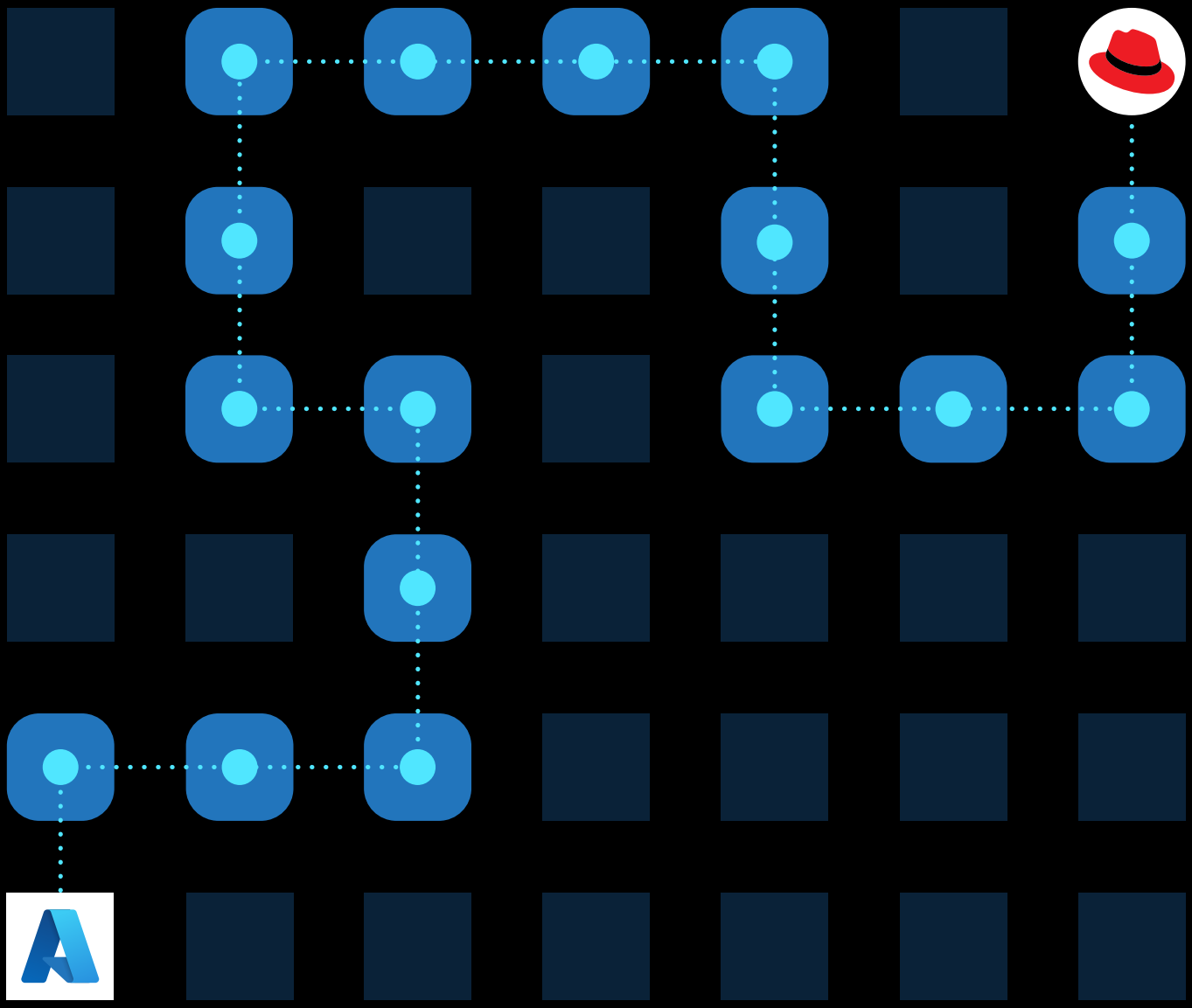
Serie de ebooks

Microsoft Azure

Primeros pasos con Azure Red Hat OpenShift

Desde la prueba de conceptos hasta la producción



Primeros pasos con Azure Red Hat OpenShift

Capítulo 1

Comuníquese con Microsoft y Red Hat

Si está interesado en Azure Red Hat OpenShift, queremos hablar con usted. En esta guía encontrará instrucciones útiles para usar la plataforma, pero si desea obtener aún más información, puede consultar la gran variedad de recursos adicionales que ofrecen Microsoft y Red Hat. Queremos asegurarnos de que Azure Red Hat OpenShift sea la plataforma que estaba buscando para satisfacer sus necesidades de innovación de las aplicaciones.

Azure Red Hat OpenShift se creó por una razón sencilla: la pidieron los clientes como usted. Hoy más que nunca, nuestros clientes en común, sean empresas grandes o pequeñas, están implementando los productos de Red Hat en Microsoft Azure. Si bien OpenShift en Azure tuvo compatibilidad total durante muchos años como oferta autogestionada, la configuración, la implementación y las operaciones del día 2 necesarias a la hora de gestionar los clústeres consumen tiempo y requieren experiencia en Kubernetes, lo cual deja los objetivos empresariales en segundo plano.

Cada vez son más los clientes que lanzan implementaciones exitosas con esta oferta, Azure Red Hat OpenShift, y notamos que la configuración y las operaciones del día 2 les llevan mucho menos tiempo y se pueden enfocar más en las aplicaciones.

Creamos esta guía en función de nuestra experiencia, de modo que nuestros clientes contaran con las prácticas recomendadas para diseñar aplicaciones en Azure Red Hat OpenShift por sí mismos. Puede leer los capítulos de manera secuencial, desde la introducción hasta la conclusión, o simplemente buscar la información específica que necesite.

A quién está dirigida esta guía

Esta guía está dirigida a audiencias técnicas (desarrolladores, operadores y arquitectos de plataforma) que buscan potenciar su capacidad de diseño e implementación de aplicaciones con Azure y Red Hat OpenShift para disponer de clústeres totalmente gestionados de RHOS.

Los temas que abarca esta guía

En esta guía, revisaremos los aspectos clave para entender y adoptar Azure Red Hat OpenShift, desde la realización de la prueba de concepto dentro de la empresa hasta la implementación de la producción.

Capítulo 2 *Introducción a Red Hat OpenShift:* se presenta Red Hat OpenShift y se mencionan los motivos por los cuales tantos desarrolladores, operadores y arquitectos de plataforma la eligen, la prefieren y les resulta tan útil.

Capítulo 3 *Azure Red Hat OpenShift:* se explica el servicio gestionado y la manera en que se ofrece a los clientes.

Capítulo 4 *Etapa previa a la implementación: preguntas sobre la arquitectura empresarial*: se abordan preguntas importantes que debe considerar antes de implementar Azure Red Hat OpenShift. Aquí se incluye mucha orientación en cuanto a las prácticas recomendadas, las cuales aprendimos manteniendo conversaciones con clientes que ya lo han hecho.

Capítulo 5 *Implementación de un clúster de Azure Red Hat OpenShift:* se señalan los recursos clave necesarios para implementar un clúster de Azure Red Hat OpenShift en la documentación oficial.

Capítulo 6 *Etapa posterior a la implementación: el día 2* se abordan las tareas posteriores a la implementación, las cuales se suelen llamar "Día 2". Esta guía intenta remarcar la comodidad de este servicio gestionado, Azure Red Hat OpenShift, el cual permite que el proceso sea más sencillo que si usted tuviera que hacer todo de manera manual.

Capítulo 7 *Implementación de aplicaciones en Red Hat OpenShift:* es una guía breve sobre la implementación de aplicaciones en RHOS, un proceso que, de todas maneras, no difiere en otros entornos.

