UNIVERSIDAD NACIONAL DE LA PLATA

FACULTAD DE INFORMÁTICA

TESINA DE LICENCIATURA

Programa de Apoyo al Egreso de Profesionales en Actividad

TÍTULO: Evolución del uso de herramientas para el despliegue de un proyecto a gran escala.

AUTOR: Leticia Specogna

DIRECTOR ACADÉMICO: Rodolfo Bertone

DIRECTOR PROFESIONAL: Javier A. Petruccelli

CARRERA: Licenciatura en Sistemas

Resumen

eSidif es un proyecto que pertenece al Ministerio de Economía de la Nación. Se encarga de la formulación del presupuesto nacional y el registro la ejecución presupuestaria.
Lleva más de 15 años de desarrollo y aún se siguen sumando subsistemas que surgen a través de nuevos requerimientos.
Se va a mostrar uno en particular, que es web y también tiene una manera de desplegar novedosa, aplicando tecnologías más modernas de integraciones continuas.

Palabras Clave

- *Proyecto a gran escala*
- *Metodologías tradicionales*
- *Metodologías ágiles*
- *Scrum*
- *Planificación*
- *Despliegue*
- *Openshift*

Conclusiones

eSidif comenzó hace algunos años, y evolucionó su organización hasta lo que tiene hoy, que es una convivencia entre metodologías tradicionales, más duras, y metodologías ágiles.
Al ser tan grande, se permiten cambios y modernizaciones en cada pequeño proyecto que surge, que sirve también de prueba piloto a los que vendrán, además de evolucionar el proceso de ingeniería y estar a la altura de los avances tecnológicos.

Trabajos Realizados

Se presentó eSidif, se hizo un recorrido histórico por el proyecto y por las metodologías usadas, el proceso de despliegue y finalmente SIRECO que es un proyecto web desplegado en Openshift.

Trabajos Futuros

Como trabajo futuro podría hacerse una comparación de procesos de despliegue para sistemas web, analizando las alternativas ya conocidas y los costos/beneficios de cada una, eficiencia, simpleza y robustez.

Fecha de la presentación: Octubre 2020

UNIVERSIDAD NACIONAL DE LA PLATA

Facultad de Informática

Evolución del uso de herramientas para el despliegue de un proyecto a gran escala.

Alumna: Leticia Specogna.
Director: Rodolfo Bertone.

AGRADECIMIENTOS

A mis padres, que me dieron la posibilidad de venir a La Plata a estudiar, y con eso cambiaron mi vida. Siempre les agradeceré.

A mis hermanos, que son verdaderos amigos, por los momentos vividos durante la época de estudio, ¡gracias!

A mis amigos, los que coseche a lo largo de la vida, los que bancan en las malas pero más los que celebran conmigo cada alegría, los quiero siempre en mi equipo.

A Mariano, que me alentó y me hizo el aguante este último tiempo, gracias amor.

A mi director, Pampa, por su dedicación y entrega, ¡muchas gracias!

Al Lifia, que me dio el primer trabajo estable en el mundo informático, en el cual aprendí mucho y conocí gente muy valiosa.

No hubiera podido sin el aguante de cada uno desde su lugar.

Contenido

Capítulo 1 – eSidif 5
1.1 Contexto 5
1.2 Características de eSidif 5
1.3 Evolución Histórica 5
1.4 Tecnología, lenguaje y metodología 8
1.5 Distribución de los equipos 9
1.5.1 Dirección de gestión de proyectos 10
1.5.2 Dirección análisis y atención a usuarios 11
1.5.3 Dirección de desarrollo 11
1.5.4 Dirección ingeniería 13
1.5.5 Dirección técnica 15
1.6 Resumen 15
Capítulo 2 – Metodologías ágiles 16
2.1 Contexto 16
2.2 Metodología 16
2.2.1 Objetivos de una metodología 16
2.2.2 Características de una metodología 17
2.3 Metodología tradicional - RUP 17
2.3.1 Definición. 17
2.3.2 Características 18
2.4 Metodología ágil - Scrum 18
2.4.1 Definición 18
2.4.2 Características 19
2.4.3 Manifiesto de metodologías ágiles 19
2.4.4 Dinámica de Scrum 21
2.5 RUP y Scrum 28
2.5.1 Diferencias 28
2.5.2 Contratos 30
2.6 Resumen 31
Capítulo 3 – Proceso de despliegue. Introducción a Openshift 32

3.1 - Contexto....32
3.2 - Despliegue....32
3.3 - Fases del despliegue ágil....33
3.3.1 Sprint de Relevamiento....33
3.3.2 Sprint de Construcción....34
3.3.3 Sprint de Ejecución....35
3.3.4 Sprint de Cierre....35
3.4 Actividades del despliegue ágil....35
3.5 Contenedores de software....36
3.6 Orquestador de contenedores....38
3.6.1 Características....38
3.6.2 Docker y Kubernetes....39
3.7 Introducción a Openshift Container Platform....40
3.7.1 Arquitectura....40
3.7.2 Seguridad....48
3.8 Resumen....52
Capítulo 4 – Casos de prueba en subsistemas conectados a eSidif....53
4.1 Contexto....53
4.2 Sireco....53
4.3 Pipeline de Jenkins....54
4.4 Pipeline aplicado a SIRECO....56
4.5 Configuraciones....64
4.5.2 Servicio de mail....65
4.6 Resumen....66
Capítulo 5 – Conclusiones....67
REFERENCIAS....69

Capítulo 1 – eSidif

1.1 Contexto

ESidif es un proyecto del Ministerio de Economía de la Nación, que se encarga de la formulación del presupuesto nacional y registro de la ejecución presupuestaria. Surgió en 2004 como una evolución de sistemas anteriores, los cuales serán descriptos en forma general en la presente sección.

Comprende las áreas de compras, gastos, pagos, recursos, contabilidad general, conciliación bancaria, fondos rotatorios, pasajes y viáticos, entes, cuenta única del tesoro. Cada uno de estos grupos permite la gestión completa y auditada del presupuesto nacional y su ejecución. También cuenta con un inicio de ejercicio presupuestario y firma digital.

1.2 Características de eSidif

A continuación se mencionan brevemente algunas características del proyecto eSidif:

- Posee una base de datos única.
- Provee de gestión, registro y control simultáneos.
- Presenta mejoras en las buenas prácticas de administración financiera.
- Beneficio de ser el único sistema de administración financiera.
- Es de fácil uso.
- Incorpora seguridad y auditoría.
- Se tiene la posibilidad de acceder a distintas visiones de la información.
- Autonomía del usuario.
- Provee descentralización operativa.
- Presenta facilidad para los usuarios a la hora de la toma de decisiones.

1.3 Evolución Histórica

En esta sección se hará un recorrido por la trayectoria de la Dirección General de Sistemas Informáticos de la Administración Financiera (DGSIAF).

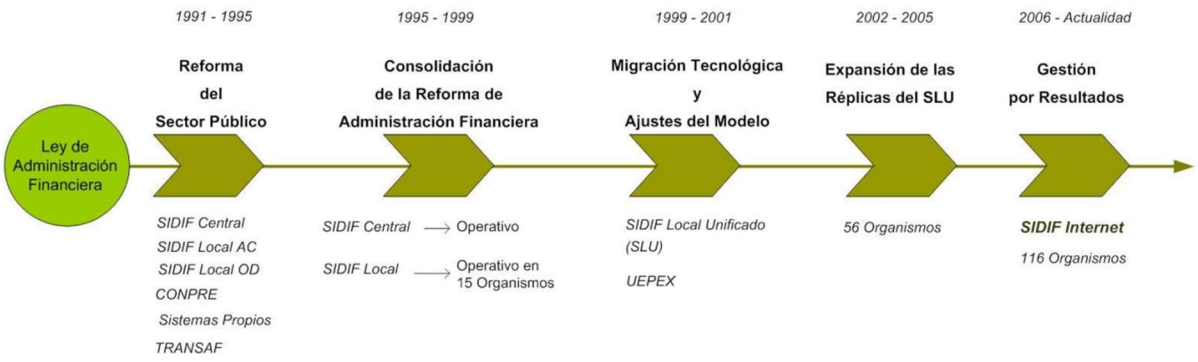


Imagen 1. Línea del tiempo

El primer hito dentro de la línea del tiempo presentada en la imagen 1, muestra la situación con la que se contaba antes de 1991. Esta época estuvo caracterizada por:

- Ausencia de estrategia informática y plan de sistemas.
- Sistemas aislados desarrollados en diferentes ambientes, diferente alcance, nivel de computarización y seguridad, sin documentación, no compatibles con la Ley de Administración Financiera.
- Escaso o nulo flujo de información.
- Multiplicidad en el registro de transacciones.
- Informaciónprocesadapococonfiable.
- Equipos obsoletos con recursos limitados.
- Personal informático escaso sin capacitación en el ambiente informático a utilizar.

Hubo luego una etapa de consolidación de la Reforma de Administración Financiera comprendida entre 1995 y 1999. Durante este lapso se desarrolló el Sistema Integrado de Información Financiera (SIDIF).

Este sistema, a partir de los registros efectuados en cada una de las jurisdicciones y entidades, permitió realizar en la Secretaría de Hacienda controles centralizados sistemáticos y permanentes del gasto público, disponiendo, a la vez, de información oportuna y abarcativa de toda la Administración Nacional.

Se incorporó una concepción interactiva de los procesos. Se manejaba la información en una base de datos central, operada SIDIF CENTRAL, que se ocupaba de inter-operar con las bases de datos locales de los Servicios Administrativos Financieros (SAF) operadas en cada organismo por su SIDIF LOCAL.

Durante 1999 y 2001 se produjo la migración tecnológica y los ajustes en el modelo, que marcaron un nuevo hito en la evolución de este sistema. En ese momento la Subsecretaría de Presupuesto comenzó a desarrollar un nuevo sistema de administración financiera para los SAF, llamado SIDIF Local Unificado (SLU). Los aspectos más importantes de esta etapa fueron:

- Gestión de transacciones con impacto económico y financiero.
- Modernización y ampliación de requerimientos funcionales.
- Renovación de equipamiento que soportaba el crecimiento de usuarios y operaciones.
- Dedicación de equipos interdisciplinarios a trabajar estrechamente con los SAF para relevar las particularidades de cada organismo.
- Capacitación de usuarios.
- Puesta en marcha del sistema.

En octubre de 2002 la Secretaría de Hacienda comenzó un proceso de réplica masiva del SLU en Organismos de la Administración con el objeto de reemplazar los sistemas existentes y racionalizar el mantenimiento de los mismos. A fines de 2005 el SLU era usado por 56 organismos.

En ese período, en 2004, empezó a gestarse la idea de eSidif. Este proyecto generó una profunda transformación de los sistemas de administración financiera sobre la base de tres lineamientos estratégicos fundamentales:

- Ampliación del alcance funcional.
- Actualización tecnológica e incorporación de herramientas.
- Promoción y facilidad de gestión de la administración financiera orientada a resultados.

Desde el punto de vista tecnológico, eSidif se concibió como un producto adaptable a diferentes entornos y tamaños organizacionales, con características visuales y de navegación que facilitan su uso, con una base de datos que centraliza toda la información de la gestión de los organismos ejecutores y de los órganos rectores, y con una arquitectura en capas que facilita y reduce los costos de mantenimiento y evolución del sistema. Incorporó además la capacidad de inter-operar con sistemas de otros organismos del Estado Nacional, a través del uso de servicios web.

Ya dentro del último período presentado en la línea del tiempo, en 2009, una vez logrado un avance significativo en el desarrollo, despliegue y madurez de eSidif, la subsecretaría de Presupuesto promovió la incorporación de soluciones modernas de inteligencia de negocios (BI).
Más tarde, en 2011, se comenzó con la implementación gradual de firma digital. Este avance profundizó el proceso de despapelización de la Administración Nacional y la descentralización de la carga de información.
Acorde a las innovaciones tecnológicas, en el marco del gobierno electrónico, en 2012, se implementó un servicio de consulta por internet accesible desde computadoras o equipos móviles para los proveedores y contratistas del Estado, llamado e-Prov.

Todo lo mencionado hasta aquí ha sido implementado exitosamente en la Administración Nacional. Es por ello, que respondiendo a los Lineamientos estratégicos de la Subsecretaría, en lo que respecta a promover el conocimiento de la información entre diversas instituciones y gobiernos, se dio origen al proyecto de implementación de sistemas de administración financiera para la gestión de finanzas públicas en las provincias.

1.4 Tecnología, lenguaje y metodología

Está desarrollado en JAVA, usando Eclipse como IDE, posee un framework propio llamado Toolkit que provee sistematización para el armado de ambiente compilación de código.

También hay un conjuntos de herramientas propias que complementan el desarrollo y facilitan la generación de entregas continuas, tales como:

- Nota de Entrega Automatizada (NEA): permite documentar las entregas con los alcances que tiene cada release, los bugs corregidos, los scripts incluidos.
- Build with Ant (BWA): genera la entrega a nivel de código. Se encarga de integrar cada subproyecto que conforma eSidif. De esta manera prevé errores de compilación y prepara la entrega en un archivo comprimido listo para ser instalado en un ambiente de pruebas locales.
- Sistema de instalación centralizado (SIC): recibe el archivo comprimido generado en el paso anterior y lo instala. Esta implantación garantiza una realease del sistema listo para ser probado.

La metodología utilizada en el proyecto es RUP. El desarrollo tiende a utilizar características de los métodos ágiles, particularmente Scrum. Este tema será desarrollado con más detalle en el próximo capítulo.

1.5 Distribución de los equipos

Para que un proyecto tan grande sea organizado es necesario dividirlo. Marcar bien los límites de cada equipo, que inevitablemente va a convivir o interrelacionarse con otro, en todas las etapas del proyecto.

En esta sección de presentará la conformación de los equipos de trabajo y los roles que cumplen dentro del proyecto.

Tomando como base el organigrama presentado en la imagen 2, extraido del sitio odifical de la DGSIAF (Dirección General de Sistemas de Información de la Administración Financiera) [1] que representa la situación actual.

Imagen 2 - organigrama DGSIAF

1.5.1 Dirección de gestión de proyectos

Administración y project management office (pmo)

Entre las tareas de administración se encuentran:

- Administrar el personal.
- GestionarExpedientes.
- Generardocumentación.

PMO se encarga de ejecutar con los distintos coordinadores del proyecto, los aspectos específicos relacionados con la gestión, haciendo hincapié en la planificación y seguimiento del proyecto.

Entre sus actividades principales están:

- Gestionar la planificación.
- Evaluar y proponer planes de mitigación.
- Gestionar el Seguimiento.
- Gestionar la contratación de recursos.
- Gestionar y hacer seguimiento de riesgos.
- Gestionar el despliegue.

1.5.2 Dirección análisis y atención a usuarios

Análisis

Se encarga de interpretar los requerimientos del usuario y trasladarlos al área de diseño y desarrollo traducidos en lenguaje unificado de modelado (UML).

