10. Plataformas para el Despliegue de Sistemas Distribuidos

10.1. Introducción y objetivos

En este tema se estudian algunas de las plataformas que nos ayudan a automatizar el despliegue de sistemas distribuidos.

En primer lugar, estableceremos el contexto de conceptos como integración y distribución continuas (CI/CD), desarrollo y operaciones (DevOps), contenedores y monitorización de sistemas, entre otros. Detallaremos las principales características de las arquitecturas basadas en eventos (*event-driven architecture*, EDA), como una solución de diseño para poder dar respuesta a unas elevadas prestaciones de escalabilidad y rendimiento.

Posteriormente, veremos algunas de las plataformas y herramientas que facilitan el despliegue y la monitorización de estos sistemas en diversas infraestructuras computacionales junto con los beneficios que presentan.

Con el estudio de este tema, pretendemos alcanzar los siguientes objetivos:

* Comprender el contexto, el valor y la utilidad que tiene la automatización del despliegue de sistemas distribuidos.
* Conocer los principales conceptos y técnicas como CI/CD, DevOps y EDA.
* Introducir algunas herramientas de automatización para el despliegue de sistemas distribuidos y middleware para EDA.
* Conocer algunas plataformas de soporte a la monitorización de sistemas distribuidos.

10.2. Contexto de las plataformas para el despliegue de sistemas distribuidos

En el ámbito de la ingeniería del software, una de las tareas críticas es integrar los desarrollos realizados, empaquetar la solución y desplegarla para hacerla accesible a los usuarios en un entorno operativo. Existen una serie de conceptos, técnicas, métodos y plataformas que facilitan la automatización de estas actividades para minimizar esfuerzos, errores y agilizar todo el proceso de despliegue de las aplicaciones.

DevOps (acrónimo en inglés de *development and operations*) es una combinación de los términos desarrollo y operaciones, con la intención de representar un enfoque colaborativo o compartido de las tareas realizadas por los equipos de desarrollo de aplicaciones y operaciones de tecnologías de la información (TI) de una empresa. En su significado más amplio, DevOps es una filosofía que promueve una mejor comunicación y colaboración entre estos equipos y también en una organización. En su interpretación más estrecha, DevOps describe la adopción de desarrollo de software iterativo, automatización e implementación y mantenimiento de infraestructura programable. El término también cubre los cambios culturales, como generar confianza y cohesión entre los desarrolladores y los administradores de sistemas y alinear los proyectos tecnológicos con los requisitos comerciales. DevOps puede cambiar la cadena de entrega de software, los servicios, los roles laborales, las herramientas TI y las mejores prácticas. Si bien DevOps no es una tecnología, sus entornos generalmente aplican metodologías y tecnologías como (Courtemanche, Mell y Gillis, 2021):

* Herramientas de integración y distribución continuas (*continuous integration/continuous delivery,*CI/CD), con énfasis en la automatización de tareas.
* Sistemas y herramientas que respaldan la adopción de DevOps, incluido el monitoreo en tiempo real, la gestión de incidentes, la gestión de configuración y las plataformas de colaboración.
* Computación en la nube, microservicios y contenedores implementados simultáneamente con metodologías DevOps.

La integración y la distribución continuas (CI/CD) es un método para distribuir las aplicaciones a los clientes con frecuencia mediante el uso de la automatización en las etapas del desarrollo de aplicaciones. Los principales conceptos que se le atribuyen son la integración, la distribución y la implementación continuas (Red Hat, 2018):

* CI se refiere a la integración continua, que es un proceso de automatización para los desarrolladores. El éxito de la CI implica que se diseñen, prueben y combinen los cambios nuevos en el código de una aplicación con regularidad en un repositorio compartido. Supone una solución al problema de que se desarrollen demasiadas divisiones de una aplicación al mismo tiempo, que luego podrían entrar en conflicto entre sí.
* CD se refiere a la distribución o la implementación continuas y se trata de conceptos relacionados que suelen usarse indistintamente. Ambos se refieren a la automatización de las etapas posteriores del proceso, pero a veces se usan por separado para explicar hasta dónde llega la automatización. Como distribución continua se refiere a que los cambios que implementa un desarrollador en una aplicación se someten a pruebas automáticas de errores y se cargan en un repositorio para que luego el equipo de operaciones pueda implementarlos en un entorno de producción en vivo. Es una solución al problema de la falta de supervisión y comunicación entre los equipos comerciales y de desarrollo, así que su propósito es garantizar que la implementación del código nuevo se lleve a cabo con el mínimo esfuerzo. Como implementación continua hace referencia al lanzamiento automático de los cambios que implementa el desarrollador desde el repositorio hasta la producción, para ponerlos a disposición de los clientes. Así ya no se sobrecarga a los equipos de operaciones con procesos manuales que retrasan la distribución de las aplicaciones.

El ciclo de vida del desarrollo de sistemas finaliza con su despliegue en un entorno operacional o productivo, lo que se conoce como *delivery* o puesta en producción. Este proceso ha tenido una evolución en los últimos años:

* Despliegue tradicional: las organizaciones ejecutaban aplicaciones en servidores físicos. Era la única forma de desplegar aplicaciones y no había forma de definir los límites de recursos para las aplicaciones en un servidor físico y esto causaba problemas de asignación de recursos. Por ejemplo, si varias aplicaciones se ejecutan en un servidor físico, puede haber instancias en las que una aplicación consuma la mayoría de los recursos y, como resultado, las otras aplicaciones tengan un rendimiento inferior. Una solución para esto sería ejecutar cada aplicación en un servidor físico diferente, pero esta opción no era factible en muchas ocasiones ya que era costoso para las organizaciones mantener muchos servidores físicos.
* Despliegue en entornos virtuales: como solución a los problemas de los servidores físicos, se introdujo la virtualización. Esto nos permite ejecutar varias máquinas virtuales (*virtual machines,*VM) en la CPU de un solo servidor físico. La virtualización permite que las aplicaciones se aíslen entre VM y proporciona un nivel de seguridad, ya que otra aplicación no puede acceder libremente a la información de una aplicación. La virtualización permite una mejor utilización de los recursos en un servidor físico y permite una mejor escalabilidad porque una aplicación se puede agregar o actualizar fácilmente y reduce los costos de hardware. Con la virtualización, se puede presentar un conjunto de recursos físicos como un clúster de máquinas virtuales desechables. Cada VM es una máquina completa que ejecuta todos los componentes, incluido su propio sistema operativo, sobre el hardware virtualizado.
* Despliegue en contenedores: los contenedores son similares a las máquinas virtuales, pero tienen propiedades de aislamiento relajadas para compartir el sistema operativo (SO) entre las aplicaciones. Por lo tanto, los contenedores se consideran ligeros. Al igual que una VM, un contenedor tiene su propio sistema de archivos, cuota de CPU, memoria, espacio de proceso y más. Como están desacoplados de la infraestructura subyacente, son portátiles entre nubes y distribuciones de SO. De esta forma podemos encapsular nuestra aplicación junto con todas las dependencias, librerías, herramientas o sistemas externos que necesite para operar.

En el contexto de los sistemas distribuidos, para poder dar respuesta a las necesidades de escalabilidad y rendimiento, además de las arquitecturas de microservicios se han vuelto populares aquellas basadas en eventos (*event-driven architectures,*EDA).