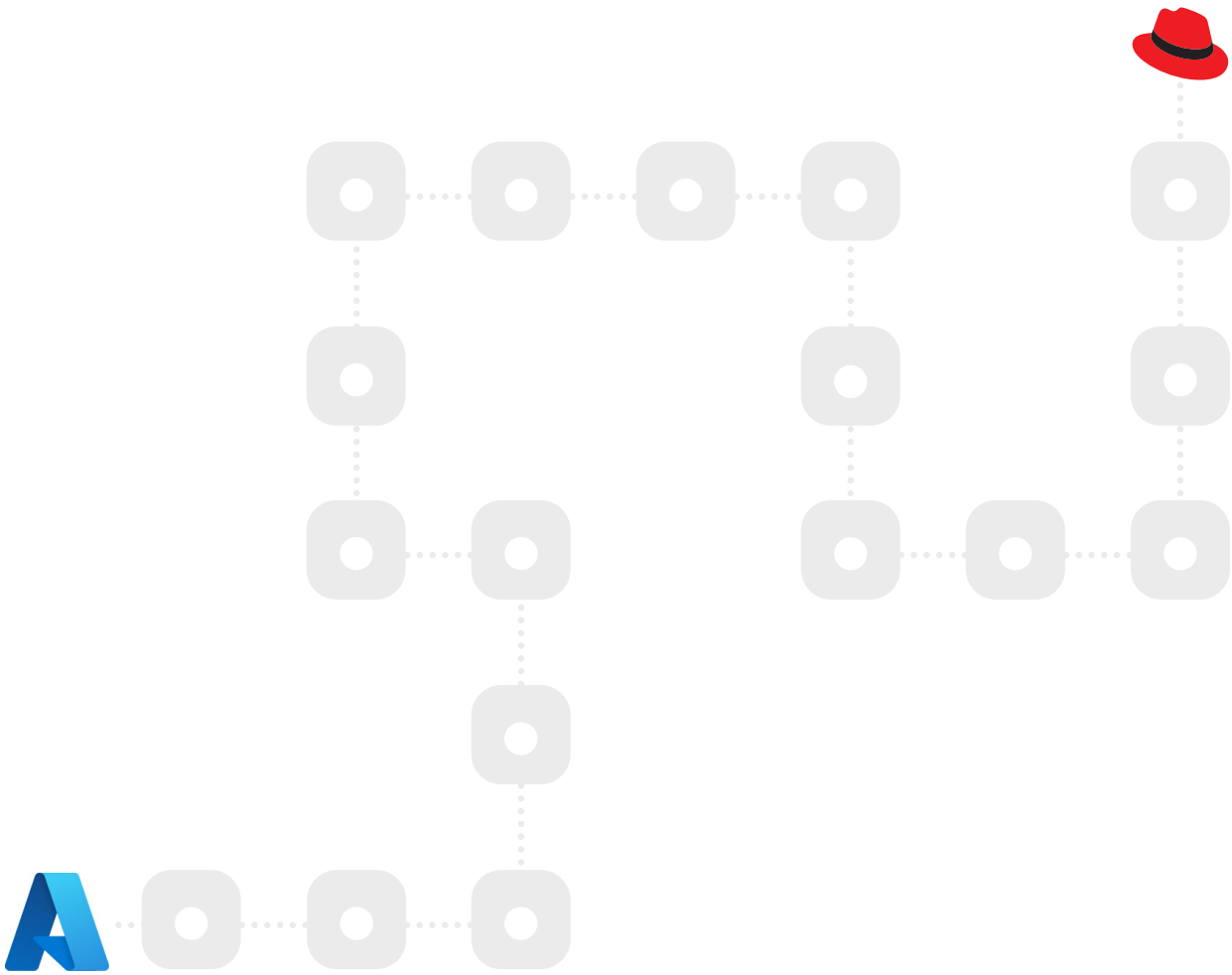
Capítulo 8 *Análisis de la plataforma de la aplicación:* brinda una visión general guiada de algunas funciones clave de la plataforma, como Container Registry (el registro de contenedores), Pipelines (los canales), Serverless (la informática sin servidor) y otras similares.

Capítulo 9 *Integración con otros servicios:* cubre la integración de la plataforma con servicios tales como Azure Service Operator y Azure DevOps.

Capítulo 10 *Incorporación de las cargas de trabajo y los equipos:* proporciona algunas prácticas recomendadas y sugerencias sobre el modo de incorporar los equipos de aplicaciones y ganar terreno en la adopción del servicio dentro de su empresa.

Capítulo 11 *Conclusión:* contiene la conclusión.

Capítulo 12 *Glosario:* cumple la función de un glosario.



Capítulo 2

Introducción a Red Hat OpenShift

Este capítulo proporciona una breve introducción sobre Red Hat OpenShift, la forma en que se usa y las ventajas que destacan los clientes de Red Hat. Es un manual útil si está dando sus primeros pasos con OpenShift y cumple la función de un repaso rápido si lo que busca es aprender un poco más sobre la plataforma.

Descripción general de Red Hat OpenShift

Red Hat OpenShift es una plataforma empresarial de aplicaciones que se encuentra optimizada para mejorar la productividad de los desarrolladores y promover la innovación. Además, cuenta con gestión simplificada del ciclo de vida y con operaciones automatizadas integrales que permiten gestionar las implementaciones en los entornos de nube híbrida y multicloud. Gracias a esto, los equipos de desarrollo pueden diseñar e implementar las aplicaciones nuevas, mientras que los de operaciones pueden preparar, gestionar y ajustar las plataformas de Kubernetes.

Por un lado, los equipos de desarrollo tienen acceso a las imágenes y las soluciones validadas de cientos de partners con análisis de seguridad y firmas criptográficas en todo el proceso de distribución. Asimismo, pueden acceder a imágenes por solicitud, así como a una amplia variedad de servicios de nube de terceros, desde una sola plataforma.

Por el otro, los equipos de operaciones pueden observar las implementaciones donde quiera que estén, incluso entre equipos, y cuentan con funciones de registro y supervisión integrados. Los operadores de Kubernetes de Red Hat incorporan una lógica de aplicaciones única, la cual permite que el servicio sea funcional. Esto quiere decir que no solo está configurado, sino que también está optimizado para el rendimiento y se puede actualizar y aplicar parches desde el SO con tan solo un clic. En pocas palabras, Red Hat OpenShift es una solución única con la que puede liberar el potencial de los equipos de TI y desarrollo.

Servicios de la nube de OpenShift: ahorro de costos y ventajas empresariales

Hay miles de usuarios que confían en Red Hat OpenShift para mejorar la relación con sus clientes, cambiar el modo en que distribuyen las aplicaciones y obtener ventajas competitivas para ser líderes de sus sectores. La investigación de IDC, descrita en El valor empresarial de Red Hat OpenShift, marzo de 2021, demuestra el gran valor que alcanzaron las empresas entrevistadas gracias a Red Hat OpenShift. La plataforma les permitió distribuir aplicaciones y características más oportunas y de mayor calidad y, a la vez, optimizar los costos relacionados con la TI y el desarrollo y ahorrar el tiempo del personal.

Algunos indicadores de la investigación son los siguientes:

- Hay un retorno sobre la inversión del 636 % en cinco años.
- Hay un tiempo de amortización de 10 meses.
- Los costos operativos se reducen en un 54 % en cinco años.
- Las características nuevas se triplican cada año.
- La productividad de los desarrolladores de aplicaciones aumenta en un 20 %.
- El tiempo de inactividad imprevisto se reduce un 71 %.
- Los equipos de infraestructura de TI tienen un 21 % más de eficiencia.

Fuente: Whitepaper de IDC, patrocinado por Red Hat: *El valor empresarial de Red Hat OpenShift*, documento n.° US47539121 de marzo de 2021.

Además del valor que demostramos que ofrece RHOS, sus servicios de nube, como Azure Red Hat OpenShift, brindan ventajas comerciales adicionales. Se destacaron algunas en el estudio de Forrester llamado El Total Economic Impact™ de los servicios de nube de Red Hat OpenShift: ahorro de costos y beneficios de negocios utilizando Red Hat.

Las ventajas comerciales clave de los servicios de nube de OpenShift son las siguientes:

- Un retorno sobre la inversión (ROI) del 468 %
- Un valor actual neto (VAN) de USD 4,08 millones
- Un tiempo de amortización de 6 meses

Asimismo, el informe presenta beneficios importantes que se pueden cuantificar:

- **Velocidad de desarrollo mejorada**: usar Red Hat OpenShift permite que las empresas reduzcan su ciclo de desarrollo hasta un 70 %.
- **Recuperación del 20 % del tiempo que los desarrolladores dedican al mantenimiento de la infraestructura**: los entrevistados notaron que los servicios de nube de RHOS eliminaron la necesidad de que los desarrolladores mantuvieran la infraestructura de desarrollo de la aplicación, gracias a lo cual pudieron centrarse en diseñar el producto o la solución. En tres años, este tiempo tiene un valor de más de USD 2,3 millones.
- **Mejora del 50 % en la eficiencia operativa**: gracias a que los servicios de nube de RHOS son gestionados, los entrevistados notaron que el 50 % de los empleados de DevOps, que antes debían gestionar la infraestructura, podían hacer otros trabajos más productivos. En tres años, el aumento de la eficiencia operativa vale más de USD 1,3 millones.*

Pero eso no es todo: el informe también encontró beneficios que no se pueden cuantificar:

- **Satisfacción y permanencia de los desarrolladores**: los entrevistados resaltaron que, gracias a los servicios de nube de Red Hat OpenShift, los desarrolladores pudieron dividir las actualizaciones en partes más pequeñas. Por consiguiente, no solo disminuyó la presión de tener que realizar pruebas extensas en plazos limitados, sino también la necesidad de responder a los simulacros en la producción.
- **Seguridad y disminución del riesgo**: los entrevistados expresaron que los servicios de nube de Red Hat OpenShift automatizaron ciertas características y actualizaciones de seguridad, lo cual eliminó la necesidad de hacer tareas de mantenimiento manual sin dejar de garantizar un entorno seguro.
- **Confiabilidad**: los entrevistados señalaron que los servicios de nube de RHOS aumentaron la confiabilidad de su plataforma de aplicaciones a largo plazo, ya que presenta menos interrupciones o fallas en el sistema, aun con un entorno en expansión.
- **Portabilidad y continuidad empresarial**: los entrevistados también destacaron que los servicios de nube de Red Hat OpenShift garantizaron la continuidad empresarial y brindaron asistencia en cuanto a la estrategia de recuperación ante desastres gracias a la portabilidad, la capacidad de ajuste y la flexibilidad que ofrecen.

Fuente: Forrester, *El Total Economic Impact™ de los servicios de nube de Red Hat OpenShift*, diciembre de 2021

*** Lectura adicional:** Las empresas aceleran la agilidad con servicios gestionados en la nube

"¿Red Hat OpenShift o solo Kubernetes?": El costo de diseñar su propia plataforma de aplicaciones de Kubernetes

A menudo, nos referimos a Red Hat OpenShift como "Kubernetes empresarial", pero puede ser difícil entender lo que significa esto en una primera instancia. Los clientes suelen preguntar: "¿OpenShift o Kubernetes solo?". Sin embargo, es importante que entendamos que **OpenShift usa Kubernetes**. Kubernetes proporciona la base sobre la cual se diseña la plataforma OpenShift, así como la mayoría de las herramientas necesarias para ejecutarla.

Kubernetes es un proyecto open source muy importante de Cloud Native Computing Foundation y una tecnología fundamental para ejecutar contenedores.