Entre sus actividades se destacan:

- Explicar de forma detallada y precisa los requisitos.
- Evaluar la viabilidad técnica de los procesos que intervienen en el desarrollo.
- Modelar la solución en diferentes diagramas.
- Estimar el tamaño y esfuerzo de los negocios.
- Implementar los diagramas en UML.
- Definir de la estrategia y documentación del análisis de convivencia y migración.
- Generar la documentación del despliegue.
- Dar soporte ante inquietudes entre los usuarios y los desarrolladores.
- En eSidif hay subgrupos de análisis para cada equipo/negocio.

Réplicas

Tiene distintas funciones de acuerdo a las etapas en que se encuentre el proyecto:
cuando el proyecto está en pasaje a producción, se encarga de:

- Realizar demos y capacitar a los usuarios hasta que alcancen autonomía en el uso de la aplicación.

En cambio, cuando el proyecto se encuentra durante las etapas de desarrollo su función es:

- Participar en talleres de definición funcional y reuniones de avance y evaluación de negocios implementados y a implementar.

1.5.3 Dirección de desarrollo

Diseño & desarrollo

El área de Diseño y Desarrollo, tiene como misión materializar los requerimientos del usuario en artefactos visibles y ejecutables por el usuario.

Entre sus funciones se destacan:

- Modelar y construir componentes para ejecutar la funcionalidad del sistema.
- Construir los algoritmos de convivencia entre sistemas legados desarrollados por el equipo de convivencia.

Diseño y desarrollo a su vez, se divide en equipos dedicados a diferentes partes del dominio que en conjunto completan el circuito de ejecución, estas son: Pagos, Gastos, Compras, Fondos Rotatorios, Contabilidad General, Cuenta Única del Tesoro, Mantenimiento, Entes, Pasajes y Viáticos, Recursos, Deuda Pública.

Convivencia

Por su parte, el equipo de convivencia tiene a su cargo la construcción de algoritmos en lenguajes PL/SQL que garanticen la actualización y el control de los datos, entre todos los sistemas que conforman el eSidif.

Arquitectura

Su función es definir la arquitectura de software del e-Sidif y la plataforma de desarrollo sobre la cual se construye el producto. Se encarga de proveer de actualización tecnológica de la plataforma, y de mantenimiento permanente de la misma, así como también de brindar soporte a los demás equipos.

Entre sus funciones principales se encuentran:

- Definir la arquitectura de software.
- Gestionar la plataforma de desarrollo, que incumbe desde la definición, diseño, implementación y mantenimiento de la misma.
- Investigar y evaluar de las nuevas tecnologías que pueden ser de utilidad para la plataforma.
- Realizar manuales de uso para cada uno de los componentes de la plataforma.

Mantenimiento

Comprende el mantenimiento de e-Sidif y del SLU/UI (SIDIF Local Unificado/Unidad Informática). Se encuentra subdividida en distintos tipos, dependiendo del tipo del proyecto en el cual se esté participando.

Sus objetivos son agregar funcionalidad o corregir módulos que se encuentran productivos.

Entre sus funciones principales se encuentran:

- Interpretar y modelar modificaciones de los requerimientos del usuario.
- Modificar los ejecutables.
- Participar en las reuniones de priorización de los requerimientos.
- Participar en los emprendimientos de mejoras de los procesos.

Business Intelligence (BI)

BI y se encarga de:

- Interpretar requerimientos de niveles ejecutivos.
- Desarrollar e implementar con herramientas gerenciales.

1.5.4 Dirección ingeniería

Ingeniería

Este área se encarga de definir el proceso de desarrollo del proyecto, así como también de proveer al equipo del proyecto un ambiente estándar de herramientas que soporten el proceso de desarrollo definido.

Sus actividades son:

- Definirprocesos de desarrollo.
- Definirguías de trabajo.
- Proveer un ambienteestándar.
- Asegurar el cumplimiento de Calidad.

- Investigar las disciplinas del proceso Rup, adaptándolas al proyecto y definiendo las actividades, roles y artefactos.
- Estandarizar cada tipo de artefacto: templates, Ejemplos, Guías de buenas prácticas, Material de capacitación, etc.
- Gestionar Control de calidad de los artefactos generados.
- Elaborar de guías de administración, instalación y uso de herramientas.
- Capacitar y dar soporte al equipo de desarrollo, a través de sus disciplinas, artefactos y herramientas de soporte.
- Investigar y adaptar prácticas de procesos ágiles al proyecto

Testing

Es un área que se encarga de hacer pruebas integrales, de regresión y circuitos. Su trabajo se complementa con el de diseño y desarrollo. En el próximo capítulo se explicará la interacción de ambos grupos.

Entre sus actividades se destacan:

- Control de la calidad de los ejecutables.
- Verificar el cumplimiento de los requerimientos del usuario, esto es, a través de casos de prueba.

Grupos de orientación de diseño (GOD)

Es un área pequeña pero también tiene actividades interesantes. Entre ellas se destacan:

- Optimizar de los productos, tanto intermedio como final.
- Promocionar del reuso de código normalizado.
- Fomentar el reuso de pantallas comunes.
- Participar en los procesos de aseguramiento de calidad de modelado de Análisis y Diseño.
- Colaborar con el área de Arquitectura para implementar componentes generales.

1.5.5 Dirección técnica

Administra un centro de cómputos que asegura la operatoria diaria no sólo de nuestros sistemas productivos, sino que también posibilita el proceso interno de desarrollo. Provee además la conectividad a los usuarios dentro de la Administración Pública y brinda la infraestructura tecnológica para posibilitar el acceso desde múltiples dispositivos.

Está comprendida por grupos de bases de datos, windows, unix, redes, área técnica, producción que se encargan de dar soporte en las diferentes áreas mencionadas anteriormente

1.6 Resumen

En este capítulo se hizo una presentación del proyecto e-Sidif, un breve repaso a través del tiempo, la conformación de los equipos que están involucrados durante todo el proceso de ingeniería de software con sus tareas principales. En el próximo capítulo se explicará la metodología utilizada en este proceso.

Capítulo 2 – Metodologías ágiles

2.1 Contexto

En el capítulo anterior se presentó el proyecto y su evolución a lo largo del tiempo. Además, se presento la conformación del equipo de desarrollo en función de la magnitud del trabajo a resolver.

En este capítulo describirá las metodologías usadas en eSidif. Dichas metodologías fueron evolucionando con el paso de los años y ahora conviven dos: RUP y Scrum, una tradicional y una ágil. Se hará una breve descripción de la primera para luego profundizar en las ágiles.

Es importante dar una definición de metodología, para poder después clasificar, comparar y por último enfocar en las ágiles, que darán lugar al desarrollo del próximo capítulo.

2.2 Metodología

Se define como un conjunto de pasos y procedimientos que deben seguirse para desarrollar software. Está compuesta por:

- Cómo dividir un proyecto en etapas.
- Qué tareas se llevan a cabo en cada etapa.
- Qué restricciones deben aplicarse.
- Qué técnicas y herramientas se emplean.
- Cómo se controla y gestiona un proyecto.

2.2.1 Objetivos de una metodología

La metodología busca cumplir con los siguientes objetivos:

- Asegurar la uniformidad y calidad tanto del desarrollo como del sistema en sí.
- Satisfacer las necesidades de los usuarios del sistema.
- Conseguir un mayor nivel de rendimiento y eficiencia del personal asignado al desarrollo. Ajustarse a los plazos y costes previstos en la planificación.
- Facilitar el mantenimiento posterior de los sistemas.

- Definir actividades a llevarse a cabo en un Proyecto de Sistema de Información.
- Unificar criterios en la organización para el desarrollo del Sistema de Información.
- Proporcionar puntos de control y revisión.
- Permitir construir un sistema documentado y que sea fácil de mantener.
- Ayudar a identificar, lo antes posible, cualquier cambio que sea necesario realizar dentro del proceso de desarrollo

2.2.2 Características de una metodología

Una metodología es un marco de trabajo para organizar, planificar y acompañar el proceso de desarrollo. Para llevarlas a cabo hay reglas predefinidas, tareas, productos intermedios, técnicas, fases y sub-fases. Requieren de comunicación efectiva, herramientas (por ejemplo herramientas *CASE*, que acompañen la documentación). Todo esto se combina en pos de mejorar el desarrollo, dar soporte al mantenimiento y reutilización de software, no sólo a nivel de código.

2.3 Metodología tradicional - RUP

Una de las metodologías utilizadas en eSidif es RUP. A continuación se va a hacer un repaso de sus características.

2.3.1 Definición.

La sigla RUP provienede Rational Unified Process, traducido es Proceso Unificado de Rational (la empresa que lo desarrolló). Es una de las principales metodologías tradicionales (o no ágil), está guiada por una fuerte planificación durante todo el proceso de desarrollo.

Su objetivo principal es asegurar la producción de software de alta calidad que cumpla con las necesidades de los usuarios, con una planeación y presupuesto predecible.

2.3.2 Características

Propone un proceso iterativo e incremental dividido en cuatro fases, inicio, elaboración, construcción y transición. Usa el Lenguaje de Modelado Unificado (UML), está dirigido por casos de uso y centrado en la arquitectura.

Por otro lado desarrolla flujos de trabajo del proceso los que están asociados a la construcción propiamente dicha del software que son:

- Modelado del negocio
- Requisitos
- Análisis y diseño
- Implementación
- Pruebas y despliegue

Y también los flujos de trabajo de soporte que incluyen:

- Gestión del cambio y configuraciones
- Gestión de proyecto y entorno

2.4 Metodologíaágil - Scrum

Scrum es la metodología ágil que se incorporó al proyecto en los últimos años y convive actualmente con RUP.

2.4.1 Definición

La terminología “Scrum” procede delrugby, donde se designa al acto depreparar el avance del equipo en unidad pasando la pelota a uno y otro jugador. Scrum es adaptable, ágil, auto-organizado y con pocos tiempos muertos. Lo ágil se define como la habilidad de responder de forma versátil, al cambio para maximizar los beneficios.

Esta metodología fue desarrollada por Jeff Sutherlandix y elaborada más formalmente por Ken Schwaberx. Poco tiempo después Sutherland y Schwaber se unieron para refinar y extender Scrum.

Puede ser aplicado teóricamente a cualquier contexto en donde un grupo de gente necesita trabajar junta para lograr una meta común.

2.4.2 Características

Las metodologías ágiles varían en su forma de responder al cambio, pero en general comparten las siguientes características:

- Los individuos y sus interacciones son más importantes que los procesos y las herramientas.
- El software que funciona correctamente es más importante que la documentación exhaustiva.
- La colaboración con el cliente en lugar de la negociación de contratos.
- La respuesta al cambio en lugar de aferrarse a un plan.
- Se basan en el desarrollo iterativo e incremental, donde los requerimientos y las soluciones evolucionan mediante la colaboración de grupos auto-organizados y multidisciplinarios.
- Los clientes reciben prototipos o versiones periódicamente a medida que avanza el proyecto, lo que les permite evaluar el trabajo realizado.

2.4.3 Manifiesto de metodologías ágiles

Luego de una reunión en la que participaron 17 expertos de la industria del software se creó “The Alliance”: una organización dedicada a promover los conceptos relacionados con el desarrollo ágil de software y ayudar a las organizaciones para que los adopten. El punto departida fue el Manifiesto Ágil, un documento que resume la filosofía ágil.
Éstas son las características que diferencian un proceso ágil de uno tradicional. Los dos primeros son generales y resumen gran parte del espíritu ágil.[2] y [3]

1. La prioridad es satisfacer al cliente mediante tempranas y continuas entregas de software que le aporten un valor. Un proceso es ágil si a las pocas semanas de empezar, ya se entrega software que funcione aunque sea rudimentario. El cliente decide si pone en marcha dicho software con la funcionalidad que ahora se proporciona o simplemente lo revisa e informa deposibles cambios a realizar.

2. Dar la bienvenida a los cambios. Se capturan los cambios para que el cliente tenga una ventaja competitiva. Los cambios en los registros deben verse como algo positivo. Estos cambios le van a permitir al cliente aprender más, y lograr una mayor satisfacción.

3. Entregar frecuentemente software que funcione desde un par de semanas a un par de meses, con el menor intervalo de tiempo posible entre entregas. Se insiste en que las entregas al cliente sean software y no planificaciones, ni documentación de análisis o de diseño.

4. La gente del negocio y los desarrolladores deben trabajar juntos a lo largo del proyecto. El proceso de desarrollo necesita ser guiado por el cliente, por lo que la interacción con el equipo es muy frecuente.

5. Construir el proyecto entorno a individuos motivados. Darles el entorno y el apoyo que necesitan y confiar en ellos para conseguir finalizar el trabajo.

6. El diálogo cara a cara es el método más eficiente y efectivo para comunicar información dentro de un equipo de desarrollo. Los miembros de equipo deben hablar entre ellos. Éste es el principal y mejor modo de comunicación.

7. El software que funciona es la medida principal de progreso. El estado de un proyecto no viene dado por la documentación generada o la fase en la que se encuentre, sino por el código generado y el funcionamiento.

8. Los procesos ágiles promueven un desarrollo sostenible. Los desarrolladores y usuarios deberían ser capaces de mantener el ritmo de desarrollo durante toda la ejecución del proyecto, asegurando en todo momento que la calidad es máxima.

9. La atención continua a la calidad técnica y al buen diseño mejora la agilidad. Producir código claro y robusto es la clave para avanzar más rápidamente en el proyecto.

10. La simplicidad es esencial. Tomar los caminos más simples que sean consistentes con los objetivos perseguidos. Si el código producido es simple y de alta calidad será más sencillo adaptarlo a los cambios que puedan surgir.

11. Las mejores arquitecturas, requisitos y diseños surgen de los equipos organizados por sí mismos. Todo el equipo es informado de las responsabilidades y éstas recaen sobre todos sus miembros. Es el propio equipo el que decide la mejor forma de organizarse, de acuerdo a los objetivos que se persigan.

12. En intervalos regulares, el equipo reflexiona respecto en cómo llegar a ser más efectivo, y según esto ajusta su comportamiento. Puesto que el entorno está cambiando continuamente, el equipo también debe ajustarse al nuevo escenario de forma continua. Puede cambiar su organización, sus reglas, sus convenciones, sus relaciones, etc., para seguir siendo ágil.

2.4.4 Dinámica de Scrum

En la siguiente imagen, se puede visualizar el flujo de trabajo de la metodología Scrum. Esdecir, desde la captura de requerimiento, el product backlog, la duración del sprint, las iteraciones, etc

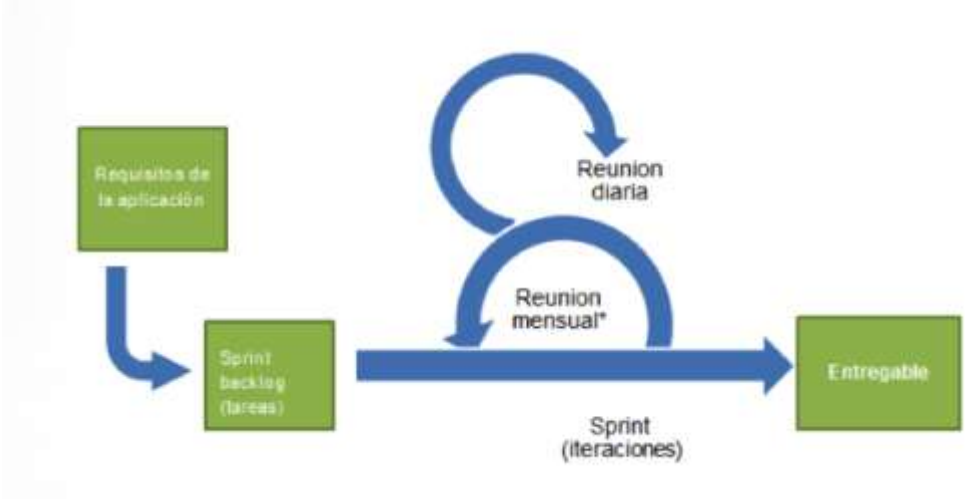


Imagen 3 - Dinámica de Scrum

2.4.4.1 Elementos de Scrum

A continuación se mencionan los elementos que componen Scrum y luego se realiza unadescripción de los mismos.