Una arquitectura basada en eventos es una arquitectura de software y un modelo para el diseño de aplicaciones. Con un sistema controlado por eventos, la captura, comunicación, procesamiento y persistencia de eventos son la estructura central de la solución, lo que difiere del modelo tradicional basado en un modelo petición-respuesta cliente-servidor o entre servicios. Su principal ventaja es que facilita un acoplamiento mínimo, lo que la convierte en una buena opción para las arquitecturas de aplicaciones modernas y distribuidas, ya que los productores de eventos no saben qué eventos están escuchando los consumidores para un evento, y el evento no sabe cuáles son las consecuencias de su ocurrencia.

En estos modelos de arquitectura, el concepto de evento es la base que coordina los diversos bloques funcionales. Un evento es cualquier suceso significativo o cambio de estado del hardware o software del sistema. No es lo mismo que una notificación de evento, que es un mensaje o notificación enviada por el sistema para notificar a otra parte del sistema que se ha producido un evento. Nos referimos, por tanto, a evento como al contenido del mensaje que será notificado entre las partes. La fuente de un evento puede ser de entradas internas o externas. Los eventos pueden generarse a partir de un usuario, como un clic del ratón o una pulsación de tecla, una fuente externa, como la salida de un sensor, o provenir del sistema, como cargar un programa o que se produzca un error.

Los principales actores de una arquitectura basada en eventos son:

* El productor de eventos: detecta un evento y lo representa como un mensaje. No conoce el consumidor del evento o el resultado del procesado de este.
* El consumidor de eventos: generalmente a través de una plataforma de mediación de mensajes o eventos basada en canales, reciben los eventos y realizan las actuaciones pertinentes de forma asíncrona.
* La plataforma de procesamiento de eventos: es el canal de comunicación o bus de mensajes que facilita la interacción desacoplada, distribuida e independiente de la tecnología entre productores y servidores.

Existen dos grandes formas de gestionar una arquitectura basada en eventos:

* Modelo publicación-subscripción: se dispone de una infraestructura de mensajería basada en suscripciones a un flujo de eventos. Con este modelo, después de que ocurre o se publica un evento, se envía a los suscriptores que necesitan ser informados.
* Modelo de transmisión de eventos: en este modelo los eventos se escriben en un registro. Los consumidores de eventos no se suscriben a un flujo de eventos, sino que pueden leer desde cualquier parte de la transmisión y pueden unirse a la transmisión en cualquier momento.

Una vez que hemos desplegado nuestro sistema en un entorno productivo, es necesario monitorizar la infraestructura y los componentes lógicas para asegurar que el sistema esté funcionando. En aplicaciones de reducido tamaño que operan en una infraestructura simple es una tarea sencilla, que puede llegar a ser extremadamente compleja en el caso de sistemas distribuidos desplegados sobre múltiples infraestructuras computacionales, para lo que es necesario disponer de plataformas y herramientas que automaticen la monitorización y lancen alertas.

Dentro del ámbito de la monitorización podemos supervisar diversos aspectos:

* Supervisión del servidor en tiempo real.
* Supervisión del rendimiento de la red.
* Supervisión de contenedores.
* Supervisión de servicios de infraestructura en la nube, como mediadores de mensajes, sistemas de registro, bases de datos *o*API*gateways* entre otros.
* Supervisión de los componentes de la propia aplicación, como servicios o*front-ends*.

10.3. Valor y utilidad de las plataformas para el despliegue de sistemas distribuidos

Las plataformas que nos facilitan la automatización del despliegue de sistemas distribuidos presentan una serie de beneficios, tanto en su visión global como a nivel específico en algunas de las fases, como CI/CD, enfoque DevOps, uso de contenedores, arquitecturas de eventos y monitorización de sistemas.

El uso de plataformas de integración y la entrega continuas presentan una serie de beneficios como:

* Cambios de código más pequeños: permite integrar pequeños fragmentos de código a la vez que son más simples y fáciles de manejar que grandes fragmentos de código y, como tales, tienen menos problemas que deban repararse en una fecha posterior.
* Aislamiento de errores: es la práctica de diseñar sistemas de tal manera que cuando ocurre un error, los resultados negativos tienen un alcance limitado. El diseño de un sistema con CI/CD garantiza que los aislamientos de fallos sean más rápidos de detectar y más fáciles de implementar. Los aislamientos de fallos combinan el monitoreo del sistema, la identificación de cuándo ocurrió el error y la detección de su ubicación. Por lo tanto, las consecuencias de los errores que aparecen en la aplicación tienen un alcance limitado.
* Tiempo medio de resolución más rápido: CI/CD reduce el tiempo de resolución de errores porque los cambios de código son más pequeños y los aislamientos de fallos son más fáciles de detectar.
* Más confianza en las pruebas: al añadir nuevo código con sus pruebas y poder automatizar las pruebas de regresión, aumenta la confianza en que los posibles errores en producción disminuyan.
* Tasa de liberación más rápida: los errores se detectan más rápido y, como tales, se pueden reparar más rápido, lo que lleva a un aumento de las tasas de liberación.
* Aumento de la transparencia y responsabilidad del equipo: CI/CD es una excelente manera de obtener comentarios continuos no solo de sus clientes, sino también de su propio equipo. Esto aumenta la transparencia de cualquier problema en el equipo y fomenta la rendición de cuentas responsable. CI se enfoca principalmente en el equipo de desarrollo, por lo que los comentarios de esta parte de la canalización afectan las fallas de construcción, los problemas de fusión, los contratiempos arquitectónicos, etc. CD se enfoca más en llevar el producto rápidamente a los usuarios finales para obtener el cliente que tanto necesita. comentario. Tanto CI como CD brindan retroalimentación rápida, lo que le permite mejorar el producto de manera constante y continua.
* Reducción de costes: CI/CD reduce la cantidad de errores que pueden ocurrir en los muchos pasos repetitivos, libera tiempo del desarrollador que podría dedicarse al desarrollo del producto, ya que no hay tantos cambios de código para corregir en el futuro si el error se detecta rápidamente.
* Mayor facilidad de mantenimiento y actualizaciones: con CI/CD podemos automatizar las tareas de mantenimiento durante los períodos de inactividad, también conocidos como la hora no crítica.

El uso de un enfoque de DevOps deriva en una serie de ventajas como:

* Menores tamaños en entregas: DevOps se asocia con una entrega rápida y frecuente, así como con ciclos de desarrollo más cortos. La integración y la entrega continuas que son de vital importancia para alcanzar los objetivos de DevOps son las prácticas que permiten pequeños incrementos de entregas y ciclos de retroalimentación rápidos. Los cambios de software se realizan siempre que sea necesario para hacerlo. Básicamente, la adopción de DevOps aumenta la cadencia de los lanzamientos de aplicaciones a producción, lo que influye positivamente en el proceso de desarrollo de software.
* Equipos dedicados en lugar de silos centrados en habilidades: los departamentos de ingeniería y operaciones tradicionalmente aislados ahora pueden comunicarse entre sí sin un gran retraso entre una parte de la organización y la siguiente. Una mayor exposición y transparencia son más productivas y menos disruptivas para sus equipos, lo que permite que un solo equipo de ingenieros trabaje durante todo el ciclo de vida de la aplicación y pueda desarrollar así sus habilidades.
* Mejora de la comunicación y la colaboración: DevOps significa un gran cambio cultural que no solo elimina las barreras de la comunicación, sino que también permite a los empleados colaborar y compartir recursos fácilmente. Para que las operaciones de DevOps tengan éxito, todas las partes interesadas deben participar. Coordinar diferentes equipos para que todos puedan cooperar lo ayudará a evitar señalar con el dedo y habilitará la confianza y la colaboración.
* Reducción del tiempo de recuperación: las prácticas de DevOps son particularmente adecuadas para minimizar el impacto de los cuellos de botella, los retrocesos y los fallos de implementación. Pero cuando se encuentran errores de todos modos, se necesita menos tiempo para recuperarse de ellos. La detección de errores más rápida proporciona un ciclo de retroalimentación eficiente entre las operaciones y los desarrolladores por un lado y los clientes, por el otro.
* Mayores eficiencias: la automatización es el núcleo de la cultura DevOps, ya que ayuda a garantizar que también se observen todos los demás principios clave de DevOps. Tradicionalmente, un entorno de desarrollo se crea desde cero, pero se puede crear un entorno DevOps mediante procesos automatizados para que pueda manejar el rápido crecimiento del producto, las distintas cargas de trabajo y las necesidades cambiantes a lo largo del tiempo. La velocidad en la que se mueven las soluciones DevOps es de suma importancia.
* Fiabilidad y escalado con mínimos riesgos: garantizar la confiabilidad durante las horas pico es fundamental. Sin embargo, cuando los procesos y los sistemas se enfrentan a una mayor presión, puede resultar más difícil. Es importante desarrollar una estrategia integral que se centre en la confiabilidad, la escalabilidad y el apoyo al crecimiento continuo. DevOps permite a los equipos implementar rápidamente nuevo software mientras protegen los datos locales existentes en un entorno distribuido. Las prácticas de DevOps enfatizan la orquestación de los esfuerzos de un equipo mientras mantienen la integridad del entorno de desarrollo. Por lo tanto, un equipo podrá mantener una excelente experiencia para los usuarios mientras realiza actualizaciones y cambios en un producto.
* Ahorro de costes: las iniciativas DevOps no solo impulsan innovaciones que aumentan el valor comercial, sino que también ahorran costos de mantenimiento y actualizaciones, lo que elimina gastos de capital innecesarios y reduce la complejidad de los proyectos y los excesos de presupuesto. Podría decirse que, cuando existe una estrategia integral de DevOps, una empresa puede reducir significativamente los costos y aumentar su rentabilidad. En particular, se pueden aprovechar las economías de escala integradas en los procesos de producción y prueba. Al automatizar las canalizaciones de lanzamiento, DevOps no solo permite un lanzamiento más rápido, sino que también reduce la mano de obra y, por lo tanto, reduce los costos. En muchos casos, esto puede significar reducir algunos de los gastos generales asociados con las implementaciones, así como reducir el tiempo de inactividad o el costo de las medidas de control de cambios.

El despliegue de las aplicaciones en contenedores presenta una serie de beneficios, como:

* Creación e implementación ágil de aplicaciones: mayor facilidad y eficiencia en la creación de imágenes de contenedores en comparación con el uso de imágenes de máquinas virtuales.
* Desarrollo e integración continua: proporciona creación e implementación de imágenes de contenedores confiables y frecuentes con reversiones rápidas y eficientes (debido a la inmutabilidad de la imagen).
* Separación de preocupaciones de desarrollo y operaciones: cree imágenes de contenedores de aplicaciones en el momento de la compilación/lanzamiento en lugar del momento de la implementación, para así desvincular las aplicaciones de la infraestructura.
* Monitorización: no solo muestra información y métricas a nivel del sistema operativo, sino también el estado de la aplicación y otras señales.
* Portabilidad de distribución en la nube y el sistema operativo: se ejecuta en Ubuntu, RHEL, CoreOS, en servidores físicos y en las principales nubes públicas.
* Gestión centrada en la aplicación: eleva el nivel de abstracción desde la ejecución de un sistema operativo en hardware virtual hasta la ejecución de una aplicación en un sistema operativo utilizando recursos lógicos.
* Microservicios libremente acoplados, distribuidos, elásticos y liberados: las aplicaciones se dividen en piezas más pequeñas e independientes y se pueden implementar y administrar dinámicamente, no una pila monolítica que se ejecuta en una gran máquina de un solo propósito.
* Aislamiento de recursos: rendimiento predecible de las aplicaciones.
* Utilización de recursos: alta eficiencia y densidad.

El desarrollo de sistemas distribuidos basados en arquitecturas dirigidas por eventos (EDA) presenta una serie de beneficios:

* Mejor escalabilidad y tolerancia a fallos: una EDA es escalable porque se implementa utilizando una arquitectura moderna distribuida y tolerante a fallos. Los nodos informáticos débilmente acoplados trabajan juntos para formar una solución integrada. Si un nodo de procesamiento de eventos falla, otro cobrará vida de inmediato y continuará donde lo dejó el nodo anterior. Este entorno desacoplado y distribuido facilita la capacidad de un sistema de dar soporte a una gran carga de peticiones y recuperarse ante caídas parciales del sistema.
* Información más robusta: dado que los eventos se registran a medida que ocurren, desde fuentes como dispositivos, aplicaciones y redes vinculadas de Internet de las cosas (IoT), las empresas tienen acceso a todos los datos y el contexto que necesitan para tomar las mejores decisiones.
* Habilita la funcionalidad asíncrona: todas las funciones son completamente asíncronas, lo que descarta por completo la naturaleza lineal del diseño de software tradicional. Los desarrolladores ya no están sujetos a un código altamente específico que predetermina cada acción. En cambio, la ocurrencia de un evento determina qué acciones se tomarán a continuación. Si bien esto puede ser más difícil de documentar para los desarrolladores, proporciona una flexibilidad y una apertura sin precedentes en el diseño del sistema.
* Manejo más consistente de la concurrencia: la inmutabilidad de los eventos y el orden de los mensajes facilitan llegar a la concurrencia dentro de los sistemas distribuidos.
* Uso de recursos de red optimizado: cuando tenemos que integrar varios servicios para una funcionalidad, en lugar de tener que realizar múltiples peticiones para comprobar si se ha alcanzado un cierto evento (modo encuesta), es más eficiente no estresar la infraestructura con peticiones no necesarias y esperar la notificación asíncrona cuando el evento en sí tenga lugar.

La monitorización de un sistema desplegado sobre una infraestructura tiene, entre otros, estos beneficios:

* Proporcionar indicadores sobre interrupciones y degradación del servicio.
* Detectar cortes de servicio de forma temprana.
* Generar alertas en tiempo real sobre incidencias que agilicen su resolución.
* Registrar y analizar actividades no autorizadas o intentos de ataques que puedan comprometer la seguridad del sistema.
* Medir la escalabilidad del sistema mediante la recopilación de métricas del sistema en tiempo real.
* Obtener datos de uso del sistema en tiempo real que permitan optimizar aquellas transacciones más comunes.
* Asegurar la alta disponibilidad del sistema.
* Facilitar el cumplimiento de acuerdos de nivel de servicio (*service level agreements,*SLA) con los clientes.

10.4. Plataformas para automatizar el despliegue

A nivel de automatización de despliegue, podemos incluir aquellas plataformas de base que nos permiten ejecutar aplicaciones en sistemas operativos distribuidas o en entornos basados en contenedores y, por otro lado, aquellas herramientas que facilitan la automatización de la fase de despliegue en diversas infraestructuras.

Las principales plataformas que dan soporte a la ejecución de contenedores o sistemas operativos distribuidos y que facilitan la automatización del despliegue se exponen en la siguiente figura:

A blue and white rectangular box with text

Description automatically generated with medium confidence

Tabla 1. Plataformas para automatizar el despliegue. Fuente: elaboración propia.

Plataformas contenedoras o sistemas operativos distribuidos

A nivel de plataformas de contenedores o sistemas operativos distribuidos, las más destacadas se detallan a continuación.