No obstante, es posible que los usuarios potenciales se pregunten: "¿**Puedo ejecutar mis aplicaciones con Kubernetes solo**?". Muchas empresas comienzan implementando Kubernetes y se encuentran con que pueden ejecutar un contenedor o, incluso, una aplicación empresarial en solo unos días. A medida que inician las operaciones del Día 2, que aumentan los requisitos de seguridad y que se implementan más aplicaciones, algunas caen en la trampa de diseñar su propia **plataforma como servicio (PaaS)** con esta tecnología. Agregan un controlador de entrada open source, escriben algunos scripts para conectar a sus canales de **integración continua/implementación continua (CI/CD)** y luego intentan implementar una aplicación más compleja. Y es ahí cuando comienzan los problemas, ya que, si la implementación de Kubernetes representa la punta del iceberg, la complejidad de la gestión del Día 2 es la gran masa de hielo escondida debajo del agua que es capaz de hundir un barco.

Es posible resolver estos desafíos y problemas, pero se suele necesitar un equipo de operaciones compuesto de varias personas, así como muchas semanas y meses de esfuerzo, para diseñar y mantener esta "PaaS personalizada creada en Kubernetes". Esto afecta la eficiencia de la empresa y presenta complicaciones en el mantenimiento desde una perspectiva de seguridad y certificación, además de crear la necesidad de diseñar todo de cero cuando se incorporan equipos de desarrolladores. Si una empresa enumerara todas las tareas de diseño y mantenimiento que esta plataforma personalizada de Kubernetes (es decir, Kubernetes) requiere, así como todos los elementos adicionales necesarios para ejecutar contenedores con éxito, se agruparían de la siguiente manera:

- **Gestión de clústeres**: incluye instalar sistemas operativos y aplicarles parches; instalar Kubernetes; integrar la autenticación; fortalecer los nodos; aplicar parches de seguridad y configurar las redes de CNI, los ingresos y salidas, el almacenamiento permanente y la nube o el entorno multicloud subyacente.
- **Servicios de aplicaciones**: estos incluyen la incorporación de registros, el control de la seguridad, la supervisión del rendimiento, la aplicación de los parches de seguridad, el registro de los contenedores y la configuración del proceso de organización de la aplicación.
- **Integración de desarrolladores**: esto comprende la compatibilidad del middleware; la provisión de paneles de rendimiento de aplicaciones; el RBAC y la integración CI/CD, de herramientas de desarrollo/IDE y de marcos.

Hay muchas más tareas y tecnologías para agregar a la lista, las cuales son fundamentales para que cualquier empresa use los contenedores con seriedad. Sin embargo, la complejidad, el tiempo y el esfuerzo de configuración de todo es insignificante si se compara con el mantenimiento constante de esas piezas individuales. Cada integración necesita pruebas minuciosas y cada elemento y actividad tendrá ciclos de lanzamiento, políticas de seguridad y parches distintos.

¿Qué obtiene con Red Hat OpenShift en comparación con Kubernetes solo?

Si una empresa ejecuta contenedores en producción que se diseñaron con Kubernetes solo, los elementos mencionados en la sección anterior en general se instalan e integran juntos para crear una especie de plataforma de aplicaciones con características propias.

Sin embargo, Azure Red Hat OpenShift combina todas estas partes en una única plataforma, lo cual facilita el trabajo de los equipos de operaciones y TI y, al mismo tiempo, les brinda a los de aplicaciones todo lo necesario para realizar sus tareas. Cubriremos estos temas en detalle más adelante, pero, con esto en mente, observemos algunas de las diferencias más importantes entre las dos tecnologías:

- **Implementación más sencilla**: implementar una aplicación en Kubernetes puede llevar mucho tiempo. El proceso consiste en incorporar el código de GitHub a una máquina, poner en marcha un contenedor, alojarlo en un registro como Docker Hub y, finalmente, entender su canal de CI/CD, el cual puede ser muy complejo. En cambio, con OpenShift puede directamente automatizar las tareas complejas y de backend: solo necesita crear un proyecto y cargar su código.
- **Seguridad**: actualmente, vemos que la mayoría de los proyectos de Kubernetes se realizan en equipos compuestos de varios desarrolladores y operadores. Kubernetes es compatible con controles tales como RBAC e IAM, pero aun así requiere una instalación y configuración manuales, lo cual lleva tiempo. Después de años de experiencia, Red Hat y OpenShift lograron identificar las prácticas recomendadas en materia de seguridad, las cuales están disponibles para los clientes desde el comienzo. Solo debe agregar usuarios nuevos y OpenShift se encargará de gestionar los espacios de nombres y crear diferentes políticas de seguridad.
- **Flexibilidad**: con Azure Red Hat OpenShift, podrá aprovechar las prácticas recomendadas de implementación, gestión y actualización conocidas. Además, eliminaremos las tareas complejas de backend, por lo que podrá distribuir sus aplicaciones con más rapidez. La plataforma de Kubernetes no solo es útil para equipos que prefieren que les indiquen cómo realizar las tareas y que se benefician de un enfoque simplificado, sino que también les permite personalizar de forma manual su canal de DevOps de CI/CD, lo cual ofrece mayor flexibilidad y creatividad a la hora de desarrollar los procesos.

- **Operaciones cotidianas**: los clústeres están conformados por un conjunto de máquinas virtuales, así que, inevitablemente, el equipo de operaciones deberá poner en marcha otras nuevas y agregarlas. Es posible que en el proceso de configuración con Kubernetes se deban desarrollar scripts para configurar el autorregistro o la automatización de la nube, por ejemplo, lo cual puede resultar complejo y consumir mucho tiempo. En cambio, la plataforma Azure Red Hat OpenShift automatiza y gestiona las operaciones de implementación, ajuste y actualización de clústeres.
- **Gestión**: si bien puede aprovechar las herramientas y los paneles estándares de Kubernetes presentes en cualquier distribución, la mayoría de los desarrolladores necesitan una plataforma más completa y sólida. Azure Red Hat OpenShift ofrece una consola web muy buena que se basa en las API y las funciones de Kubernetes para que los equipos de operaciones gestionen sus cargas de trabajo.

En el *Capítulo 8, Análisis de la plataforma de la aplicación*, puede encontrar una descripción guiada de muchas de las funciones complementarias importantes de esta plataforma.

Resumen

Muchos clientes de Microsoft y Red Hat eligen Azure Red Hat OpenShift como su plataforma preferida para adoptar aplicaciones en contenedores.

En el siguiente capítulo, analizaremos Red Hat OpenShift como un servicio de nube y hablaremos de su arquitectura, integración y gestión.

Capítulo 3

Azure Red Hat OpenShift

Con **Azure Red Hat OpenShift**, puede implementar los clústeres completamente gestionados de la plataforma sin tener que preocuparse por diseñar ni gestionar la infraestructura necesaria para ejecutarlos.

Azure Red Hat OpenShift es una plataforma de aplicaciones desarrollada en la nube a la que puede acceder según lo solicite. Microsoft y Red Hat se encargan en conjunto de su diseño, ejecución y soporte. Además, un equipo especializado de **ingeniería de confiabilidad del sitio** (**SRE**) automatiza los clústeres de OpenShift, ajusta su capacidad y los protege mientras trabaja para ofrecer una experiencia integrada de soporte. En esta plataforma, no es necesario ejecutar máquinas virtuales ni aplicar parches. Microsoft y Red Hat se encargan de aplicarlos en los nodos de plano de control y de trabajo, los cuales también actualizan y supervisan. Los clústeres de Azure Red Hat OpenShift se incluyen en la suscripción de Azure y en la factura del servicio.

Agilice la distribución de las aplicaciones con Azure Red Hat OpenShift:

- Permita que los desarrolladores generen innovaciones mediante los canales integrados de CI/CD y, luego, conecte fácilmente sus aplicaciones a cientos de servicios de Azure, como MySQL, PostgreSQL, Redis y Azure Cosmos DB.
- Automatice la implementación, la configuración y la ejecución y deshágase de las complejidades que obstaculizan su productividad.
- Ajuste la capacidad a su propio ritmo: en tan solo minutos, puede implementar un clúster de gran disponibilidad que contenga tres nodos de trabajo y ajustar su capacidad a medida que cambie la demanda de aplicaciones. Elija entre nodos de trabajo que sean estándares, que ofrezcan alta capacidad de memoria o que permitan un uso elevado de la CPU.
- Aproveche las operaciones, la seguridad y el cumplimiento de nivel empresarial y disfrute de una experiencia integrada de soporte.

A continuación, profundizaremos en la información técnica acerca del diseño y la ejecución de Red Hat OpenShift.

Arquitectura

Azure RHOS utiliza los servicios de infraestructura de Azure como base para la instalación de RHOS: las máquinas virtuales, los grupos de seguridad de red, las cuentas de almacenamiento, entre otros. La arquitectura de Azure se implementa por completo con la suscripción, lo que facilita la integración con los otros servicios que ya ejecuta en su cuenta.

La propia plataforma OpenShift está diseñada a partir de Red Hat Enterprise Linux CoreOS, tecnología en la cual se aloja la arquitectura basada en microservicios de pequeñas unidades individuales que funcionan en conjunto. El servicio `Kubelet`, la base del clúster de Kubernetes, se ejecuta en cada máquina virtual. La base de datos de esta arquitectura, la cual se ejecuta en el plano de control, es **etcd**, un tipo de almacenamiento confiable de clave-valor agrupado en clústeres.