- Roles
  - Product owner
  - Scrum master
  - Team
- Poda de requerimientos
- Product backlog
- Sprint
  - Planificación.
  - Sprint backlog.
  - Scrum diario.
  - Estimaciones.
  - Builds continuos.
  - Revisión del sprint.
  - Reuniónretrospectiva.
- Valores
  - Foco, comunicación, respeto y coraje.

Roles

Los grupos no deberían estar integrados por más de veinte personas. Si hay más, lo más recomendable es formar varios equipos. No hay una técnica oficial para coordinar equipos múltiples, pero se han documentado experiencias de hasta 800 miembros, divididos en Scrum de Scrum definiendo un equipo central que se encarga de la coordinación, las pruebas cruzadas y la rotación de los miembros. Scrum tiene una estructura muy simple. Todas las responsabilidades del proyecto se reparten en 3 roles:

● Product owner (Dueño del producto) Representa a todos los interesados en el producto final. Es el responsable oficial del proyecto, gestión, control y visibilidad de la lista de acumulación o lista de retraso del producto (Product Backlog). Toma las decisiones finales de las tareas asignadas al registro y convierte sus elementos en rasgos a desarrollar.

Sus áreas de responsabilidad son:
- Financiación del proyecto.
- Requisitos del sistema.
- Retorno de la inversión del proyecto.
- Lanzamiento del proyecto.

● Scrum Master (Líder del proyecto) Responsable del proceso Scrum, de cumplir la meta y resolver los problemas. Así como también, de asegurarse que el proyecto se lleve a cabo de acuerdo con las prácticas, valores y reglas de Scrum y que progrese según lo previsto. Interactúa con el cliente y el equipo. Coordina los encuentros diarios, y se encarga de eliminareventuales obstáculos. Debe ser miembro del equipo y trabajar a la par.

● Team (Equipo) Responsables de transformar el Backlog de la iteración en un incremento de la funcionalidad del software, es decir, de convertir el product backlog en

un software entregable. El equipo tiene la autoridad para reorganizarse y definir las acciones necesarias o sugerir eliminación de impedimentos.

Características del equipo:
- Auto-gestionado.
- Auto-organizado.
- Multifuncional.

El número ideal para la conformación de un equipo es entre 8 y 10 personas.

Poda de requerimientos

La primera actividad es armar una lista exhaustiva de los requerimientos originales del sistema. Luego se procede a ver qué requerimientos son realmente necesarios, cuáles pueden posponerse y cuáles eliminarse. Para ello debe identificarse un representante con capacidad de decisión, priorizar los requerimientos en base a su importancia y acordar cuáles son los prioritarios para la fecha de entrega. La poda de requerimientos es una buena práctica en los modelos ágiles, se hace lo que el cliente realmente desea, no más. Es decir, se efectúa una priorización de requerimientos.

Product Backlog

Con los requerimientos priorizados se arma el Backlog de Producto. Este es una forma de registrar y organizar el trabajo pendiente para el producto (actividades y requerimientos). Es un documento dinámico que incorpora constantemente las necesidades del sistema. Por lo tanto, nunca llega a ser una lista completa y definitiva. Se mantiene durante todo el ciclo de vida (hasta finalizar el sistema) y es responsabilidad del ProductOwner.

Sprint Scrum

Un Sprint es el período de tiempo durante el que se desarrolla un incremento de funcionalidad. Constituye el núcleo de Scrum, que divide de esta forma el desarrollo de un proyecto en un conjunto de pequeñas “carreras”.

Características del Sprint:

- Duración máxima de 30 días.
- Durante el Sprint no se puede modificar el trabajo que se ha acordado en el Backlog.
- Sólo es posible cambiar el curso de un Sprint, abortándolo, y sólo lo puede hacer el Scrum Master si decide que no es viable por alguna de estas razones:
    - La tecnología acordada no funciona.
    - Las circunstancias del negocio han cambiado.
    - El equipo ha tenido interferencias.

Planificación

Se planifica en detalle el trabajo al inicio de cada Sprint asumiendo que los objetivos no van a cambiar durante el mismo. De esta manera se atenúa el riesgo.

Aspectos a tener en cuenta sobre la planificación de un Sprint:

- Una determinación general de alcance, frecuentemente basada en una EDT (Estructura de División del Trabajo).
- Estimaciones de esfuerzo de alto nivel realizadas durante la etapa de concepción del proyecto.
- Esfuerzo dedicado a labores de soporte o de preparación de los ambientes requeridos por el proyecto.
- Esfuerzo asociados a las reuniones diarias, de planificación y de revisión.
- Requerimientos de recursos de infraestructura o logísticos (máquinas, redes, licencias, papel, pizarras, etc.).
- Habilidades presentes y necesarias en el equipo.
- Restricciones asociadas al conocimiento del negocio, la tecnología o externas (legales, reglamentarias, estándares, etc.).

Rol del Scrum Master durante la planificación:

- Dirige la planificación.
- Es vínculo entre el equipo y el ProductOwner del proyecto.
- Registra problemas y riesgos detectados durante la planificación.

- Registra las tareas, asignaciones y estimaciones.
- Inicia el Backlog del Sprint.

Sprint Backlog

Se refiere al trabajo o tareas determinadas por el equipo para realizar en un Sprint. Es decir, la conversión de tareas a un producto funcional.

Características:

- Las tareas se estiman en una duración entre 1 a 20 horas de trabajo. Las de mayor duración deben intentar descomponerse en sub-tareas de ese rango de tiempo.
- La estimación se actualiza día a día.

Scrum diario (Daily)

Scrum asume que el proceso es complejo y que es necesario revisarlo frecuentemente, por eso se realiza una reunión diaria de seguimiento. El encuentro diario impide caer en el dilema señalado por Fred Brooks: “¿Cómo es que un proyecto puede retrasarse un año? Un día a la vez”.

El foco de la reunión es determinar el avance en las tareas y detectar problemas o “bloqueos” que están haciendo lento el progreso del equipo o que eventualmente impidan a un equipo cumplir con la meta del Sprint. La idea es que ningún problema quede sin resolver o, por lo menos, sin iniciar alguna acción de respuesta dentro de las 24 horas después de su detección. La reunión es además un espacio definido para que cada miembro del equipo comunique a los demás el estado de su trabajo y por lo tanto reafirme el compromiso.

Rol del Scrum Master durante el Scrum

Dentro de los roles del Scrum Master durante el Scrum encontramos los siguientes:

- Dirigir la reunión y mantener el foco de atención.
- Realizar preguntas para aclarar dudas.
- Registrar, escribir y/o documentar los problemas para su resolución después de la reunión.

● Asegurarse que los miembros cuenten con el ambiente adecuado para la reunión.

Estimaciones

Las estimaciones se realizan por primera vez en la reunión de planificación al inicio del Sprint. Posteriormente, las tareas se re-estiman todos los días y se registran sus cambios en el Backlog del Sprint.

Esta actividad puede ser realizada inmediatamente antes o después del Scrum diario.

Algunas claves para la estimación son las siguientes:

● Siempre se realizan estimaciones de esfuerzo, no de duración.

● Siempre se estima el esfuerzo total pendiente para terminar la tarea, no se estima el esfuerzo consumido.

● Se buscan unidades manejables, lo usual es que estén en un mínimo de 2 horas y un máximo de 20. Si la tarea es muy corta se trata de juntarla con otras relacionadas. Si la tarea es muy grande se trata de descomponerla.

Builds continuos y pruebas básicas

La estrategia que generalmente se utiliza es la de Builds continuos y “smoke test” (prueba básica para la funcionalidad del sistema). El procedimiento de Builds continuos es el siguiente:

● Los programadores desarrollan según el Backlog del Sprint, y al finalizar, notifican al integrador.

● El integrador toma el código y lo integra con el resto del producto.

● Se compila el software y se prueba “por arriba” el producto, para verificar que no se haya roto.

● Si se encuentran problemas se devuelve al desarrollador.

● Se notifica al equipo que hay una nueva versión “estable” del código para usar como base.

Revisión del Sprint

El objetivo de la reunión de revisión es presentar el producto o porción del producto desarrollada por el equipo a los usuarios. La reunión se utiliza para detectar

inconformidades mayores que se vuelven elementos del Backlog de Producto y que eventualmente se resuelven en el siguiente Sprint. A la reunión asisten el equipo, el Scrum Master, el Product Owner y todas las personas implicadas en el proyecto.

Reunión retrospectiva

Scrum está involucrado con la 'mejora continua' y se aplica en la práctica a través de reuniones de retrospección. En ellas se pretende detectar puntos débiles y fuertes en los que hay que enfocarse en futuros sprints.

2.5 RUP y Scrum

Las metodologías de desarrollo ágiles como Scrum y las llamadas tradicionales como RUP, no son competidoras, está claramente demostrado en eSidif, donde conviven.

Es necesario tener en cuenta que ante una externalización del desarrollo hay que adoptar la metodología que más se ajuste a las necesidades de la organización.

A pesar de que las metodologías ágiles valoran más la colaboración con el clientes que los contratos, es necesario definir contratos para determinar reglas mínimas, dado que un cambio de planes en contratos establecidos, es un riesgo que hay que evitar.

2.5.1 Diferencias

En este apartado de Beatriz González [4] se van a presentar las diferencias entre ambas metodologías.

RUP	SCRUM
Previsibilidad: de pocos cambios, estables y enfocados en la organización.	Adaptabilidad: de alto cambio, turbulento y enfocados en el proyecto.
Cierta resistencia a los cambios.	Preparados para cambios durante el proyecto.

Cliente en “espera”, que interactúa con el equipo de desarrollo. Las interacciones se dan según necesidad, guiadas por la planificación y contratos.	Cliente comprometido, es parte del equipo de desarrollo. Las interacciones están centradas en las prioridades de cada iteración.
Reuniones de seguimiento.	Reuniones diarias (dailies)
Proceso muy controlado, con numerosas políticas/normas.	Proceso poco controlado, con pocas normas.
Existe un contratoprefijado.	No existe un contrato, hay reglas pero son flexibles.
Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo.	Basadas en heurísticas provenientes de prácticas de producción de código.

En la imagen X se puede ver gráficamente la diferencia de ambas metodologías en las diferentes etapas de ingeniería de software.

Imagen 4 - Descripción gráfica de etapas de ingeniería de software.

2.5.2 Contratos

En este apartado se explicarán las definiciones a nivel de contrato convenientes de acuerdo a cada metodología.

2.5.2.1 Metodologías ágiles

Es conveniente explicitar reglas del juego en forma de cláusulas de contrato, como:

- Priorización inicial por valor para el negocio de la lista de objetivos del proyecto o servicio (Product Backlog), que establecerá un primer punto de partida pero no será vinculante.
- Revisiones regulares del producto final de modo que se disponga de ciclos cortos de aprendizaje, con posibilidad de repriorización del Product Backlog.
- Establecimiento de cómo hacer cambios de requisitos a cuenta del esfuerzo pendiente en el proyecto.
- Definición de completado (DoD: Definition of Done), que explique qué es un producto preparado para ser utilizado.
- Cláusulas específicas de Cambios de requisitos y de Finalización anticipada del contrato.

2.5.2.2 Metodologías tradicionales

Por su parte, las metodologías tradicionales tienen las propias y son las siguientes:

- Una firma inicial de requisitos o análisis funcional (para asegurar que se desarrollará todo lo esperado según lo especificado en las primeras etapas del proyecto). Lo anterior crea un contexto que favorece que el cliente pida todo lo que crea que en algún momento sea necesario.
- Revisión intermedia de sub-productos (principalmente documentación).
- Una aceptación final del producto. Esto puede ser riesgoso porque se deja para el final la parte crítica de verificación del cumplimiento de las expectativas.

2.6 Resumen

En este capítulo se presentaron las metodologías aplicadas a ESIDIF Se caracterizó cada una y sobre el final del capítulo se hizo una comparación para resaltar las diferencias. En pocas líneas: las metodologías tradicionales (RUP) para el desarrollo de software se caracterizan por tener un proceso disciplinado y sistemático con el objetivo de hacer el trabajo más predecible, eficiente y planificado. Es un proceso riguroso de seguimiento de pasos metódicos y propulsor de una documentación extensa. No permite entregar un producto útil al cliente en etapas tempranas de su proceso de desarrollo y tampoco permite realizar cambios en los requerimientos. Mientras que las metodologías ágiles (Scrum) son procesos iterativos en los que se entrelazan la especificación, el diseño, el desarrollo y las pruebas. El software se vadesarrollando a través de incrementos, en donde cada incremento incluye nuevas funcionalidades al sistema.

Se puso particular énfasis en las metodologías ágiles, dado que a partir del próximo capítulo se aplicarán conceptos ágiles al proceso de despliegue.

Capítulo 3 – Proceso de despliegue. Introducción a Openshift

3.1 - Contexto

Dentro de la metodología RUP, la disciplina de despliegue se encarga de que el producto construido sea transferido al ambiente de producción para ser utilizado por el usuario. Para ello, se debe elaborar un plan detallando con una serie de tareas que permitan, mediante su ejecución, dejar disponible el sistema al usuario final.

A priori resulta difícil relacionar una disciplina surgida dentro de una metodología estructurada como RUP, con los principios que pregonan las metodologías ágiles. Pero dentro de este contexto, en el proyecto se combinan ambas logrando una convivencia armónica entre ellas. Por un lado la definición de los Artefactos se realiza siguiendo los principios que describe RUP. Mientras, para la gestión de procesos, en la mayoría de los equipos se utiliza Scrum. Scrum define un marco de trabajo para la gestión y desarrollo de software basándose en un proceso iterativo e incremental y que puede ser utilizado en proyectos, como este, donde se optó por el desarrollo ágil de software.

Tomando como punto de partida la convivencia de las metodologías mencionada, se trabajó en la adaptación de la disciplina de despliegue. Se realizaron los ajustes necesarios, sin perder la rigurosidad que la misma detalla, pero adaptándose a los fundamentos ágiles.

3.2 - Despliegue

Siguiendo los lineamientos que define scrum, en el contexto del desarrollo de un producto, al final de cada sprint se realiza el despliegue de la funcionalidad implementada al usuario final. De este modo, finalizado el último sprint se contará con toda la funcionalidad implementada en producción. En este marco, tiene sentido plantear la disciplina de despliegue como un conjunto de actividades que permita realizar este pasaje al entorno productivo.