Docker

Es una plataforma abierta para desarrollar, enviar y ejecutar aplicaciones sobre contenedores. Permite separar las aplicaciones de la infraestructura para que pueda entregar software rápidamente. Brinda la capacidad de empaquetar y ejecutar una aplicación en un entorno vagamente aislado llamado contenedor. El aislamiento y la seguridad le permiten ejecutar muchos contenedores simultáneamente en un *host* determinado. Los contenedores son livianos y contienen todo lo necesario para ejecutar la aplicación, por lo que no necesita depender de lo que está instalado actualmente en el *host*. Podemos compartir contenedores fácilmente mientras trabaja y asegurarse de que todas las personas con las que comparte obtengan el mismo contenedor que funciona de la misma manera. Proporciona herramientas y una plataforma para administrar el ciclo de vida de los contenedores y utiliza una arquitectura cliente-servidor.

El cliente de Docker se comunica con el demonio de Docker, que hace el trabajo pesado de crear, ejecutar y distribuir los contenedores de Docker. El cliente y el demonio de Docker pueden ejecutarse en el mismo sistema o puede conectar un cliente de Docker a un demonio de Docker remoto. Se comunican mediante una API REST, a través de *sockets* UNIX o una interfaz de red. Otro cliente es Docker Compose, que le permite trabajar con aplicaciones que consisten en un conjunto de contenedores. El uso comercial de Docker Desktop en empresas más grandes (más de 250 empleados O más de diez millones de dólares en ingresos anuales) ahora requiere una suscripción (Docker, s. f.).

A diagram of a docker host

Description automatically generated

Figura 1. Arquitectura de componentes de Docker. Fuente: Docker Docs, s. f.

Kubernetes (k8s)

Es una plataforma portátil, extensible y de código abierto para administrar cargas de trabajo y servicios en contenedores, que facilita tanto la configuración declarativa como la automatización. Los servicios, el soporte y las herramientas de Kubernetes están ampliamente disponibles. Su nombre proviene del griego y significa timonel o piloto. K8s como abreviatura resulta de contar las ocho letras entre la «K» y la «S». Google abrió el código del proyecto Kubernetes en 2014. Kubernetes combina más de quince años de experiencia de Google en la ejecución de cargas de trabajo de producción a escala con las mejores ideas y prácticas de la comunidad. Proporciona un marco para ejecutar sistemas distribuidos de forma resiliente. Algunas de sus características son:

* El descubrimiento de servicios y el balanceo de carga, ya que si el tráfico a un contenedor es alto, Kubernetes puede equilibrar la carga y distribuir el tráfico de la red para que la implementación sea estable.
* Orquestación de almacenamiento, permite montar automáticamente un sistema de almacenamiento local o en la nube.
* Implementaciones y reversiones automatizadas.
* Empaquetado automático en contenedores, gestionando un grupo de nodos que puede usar para ejecutar tareas en contenedores, indicando cuánta CPU y memoria (RAM) necesita cada contenedor.
* Reparación automática, ya que reinicia los contenedores que fallan, reemplaza los contenedores, elimina los contenedores que no responden a su verificación de estado definida por el usuario y no los anuncia a los clientes hasta que están listos para servir.
* Administración de configuraciones y secretos, administrando información como contraseñas, *tokens* de OAuth y claves SSH.

Cuando se implementa Kubernetes, se obtiene un clúster, que consta de un conjunto de máquinas de trabajo, llamadas nodos, que ejecutan aplicaciones en contenedores. Cada clúster tiene al menos un nodo trabajador. Los nodos trabajadores alojan los *pods* que son los componentes de la carga de trabajo de la aplicación. El plano de control administra los nodos trabajadores y los *pods* en el clúster. En entornos de producción, el plano de control generalmente se ejecuta en varias computadoras y un clúster generalmente ejecuta varios nodos, lo que proporciona tolerancia a fallas y alta disponibilidad (Kubernetes, s. f.).

A diagram of a cloud server

Description automatically generated

Figura 2. Arquitectura de componentes de un clúster Kubernetes. Fuente: Kubernetes, 2022.

Las dos plataformas más comunes para el despliegue en contenedores son *Docker* y *Kubernetes*. *Docker* se trata de empaquetar aplicaciones en contenedores en un solo nodo y *Kubernetes* está destinado a ejecutarlas en un clúster.

Apache Mesos

Es un núcleo (*kernel)* de sistemas distribuidos Está construido usando los mismos principios que el *kernel* de Linux, solo que en un nivel diferente de abstracción. El *kernel* se ejecuta en cada máquina y proporciona aplicaciones (por ejemplo, Hadoop, Spark, Kafka, Elasticsearch) con API para la gestión y programación de recursos en todo el centro de datos y entornos de nube. Permite abstraer la CPU, la memoria, el almacenamiento y otros recursos informáticos de las máquinas (físicas o virtuales), lo que permite que los sistemas distribuidos elásticos y tolerantes a fallas se construyan fácilmente y se ejecuten de manera efectiva. Los principales componentes principales de Mesos son un demonio maestro que administra los demonios de agente que se ejecutan en cada nodo del clúster y los marcos de trabajo de Mesos que ejecutan tareas en estos agentes (Hindman et al., 2011).

A diagram of a task management

Description automatically generated

Figura 3. Arquitectura de componentes de Apache Mesos. Fuente: Apache Mesos, s. f.

Red Hat OpenShift

Es una de las principales plataformas de Kubernetes a nivel empresarial, ya que cuenta con el soporte técnico de Red Hat. Permite diseñar, implementar y ejecutar las aplicaciones de manera uniforme en la nube o en nuestros propios equipos físicos. Gracias a que ofrece operaciones integrales automatizadas y la implementación de autoservicio para los desarrolladores, los equipos pueden trabajar en conjunto para llevar las ideas de la etapa de desarrollo a la de producción de manera más eficiente. Algunas de sus principales características y beneficios son:

* Escalabilidad, las aplicaciones pueden escalar a miles de instancias en cientos de nodos en segundos.
* Flexibilidad: simplifica la implementación y la administración de una infraestructura híbrida, lo que le brinda la flexibilidad de tener un servicio autoadministrado o completamente administrado.
* Basado en estándares de código abierto, ya que incorpora contenedores *open container initiative*(OCI) y Kubernetes certificados por Cloud Native Computing Foundation para la orquestación de contenedores; portabilidad de contenedores: las imágenes de contenedores creadas en el estándar de la industria OCI garantizan la portabilidad entre las estaciones de trabajo de los desarrolladores y los entornos de producción de Red Hat OpenShift.
* Experiencia de desarrollador mejorada: ofrece un conjunto integral de herramientas para desarrolladores, compatibilidad con varios idiomas e integraciones de línea de comandos y entorno de desarrollo integrado (IDE).
* Instalación y actualizaciones automatizadas: la instalación automatizada y las actualizaciones inalámbricas de la plataforma son compatibles en la nube con Amazon Web Services, Google Cloud Platform, IBM Cloud y Microsoft Azure.
* Como automatización: incluye compilaciones, implementaciones, escalado, administración de salud y más de contenedores y aplicaciones optimizados y automatizados.
* Gestión multiclúster: con Red Hat Advanced Cluster Management for Kubernetes puede implementar fácilmente aplicaciones, administrar varios clústeres y aplicar políticas en todos los clústeres a escala (Red Hat, 2022).

A diagram of a server

Description automatically generated

Figura 4. Arquitectura de capas de Red Hat Open Shift. Fuente: Red Hat Open Shift, s. f.