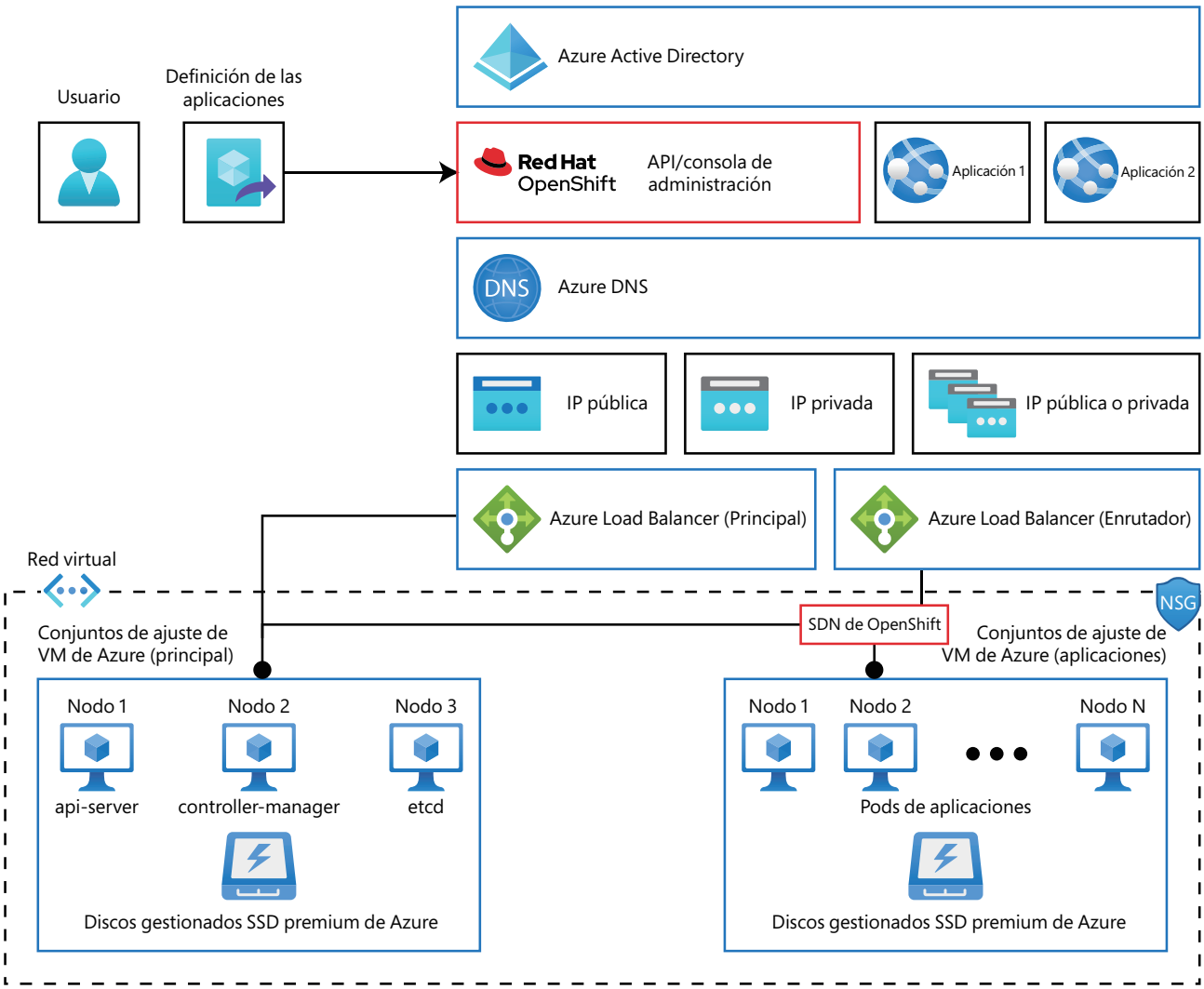


Figura 3.1: la arquitectura de Azure Red Hat OpenShift

En las dos secciones siguientes, *Informática: nodos de control y de trabajo,* y *Redes*, se explican en más detalle los elementos incluidos en el diagrama.

Informática:nodos de control y de trabajo

La arquitectura de RHOS se ejecuta en máquinas virtuales (nodos) que cumplen funciones de control, de infraestructura o de aplicaciones. Sin embargo, la versión 4 de Azure RHOS todavía no es compatible con los nodos de infraestructura, por lo tanto, en este e-book, solo se incluye información acerca de los dos restantes: los de control y de aplicaciones.

Los nodos de **control**, tradicionalmente llamados nodos **maestros**, son máquinas virtuales que contienen elementos esenciales para el clúster de Kubernetes, como el servidor de la API o el de controlador y administrador, el programador y el almacenamiento etcd. Estos elementos gestionan el clúster de Kubernetes y programan la ejecución de pods en los nodos de aplicaciones o trabajo.

- **El servidor de la API de Kubernetes** se encarga de validar y configurar los datos necesarios para los pods, los servicios y los controladores de replicación, además de funcionar como punto focal del estado compartido del clúster.
- **El controlador y administrador de Kubernetes** supervisa etcd en busca de cambios en la replicación, los espacios de nombres y los objetos de controlador de las cuentas de servicio, para luego fortalecer el estado especificado con la API. Este conjunto de procesos conforma un clúster, que posee un líder activo a la vez.
- **El programador de Kubernetes** supervisa los pods que se crean sin un nodo asignado y selecciona el más adecuado para alojarlos.
- La base de datos **etcd** almacena el estado maestro permanente, mientras que el resto de los elementos lo observan para detectar cambios y ajustarse al estado especificado.

En los nodos **de aplicaciones** o trabajo, se ejecutarán, justamente, las aplicaciones.

Cada nodo de un clúster de Kubernetes ejecuta `Kubelet`, que mantiene la **interfaz de tiempo de ejecución de contenedores** conocida como **cri-o**; un proxy de servicios; entre otros servicios esenciales. Todos ellos están conectados con la tecnología definida por software de OpenShift. En el caso de Azure RHOS, se trata de una **red virtual abierta (OVN)** que se ejecuta en la red virtual de Azure.

La plataforma crea nodos que se ejecutan en máquinas virtuales conectadas a discos de Azure con grandes capacidades de almacenamiento, de 1 TB inclusive. Esto se debe a que, en Azure, el rendimiento del almacenamiento, en términos de operaciones de E/S por segundo, depende de la capacidad del disco. Por ejemplo, una capacidad de 1 TB ofrece un ancho de banda suficiente para contener el disco que utiliza la base de datos etcd.

Redes

Azure Red Hat OpenShift requiere una sola red virtual de Azure con dos subredes configuradas: una para los nodos de plano de control y otra para los de trabajo. El tamaño mínimo de ambas es /27 (32 direcciones), pero tenga en cuenta que, si desea ampliar el clúster en un futuro, las subredes no deberían ser muy pequeñas.

Los dos equilibradores de carga de Azure (que aparecen como **Maestro** y **Enrutador** en el diagrama) dirigen el tráfico de la siguiente manera:

- Equilibrador de carga maestro (o del plano de control): dirige el tráfico entrante para los usuarios de la API de OpenShift y brinda conectividad de salida para los nodos de dicho plano.
- Equilibrador de carga enrutador (o de las aplicaciones): dirige el tráfico entrante hacia las aplicaciones que se ejecutan en OpenShift por medio de un enrutador o controlador de ingreso de OpenShift y brinda conectividad de salida para los nodos de dicho plano.

La documentación sobre redes incluye un artículo excelente sobre la configuración, los requisitos y las limitaciones de las redes.

Integracióncon otros servicios de Azure

Azure RHOS es un servicio desarrollado en Azure, lo que permite que pueda implementarse junto con muchos de los servicios de Azure que usted acostumbra usar e integrarse fácilmente con ellos. A continuación, haremos una descripción general de algunos de los servicios de infraestructura de Azure con integraciones comunes.

En la *figura 3.2* podemos ver muchos de los puntos de integración del servicio más comunes para OpenShift en Azure.

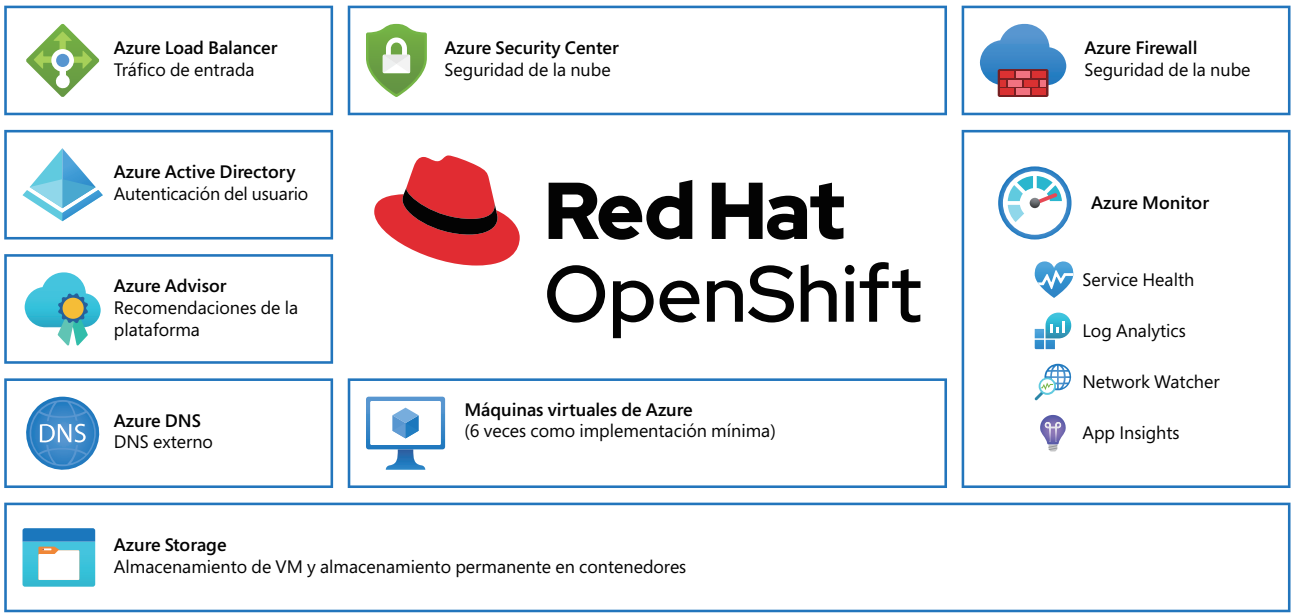


Figura 3.2: puntos de integración del servicio más comunes en Azure

Además, las aplicaciones que se ejecutan en OpenShift pueden integrarse muy estrechamente con los servicios de Azure por medio de Azure Service Operator. Veremos este tema de forma más detallada en el *Capítulo 9: Integración con otros servicios*.