Naturalmente cada proyecto tendrá diferentes plazos y limitaciones para realizar estas implementaciones. Puede optarse por no realizar esta actividad al fin de cada sprint, dejándola para cuando se hayan acumulado las suficientes partes del producto que tengan relevancia para implementarse en un entorno productivo.

Aquí podrían aparecer dos tipos diferentes de implementaciones en un entorno productivo.Por un lado las que corresponden al fin de un sprint y por otro las que se corresponden a la acumulación de porciones de producto y que constituyen un hito a implementar.

Las porciones del producto, pueden corresponder a diferentes negocios (funcionales) en los cuales diferentes equipos realizan el análisis, el desarrollo y el testing dentro del marco del desarrollo ágil con scrum.
Nos encontramos entonces en un escenario donde varios equipos de diferentes disciplinas interactúan para construir un producto, y donde cada uno de ellos usa scrum. Si a esto se le agregan varios productos desarrollándose en paralelo; se encuentran con varios equipos de diferentes disciplinas interactuando para construir varios productos en donde además el nivel de acoplamiento puede ir variando. -
Este caso es el de mayor complejidad a considerar al momento de realizar las actividades de un despliegue.

3.3 - Fases del despliegue ágil

Se dividió la gestión de un despliegue en 4 sprints, que no necesariamente tienen la misma duración. El objetivo fue poder cerrar cada uno de estos sprints con un producto entregable. A cada sprint se le dio un nombre: relevamiento, construcción, ejecución, cierre.

3.3.1 Sprint de Relevamiento

Durante este sprint (o etapa), se debe tomar conocimiento de las característicaspropias que tendrá el despliegue a realizar. Identificar las

particularidades, conocer las fechas relevantes y realizar la comunicación a todas las áreas de estas novedades.

Paralelamente se comienza a trabajar en el armado de los artefactos que utilizarán al momento de ejecutar el despliegue y en el acompañamiento de la elaboración de los documentos que se enviarán al usuario. Al finalizar este sprint se presenta a los involucrados e interesados en la implementación un documento que detalla resumidamente las características del despliegue.

3.3.2 Sprint de Construcción

Este sprint podría iniciarse aún sin que haya terminado el anterior.

Durante esta etapa, se gestiona el envío de avisos a todos los usuarios, indicando que se realizarán tareas de puesta en producción que requerirán realizar la baja del sistema durante un determinado tiempo. Del mismo modo y dado que en algunas ocasiones hay un número considerable de personas involucradas, se revisan y evalúan las acciones para asegurar la disponibilidad de los recursos técnicos necesarios para que se puedan realizar las tareas. Paralelamente se gestiona una serie de documentos asociados que sirven de presentación del producto que se está implementado.

Además y analizando la construcción propia del despliegue, se realiza la elaboración del cronograma detallado de tareas que se deben llevar a cabo previamente a la fecha de implementación, las que ocurrirán el día de la implementación y las tareas posteriores. También se trabaja en la generación y seguimiento de lo que se debe instalar, mediante la elaboración de una lista de entregas a instalar en el entorno productivo. Esta lista de tareas se referencia desde el cronograma, que es en donde se representa el flujo de control del despliegue.

3.3.3 Sprint de Ejecución

Se realiza en este sprint la ejecución de las tareas previamente definidas y planificadas, tanto si están a cargo del equipo de despliegue o a cargo de otros equipos asignados al despliegue.

¿Qué representa esta ejecución? Concretamente es indicar a cada responsable o a cada área lo que debe realizar, en qué momento y bajo qué condiciones debe realizarla. Idealmente antes de la implementación en el entorno productivo se realiza una simulación de todo el proceso en un entorno similar al productivo. Esto permite repasar y depurar el proceso, realizando los ajustes necesarios para su ejecución.

3.3.4 Sprint de Cierre

En este punto se realiza lo que podría considerarse como una analogía a una retrospectiva. Por un lado el propio equipo de despliegue repasa lo ocurrido durante todo el proceso de despliegue, tomando estas conclusiones para futuras implementaciones.

Asimismo, se realizan reuniones con las áreas intervinientes a fin de repasar y sacar conclusiones, similar a un espacio de retrospectiva.

Adicionalmentese realiza un breve informe de situación sobre la implementación realizada.

3.4 Actividades del despliegue ágil

Dentro del contexto de este proyecto, con un despliegue de gran funcionalidad, las actividades principales son:

- Relevar y consensuar las fechas en que se realizará el despliegue.
- Relevar el alcance que tendrá el despliegue (en cuanto a funcionalidad que incluye, tareas técnicas particulares y cualquier otra consideración especial a tener en cuenta).

- Gestionar la generación de los documentos que permitan al usuario conocer la funcionalidad entregada.
- Planificar la ejecución en los entorno de preproducción
- Colaborar con la planificación del mantenimiento de los entornos de preproducción.
- Planificar los momentos del despliegue en cada una de sus instancias mediante laelaboración de un cronograma con las tareas a realizarse para la puesta en producción. Adicionalmente, participar en la elaboración de los artefactos que acompañan a la ejecución (planilla de entregas a instalar).
- Mantener informados a los integrantes del proyecto sobre las características propias y las novedades del despliegue y fomentar la interacción con ellos.
- Coordinar y Ejecutar el despliegue.
- Acompañar el post despliegue, generando un informe de situación e incidentes. derivado de la implementación.
- Realizar reuniones de mejora continua que permitan tener una abierta y clara comunicación con cada una con los integrantes de las áreas involucradas.
- Revisar y mejorar la automatización de los procesos que acompañan la disciplina, tanto en su planificación y ejecución como en las tareas anexas.
- Realizar el acompañamiento en los despliegues no funcionales.

3.5 Contenedores de software

Las formas de despliegue de aplicaciones también se fueron actualizando, respondiendo a las necesidades que surgen en cada caso.

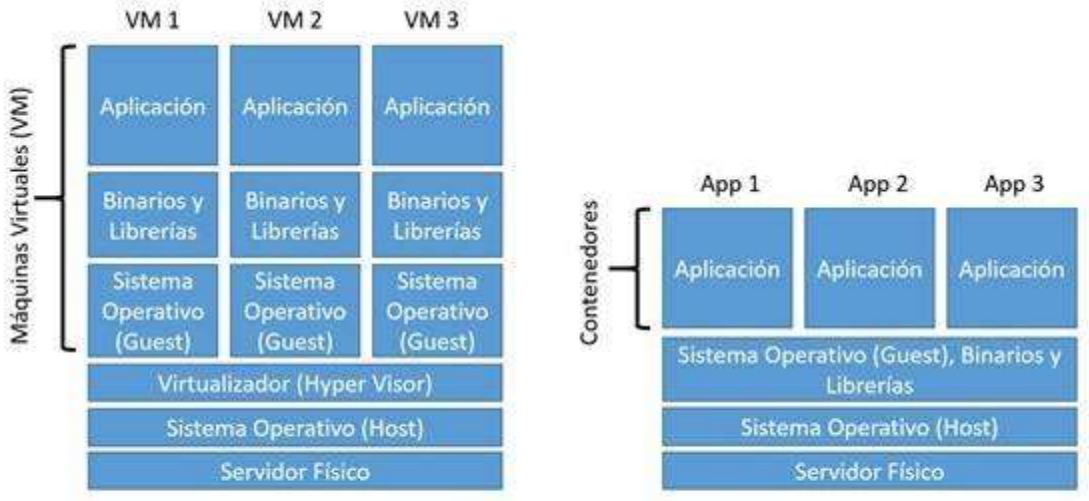
Actualmente, políticas como contenedores de software están siendo utilizadas. Se ha cambiado la manera de desplegar aplicaciones sobre las infraestructuras y se han utilizado para automatizar estos despliegues y operaciones gestionando su creación, empaquetado y distribución de una forma más eficiente.

A continuación se define someramente qué son los contenedores de software, luego los orquestadores de contenedores, para finalmente introducir Kubernetes y openshift sobre el final del capítulo. [5]

Los contenedores de software son un mecanismo de empaquetado en el que las aplicaciones pueden extraerse del entorno donde se ejecutan y esto genera el despliegue basado en contenedores independientes. Esto se logra gracias a que tienen la capacidad de replicar el software en diferentes contextos, aunque el ambiente en el que lo haga no cumpla con las configuraciones determinadas de las aplicaciones.

Los contenedores otorgan la habilidad de correr aplicaciones en un ambiente aislado. Este aislamiento es una manera de asegurar las aplicaciones porque permite ejecutar los contenedores de manera simultánea o con un host específico. Esta tecnología es muy independiente y no necesita la carga adicional de un hipervisor, sino que se ejecutan directamente en el núcleo del equipo del host.

Estos contenedores son ambientes de ejecución livianos que proveen a las aplicaciones con los archivos, variables y librerías que necesitan para operar. Se utilizan para garantizar que una aplicación se ejecute correctamente aún cuando su entorno cambia, disminuyendo la posibilidad de fallas y aumentando su portabilidad. Se los suele comparar con la virtualización clásica, y si bien se asemejan, los contenedores funcionan en un plano más bajo: las máquinas virtuales habilitan la virtualización de la infraestructura computacional, los contenedores permiten la virtualización de las aplicaciones. Además, a diferencia de las máquinas virtuales, los contenedores utilizan el sistema operativo de su host, en lugar de integrar uno propio.



Máquinas Virtuales versus Contenedores

Imagen 5 - Diferencia entre virtualización y contenedores. (ref: fayerwayer.com)

Al incluir un sistema operativo completo, los contenedores requieren de recursos computacionales mínimos y son rápidos y fáciles de instalar. De esta forma reducen la carga en los servidores y permiten desplegar un número mucho mayor de aplicaciones.
Los contenedores pueden ser desplegados en clústeres. Un contenedor individual puede así encapsular componentes únicos de aplicaciones más complejas. Al separar los componentes de esta forma los desarrolladores pueden actualizar componentes individuales sin re trabajar la aplicación completa.
La facilidad de despliegue puede resultar en un número elevado y crecientemente complejo de aplicaciones en contenedores múltiples. La mantención de un buen funcionamiento y administración en ese punto se transforma en un trabajo demasiado arduo para una persona. Para simplificar este proceso existe software de gestión de contenedores, que se encarga de manejar tareas asociadas con la administración de aplicaciones contenerizadas y componentes de aplicación, de esta forma se facilita la operación básica y se asegura la interoperabilidad. Estas características pueden potenciarse si se complementan con una orquestación automatizada de contenedores, apoyada en una estrategia de contenedores completa.

3.6 Orquestador de contenedores

Los orquestadores de contenedores son un complemento necesario a los contenedores de software, que apoyan la gestión, creación y distribución de los contenedores y lo logra controlando y dirigiendo cada aspecto del sistema.

3.6.1 Características

Un orquestador de contenedores se ocupa de cuestiones como:

- Configuración automática.
- Despliegue y “levantado” automático de servicios basados en contenedores.
- Balanceado de carga.

- Auto-escalado y auto-reinicio de contenedores.
- Control de la “salud” de cada contenedor.
- Intercambio de datos y networking.
- Mantenimiento de parámetros “secretos” y configuraciones.

3.6.2 Docker y Kubernetes

Docker es una plataforma de código abierto para desarrollar, enviar y correr aplicaciones al permitir separar la infraestructura para ejecutar el software rápidamente.

Kubernetes es un software de orquestación de código abierto que ofrece una API para controlar la forma y el lugar en que se ejecutarán los contenedores. Le permite ejecutar contenedores y cargas de trabajo de Docker y dar solución a algunas de las complejidades de funcionamiento al escalar varios contenedores implementados en varios servidores.

Organiza los clústeres de máquinas virtuales y programa los contenedores para que se ejecuten en esas máquinas en función de los recursos de proceso disponibles y de los requisitos de recursos de cada contenedor.

Los contenedores se agrupan en pods (la unidad operativa básica de Kubernetes) que se pueden escalar hasta el estado deseado.

Kubernetes también administra automáticamente la detección de servicios, incorpora el equilibrio de carga, realiza un seguimiento de la asignación de recursos y los escala en función del uso de la capacidad de proceso. Además comprueba el estado de los recursos individuales y permite que las aplicaciones se recuperen automáticamente reiniciando o replicando los contenedores.

Con Kubernetes se puede organizar un clúster de máquinas virtuales y programar los contenedores para que se ejecuten en esas máquinas según los recursos de proceso disponibles y los requisitos de recursos de cada contenedor.

Ayuda a eliminar procesos manuales que complican la implementación y la escalabilidad en los contenedores de software. Esto lo hace a través de un clúster de

grupos de hosts y ayuda a administrarlos fácil y eficazmente, que pueden ser host públicos, privados o híbridos.

3.7 Introducción a Openshift Container Platform

Openshift es una plataforma de aplicaciones integral creada para contenedores con Kubernetes. Fue desarrollada por Red Hat en el año 2011.

Se puede utilizar para ejecutar, orquestar, monitorear y escalar contenedores. Brinda un esquema de organización para controlar, defender y extender la plataforma de aplicaciones a lo largo del ciclo de vida de una aplicación.

Permite una cadena segura de suministro de software para ayudar a que las aplicaciones sean más seguras sin reducir productividad del desarrollador y proporcionar una experiencia de gestión y operaciones consistentes en diferentes infraestructuras.

3.7.1 Arquitectura

A continuación se listarán los componentes y roles que admite Openshift. [6]

Sistema Operativo

Openshift se empaqueta y se implementa en RHEL (Red Hat Enterprise Linux) CoreOS. RHEL CoreOS es una tecnología de sistema operativo de un sólo propósito, optimizada para contenedores y Kubernetes, con una huella mínima accionado por los mismo binarios que RHEL. Este SO reforzado puede ayudar a las organizaciones a reunir requisitos para el endurecimiento del sistema con la menor funcionalidad a través de su naturaleza ligera y especialmente diseñada, dado que solo incluye las características, funciones y servicios necesarios para alojar contenedores en un entorno Openshift.

Red Hat CoreOS brinda la capacidad de soportar actualizaciones transaccionales utilizando el sistema de actualización rpm -ostree. Las actualizaciones se entregan a través de imágenes de contenedor y son parte del proceso de actualización de Openshift. Cuando se implementa, la imagen del contenedor se extrae, extrae y escribe

el disco, y el gestor de arranque se modifica para iniciar en la nueva versión. Las máquinas en el clúster se reiniciarán para actualizar de forma continua y así garantizar que la capacidad del clúster se vea mínimamente afectada.
SI hubiera actualizaciones de RHEL CoreOS en Openshift se realizarían durante las actualizaciones del clúster.

Entorno Operativo

Openshift se puede implementar en hardware físico, en una infraestructura virtual, o en la nube. Se puede implementar en entornos privados o públicos, dependiendo del uso específico de la organización.

Tiempo de ejecución de OCI (Open Container Iniciative)

CRI-O (Container Runtime Interface) es el único motor de contenedor disponible dentro de los clústeres de Openshift.

Kubernetes

Proporciona orquestación para servicios complejos de contenedores múltiples. Kubernetes también proporciona programación para servicios en un clúster de host de contenedor. Openshift agrega a Kubernetes herramientas de desarrollo y centradas en las operaciones que permiten el desarrollo rápido de aplicaciones, fácil implementación y escalado, y mantenimiento de ciclo de vida a largo plazo para aplicaciones y para la plataforma misma. Openshift también aprovecha la integración de componentes de Kubernetes para automatizar compilaciones de aplicaciones, implementaciones, escalado, administración de salud, y más.

