance: Mucho más livianos que VM, pierde muy poca performance que metal.

Aprovisionamiento: Utiliza capas de aprovisionamiento, soporta snapshots, el bootup es en segundos o minutos. Fácil de mover cambios desde y hacia producción.

Máquina Virtual



Hardware: Cuasi independiente – hardware virtualizado, solo respeta arquitectura (x86, ARM,..)

Aislamiento: Todos los servicios comparten el SO y kernel pudiendo afectar a otros.

Cambios: Más fáciles que *metal*, existen **snapshots**, y por ende se puede versionar una VM.

Performance: Hay una gran pérdida de performance ya que cada máquina tiene varios S.O.

Aprovisionamiento: Al tener las imágenes de base y snapshots de una máquina virtual, el aprovisionamiento es más rápido y simple. Un developer podría tener su entorno similar a producción.

Metal / Físico



Hardware: Dependiente (drivers de placas, video, controladores, etc.)

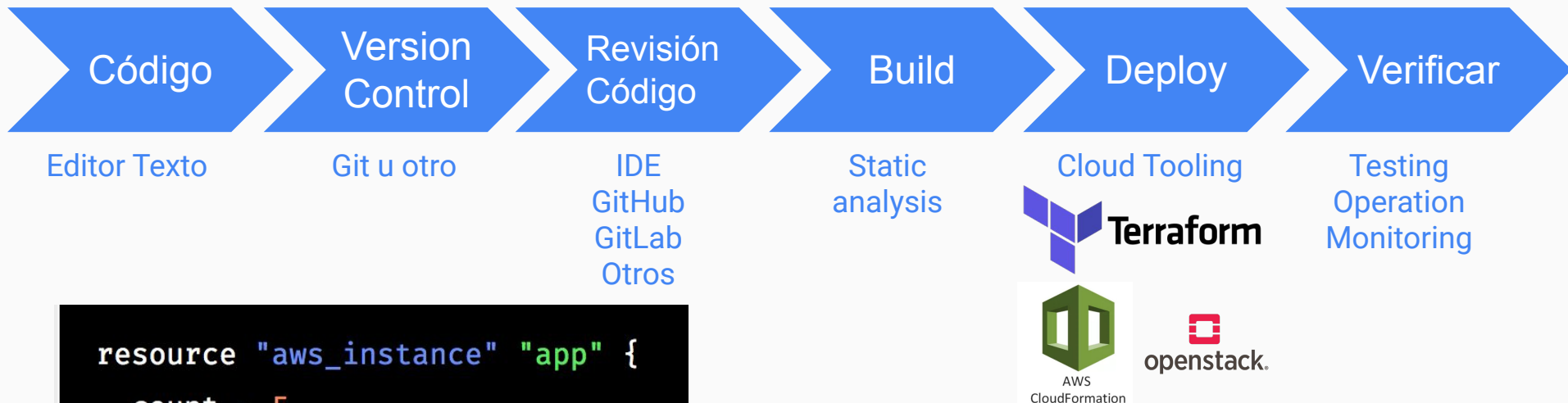
Aislamiento: Todos los servicios comparten el SO y kernel pudiendo afectar a otros.

Cambios: No hay snapshots, al romperse hay que reinstalar.

Performance: No hay overhead de performance

Aprovisionamiento: Requiere scripts con herramientas para gestión de paquetes, dependencias, acceso al servidor físico.

DevOps: Infrastructure as Code



```
resource "aws_instance" "app" {  
  count = 5  
  
  ami           = "ami-408c7f28"  
  instance_type = "t1.micro"  
}
```


An aerial photograph of the New York City skyline at dusk. The sky is a mix of dark purple, blue, and orange. The city is densely packed with skyscrapers, many of which are illuminated with lights. The Empire State Building is prominent in the center, with its top lit in red and green. The Hudson River is visible on the right side of the image. The text '¿Preguntas?' is overlaid in the center in a large, white, sans-serif font.

¿Preguntas?

Referencias

- SWEBOK - Capítulo 7 - Software Configuration Management
- The Importance of Branching Models in SCM
 - Chuck Walrad y Darrel Strom. IEEE Magazine, Sept. 2002

Devops