Gestión

Para comprender de manera sencilla qué parte del servicio Azure Red Hat Openshift está gestionada, podemos pensar que todo, desde el centro de datos hasta los operadores del clúster, está gestionado. Los operadores son servicios que se ejecutan en los nodos del plano de control y se encargan de la supervisión, las actualizaciones y el estado del clúster. Esto significa que tanto Microsoft como Red Hat supervisan, mantienen y gestionan de manera conjunta estos elementos para que el clúster continúe en línea. El cliente es responsable de todas las tareas que no formen parte de los operadores del clúster.

Como usuario de los servicios de Azure RHOS, posee acceso completo de **cluster-admin** (administrador del clúster), es decir, comparte la responsabilidad de conservar la integridad de su clúster. Para comprender lo que puede hacer o no con el clúster, es importante que tenga en cuenta las políticas de soporte.

En la Matriz de responsabilidades de Azure Red Hat OpenShift, puede encontrar una descripción detallada y precisa de las acciones que corresponden a Microsoft y Red Hat como parte del servicio de nube.

Autenticacióny autorización

Para proveer autenticación a los clústeres de Azure RHOS, puede utilizar Azure Active Directory, pero no es obligatorio. Existen otras alternativas, como el inicio de sesión de GitHub y el uso de un archivo de contraseña.

Si utiliza Azure Active Directory, las API de Azure RHOS y de Kubernetes se encargan de reenviar las solicitudes de autenticación. Los usuarios solo deben presentar sus credenciales y se los autoriza según su función.

La capa de autenticación identifica al usuario asociado con la solicitud enviada a la API de Azure RHOS y, luego, determina si debe aprobarla en función de su información.

En la sección *Autenticación*: *Azure Active Directory*, puede encontrar las instrucciones para configurar el servicio. El proceso funciona de forma similar con otros proveedores de autenticación.

En el caso de la autorización, esta se lleva a cabo en el motor de políticas de Azure RHOS, que define acciones, "crear pods" o "enumerar servicios", por ejemplo, y las agrupa en funciones dentro de un documento de políticas. Las funciones se vinculan a los usuarios o a los grupos mediante el identificador correspondiente. Cuando la cuenta de un usuario o un servicio intenta realizar una acción, el motor de políticas verifica las funciones asignadas al usuario (por ejemplo, administrador cliente o administrador del proyecto actual) antes de permitirle continuar.

La relación entre las funciones del clúster y sus enlaces, las funciones locales y sus enlaces, los usuarios, los grupos y las cuentas de servicios se detallan en la *figura 3.3*:

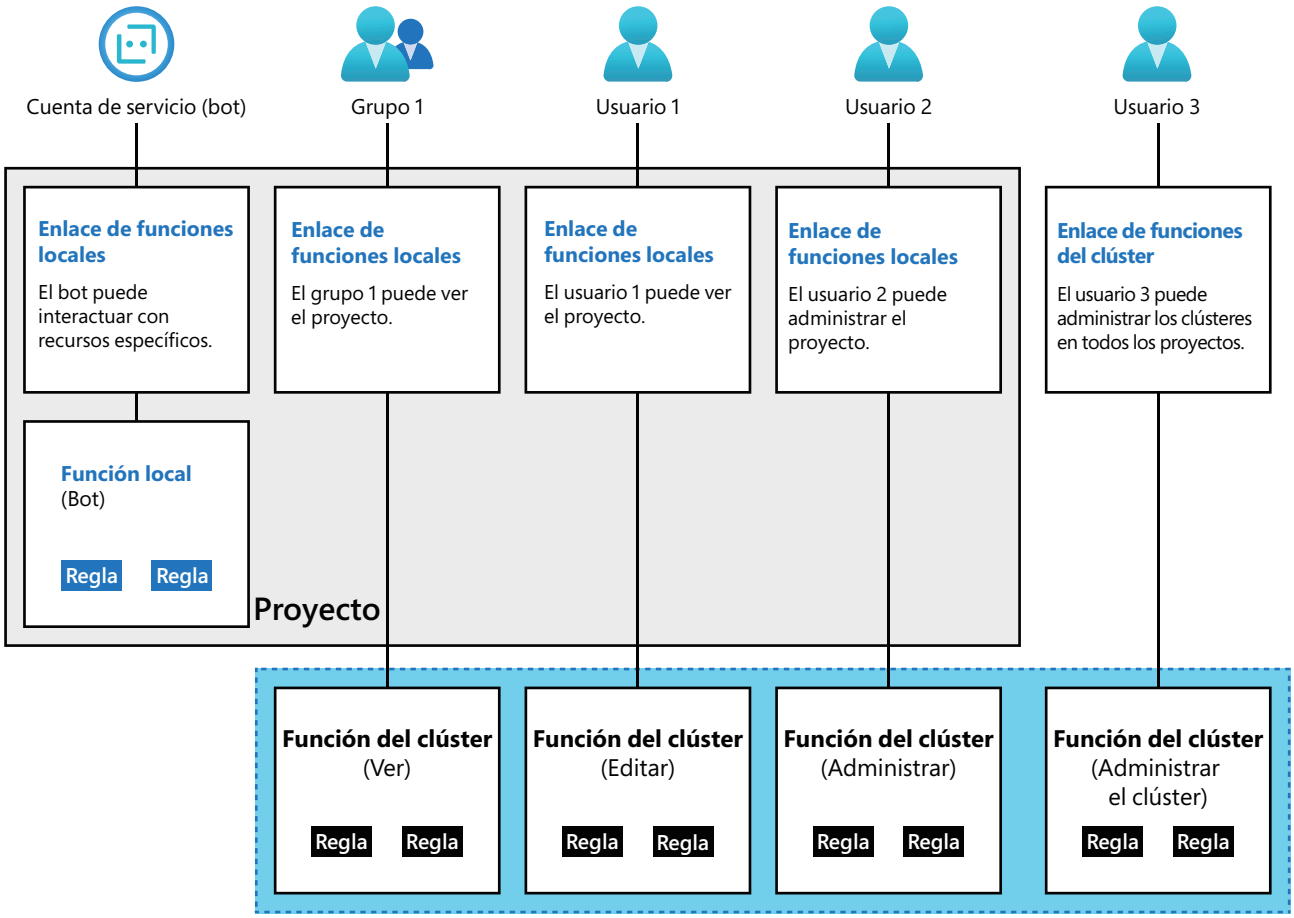


Figura 3.3: relación entre las funciones

En la documentación de RHOS puede encontrar una lista completa de los proveedores de autenticación.

Soporte

Azure Red Hat OpenShift gestiona el soporte de manera inigualable. Los equipos de soporte de Microsoft y Red Hat trabajan de forma conjunta, a la par del equipo global de ingeniería de confiabilidad del sitio (SRE), para garantizar que funcione el servicio.

Los ingenieros de Microsoft y Red Hat clasifican las solicitudes de soporte que los clientes realizan en el portal de Azure y se ocupan de ellas rápidamente, independientemente de que el problema se origine a nivel de la plataforma de Azure o de OpenShift.