Plataformas que automatizan el despliegue de las aplicaciones

A nivel de herramientas que automatizan el despliegue de las aplicaciones en las infraestructuras físicas o en las lógicas anteriormente mencionadas, las más destacadas se detallan a continuación.

Argo Continuos Deployment

Es una solución de implementación continua de código abierto que proporciona compatibilidad con Kubernetes primero. Sigue flujos de trabajo basados en Git para automatizar la implementación de servicios dentro de Kubernetes. Como controlador de Kubernetes, garantiza que el estado actual de sus recursos coincida con el estado de recursos de destino deseado o especificado. Se puede encontrar en GitHub bajo argoproj. Argo CD aprovecha lo que se llama Argo GitOps Engine, una biblioteca de código abierto que implementa las funciones principales de GitOps, lo que le permite administrar, automatizar, auditar y comprender el ciclo de vida de su implementación de software. Argo CD es una de las herramientas de implementación de software más sólidas que existen (Argo CD, s. f.).

Atlassian Bamboo

Bamboo Data Center es una canalización de entrega continua que ofrece resistencia, fiabilidad y escalabilidad a equipos de todos los tamaños. Como solución empresarial de Atlassian, Bamboo, Bitbucket y Jira Software están completamente integrados y nos ofrecen total trazabilidad desde el momento en que se solicita una función hasta la implementación. Entre sus funcionalidades destacan (Atlassian, s. f.):

* Automatización del flujo de trabajo: aprovecha las ventajas de la implementación ágil con flujos de trabajo automatizados desde el código hasta la implementación.
* Recuperación ante desastres integrada: mantén a los equipos conectados y en la dirección correcta gracias a la resistencia de compilación y la alta disponibilidad.
* Escala con confianza: aumenta la capacidad y mantén el rendimiento a medida que crece su organización.

AWSCodeDeploy

Es un servicio de despliegue completamente administrado que automatiza las implementaciones de software en una variedad de servicios informáticos, como Amazon EC2, AWS Fargate, AWS Lambda y sus servidores locales. Es independiente de la plataforma y el lenguaje, por lo que funciona con cualquier aplicación (aunque existe un requisito para que sus instancias locales se conecten a puntos finales públicos de AWS). Tampoco parece haber ningún soporte para la implementación en otros proveedores de la nube, como GCP y Azure, por lo que en realidad es una solución muy acoplada a la infraestructura AWS (AWS, s. f.).

Azure Automation

Ofrece un servicio de configuración, actualizaciones del sistema operativo y automatización basado en la nube que admite una administración coherente en sus entornos Azure y no Azure. Incluye automatización de procesos, gestión de configuración, gestión de actualizaciones, capacidades compartidas y características heterogéneas. Process Automation en Azure Automation le permite automatizar tareas de administración frecuentes, lentas y propensas a errores. Este servicio lo ayuda a concentrarse en el trabajo que agrega valor comercial. Al reducir los errores y aumentar la eficiencia, también ayuda a reducir los costos operativos. El entorno operativo de automatización de procesos se detalla en la ejecución de *Runbook* en Azure Automation (Microsoft, 2022).

Cloud Foundry BOSH

Es un proyecto que unifica la ingeniería de lanzamiento, la implementación y la gestión del ciclo de vida del software en la nube a pequeña y gran escala. BOSH puede aprovisionar e implementar software en cientos de máquinas virtuales. También realiza monitoreo, recuperación de fallas y actualizaciones de software con un tiempo de inactividad de cero a mínimo. Si bien BOSH se desarrolló para implementar Cloud Foundry PaaS, también se puede usar para implementar casi cualquier otro software (Hadoop, por ejemplo). BOSH es particularmente adecuado para grandes sistemas distribuidos. Además, BOSH es compatible con múltiples proveedores de infraestructura como servicio (IaaS), como VMware vSphere, Google Cloud Platform, Amazon Web Services EC2, Microsoft Azure, OpenStack y Alibaba Cloud. Hay una interfaz de proveedor de nube (CPI) que permite a los usuarios ampliar BOSH para admitir proveedores de IaaS adicionales, como Apache CloudStack y VirtualBox (Cloud Foundry BOSH, s. f.).

GitLab

Originalmente una herramienta de administración de código fuente basada en Git, introdujo una solución de CI/CD en su conjunto de productos. Admiten herramientas de orquestación de contenedores, como Kubernetes y ECS, así como los tres principales proveedores de nube (GCP, Azure, AWS). En términos de infraestructura, GitLab lanzó recientemente una integración de Terraform. La plataforma también ofrece funciones de buen gobierno y cumplimiento disponibles en sus planes de nivel empresarial. En lo que respecta a los idiomas, GitLab tiene como objetivo admitir todos los idiomas, antiguos y nuevos, incluidos PHP, Go, Ruby, .NET, Java, JS y más. Se califica a sí mismo como una plataforma de CI/CD, pero el CD se logra al extender CI con *scripts*, por lo que no es del todo una plataforma en el verdadero sentido de la palabra. Adicionalmente, hay una curva de aprendizaje relevante para utilizar la herramienta, no tiene ninguna capacidad de verificación de implementación, pero puede integrarse con herramientas como Prometheus para fines de monitoreo (GitLab, s. f.).

GoCD

Es un servidor de integración y despliegue continuo gratuito y de código abierto, que permite modelar y visualizar fácilmente flujos de trabajo complejos. Proporciona visualización de canalización de extremo a extremo, implementaciones nativas de la nube, modelado de flujo de trabajo complejo y trazabilidad avanzada. Algunas de sus funcionalidades son (GoCD, s. f.):

* Visualización de un extremo a otro: muestra todo el camino desde la creación hasta el compromiso y la producción en una sola vista para permitir que los usuarios naveguen entre trabajos, detecten ineficiencias y optimicen procesos.
* Implementaciones nativas de la nube: permite optimizar el flujo de trabajo de CD en entornos de nube comunes como Kubernetes, Docker y AWS.
* Modelado de flujo de trabajo complejo: ejecución paralela, gestión de dependencias, construcciones de modelado de flujo de trabajo de CD.
* Trazabilidad avanzada: resuelva problemas de tuberías rotas con pistas de auditoría y diferencias.

Google Cloud Deployment Manager

Es un servicio de implementación de infraestructura que automatiza la creación y la administración de recursos de Google Cloud. Facilita la escritura de plantillas flexibles y archivos de configuración que sirven de base para crear despliegues que tengan una variedad de servicios de Google Cloud, como Cloud Storage, Compute Engine y Cloud SQL, configurados para funcionar en conjunto (Google Cloud Deployment Manager, s. f.).

Harness

Es una plataforma de entrega de software moderna que ofrece integración e implementación continua, indicadores de funciones y administración de costos en la nube. Esta herramienta se creó para reducir el tiempo, el esfuerzo y la administración necesarios para crear nuevos artefactos. Se basa en el uso de aprendizaje automático para optimizar la cantidad de pruebas necesarias durante el proceso de construcción y despliegue. Harness Feature Flags ayuda a administrar los lanzamientos al permitir que los desarrolladores, especialistas en *marketing*, gerentes y otros líderes de la empresa activen y desactiven instantáneamente las funciones en producción. Esto permite pruebas A/B fáciles y puede aumentar drásticamente la velocidad de implementación. Los indicadores de características se están convirtiendo rápidamente en una parte clave del proceso de implementación de software. Harness Cloud Cost Management brinda a los usuarios visibilidad instantánea del coste proveniente del servicio en la nube elegido. También permite a los usuarios controlar o detener automáticamente los recursos que se aprovisionaron en exceso o se dejaron en ejecución accidentalmente (Harness, s. f.).