Contenedores: las instancias de aplicación para el usuario final, los componentes de la aplicación u otros servicios se ejecutan en contenedores Linux. El contenedor sólo incluye las librerías, funciones, elementos y código necesarios para ejecutar la aplicación.

Pods: mientras los componentes de la aplicación se ejecutan en contenedores, Openshift organiza y administra los Pods. Un pod es un grupo de contenedores que se implementan juntos en el mismo host, es una unidad orquestada en Openshift compuesta por uno o más contenedores. En general, un pod sólo debe contener una función única, como un servidor de aplicaciones o un servidor web, y no debe incluir múltiples funciones como una base de datos y un servidor de aplicaciones.

Componentes de Openshift

Operadores Openshift: un operador es un método para empaquetar, implementar y administrar una aplicación nativa de Kubernetes. Los operadores automatizan la gestión del ciclo de vida de las aplicaciones en contenedores dentro de Kubernetes. Una aplicación nativa de Kubernetes es una aplicación que se implementa en Kubernetes y es administrada usando las API de Kubernetes y las herramientas de KUBECTL. Un controlador es un concepto central de Kubernetes y se implementa como un bucle de software que se ejecuta continuamente en los nodos maestros de Kubernetes, compara, y si es necesario, concilia el estado deseado expresado y el estado actual de un objeto. Los objetos son recursos como Pods, servicios, mapas de configuraciones, entre otros. Los operadores aplican el modelo de controlador a nivel de aplicaciones completas, y son, en efecto, controladores personalizados específicos de la aplicación.

Dado que Openshift es una plataforma plena en contenedor que consta de muchos componentes diferentes, Openshift toma ventaja de los operadores para impulsar la instalación y las actualizaciones de Openshift y de sus servicios.
Los operadores sirven como la base de la plataforma que elimina la necesidad de actualizaciones manuales de los sistemas operativos y aplicaciones de plano de control Openshift. Todo lo que compone la plataforma es gestionado a lo largo de su ciclo de vida con operadores.

Administrador de ciclo de vida de operadores: el operador proporciona facilidades para almacenar y distribuir operadores a personas desarrollando y desplegando aplicaciones. El administrador del ciclo de vida del operador es el plano posterior que facilita la gestión de operadores en un clúster de Kubernetes. Con el administrador del ciclo de vida de operadores, los administradores del clúster pueden controlar qué operadores están disponibles en qué espacios de nombres y quién puede interactuar con la ejecución de operadores. Los permisos de un operador se configuran automáticamente para seguir un enfoque de privilegios mínimos. El administrador ayuda a los usuarios a instalar, actualizar y administrar el ciclo de vida de todos los operadores y sus servicios asociados que se ejecutan en sus clústeres.

Nodos Openshift: las aplicaciones se ejecutan en nodos en los contenedores. Contienen el nodo daemon necesario de openshift, Kubelet, el tiempo de ejecución del contenedor y otros servicios necesarios para soportar el alojamiento de contenedores. La mayoría de los componentes de software que se ejecutan por encima del sistema operativo se ejecutan en contenedores en el nodo.

Openshift Máster: son el plano de control para Openshift. Las máquinas maestras mantienen y entienden el estado del entorno y organizan toda la actividad que ocurre en los nodos. Los masters se ejecutan en RHEL Core OS. Las máquinas maestras están definidas por una serie de recursos API de máquinas independientes. Se aplican controles adicionales a las máquinas máster para evitar que los clientes eliminen todas las máquinas Máster y rompan el clúster. Los Clúster Masters contienen más que los servicios de Kubernetes para administrar el clúster de openshift; los servicios que se incluyen en la categoría Kubernetes en el máster incluyen el servidor API, el planificador, el servidor administrador del controlador y los controladores de ingreso. Algunos de los servicios en las máquinas Máster se ejecutan como servicios del sistema, mientras que otros se ejecutan como Pods estáticos. Los servicios del sistema que se ejecutan en máquinas Máster incluyen sshd, motor de contenedor CRI-O y Kubelet. Las siguientes son las cuatro funciones de Openshift Masters:

- *API y autenticación:* las máquinas máster proporcionan la API única con la que interactúan todas las herramientas y sistemas. Todo lo que interactúa con openshift debe pasar por esta API. Todas las solicitudes de API están encriptadas con Secure Sockets Layer (SSL) y deben autenticarse. La autenticación es manejada por un servidor OAuth incorporado. Red Hat recomienda vincular los máster a un sistema externo de administración de identidad y acceso mediante el Protocolo ligero de acceso a directorios (LDAP), OpenID Connect y otros proveedores de identidad. Las autorizaciones se manejan mediante un control de acceso basado en roles (RBAC). El máster evalúa las solicitudes de autenticación y autorización. Los usuarios de Openshift a los que se les ha otorgado acceso pueden recibir autorización para trabajar con proyectos específicos.

- *Estado deseado y actual:* el estado de openshift se mantiene en el almacén de datos de Openshift. El almacén de datos usa etcd, un almacén distribuido de clave-valor. El almacén de datos contiene información sobre el entorno Openshift y perteneciente al máster de openshift, incluida la información de la cuenta de usuario y las reglas de RBAC; el estado del entorno Openshift, incluida la información del entorno de la aplicación y los datos del usuario sin aplicación; e importantes variables de entorno, datos secretos y otra información.

- *Planificador:* El planificador determina la ubicación del pod dentro de Openshift. Utiliza una combinación de configuración y estado del entorno (ej: CPU, memoria) para determinar la mejor opción para ejecutar los pods en los nodos del entorno. El planificador está configurado con un archivo JSON simple en combinación con etiquetas de nodo para dividir openshift. Esto permite que la colocación de pods dentro de openshift se base en la topología del mundo real, haciendo uso de conceptos como regiones, zonas u otras construcciones relevantes para la empresa. Estos factores pueden contribuir a la colocación programada de pods en el entorno y pueden garantizar que los pods se ejecuten en los nodos apropiados asociados con su función.

- *Salud y escalado:* el máster de openshift también es responsable de monitorear estado y escalado de los pods según lo desee para manejar carga adicional. El máster de openshift ejecuta pruebas de vida y preparación utilizando sondas definidas por los usuarios. El máster de openshift puede detectar pods fallidos y remediar los fallos a medida que ocurren.

Capa de servicio: la capa de servicio de openshift permite que los componentes de la aplicación se comuniquen fácilmente uno con el otro. Por ejemplo, un servicio web front-end que contiene múltiples servidores web se conectaría a instancias de base de datos por comunicación a través del servicio de base de datos. Openshift de forma automática y transparente maneja el equilibrio de carga en las instancias de los servicios. En conjunto con las sondas, la capa de servicio de openshift garantiza que el tráfico solo se dirija hacia pods saludables, lo que ayuda a mantener la disponibilidad.

Almacenamiento persistente: los contenedores de Linux son nativamente efímeros y solo mantienen datos durante el tiempo que están corriendo. Las aplicaciones o los componentes de la aplicación pueden requerir acceso a un repositorio de almacenamiento persistente a largo plazo, como los motores de bases de datos. Openshift proporciona los medios para conectar pods a un almacenamiento externo, que permite utilizar aplicaciones con estado en la plataforma. El almacenamiento persistente se puede aprovisionar dinámicamente a solicitud del usuario, siempre que la solución de almacenamiento tenga una integración con openshift.

Control de acceso: se usa comúnmente para permitir el acceso externo a un clúster openshift. El operador de ingreso gestiona los controladores de ingreso. Un controlador de entrada está configurado para aceptar solicitudes externos y proxy según las rutas configuradas. Las solicitudes externas están limitadas a HTTP y HTTPS usando la indicación de nombre del servidor (SNI) y la seguridad de la capa de transporte (TLS) usando SNI, que es suficiente para aplicaciones y servicios web que funcionan sobre TLS con SNI. El acceso trabaja en sociedad con la capa de servicio para proporcionar

equilibrio de carga automatizado a pods para clientes externos. El controlador de ingreso utiliza el servicio de información de punto final para determinar dónde enrutar y equilibrar el tráfico de carga; sin embargo, no enruta el tráfico a través de la capa de servicio.

Operador de red de clúster: implementa y administra la red de clúster, incluido el complemento de interfaz de red de contenedor (CNI) y la red definida por software (SDN) seleccionado para el clúster durante la instalación.

Red definida por software (SDN) de openshift: es una red de clúster unificada que permite comunicación entre pods a través de clúster de openshift. Configura una red superpuesta que una Open vSwitch (OVS). Red Hat actualmente proporciona un modo SDN para la red de pod OCP: el modo de política de red. El modo de política de red proporciona control de acceso detallado a través de reglas definidas por el usuario. Las reglas de política de red se pueden construir en un estilo de “control de acceso obligatorio”, donde todo el tráfico es denegado por defecto a menos que exista una regla explícitamente, incluso para pods o contenedores en el mismo host. El modo de política e red es el modo predeterminado.

Registro de Openshift: proporciona almacenamiento y administración integrados para compartir imágenes de contenedor, pero Openshift puede utilizar registros de contenedor existentes que cumplen con Openshift que tienen acceso a los nodos y al máster de openshift a través de la red.

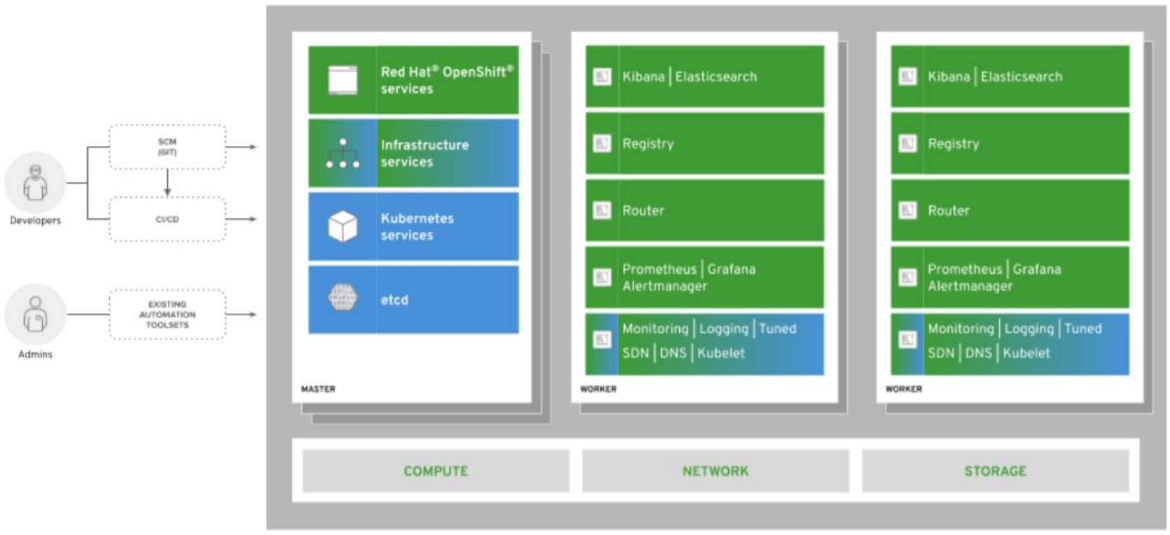


Imagen 6 - Arquitectura de alto nivel de Openshift.

Usuarios: el acceso del usuario (ej: operadores, desarrolladores, administradores de aplicaciones) a openshift se proporciona a través de interfaces estándar que incluyen la interfaz web del usuario (UI), CLI e IDE. Estas interfaces pasan por la API de manera autenticada y controlada por RBAC. Los usuarios no requieren acceso a nivel de sistema a ninguno de los hosts OCP o nodos, incluso para la depuración de aplicaciones y tareas de solución de problemas. Existen tres tipos de usuarios que pueden existir en un entorno de openshift y se nombran a continuación:

- Usuarios habituales: se crean automáticamente en el sistema al iniciar sesión por primera vez o mediante la API. También están incluidos en este tipo de usuarios los usuarios interactivos de openshift, incluidos operadores, desarrolladores y administradores de aplicaciones.

- Muchos de los usuarios del sistema se crean automáticamente cuando se define la infraestructura Openshift, principalmente con el fin de permitir que la infraestructura interactúa de forma segura con la API. Los usuarios del sistema incluyen un administrador de clúster (con acceso a todo en el entorno de Openshift), un usuario por nodo, usuarios para uso de los controladores y registros de ingreso, y varios otros.

También hay un usuario anónimo que se utiliza de forma predeterminada para solicitudes no autenticadas. Hay que tener en cuenta que incluso las solicitudes no autenticadas son gestionadas por RBAC. Las solicitudes anónimas se asignan al sistema: grupo no autenticado que está vinculado a roles con ámbito de clúster.

- Las cuentas de servicios son usuarios no humanos, generalmente asociados con proyectos, utilizados para el acceso a la API. En situaciones de automatización algunas cuentas de servicio predeterminadas se crean y asocian cuando un proyecto se crea por primera vez. Los administradores de proyectos y clústeres pueden crear cuentas de servicio adicionales para definir acceso a los contenidos de cada proyecto.

Proyectos: es un espacio de nombres de Kubernetes con anotaciones y metadatos adicionales de openshift. Es el vehículo central por el cual se gestiona el acceso a los recursos para usuarios regulares y es esencialmente el modelo de Openshift. Un proyecto permite a una comunidad de usuarios organizar y administrar su contenido de forma aislada de otras comunidades.

3.7.2 Seguridad

Openshift permite seguridad continua con capacidades de defensa en profundidad y el suministro de seguridad de software de Red Hat. Los controles de seguridad se pueden aplicar dinámicamente a la plataforma y a las aplicaciones que la plataforma soporte. Esto permite que los controles de seguridad se mantengan al día con la escala y la agilidad de las aplicaciones implementadas en la plataforma. Openshift se ejecuta en RHEL CoreOS y hace uso intensivo de las funciones de seguridad existentes integradas al sistema operativo. Red Hat gestiona los paquetes del sistema operativo y proporciona una distribución confiable de contenido. La seguridad de openshift incluye y utiliza tecnologías reforzadas como SELinux; procesos, red y almacenamiento por separado; monitoreo proactivo de los límites de capacidad (ej: CPU, disco, memoria); y comunicaciones encriptadas para soporte de infraestructura incluyendo SSH y SSL. Además openshift proporciona integración con soluciones de gestión de identidad de

terceros para admitir opciones de autenticación y autorización seguras en alineación con los requisitos de cumplimiento de la organización.

A continuación se detalla una lista de alto nivel de características y capacidades de seguridad de Openshift:

Host contenedor y plataforma multitenencia: RHEL puede gestionar multitenencia para el contenedor en tiempo de ejecución mediante el uso de espacios de nombres de Linux, SELinux, CGroups y modo de cómputo seguro para aislar y proteger los contenedores. Esto puede ser útil para mantener la separación de las cargas de trabajo de diferentes clasificaciones.

Contenido del contenedor: el catálogo de contenedores de Red Hat ofrece contenido validado de aplicaciones de Red Hat y proveedores independientes de software (ISV) certificados.