Puede encontrar un resumen del proceso integrado de soporte aquí:

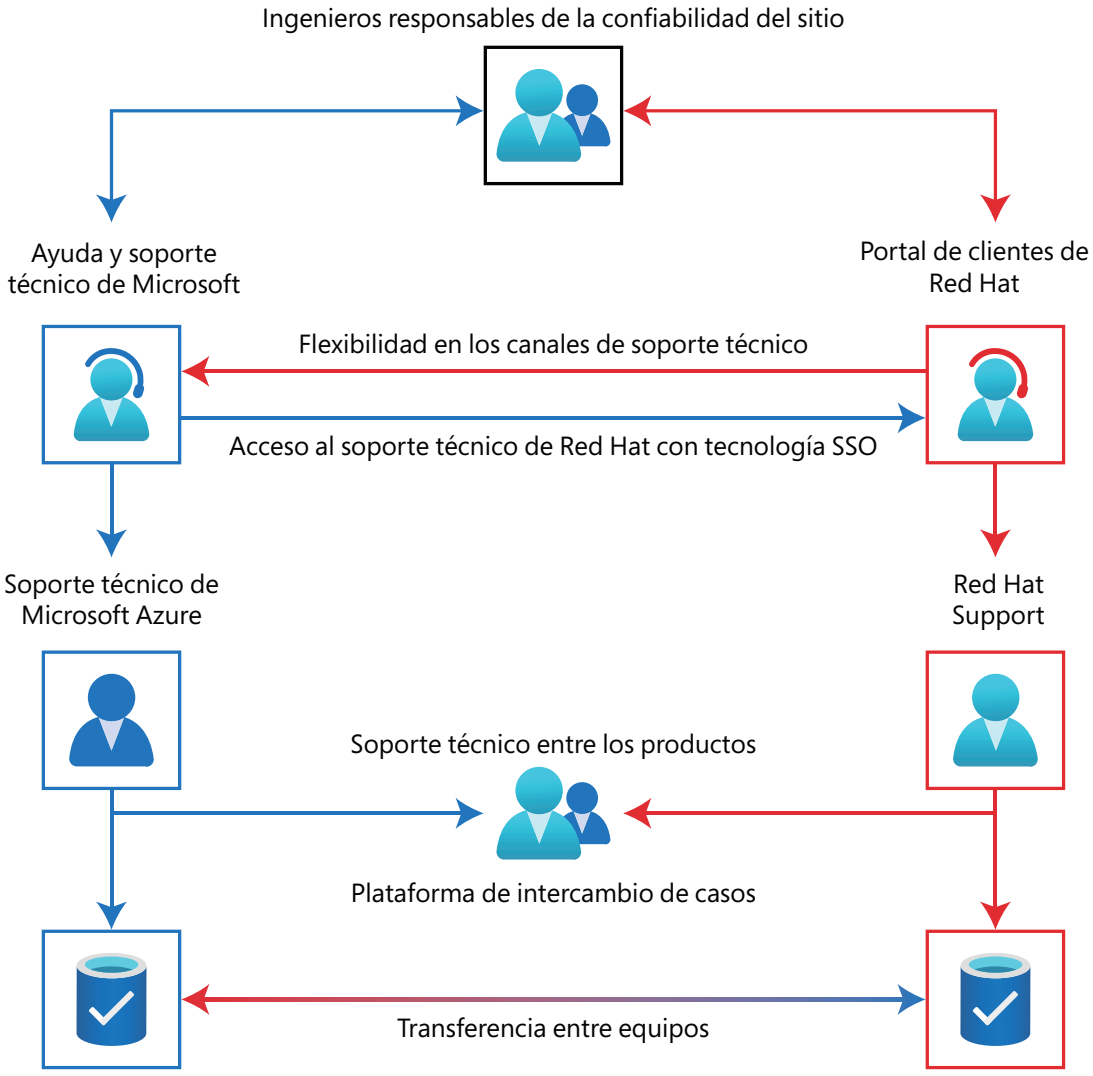


Figura 3.4: el proceso integrado de soporte

En la figura 3.4, se demuestra que los clientes pueden enviar una solicitud de soporte a los portales de ambos: Microsoft o Red Hat Sin embargo, tenga en cuenta que, para acceder al portal de soporte de Red Hat, el clúster debe estar registrado en OpenShift Cluster Manager.

Si el cliente presta su consentimiento a través de la plataforma que utiliza para hablar sobre el caso, los ingenieros de soporte de ambas empresas pueden colaborar en conjunto sin dificultad y en cualquier etapa de la resolución del problema. Los ingenieros de soporte de ambas empresas pueden comunicarse con el equipo de SRE, el cual, de ser necesario, puede aplicar medidas de corrección para reparar los clústeres.

Preciosy suscripciones

Uno de los principales beneficios de Azure RHOS frente a otras opciones con un enfoque autodidacta es que los servicios de informática, redes, almacenamiento e infraestructura de Azure, así como las suscripciones de OpenShift, se facturan juntos. Para obtener una estimación de los costos de la solución, solo debe agregar Azure Red Hat OpenShift en el simulador de presupuestos de la [Calculadora de precios de Azure](#).

A continuación, se detallan pasos para estimar costos en la página:

1. Escriba *openshift* en el cuadro de búsqueda y agréguelo a una nueva estimación.

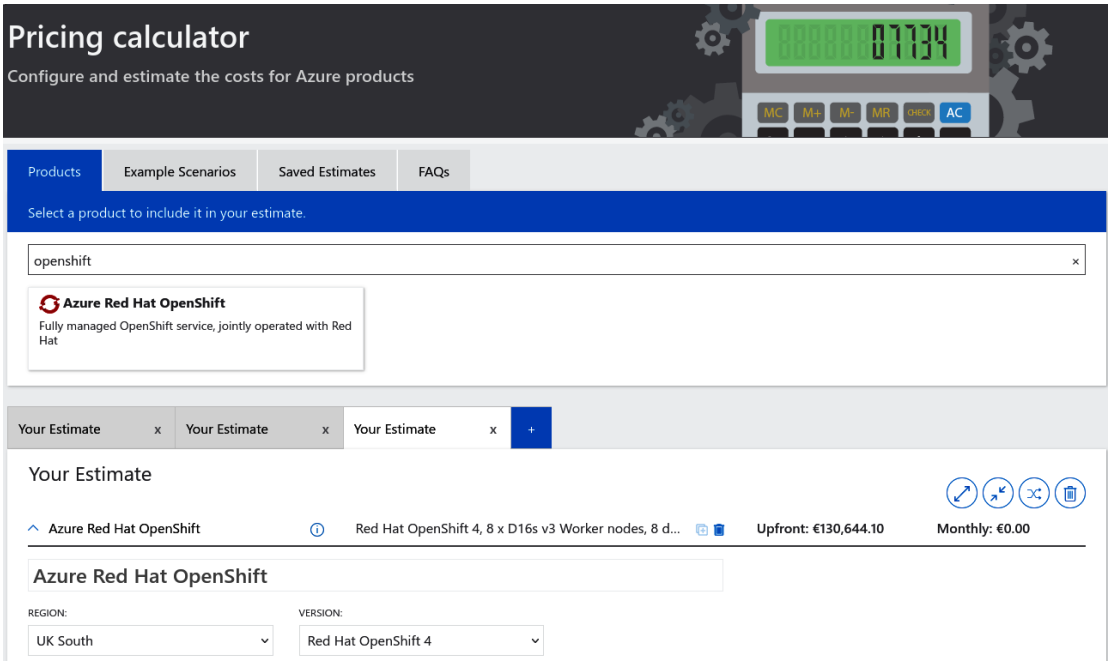


Figura 3.5: el panel de la calculadora de precios

2. Al final de la página, encontrará un menú desplegable con el que puede cambiar el precio a su moneda local. En este ejemplo, se utiliza GBP (£).
3. Elija la región de Azure en la que implementará Azure RHOS. Debido a los distintos costos informáticos, el precio varía ligeramente entre las regiones de Azure.

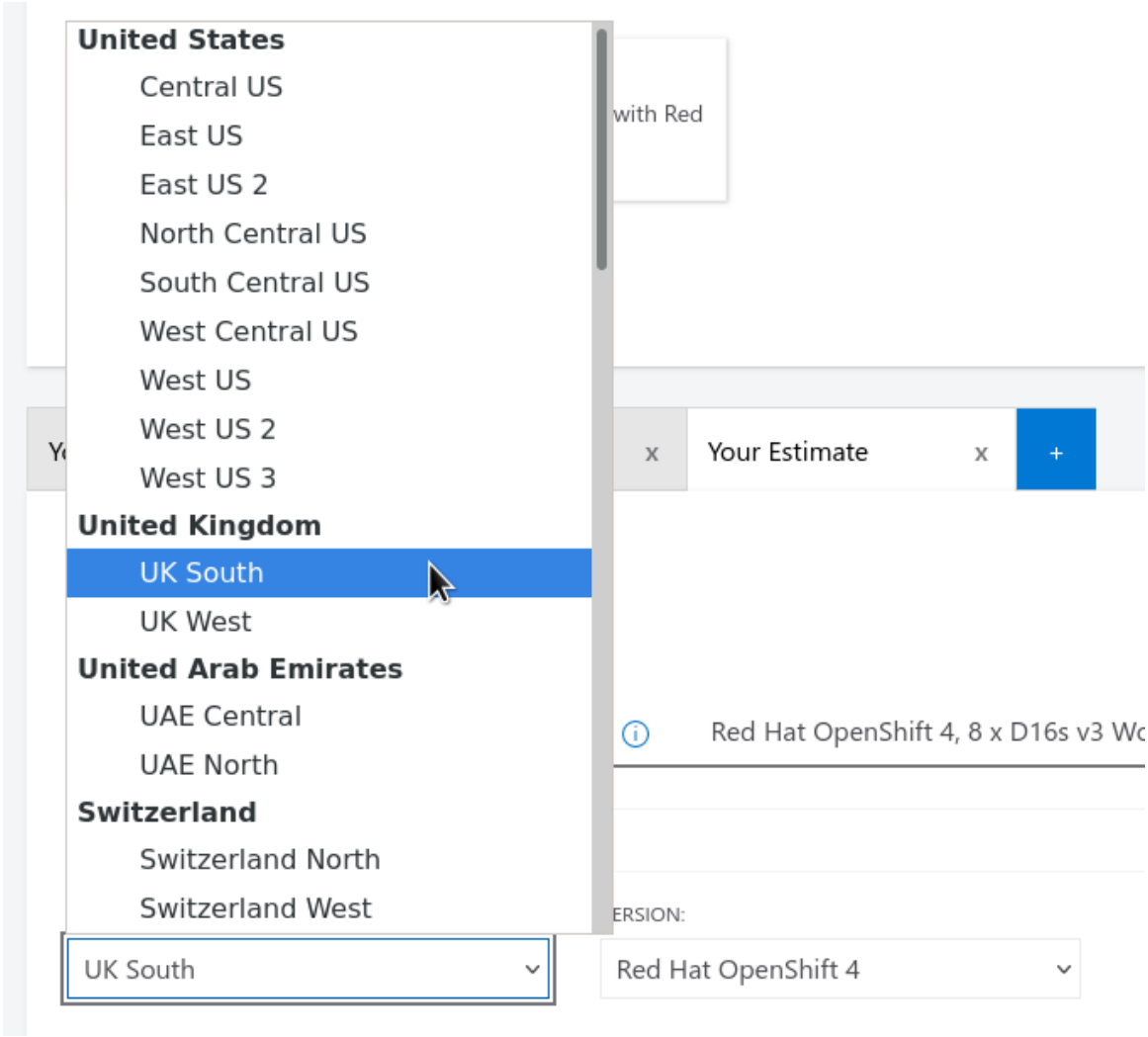


Figura 3.6: seleccionar la región de Azure

4. **Los Nodos de trabajo**, también conocidos como nodos de aplicaciones, representan la ubicación en donde se ejecutarán, justamente, sus aplicaciones. Bajo **Opciones de ahorro**, encontrará dos secciones. **Licencia** hace referencia al costo de las suscripciones de OpenShift y **Máquina virtual**, al costo de los servicios informáticos necesarios para ejecutar el clúster. Como máximo, un clúster admite 100 nodos de trabajo.

