GESTIÓN DE CONFIGURACIÓN DE SOFTWARE (SCM)

0. Origen
- Años 50: Departamento de Defensa USA
- 2001: Guía consolidada oficial
- Sistemas complejos militares
- Evolución hacia software moderno

1. Definición
- Proceso sistemático de ingeniería
- Gestiona cambios controlados
- Asegura coherencia del producto
- Supervisa metadatos de configuración
- Garantiza integridad y trazabilidad

2. Objetivos
- Control de cambios aprobados y registrados
- Rastreo de evolución del software
- Gestión de versiones múltiples
- Protección de integridad del sistema
- Reducción de errores
- Mejora de comunicación entre equipos
- Facilitación de colaboración

3. Componentes clave
- Identificación de Configuración — Determinar ítems a gestionar
- Control de configuración — Procedimientos formales, Autorización de cambios
- Gestión de versiones — Historial completo, Capacidad de reversión
- Auditoría y Validación — Cumplimiento de estándares, Verificación de modificaciones
- Gestión de Entornos — Consistencia Dev/Test/Prod, Configuraciones uniformes

4. Herramientas
- Control de versiones — Git (distribuido, líder del sector), Subversion SVN (centralizado)
- Contenedores — Docker (contenedores, Dockerfiles)
- Infraestructura como código — Terraform (multi-cloud), Ansible (Python, YAML), Puppet/Chef (gestión de infraestructura), SaltStack (automatización IT)
- Automatización CI/CD — Jenkins (construcción y despliegue)

5. Contexto DevOps
- En DevOps — Automatización de administración, Recursos bajo demanda, Elimina cuellos de botella
- En CI/CD — Pull requests automatizados, Despliegue inmediato, Flujo GitOps
- En Metodología Ágil — Clasificación de trabajo, Priorización de tareas, Gestión de dependencias, Integración en sprints

6. Beneficios
- Previene pérdida de trabajo
- Mejora calidad del software
- Facilita colaboración en paralelo
- Trazabilidad completa de cambios
- Reduce errores humanos
- Soporta continuidad operativa
- Incrementa agilidad de despliegue
- Facilita auditorías y revisiones
- Permite reversión rápida

7. Implementación
- Identificación — Recopilar información, Cifrar datos secretos
- Baseline (Referencia) — Estado conocido funcional, Configuración inicial
- Control de versiones — Añadir a repositorio Git, Cifrado de secretos
- Auditoría — Revisión de cambios, Pull requests, Responsabilidad y visibilidad

8. Ejemplo práctico
- Caso: Microservicios
  - Metadatos de configuración
  - Recursos hardware (CPU, RAM)
  - Endpoints de conexión
  - Secretos y claves
  - Problema: configuración dispersa
  - Solución: centralización en Git
  - Resultado: trazabilidad y control

Diagrama de flujo: NECESIDAD DE CAMBIO → Elaboración de Propuesta de cambio (FORMULARIO ESTÁNDAR) → Evaluación (Comités, Autoridad para disponer) → Aprobación → Sí → Implementación del Cambio (+ Documentación generada) → Verificación.
- Organización, Cliente, Proveedor
- DOCUMENTACIÓN Y JUSTIFICACIÓN DEL CAMBIO
- EVALUACIÓN DEL IMPACTO DEL CAMBIO
- APROBACIÓN / RECHAZO DEL CAMBIO
- IMPLEMENTACIÓN Y VERIFICACIÓN DEL CAMBIO
- ACTUALIZACIÓN DE LA LÍNEA DE REFERENCIA
- Isdefe

CONFIGURATION MANAGEMENT TOOLS: Terraform, SALTSTACK, puppet, git, CHEF, ANSIBLE, docker

SOFTWARE CONFIGURATION MANAGEMENT (SCM) — SOFTWARE, VERSION CONTROL, UPDATE, CONFIGURATION, SOFTWARE ENGINEERING, SYSTEM ADMINISTRATOR, BACKUP, HOST

TALLER - CLASE- 007 MAPA MENTAL

Materia: Construcción y Evolución de Software
Nombre: Javier Quilumba   Curso: GR2SW
Fecha: 09/11/2025 Docente: Vicente Eguez