Registros de contenedores: Openshift incluye un registro de contenedores integrado que proporciona funcionalidad que soporta la automatización de construcción e implementación dentro del clúster, vinculada al RBAC de Openshift.
Dentro del contexto de una organización que necesita adherirse a la salvaguarda técnica de la regla de seguridad HIPAA, Red Hat Quay es un producto adicional que proporciona un registro con capacidades tanto para RBAC y escaneo de vulnerabilidades de aplicaciones y software en imágenes.

Creación de contenedores: Openshift se integra con Jenkins y Teflon y puede integrarse fácilmente con otras herramientas de integración continua / entrega continua (CI/CD) para administrar complicaciones y realizar código, inspección, escaneo de códigos y validación.

Implementación de contenedores: Openshift evita que los contenedores se ejecuten como root u otro nombre específico por defecto. Además, openshift habilita políticas de

implementaciones granulares que permiten operaciones, seguridad y equipos de cumplimiento para hacer cumplir el aislamiento y las protecciones de acceso.

Orquestación de contenedores: openshift integra capacidades operativas seguras para respaldar la confianza entre usuarios, aplicaciones y políticas de seguridad. Ofrece orquestación de contenedores, automatización de programación y ejecución de contenedores de aplicaciones en clústeres en máquinas físicas o virtuales a través de inclusión y extensión de Kubernetes. Se puede usar para guiar cómo y dónde se pueden implementar contenedores, implementación de contenedores según la disponibilidad de capacidad, cómo los contenedores pueden acceder entre sí según el mínimo privilegio, cómo es el acceso a los recursos compartidos y la administración controlada, cómo se monitorea el estado del contenedor, cómo se pueden escalar las aplicaciones automáticamente, y puede habilitar el autoservicio para desarrolladores mientras mantiene y cumple con los requisitos de seguridad.

Aislamiento de red: Openshift utiliza un enfoque SDN para proporcionar una red de clúster unificada que permite la comunicación entre pods a través del clúster de openshift. Openshift Utiliza el complemento Multus CNI para permitir el encadenamiento de complementos CNI. Esta red de pod es establecida y mantenida por openshift SDN, que configura una red superpuesta utilizando Open vSwitch (OVS). Hay un modo de SDN de políticas de red disponible en Red Hat para que el cliente configure las políticas de red. Existen otras soluciones SDN de terceros que también pueden integrarse en Openshift.

Multus es un complemento Multi CNI para admitir la función multi networking en Kubernetes utilizando objetos de red basados en la definición de recursos del cliente (CRD) en Kubernetes. Durante la instalación del clúster, el cliente configura la red de pod predeterminada. La red predeterminada maneja todo el tráfico de red ordinario para el clúster. El cliente puede definir redes adicionales basadas en los complementos CNI disponibles y adjuntar una o más de estas redes a sus pods. EL cliente puede definir más de una red adicional para su clúster, dependiendo de sus necesidades.

Esto le da al cliente flexibilidad al configurar pods que entregan funcionalidad de red como conmutación o enrutamiento.

La capacidad de crear redes adicionales permite al cliente mejorar el aislamiento de la red, que incluye plano de datos y separación del plano de control. El cliente puede enviar tráfico sensible a un plano de red que se administra específicamente por consideraciones de seguridad y puede separar datos privados que no deben compartirse.

Seguridad de los datos: Openshift proporciona acceso e integración con una amplia gama de plataformas de almacenamientos y protocolos, que permiten que las aplicaciones almacenen y cifren de forma segura los datos de la aplicación. Openshift proporciona la opción para cifrar datos confidenciales almacenados en etcd.

Servicio de malla: una característica opcional de Openshift es el servicio de Malla de Red Hat. Esto proporciona una plataforma para información de comportamiento y control operativo sobre los microservicios en red en una malla de servicios.
Una malla de servicios es una red de microservicios que conforman aplicaciones en un microservicio de arquitectura distribuida y las interacciones entre esos microservicios. Cuando una malla de servicio crece en tamaño y complejidad puede ser más difícil de entender y gestionar. Con este servicio, el cliente puede conectar, proteger y monitorear microservicios en el entorno. El servicio de malla agrega una capa transparente en las aplicaciones distribuidas existentes sin requerir ningún cambio en el código del servicio. El cliente puede agregar soporte a los servicios mediante la implementación de un proxy especial que intercepta todas las comunicaciones de red entre microservicios. La malla de servicios se configura y administra utilizando las características del plano de control.

EL servicio de malla ofrece a los clientes una forma de crear una red de servicios desplegados para proporcionar descubrimiento, equilibrio de carga, autenticación de servicio a servicio, recuperación de fallas, métricas y capacidades de monitoreo.

Funciones operativas más complejas proporcionadas por el servicio incluyen pruebas A/B,versiones candidatas, limitación de velocidad, control de acceso y autenticación de extremo a extremo.

Gestión de API: Openshift y los servicios de malla pueden integrarse para autenticar, asegurar y limitar el acceso de la API a aplicaciones y servicios.

3.8 Resumen

En este capítulo se presentaron las fases de despliegue ágil aplicadas en eSidif, para dar paso a una nueva modalidad de despliegue a través de contenedores. Sobre el final del capítulo se hizo especial hincapié en la arquitectura y seguridad de openshift, que es la plataforma basada en contenedores aplicada en los nuevos proyectos, y la elegida para la evolución del proceso de despliegue.

El próximo capítulo se enfocará en la aplicación de openshift a un proyecto en particular que está comunicado con eSidif y el cual sirvió para hacer pruebas de concepto y aceptación de openshift.

Capítulo 4 – Casos de prueba en subsistemas conectados a eSidif

4.1 Contexto

En el capítulo anterior se presentó Openshift como una tecnología novedosa aplicable al proceso de despliegue. Tal como se mencionó, Openshift da la posibilidad de hacer integraciones continuas, lo que genera despliegues ágiles, y el objetivo de este capítulo es mostrar en detalle cómo se ejecuta el proceso de despliegue. En este caso particular el foco va a estar puesto en uno de los tantos subsistemas conectados a eSidif.

4.2 Sireco

En este apartado se presentará un subsistema conectado a eSidif que usa openshift como tecnología de despliegue ágil.

La aplicación Sireco permite realizar el Registro de Cuentas Oficiales del Sector Público Nacional. Estas cuentas pertenecen a Empresas Públicas, Fondos Fiduciarios, Universidades Nacionales, Otros Entes y Organismos de la APN (antes llamados SAF: servicios administrativos financieros), radicadas en bancos nacionales y extranjeros.

La aplicación SIRECO permite realizar el Registro de las Cuentas Oficiales del Sector Público Nacional. Estas cuentas de Empresas, Fondos Fiduciarios, Universidades, Otros Entes y Organismos de la APN (ex SAF), radicadas en bancos nacionales y extranjeros. Para ello se cuenta con diferentes roles, por un lado están los Organismos (SAF, Empresas, Fondos Fiduciarios, Otros Entes y Universidades) que cargan y actualizan la información de sus cuentas, y por otro lado TGN que valida la información cargada por los Organismos y también puede administrarla.

Cuando los Organismos necesitan abrir una cuenta bancaria solicitan a TGN la autorización de apertura de la misma para luego realizar los trámites de apertura en la entidad bancaria correspondiente. Luego, el Organismo debe informar en SIRECO los datos de la nueva cuenta abierta en el Banco (nro. de cuenta, CBU, etc.), como así también la fecha de apertura.

Para obtener algunos datos de entidades básicas, se conecta a eSidif a través de servicios REST para mantener datos actualizados, como por ejemplo: bancos, paises, ciudades, etc.

4.3 Pipeline de Jenkins

El Pipeline de Jenkins es un conjunto de plugins que admiten la implementación e integración de tuberías de entregas continuas en Jenkins.

Integración continua es una manera de llamar al proceso que incluye llevar el software desde el versionado hasta sus usuarios. Cada cambio de código fuente pasa por un proceso complejo hasta su lanzamiento. Este proceso implica construir el software de manera confiable y repetible, así como también progresar la compilación a través de múltiples etapas de prueba e implementación.

La definición de un pipeline se escribe en un archivo de texto, llamado JenkinsFile, que puede commitearse en el repositorio de control del proyecto. Crear el JenkinFile y commitearlo tiene los siguientes beneficios:

- Creación automática de un proceso de construcción de un pipeline para todos lo branchs de trabajo y solicitudes de actualización.
- Revisión e iteración de código.
- Auditoría para el pipeline.
- Única fuente de verdad para el pipeline, que puede ser vista y editada por diferentes miembros del proyecto.

Además, según la documentación de Jenkins [7], el pipeline tiene las siguientes características:

- Código: las canalizaciones se implementan en código y generalmente se registran en el control de origen, lo que brinda a los equipos la capacidad de editar, revisar e iterar sobre su canal de entrega.
- Duradero: los pipelines pueden sobrevivir a reinicios planificados y no planificados por parte de Jenkins.
- Pausable: los pipeline pueden detenerse y esperar opcionalmente la entrada o aprobación humana antes de continuar su ejecución.

- Versátil: los pipelines son compatibles con los complejos requisitos de integración continua del mundo real, incluida la capacidad de bifurcar/unir, realizar bucles y realizar trabajos en paralelo.
- Extensible: el complemento pipeline admite extensiones personalizadas a su DSL (lenguaje específico de dominio) y múltiples opciones para la integración con otros complementos.

El siguiente diagrama de flujo muestra un escenario de integración continua modelado con un pipeline.

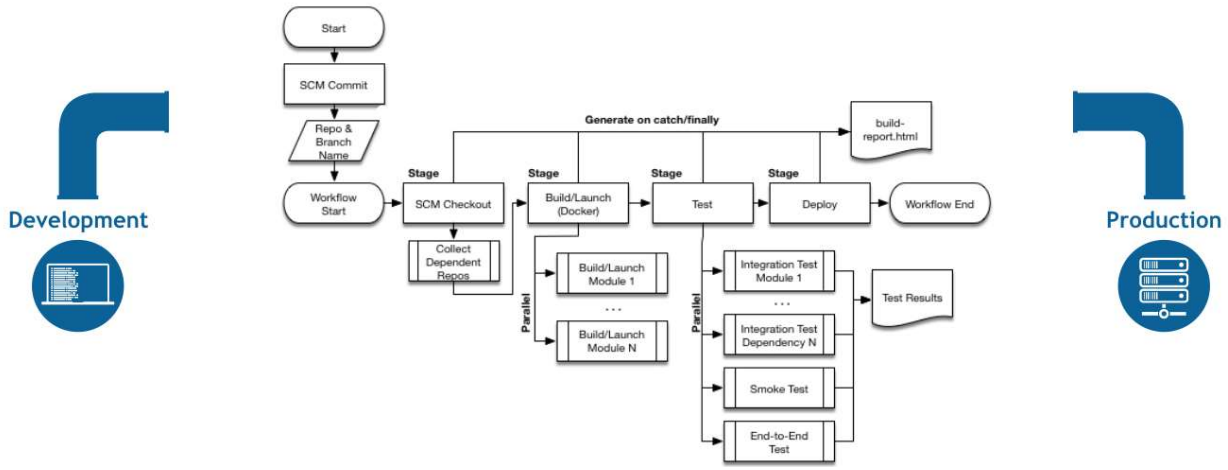


Imagen 7 - diagrama de flujo

De este diagrama se desprenden conceptos ligados al pipeline:

- Nodo: es una máquina que forma parte del entorno de Jenkins y es capaz de ejecutar una canalización.
- Etapa: es un bloque que define un subconjunto distinto conceptualmente de tareas realizadas a través de todo el pipeline (como por ejemplo “construir”, “probar”, “implementar”) que muchos plugins utilizan para visualizar o presentar el estado del pipeline.
- Paso: es una tarea. Simplemente le dice a Jenkins qué hacer en un momento determinado del proceso.

4.4 Pipeline aplicado a SIRECO

En este apartado se verán las diferentes etapas de un pipeline aplicado a sireco.

Las etapas definidas son:

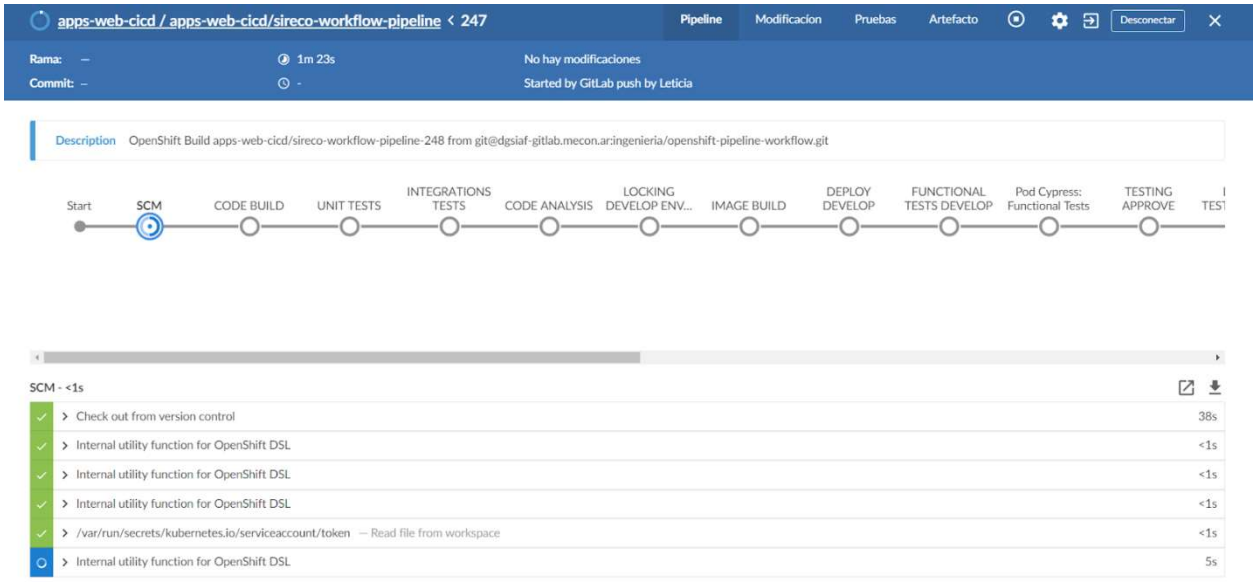


Imagen 8 – pipeline de SIRECO

SCM: es el estado inicial, SCM es en inglés el 'Source Code Management', el gestor de código fuente. Aquí Jenkins obtiene su pipeline de SCM, que será el repositorio GIt clonado localmente.

Imagen 9 – pipeline de SIRECO

CODE BUILD: Se compila el código y se realiza el armado del artefacto ejecutable que luego se incluirá en una imagen de docker.