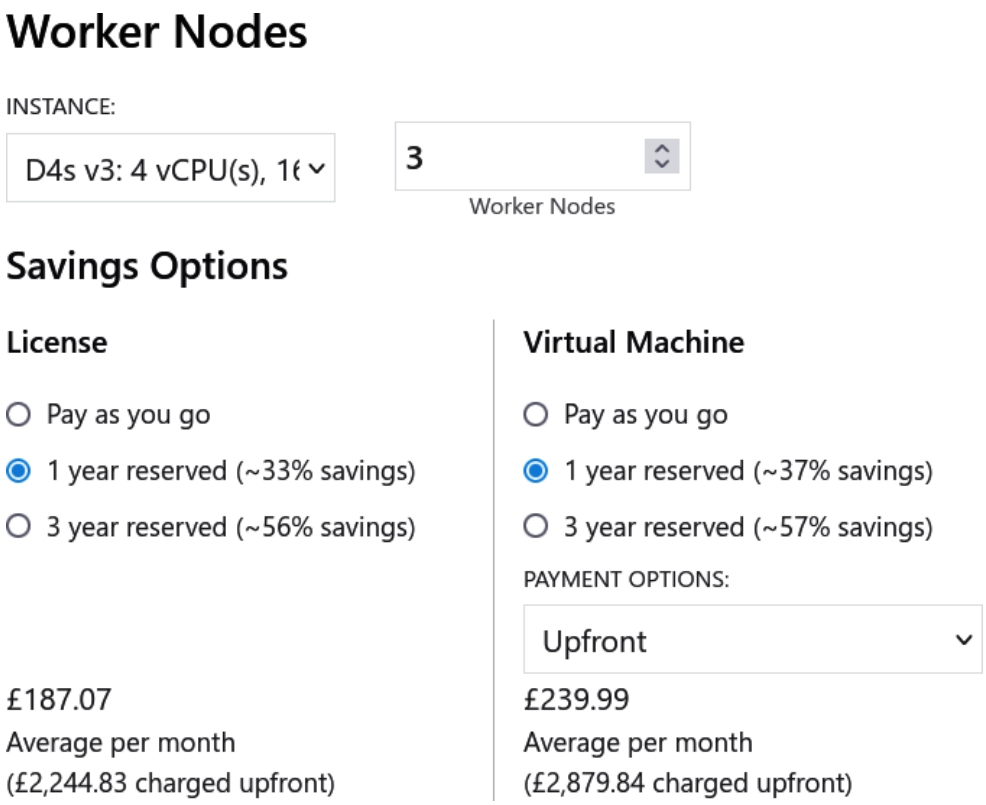


Figura 3.7: los nodos de trabajo en detalle

5. **Discos de sistema operativo administrados** hace referencia a la capacidad de los discos que utiliza Red Hat CoreOS para las particiones del sistema operativo, no para los volúmenes permanentes de las aplicaciones, los cuales se implementan de manera separada.

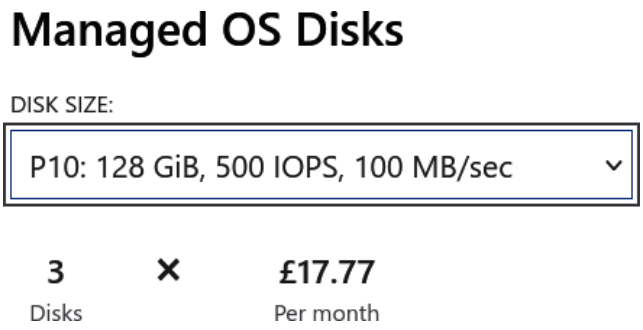


Figura 3.8: los discos del sistema operativo administrados

6. También encontrará una sección similar para los **Nodos maestros**, los cuales se conocen comúnmente como nodos de control. Tenga en cuenta que estos nodos consumen más almacenamiento en el disco que aquellos de trabajo. Esto se debe a que requieren un índice mayor de operaciones de E/S por segundo y mejor rendimiento, lo cual, en Azure, se traduce en discos de mayor capacidad. Para lograr la estabilidad del clúster, siempre debemos utilizar tres nodos.

Debe notarse que la calculadora está diseñada para mostrar precios estimados, ya que los reales variarán en función del uso del servicio.

Instancias reservadasde la máquina virtual de Azure

La expresión **instancia reservada** hace referencia al compromiso de utilizar ese recurso durante una cantidad determinada de tiempo: de uno a tres años. Existen opciones para utilizar las máquinas virtuales de Azure RHOS como instancia reservada.

Por lo general, las empresas eligen utilizar las instancias reservadas para obtener un descuento importante en la infraestructura como servicio (IaaS), cuando planean ejecutar el clúster por un largo período. Los entornos de producción, por ejemplo, suelen utilizarlas para su ejecución. Es importante aclarar que estas instancias no modifican el nivel del servicio ni la arquitectura del clúster.

Obtenga más información acerca de las instancias reservadas de Azure.

Resumen

En este capítulo, consideramos las particularidades de los servicios gestionados de nube de Azure Red Hat OpenShift. Hicimos una descripción general de su arquitectura y de la integración con otros servicios de Azure (los cuales detallaremos en el *Capítulo 9: Integración con otros servicios*), así como de la gestión, la autenticación, el soporte y los precios.

En el capítulo siguiente, nos centraremos en las preguntas que una empresa debe hacerse y las decisiones que necesitará tomar durante la etapa previa a la implementación de Azure Red Hat OpenShift.

Capítulo 4

Etapa previa a la implementación: preguntas sobre la arquitectura empresarial

Muchas empresas verán Azure Red Hat OpenShift en el portal de Azure y, si leen la documentación, aunque sea una sola vez, podrán implementar un clúster de RHOS casi sin esfuerzo y de manera exitosa. Sin embargo, si planifican la implementación con mayor detenimiento y resuelven algunas dudas primero, ahorrarán mucho más tiempo cuando tengan que eliminar los clústeres y volver a prepararlos.

Este capítulo está basado en experiencias prácticas y reales de trabajo con múltiples clientes. Aborda las preguntas más frecuentes que surgen durante la etapa previa a la implementación y que se deben responder primero. En este capítulo, cubriremos los siguientes temas:

- La cantidad de clústeres que se necesita considerando, entre otros, las etapas y la producción
- La visibilidad de las redes públicas y privadas
- La conectividad híbrida, como la que vincula a las soluciones en las instalaciones

Comenzaremos por dilucidar la cantidad de clústeres que necesita.

Cantidad necesaria de clústeres

Existen muchos patrones para implementar OpenShift, pero una pregunta común es "¿cuántos clústeres necesita mi empresa?". Esta es una decisión que depende de la empresa, por supuesto, pero en los párrafos siguientes brindamos algunos consejos que lo ayudarán a responder la pregunta.

Etapas del ciclo de vida: el desarrollo, las pruebas y la producción

La mayoría de las empresas, independientemente de su tamaño, implementan los sistemas empresariales de TI siguiendo ciertas etapas del ciclo de vida. Este enfoque también suele denominarse patrón de etapas. Las tres etapas más comunes son el desarrollo, las pruebas y la producción. Es importante dividir las tareas en diferentes etapas, ya que, de esta manera, se pueden probar los cambios que se realicen a las aplicaciones o las implementaciones en un entorno seguro y antes de que lleguen a la producción. El patrón de etapas más común y recomendable implica tener, como mínimo, tres clústeres separados de Azure RHOS:

- **Clúster de desarrollo:** aquí los desarrolladores y los operadores pueden probar lo que consideren necesario. Puede ser grande y estar conformado por un entorno de pruebas (sandbox), pero es más común y, por lo general, más seguro, tener uno más pequeño y de poca duración en el que se puedan realizar y eliminar las pruebas con frecuencia.
- **Clúster de pruebas:** aquí se prueban y validan los cambios que se planifican para el clúster, como los parches o los cambios en la configuración, antes de que se lancen a la etapa de producción. Algunas empresas suelen llamarlo "preproducción", pero este término también puede referirse a un entorno completamente distinto.
- **Clúster de producción:** aquí se ejecutan las aplicaciones.

Además de lo que ya mencionamos, algunas empresas utilizan entornos adicionales tales como los de pruebas de la integración, pero solo usted puede estimar la cantidad que necesita. Si no está seguro, puede observar los patrones de implementación más comunes que utilizan otras aplicaciones empresariales similares.