Heroku Flow

Es un flujo de trabajo de automatización del despliegue estructurado que combina integraciones estrechas con GitHub, canalizaciones presentadas visualmente, aplicaciones de revisión desechables y Heroku CI. Está diseñado para optimizar la experiencia de lanzamiento de la aplicación al hacer que la entrega continua sea fácil, visual y eficiente sobre Heroku, la plataforma en la nube como servicio (PaaS) de Salesforce que admite varios lenguajes de programación como Ruby, Java, Node.js, Scala, Clojure, Python, PHP y Go (Heroku, s. f.).

Jenkins

Es la herramienta de código abierto más conocida, que se utiliza tanto para la integración continua como para la implementación continua. Dado que es un servidor de automatización de código abierto, es de uso gratuito para todos y hay una gran comunidad de seguidores, lo que genera soporte, documentación y funciones adicionales. Este servidor/herramienta de integración continua es un programa Java independiente que es independiente de la plataforma. Está disponible para la mayoría de los principales sistemas operativos, como Linux, Windows y macOS. Aunque con base en una serie de extensiones y *plugins* los desarrolladores han hecho de Jenkins una herramienta de automatización de despliegue, en realidad no es su objetivo, no es nativo para la nube y como tal requiere de demasiados complementos para cumplir esta función, pero en todo caso lo nombramos al ser habitual en las plataformas de CI/CD (Jenkins, s. f.).

Octopus Deploy

Es una solución de software para administrar lanzamientos y automatizar implementaciones. Desde la perspectiva de la gestión de lanzamientos, ayuda a los equipos a comprender qué versiones se lanzan y en qué entornos se implementaron. A nivel de automatización de la compilación y despliegue, incorpora 450 plantillas, contiene variables para que se pueda usar el mismo proceso en entornos de desarrollo, prueba y producción.

La última gran capacidad de Octopus Deploy es la automatización de *runbooks*, de forma que es posible automatizar tareas operativas como el mantenimiento de rutina y la recuperación de incidentes de emergencia. Estos *runbooks* incluyen todos los permisos necesarios para que cualquier miembro del equipo pueda obtener permiso para ejecutar el *runbook*. En general, Octopus Deploy es una de las herramientas sólidas de implementación de software que existen, con un sólido conjunto de características y soporte para entornos modernos y heredados. Al igual que con todas las herramientas, hay algunas características importantes que Octopus Deploy no cumple. En particular, faltan por completo la verificación de implementación automatizada y la reversión de implementación automatizada (Octopus Deploy, s. f.).

Progress Chef Automate

Es una herramienta de gestión de configuración escrita en Ruby y Erlang. Utiliza un lenguaje específico de dominio de Ruby puro para escribir recetas de configuración del sistema. Chef se utiliza para agilizar la tarea de configurar y mantener los servidores de una empresa y puede integrarse con plataformas basadas en la nube como Amazon EC2, Google Cloud Platform, Oracle Cloud, OpenStack, IBM Cloud, Microsoft Azure y Rackspace para aprovisionar y configurar automáticamente maquinas nuevas. Permite automatizar (Chef, 2022):

* Aplicaciones: mediante prácticas de entrega continua e implementándolas en entornos nativos de la nube.
* Seguridad: se detectan problemas de seguridad antes de que entren en producción para reducir el riesgo, aumentar la velocidad y mejorar la eficiencia.
* Infraestructura: la gestión de configuración automatizada permite configuraciones coherentes a escala, lo que garantiza que la política de configuración sea flexible, versionable, comprobable y legible por humanos.

Puppet

Es una herramienta de gestión de sistemas DevOps de código abierto que facilita el despliegue, configuración y administración de servidores. Se utiliza para centralizar y automatizar el procedimiento de gestión de la configuración. Esta herramienta está desarrollada utilizando Ruby DSL. Está desarrollado por Puppet Labs y está disponible tanto para versiones de código abierto como para empresas. Se utiliza particularmente para la automatización de la entrega y gestión de infraestructura híbrida. Es compatible con muchas plataformas como Microsoft Windows, Debian/Ubuntu, Red Hat/CentOS/Fedora, macOS X, etc. Utiliza el paradigmacliente-servidor, en el que un sistema de cualquier clúster funciona como servidor, denominado Puppet Master, y el otro funciona como cliente en nodos denominados esclavos (Puppet, s. f.).

Red Hat Ansible Automation Platform

Es una herramienta de despliegue automatizado de aplicaciones, administración de configuración y aprovisionamiento de software de código abierto que habilita la infraestructura como código. Incluye su propio lenguaje declarativo para describir la configuración del sistema. Tiene una solución adquirida porRed Hat en 2015, no tiene agente, se conecta temporalmente de forma remota a través de SSH o Windows Remote Management (lo que permite la ejecución remota de PowerShell) para realizar sus tareas. Una de sus ventajas es que permite aprovisionar la infraestructura en la nube, *hosts* virtualizados, dispositivos de red y servidores completos, como primer paso para automatizar el despliegue de las aplicaciones (Red HatAnsible, s. f.).

10.5. Plataformas para arquitecturas basadas en eventos

Las principales plataformas para dar soporte a arquitecturas basadas en eventos facilitan la comunicaciónentre productores y consumidores de eventos y son conocidas como mediadores de mensajes o eventos. Bajo este término generalista, existen diversas interpretaciones y denominaciones como:

* *Middleware* orientado a mensajes (*message oriented middleware*, MOM): bibliotecas de software para varios lenguajes de programación con un intermediario (o no) para comunicar mensajes entre aplicaciones. Los mensajes son objetos estructurados o cadenas de texto o datos binarios. Por lo general, tiene confiabilidad adicional sobre TCP/IP o UDP.
* *Java messaging service* (JMS): es la definición de una API común para MOM, motivado por la ausencia de un estándar para interaccionar con diversas implementaciones de sistemas MOM. Si nuestro sistema está implementado contra JMS, podemos simplemente cambiar las bibliotecas y la misma aplicación que solía funcionar con un MOM (por ejemplo, TIBCO EMS) funcionará con otro (por ejemplo, ActiveMQ)
* *Enterprise dervice bus* (ESB): un ESB utiliza un MOM para integrar aplicaciones, bases de datos, intermediarios, etc. Es un MOM con una estructura de datos agregada y administración de definición de estructura. Al conectar un nuevo componente a un ESB, podemos esperar más compatibilidad lista para usar que cuando lo conectamos a un MOM. En un ESB hay estándares más altos sobre lo que debe hacer un componente para conectarse y también incorpora reglas o patrones de transformación entre los mensajes para dar soporte a los patrones de integración de aplicaciones empresariales (*enterprise integration patterns*, EIP) (Hohpe y Woolf, 2003).

Algunas de las plataformas y herramientas que dan soporte a la mensajería de eventos en EDA son:

A screenshot of a computer

Description automatically generated

Tabla 2. Plataformas para arquitecturas basadas en eventos. Fuente: elaboración propia.

Apache ActiveMQ

Es el intermediario de mensajes basado en Java, de código abierto y multiprotocolo más popular. Es compatible con los protocolos estándar de la industria para que los usuarios obtengan los beneficios de las opciones del cliente en una amplia gama de idiomas y plataformas. Es posible conectarse desde clientes escritos en JavaScript, C, C++, Python, .NET y más. Permite integrar sus aplicaciones multiplataforma utilizando el omnipresente protocolo AMQP*(advanced message queuing*), intercambiar mensajes entre aplicaciones web usando STOMP sobre *websockets*, y administrar dispositivos IoT usando MQTT (Christudas, 2019).