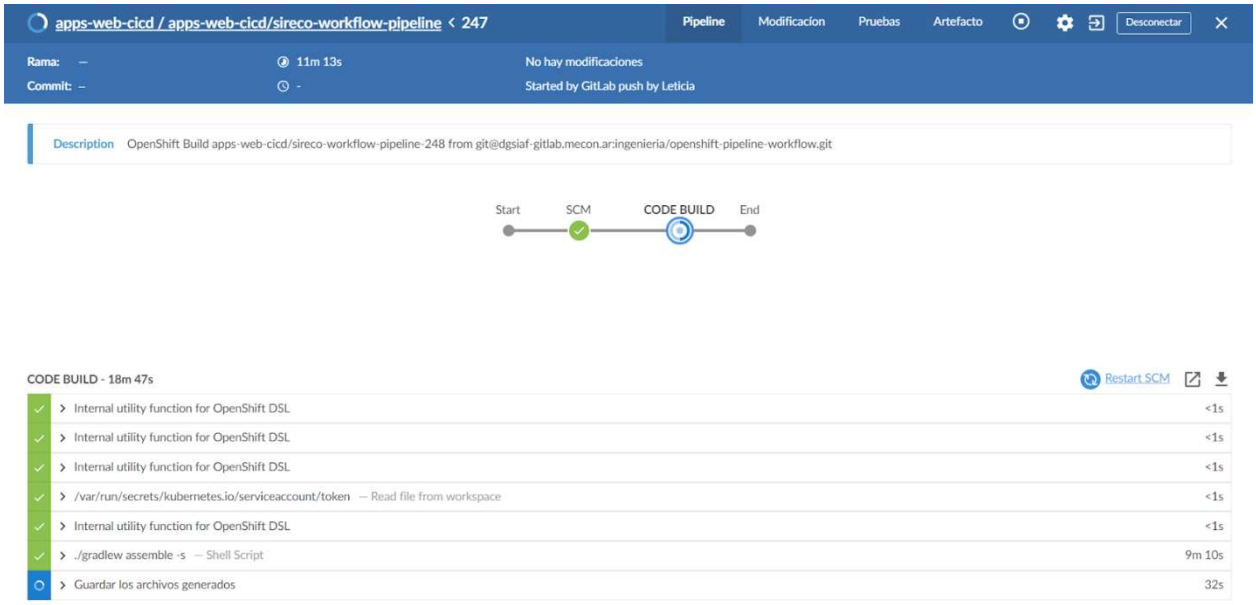


Imagen 10– pipeline de SIRECO

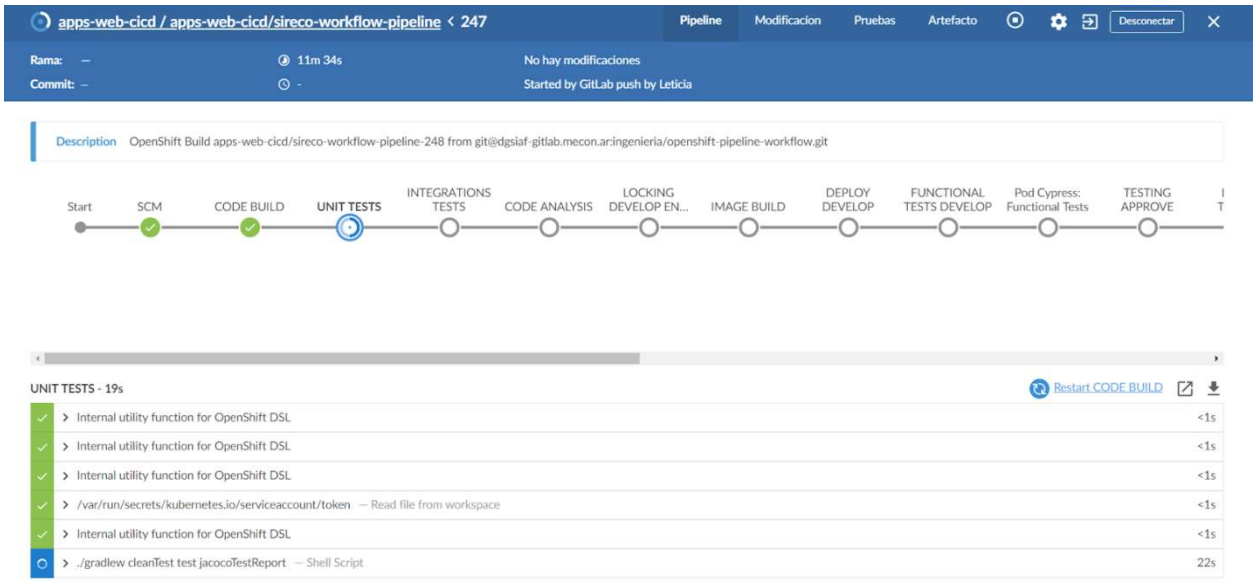


Imagen 11 – pipeline de SIRECO

UNIT TEST: se ejecutan los test de unidad, se prueba una porción de código particular, por ejemplo un método.

Imagen 12 - detalle del paso de unit test.

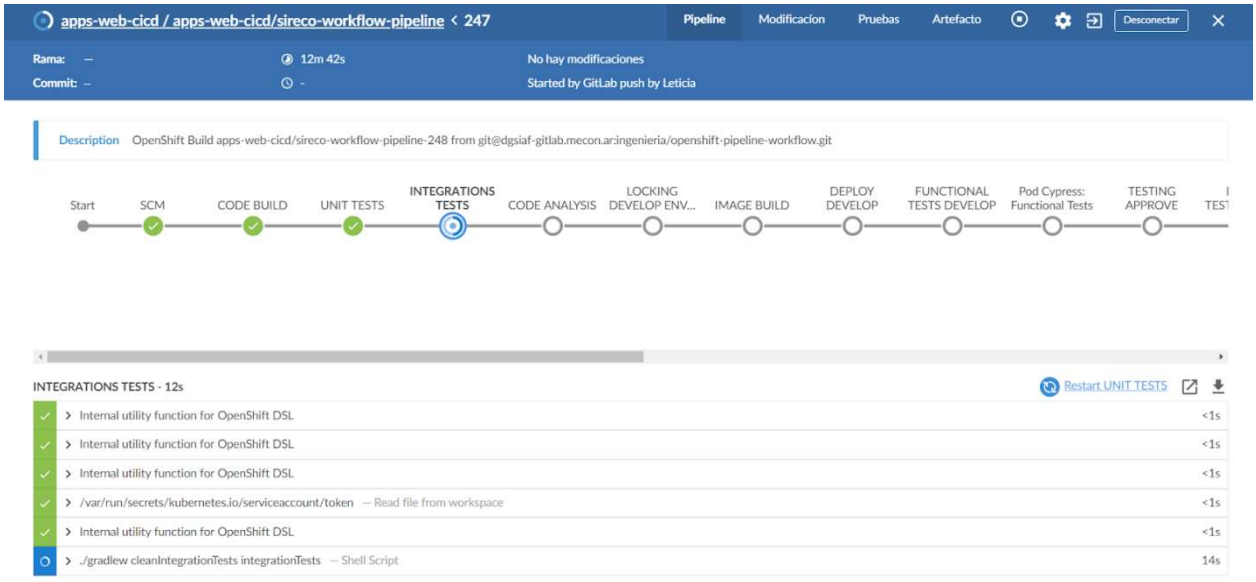


Imagen 13 – pipeline de SIRECO

INTEGRATION TESTS: se ejecutan los test de integración, conceptualmente resuelven pruebas de caja negra; el test le pega directamente a los endpoints o funciones que la aplicación publica, se requiere levantarla, correr scripts, cablear beans, para que luego el test consuma alguna funcionalidad desde afuera. Se prueba funcionalidad completa.

Imagen 14 – pipeline de SIRECO

CODE ANALYSIS: se ejecuta un análisis de código estático, en particular para este proyecto en el pipeline se usa jacoco (JAva COde COverage), con los resultados publicados en sonarqube

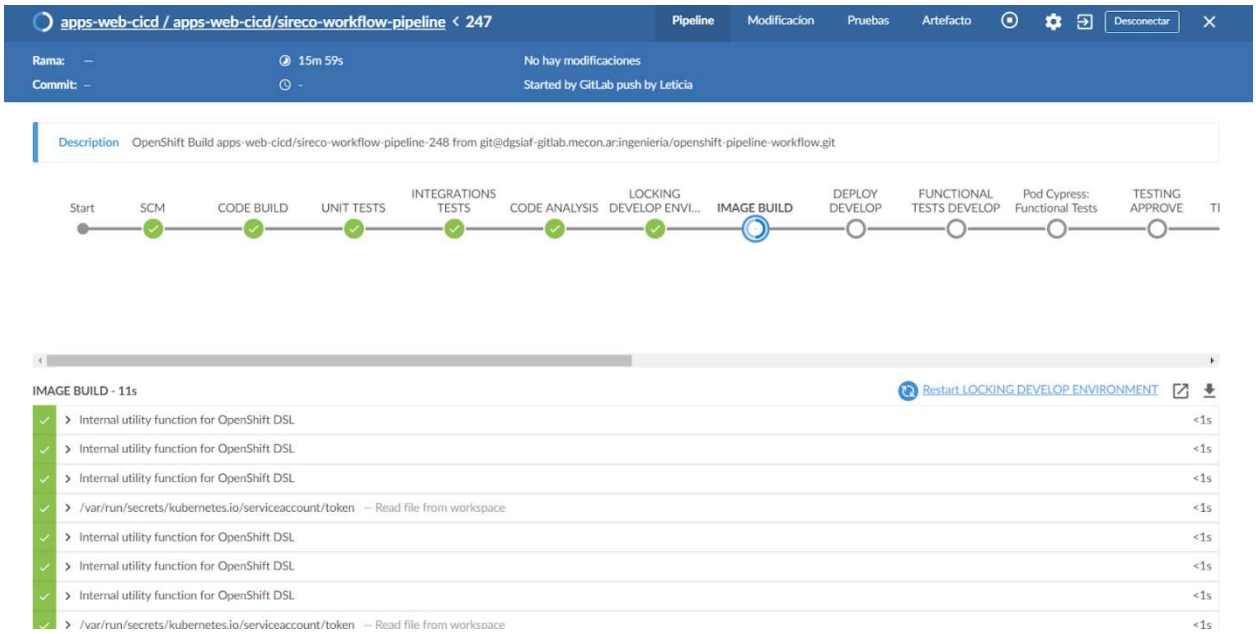


Imagen 15 – pipeline de SIRECO

IMAGE BUILD: se genera la imagen con el artefacto ejecutable incluido, la imagen docker es el template para poder instanciar múltiples contenedores.

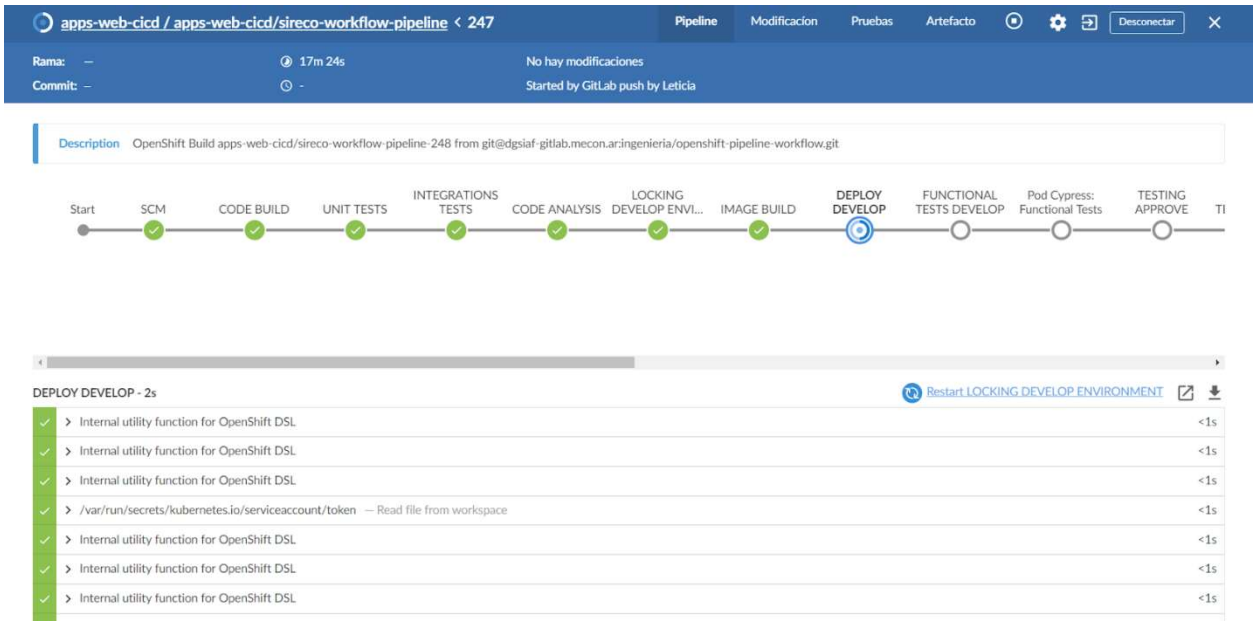


Imagen 16– pipeline de SIRECO

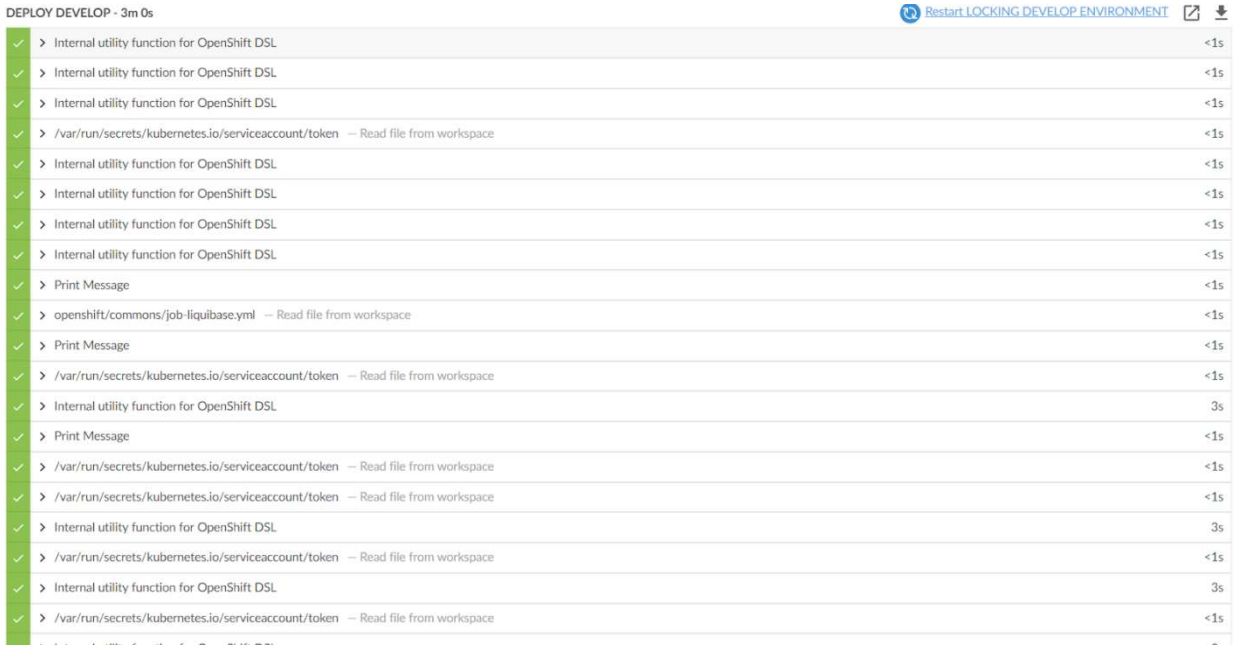


Imagen 17 – pipeline de SIRECO

DEPLOY DEVELOP: se efectiviza la implementación. Esta se realiza en dos partes: primero se ejecutan los scripts de bases de datos, usando tecnologías de migración, en este proyecto en particular se usa liquibase, y luego se hace ”deploy” en okd.

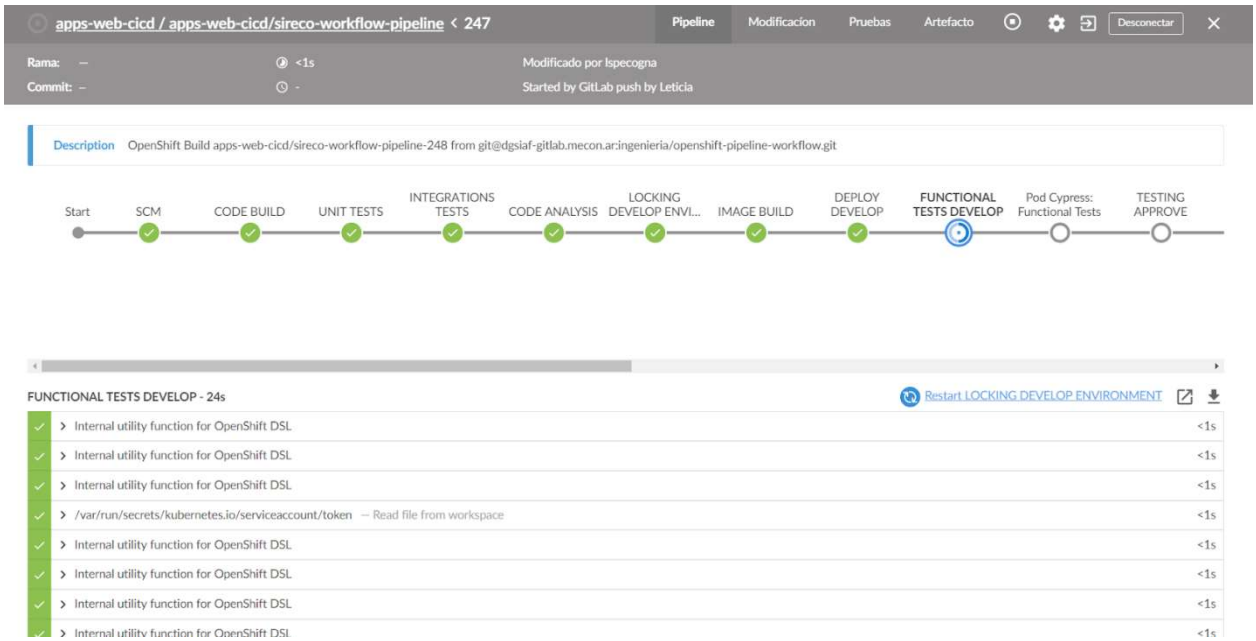


Imagen 18 – pipeline de SIRECO

FUNCTIONAL TESTS DEVELOP: se arma el ambiente para poder ejecutar test end-to-end (e2e).

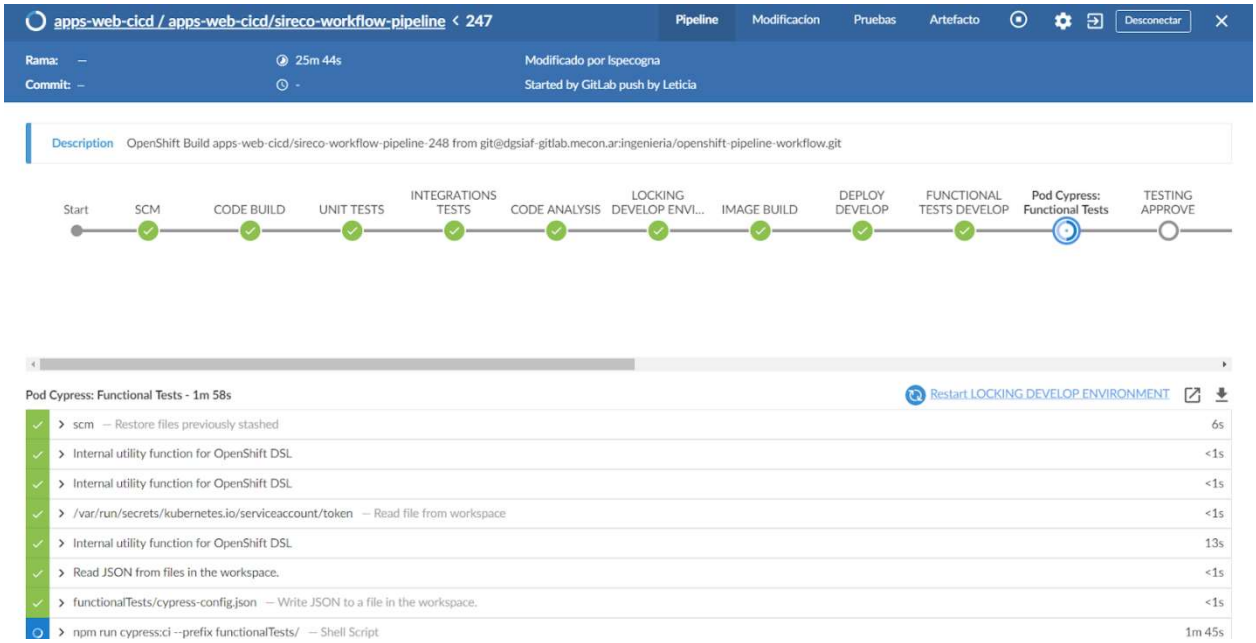


Imagen 19 – pipeline de SIRECO

Pod Cypress Functional Test: se ejecutan los test mencionados en el paso anterior (e2e), en particular se utiliza Cypress para desarrollar los test e2e para testing automático.

TESTING APPROVE: se espera la aprobación de testing para instalar la aplicación en un ambiente interno. Testing es notificado por mail, y el pipeline detiene su ejecución hasta que esto ocurra. Podría estar o no dependiendo de las reglas definidas.

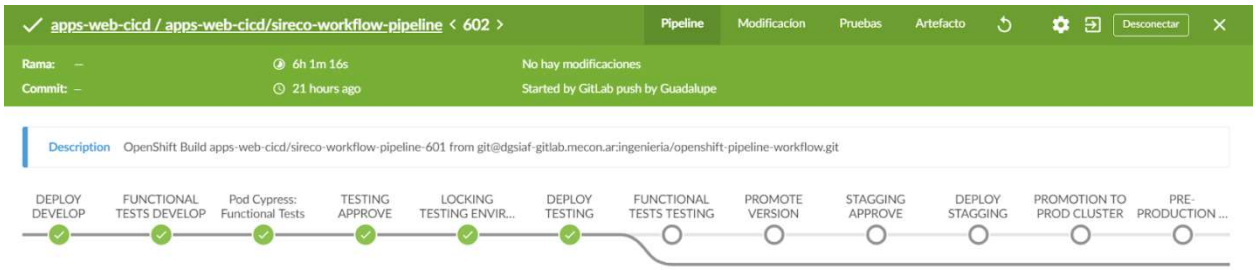


Imagen 20 – pipeline de SIRECO

DEPLOY TESTING: instalación en ambiente de testing, previo envio de mail y confirmación de usuarios habilitados.

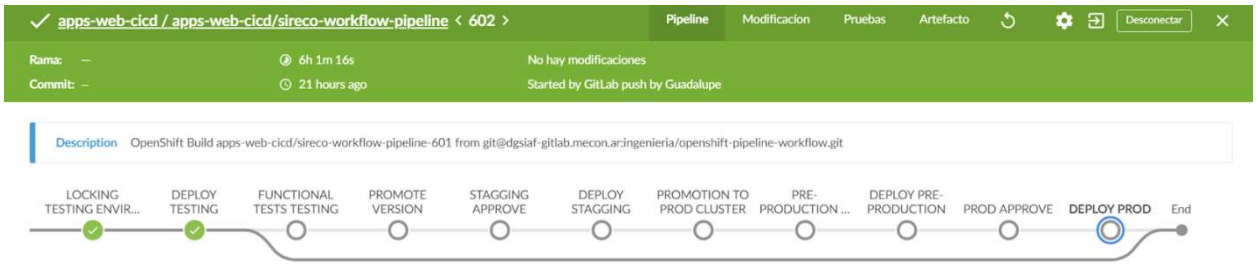


Imagen 21 – pipeline de SIRECO

STAGGING APPROVE: estado similar al anterior, se espera la aprobación para la instalación en un ambiente de aceptación, la siguiente DEPLOY STAGGING es la instalación.

Los estados siguientes son la preparación de pre-producción y prod, que es el despliegue definitivo. Para todos estos despliegues, el procedimiento es el mismo que en testing, cada área tiene su grupo autorizado para aprobar la instalación en el ambiente solicitado.
En la Imagen 21, el pipeline llegaba hasta el entorno de testing.

4.5 Configuraciones

En sireco se realizaron las siguientes configuraciones, para que, dependiendo del tag generado, la entrega llegue a diferentes instancias. A saber:

- X.Y.betaZ: la entrega llegará a un entorno de testing, esperará la aceptación por parte de este equipo para la instalación en su ambiente y realizará las pruebas preparadas.

- X.Y.RCZ: la entrega llegará a un ambiente de “stagin”, o pre-producción, donde podrán realizarse las pruebas de aceptación con los usuarios. Es importante

aclarar que esta entrega fue primero aceptada por testing y promovida a este entorno.

- X.Y.Z: la entrega se instalará en producción. Ya estará disponible a los usuarios finales. Antes de llegar a producción, esta entrega pasó por testing y stagging, es la versión definitiva.

Cada una de estas instancias tendrá a un encargado de “aceptar” la instalación en su ambiente, llegado el momento, el pipeline envía mails con el pedido de instalación y el proceso se pausa hasta la confirmación o rechazo de dicha solicitud.

4.5.2 Servicio de mail

Todo el proceso implementa la notificación vía mail. Estos mails pueden ser de solicitud de instalación en un ambiente específico, ya sea testing, mostrado en la imagen 22, aceptación o producción. Como se mencionó anteriormente, hay usuarios configurados aprueben la instalación.

También se notifica vía email la información sobre el estado final del pipeline, es decir, informar si terminó su ejecución de manera exitosa (imagen 23) o con error (imagen 24), en este último caso dando detalles del error.

Imagen 22 - mail de pedido de instalación en ambiente de testing.

Imagen 23 - mail de fin de ejecución exitosa

Imagen 24 - mail de ejecución con error

4.6 Resumen

En este capítulo se presentó el caso concreto de aplicación de despliegue ágil en un subsistema web que consume servicios de esidif. Este es el primer proyecto en el que se está usando esta modalidad, como prueba piloto, que marcará precedente en los nuevos subsistemas de eSidif que están comenzando.

Del lado del desarrollador sólo se necesita realizar un commit (un push al repositorio en términos GIT) para “disparar” el proceso del pipeline y generar la integración continua. Cuando se decida armar una entrega, ya sea a testing, pre-producción o producción, sólo será necesario generar la etiqueta que determine a qué ambiente llegará.

Capítulo 5 – Conclusiones

El despliegue de un sistema es una etapa de suma importancia, cargada de expectativas para el usuario que está a la espera, y de gran satisfacción para quienes del otro lado, se organizaron, planificaron y ejecutaron dicha planificación.

eSidif es un proyecto a gran escala, que comenzó hace algunos años, y evolucionó su organización hasta lo que tiene hoy, que es una convivencia entre metodologías tradicionales, más duras, y metodologías ágiles.
No hay una manera determinar la metodología ideal para un proyecto, cada una tiene sus puntos a favor y en contra, sumado a la complejidad que implica hacer un cambio en un proyecto que está avanzado. Ahí donde se intenta incorporar los aspectos más favorables de la nueva metodología para que el cambio sea progresivo y de bajo impacto para los equipos.
Acá queda demostrado que la convivencia es posible, a pesar de contar con dos metodologías diferentes, estas logran complementarse.
RUP se enfoca en la documentación, tiene resistencia a los cambios y se enfoca en la organización, el cliente sólo se involucra al principio del proyecto para llegar a un acuerdo y elaborar un contrato y luego queda a la espera del producto final, mientras que las metodologías ágiles se concentran en la interacción entre los participantes, la colaboración con el cliente, la respuesta al cambio y la comunicación constante.

El momento de despliegue representa el hito final en el proceso de desarrollo de software, alcanzado después de innumerables reuniones de definición, dailies, entregas iterativas a testing con devoluciones, corrección de errores, y reuniones de retrospección, que acompañaron el proceso.

Al mismo tiempo que eSidif sigue incorporando funcionalidad, van surgiendo otros proyectos que lo consultan: subsistemas ligados a eSidif, cada uno con sus particularidades y subdominios. Un nuevo proyecto abre la posibilidad de continuar con el esquema de trabajo que se viene trabajando, o bien hacer un cambio y experimentar,

ya sea en el plano de la tecnología aplicada como de las metodologías usadas. El caso de sireco en particular, que fue mencionado en este trabajo, es el ejemplo de un subsistema que consulta a eSidif. Fue desarrollado en tecnologías web, y con despliegues continuos, una manera completamente diferente, moderna y novedosa, diferente de eSidif.

Sireco ejecuta integraciones continuas disparadas de un simple *push* (sentencia que hace una entrega del código local al repositorio compartido). Esta acción activa un pipeline previamente configurado, que indicará hasta qué estadío llegará el código integrado. En todo momento se encuentra la aplicación disponible y actualizada, y las diferentes versiones son promovidas hasta llegar finalmente a ser desplegadas en producción.
Esta incorporación viene acompañada de planificación, reuniones, devoluciones constantes, con mucha comunicación entre los equipos, incluidos los usuarios, y ejecutada por herramientas que acompañan el proceso.

Al tratarse eSidif de proyecto tan grande, se permiten cambios y modernizaciones en cada pequeño proyecto que surge, que sirve también de prueba piloto a los que vendrán, además de evolucionar el proceso de ingeniería y estar a la altura de los avances tecnológicos.
No existen metodologías perfectas ni exactas para un proyecto, la ideal es la que lo hace funcionar.

REFERENCIAS

[1] https://dgsiaf.mecon.gov.ar/organigrama/

[2] https://obsbusiness.school/int/blog-project-management/metodologias-agiles/metodologia-agile-cuales-son-los-12-principios-de-su-modelo

[3] https://agilemanifesto.org/

[4] Beatriz González, 2017 - https://prezi.com/dqgwnt8ajmpz/metodologias-agiles-vs-rup-outsourcing-y-adquisicion-de-ti/?utm_campaign=share&utm_medium=copy

[5]https://www.aplyca.com/en/node/181

[6] Jason Macallister & Terilyn Floyd Carney - Red Hat OCP Applicability Guide for HIPAA Security Rule https://www.redhat.com/cms/managed-files/ve-openshift-hipaa-coalfire-analyst-materal-f23549-202005-en.pdf

[7]https://www.jenkins.io/doc/book/pipeline/

[8] Ian Sommervile, Ingeriería de software, 7ma edición, 2005.