Apache Kafka

Es una plataforma distribuida de gestión de colas de eventos, que consta de servidores y clientes que se comunican a través de un protocolo de red TCP de alto rendimiento y que se puede implementar en hardware básico, máquinas virtuales y contenedores en entornos locales y en la nube. A nivel de servidores, se ejecuta como un clúster de uno o más servidores que pueden abarcar varios centros de datos o regiones de la nube. Algunos de estos servidores forman la capa de almacenamiento, llamados intermediarios. Otros servidores ejecutan Kafka Connect para importar y exportar datos continuamente como flujos de eventos para integrar Kafka con sus sistemas existentes, como bases de datos relacionales y otros clústeres de Kafka.

Para permitirle implementar casos de uso de misión crítica, un clúster de Kafka es altamente escalable y tolerante a fallos. A nivel de clientes, permiten desarrollar aplicaciones distribuidas y microservicios que leen, escriben y procesan flujos de eventos en paralelo, a escala y con tolerancia a fallos, incluso en el caso de problemas de red o fallas de máquinas. Los clientes están disponibles para *Java* y *Scala*, incluida la biblioteca Kafka Streams de nivel superior, para Go, Python, C/C++ y muchos otros lenguajes de programación, así como API REST (Scott, 2022).

Apache ServiceMix

Es un contenedor de integración flexible y de código abierto que unifica las características y la funcionalidad de Apache ActiveMQ, Camel, CXF y Karaf en una poderosa plataforma de tiempo de ejecución que podemos usar para crear nuestras propias soluciones de integración. Proporciona un ESB completo y listo para la empresa impulsado exclusivamente por OSGi. Bajo licencia Apache License v2, sus características principales son (ServiceMix, 2022):

* Mensajería confiable con Apache ActiveMQ.
* Mensajería, enrutamiento y patrones de integración empresarial con Apache Camel.
* Servicios web WS‐\* y RESTful con Apache CXF.
* Tiempo de ejecución del servidor basado en OSGi con la tecnología de Apache Karaf.

A través de características instalables adicionales, ServiceMix también admite:

* Motor BPM a través de Activiti.
* Soporte completo de JPA a través de Apache OpenJPA.
* Gestión de transacciones *XA* a través de JTA a través de Apache Aries.

Microsoft Biztalk

Es el *middleware* ESB de Microsoft. Aprovecha .NET y permite a los desarrolladores escribir sus piezas de integración en Visual Studio. Se compone de:

* MSMQ*(Microsoft message queuing*): implementación de cola de mensajes.
* Enrutamiento: las especificaciones de enrutamiento/mensaje se implementan a través de XML, pero generalmente este XML se genera utilizando herramientas gráficas.
* Adaptadores: BizTalk tiene una variedad de adaptadores integrados. Como era de esperar, tiene un excelente soporte de adaptador para las tecnologías de *Microsoft*, como los diversos protocolos WCF.
* BizTalk Server: BizTalk requiere IIS (servicios de información de Internet) para varias funciones, como HTTP, SOAP, SSL y más. Por lo general, esto se implementa en *Windows* *Server*.
* Consola de administración de BizTalk Server: esta es una MMC (*Microsoft* *management console*) que permite una amplia configuración y administración del servidor.

MuleESB

Es un bus de servicio empresarial basado en Java que ocupa poco espacio. Es de código abierto y, como la mayoría de los ESB, permite la integración de sistemas a través de JMS, servicios web, HTTP, JDBC y más. Se compone de (MuleSoft, s. f.):

* RabbitMQJavaClient para soporta AMQP.
* Enrutadores: MuleSoft utiliza enrutadores para dividir, combinar, reordenar, evaluar y transmitir mensajes.
* Anypoint Connectors: conectores de protocolo, base de datos, transporte y base de datos prediseñados. También puedes construir el tuyo propio si es necesario.
* MuleRuntime Engine: el corazón de la plataforma MuleSoft Anypoint. Implementable en la nube o en las instalaciones.
* Mule Runtime Manager: permite la implementación, el monitoreo y la resolución de problemas de las instancias de Mule.

Oracle Service Bus

Transforma arquitecturas complejas y frágiles en redes de integración ágiles al conectar, virtualizar y administrar interacciones entre servicios y aplicaciones. Ofrece integración basada en estándares y de bajo costo para entornos SOA cruciales donde el máximo rendimiento, la escalabilidad y la fiabilidad son requisitos esenciales. Ya sea en entornos locales, en la nube, en dispositivos móviles y de cara al futuro, permite a las empresas aprovechar sus inversiones existentes de formas totalmente nuevas (Blokdyk, 2018).

RabbitMQ

Es el intermediario de mensajes de código abierto más utilizado. Admite múltiples protocolos de mensajería, colas de mensajes, acuse de recibo de entrega, enrutamiento flexible a colas o tipo de intercambio múltiple. Soporta el desarrollo de intercambio de mensajes con una gran variedad de lenguajes de programación como Java, .NET, PHP, Python, JavaScript, Ruby, Go y muchos otros. Soporta el despliegue con herramientas como BOSH, Chef, Docker y Puppet, así como la configuración en clústeres para alta disponibilidad y rendimiento y federarse en varias zonas y regiones de disponibilidad. Dispone de un API HTTP, una utilidad de línea de comandos y un entorno gráfico para gestionar la plataforma (RabbitMQ, 2022).

A screenshot of a computer

Description automatically generated

Figura 5. Vista de la consola de administración de RabbitMQ. Fuente: CloudAMQP, 2019.

En este video, *RabbitMQ: arquitecturas distribuidas dirigidas por eventos*, aprenderemos a configurar un proyecto Spring para publicar y recibir mensajes de colas sobre RabbitMQ, configurado sobre Docker.

Red Hat Fuse

* Es más que un ESB, ya que se configura como una plataforma de integración ligera de código abierto, basada en soluciones como Apache ServiceMix, que está disponible en las instalaciones o en la nube. Integra (Red Hat, s. f.).
* Apache ActiveMQ: un *broker* de mensajes rápido y de código abierto que admite JMS y clientes escritos en otros lenguajes como C y Python.
* Apache Camel: un marco de código abierto que proporciona implementaciones de EIPS (*enterprise integration patterns*).
* Apache CFX: un marco de servicios web de código abierto que permite la comunicación utilizando varios estándares *como*JAX-WS y JAX-RS, HTTP y FTP*,*así como diferentes formatos como JSON, XML, CSV, etc.
* Apache Karaf: un contenedor de tiempo de ejecución OSGi para implementar aplicaciones.
* Fabric8: una herramienta de orquestación para grandes implementaciones de middleware

10.6. Plataformas de monitorización de sistemas distribuidos

Una vez desplegado un sistema distribuido, el equipo de operaciones de una organización será el responsable de hacer la supervisión y monitorización de la infraestructura y los diversos bloques funcionales.

Algunas de las plataformas y herramientas que dan soporte a la supervisión de sistemas distribuidos son:

A blue and white rectangular box with text

Description automatically generated

Tabla 3. Plataformas de monitorización de sistemas distribuidos. Fuente: elaboración propia.

Cacti

Es una solución de monitorización de red de código abierto basada en el sistema de clasificación y trazado de datos con la herramienta RRDTool. La herramienta utiliza la funcionalidad de recopilación de datos y el sondeo de red para juntar información sobre varios dispositivos en redes de cualquier alcance. Esto comprende la capacidad de crear *scripts* personalizados para la recopilación de datos junto con la función de sondeo SNMP. Luego, muestra esta información en gráficos fáciles de comprender que se pueden organizar en cualquier jerarquía según la conveniencia de su negocio. Incluye (Cacti, 2022):

* Un marco de recopilación de datos totalmente distribuido y tolerante a fallos.
* Funciones avanzadas de automatización basadas en plantillas para dispositivos, gráficos y árboles.
* Múltiples métodos de adquisición de datos
* La capacidad de extenderse a través de complementos, funciones de administración de usuarios, grupos y dominios basadas en roles.
* Un motor de temas y compatibilidad con varios idiomas desde el primer momento.

A screenshot of a computer

Description automatically generated

Figura 6. Vista del supervisor de Cacti. Fuente: Cacti, 2022.

Inciga

Es una herramienta de supervisión de red de código abierto que calcula la disponibilidad y el rendimiento de la red. A través de una interfaz web, podemos observar aplicaciones y *hosts* alrededor de la infraestructura de red completa. La herramienta es escalable y fácilmente configurable para funcionar con cada tipo de dispositivo. Existen algunos módulos adicionales para capacidades de monitorización muy específicas, como el del entorno de nube vSphere de VMWare y el modelado de procesos comerciales (Inciga, s. f.).

LibreNMS

Es un sistema de monitorización de red de código abierto que utiliza múltiples protocolos de red para observar cada dispositivo en su red. Su API permite recuperar, administrar y trazar los datos que recopila y facilita el escalado horizontal para aumentar sus capacidades de monitoreo junto con su red. La herramienta presenta un sistema de alerta flexible que está hecho a medida para comunicarse con usted por el método que mejor se adapte a su empresa (LibreNMS, s. f.).

Nagios

Es una herramienta de monitorización de código abierto que lleva más de veinte años en disponible para la industria, por lo que existen muchos complementos disponibles. Permite monitorizar sistemas Windows, Linux, servidores, aplicaciones, *middlewares*, así como analizar *logs* e integrarse con aplicaciones de terceros para automatizar las gestiones a realizar en el caso de alertas o incidencias (Nagios, s. f. a).

A screenshot of a computer

Description automatically generated

Figura 7. Vista del analizador de Logs de Nagios. Fuente: Nagios, s. f. b.

Pandora FMS

Es una herramienta de supervisión de código abierto que ayuda a las empresas a observar toda su infraestructura de TI. No solo presenta capacidades de monitoreo de red, sino también servidores Unix y Windows e interfaces virtuales. Para redes, incluye características de primer nivel como compatibilidad con SNMP, sondeo ICMP, monitorización de la latencia de red y sobrecarga del sistema. Los agentes también se pueden instalar en los dispositivos para observar aspectos como la temperatura y el sobrecalentamiento del dispositivo, así como los sucesos de los archivos de registro (Pandora FMS, s. f.).

Prometheus

Es una solución de monitoreo de código abierto que se centra en la recopilación y el análisis de datos basados en datos de series temporales. Permite a los usuarios configurar capacidades de monitorizaciónal utilizar el conjunto de herramientas incorporado. Es ideal para entornos en contenedores como Kubernetes y es considerada como la mejor herramienta de monitorización de servidores de código abierto. Permite unificar métricas de servidores, contenedores y varios dispositivos mediante *pings* SNMP e inspeccionar el uso del ancho de banda de la red desde el punto de vista del dispositivo. Dispone de un administrador de alertas configurable en base a reglas. Se integra con Grafana para disponer de paneles y cuadros de mando de visualización de métricas (Prometheus, s. f. a).

A screenshot of a computer

Description automatically generated

Figura 8. Vista de métricas obtenidas con Prometheus sobre Grafana. Fuente: Prometheus, s. f. b.

Riemann

Es una herramienta de monitorización de código abierto ideal para sistemas distribuidos. Es un sistema de procesamiento uniforme de baja latencia capaz de recopilar métricas de una variedad de sistemas distribuidos. Está diseñado para manejar millones de pares por segundo con baja latencia, por lo que es una plataforma ideal para supervisar sistemas escalables altamente distribuidos (Riemann, s. f. a).

A screenshot of a computer

Description automatically generated

 Figura 9. Vista de métricas obtenidas con Riemann. Fuente: Riemann, s. f. b.

Sensu

Es una herramienta de monitorización de pila completa. A través de una única plataforma, se puede realizar la supervisión de servicios, aplicaciones, servidores y reportes de KPI (indicador clave de rendimiento) de negocio. Es una solución basada en agentes que instala en la infraestructura de su organización, con un *back-end* que brinda una canalización flexible y automatizada para filtrar, transformar y procesar alertas y métricas (Sensu, s. f.).

Zabbix

Es un software de supervisión de código abierto con una interfaz fácil de usar que brinda soluciones de clase empresarial a grandes organizaciones. Es un sistema centralizado que almacena los datos en una base de datos relacional para un procesamiento eficiente. Entre sus funcionalidades están (Zabbix, 2022).:

* Supervisión de red.
* Monitorización del servidor.
* Supervisión en la nube.
* Monitorización de aplicaciones.
* Supervisión del servicio.

Monitorizar diversos entornos

Debido a que es posible que sea necesario monitorizar infraestructuras o aplicaciones heterogéneas, existe una solución global que puede ser aplicada para monitorizar diversos entornos y supervisar múltiples tipos de datos de aplicaciones a través de sus *logs* o registro. En los últimos años ha adquirido una gran popularidad ELK, ahora conocido como Elastic Stack, que representa la unión de tres proyectos *open source*:

Elasticsearch

Es un motor de analítica y análisis distribuido, gratuito y abierto para todos los tipos de datos, incluidos textuales, numéricos, geoespaciales, estructurados y no estructurados. Desarrollado a partir de Apache Lucene en 2010, es conocido por sus API REST simples, naturaleza distribuida, velocidad y escalabilidad. Es el componente principal del Elastic Stack, un conjunto de herramientas gratuitas y abiertas para la ingesta, el enriquecimiento, el almacenamiento, el análisis y la visualización de datos (Elastic, s. f. a).

Logstash

Es un *pipeline* de procesamiento de datos del lado del servidor que ingesta datos de una multitud de fuentes simultáneamente, los transforma y luego los envía a un repositorio como Elasticsearch. Básicamente, ingesta, transforma y envía de forma dinámica los datos independientemente de su formato o complejidad. Deriva estructura a partir de datos no estructurados con Grok, descifra las coordenadas geográficas de las direcciones IP, anonimiza o excluye los campos sensibles y facilita el procesamiento general (Elastic, s. f. b).

En este video, *Monitorizando sistemas distribuidos con ELK*, aprenderemos a crear un entorno con Beats, Logstash, Elasticsearch y Kibana sobre Docker, para registrar y visualizar los Logs de una aplicación Spring.

Kibana

Es una interfaz de usuario gratuita y abierta que te permite visualizar los datos de Elasticsearch y navegar en el Elastic Stack. Facilita la interacción con los datos recogidos, desde rastrear la carga de búsqueda hasta comprender la forma en que las solicitudes fluyen por las aplicaciones (Elastic, s. f. c). Dispone de elementos visuales como gráficos, series temporales, mapas, gráficos geoespaciales, tablas de datos, *canvas* y un conjunto de *plugins* para gráficos 3D y más.

A screenshot of a computer

Description automatically generated

Figura 10. Vista Kibana de una monitorización con Elastic Stack. Fuente: Elastic, s. f. d.