Si utiliza Azure RHOS para aplicaciones que no considera fundamentales, puede ser aceptable que posea solo dos clústeres (si fusiona el de desarrollo con el de pruebas) o, incluso, uno solo que incluya las tres etapas. Con este enfoque, tiene dos ventajas: mantiene bajos los costos y reduce la cantidad total de clústeres que debe gestionar. Asimismo, si opera con un solo clúster, los administradores pueden utilizar espacios de nombres separados para el desarrollo, las pruebas y la producción. Sin embargo, ejecutar un solo clúster también tiene desventajas:

- Los cambios que involucran a todo el clúster (como los parches de software) pueden introducir problemas en la etapa de producción, los cuales podrían haberse detectado y prevenido si se ejecutaban en un entorno de pruebas.
- Si una aplicación tiene un comportamiento inesperado en un entorno de pruebas o de desarrollo, es decir, si genera una gran cantidad de contenedores o utiliza todo el espacio disponible en el disco, por ejemplo, ocasionará problemas para el entorno de producción.

No podemos establecer una cantidad exacta de clústeres que funcionará de manera ideal en su situación. Como ya mencionamos, debe analizar las aplicaciones empresariales similares con las que cuenta para ver cuántas etapas del ciclo de vida utilizan y, luego, tomar esta decisión.

Continuidad empresarial, recuperación ante desastres y conmutación por error

Además de los entornos estándar que ya describimos, muchas empresas crean, como mínimo, un entorno de producción para la conmutación por error. Este entorno se utiliza si surgen fallas sumamente graves que hacen colapsar la totalidad del clúster o de la región de Azure, y suele denominarse **recuperación ante desastres (DR)**. Por lo general, el entorno se implementa en una región de Azure diferente a la del clúster de producción.

El servicio gestionado de nube incluye un **acuerdo de nivel de servicio (SLA)** que garantiza el 99,95 % de disponibilidad de este, una cantidad aceptable para muchas aplicaciones fundamentales para la empresa. Sin embargo, algunas aplicaciones pueden requerir un nivel mayor, algo que no es posible si se cuenta con un solo clúster. Reflexione sobre el nivel de servicio que necesita su aplicación: ¿supera el 99,95 %?, ¿o ese porcentaje es suficiente?

Si no lo es, puede obtener mayores niveles de disponibilidad del servicio (99,999 %, por ejemplo). Solo debe ejecutar varios clústeres en paralelo y calcular el SLA compuesto. Si desea aumentarlos aún más, todos los clústeres pueden encontrarse en la misma región (por ejemplo, `europaoeste`) o en varias regiones distintas (por ejemplo, `europaoeste` y `europanorte`). En la documentación de Azure que se presenta a continuación, se describe el cálculo de los SLA compuestos cuando se ejecutan varios clústeres.

Documentación de Azure sobre los SLA compuestos

En este e-book, no abordaremos la implementación de varios clústeres de Azure RHOS en varias regiones distintas. Esto se debe a que es algo que debe pensarse detenidamente, ya que se presentan muchos desafíos al diseñar arquitecturas de tal complejidad, como, por ejemplo, hacer que el tráfico ingrese al clúster, elegir los datos que se compartirán entre ellos y decidir la manera de hacerlo.

Regiones y zonas de disponibilidad

Azure RHOS está diseñado para utilizar tres zonas de disponibilidad por cada región en la que se implementa. En Azure, las zonas de disponibilidad son centros de datos autónomos dentro de una región, con su propia fuente de alimentación, refrigeración y conectividad de red. Si inspecciona una implementación de Azure RHOS, verá un único nodo de control (una máquina virtual) y de aplicación en cada zona de disponibilidad.

En la *figura 4.1*, se muestra la distribución de los nodos de control y de aplicación (de trabajo) en tres zonas de disponibilidad dentro de una misma región de Azure:

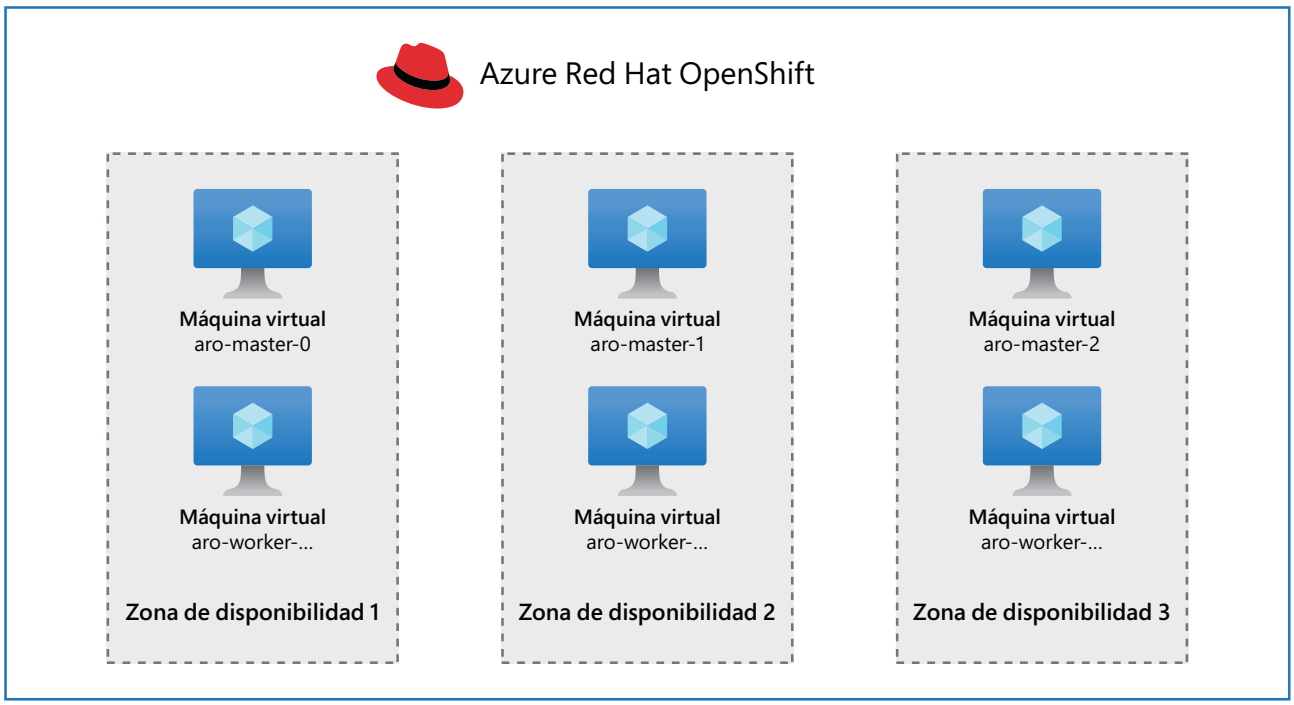


Figura 4.1: distribución de los nodos de control y de aplicación en tres zonas de disponibilidad dentro de una misma región de Azure

Azure RHOS está diseñado para ofrecerse como un servicio completamente gestionado y con alta disponibilidad, por lo que no es posible implementar menos de tres nodos de control y de trabajo.

Conceptos sobre las redes

Como mencionamos en la introducción, en el sitio de documentación de Microsoft hay una página excelente de conceptos sobre las redes:

- Conceptos sobre las redes para Azure Red Hat OpenShift

En esta página, se incluye una descripción detallada de cada elemento de las redes en Azure RHOS: los equilibradores de carga, las direcciones IP públicas y privadas, los grupos de seguridad de red y mucho más.

Algunos temas importantes para comprender son los siguientes:

- La plataforma se implementa en una red virtual actual o en una nueva. Solo se admite una red virtual, pero, de todas formas, no se beneficiaría del uso de varias redes, ya que OpenShift coloca una capa de **red definida por software (SDN)**, llamada OVN, por encima de ella.
- El tamaño mínimo de las subredes principales y de aplicación es de `/27`.
- El CIDR predeterminado del pod es `10.128.0.0/14`.
- El CIDR predeterminado del servicio es `172.30.0.0/16`.
- A cada nodo se le asigna una subred de `/23` (512 direcciones IP) para sus pods. No se puede cambiar este valor.
- Por el momento, no se admiten las IP de salida.
- Es posible controlar el enrutamiento del tráfico saliente, en especial para enviarlo a través de Azure Firewall. Por el momento, esta función está en versión de prueba disponible para el público y puede encontrar documentación sobre ella aquí: Controlar el tráfico de salida.

Visibilidad de las redes públicas o privadas

Es frecuente escuchar que Azure RHOS puede implementarse tanto en redes públicas como privadas. Si bien esto es cierto, es muy útil comprender la diferencia entre configurar el plano de control como público o privado y hacer esto mismo con las aplicaciones en su clúster.

En el caso de la visibilidad del servidor de la API, tiene que tomar la decisión cuando prepara el clúster, porque luego ya no podrá hacerlo.

Cuando llegue el momento de crear un clúster de Azure RHOS, tema que se aborda más adelante en este capítulo, ejecutará el comando `az aro create`, que acepta argumentos acerca de la visibilidad del clúster. Con el siguiente ejemplo, aprenderá a configurar la visibilidad:

Ambas privadas

```
az aro create .... --apiserver-visibility Private --ingress-visibility Private
```

Privada para el servidor de la API y pública para la entrada del nodo de trabajo

```
az aro create .... --apiserver-visibility Private --ingress-visibility Public
```

La visibilidad del mapa de entrada de aplicaciones y de `apiserver` se muestra en el diagrama de la arquitectura de Azure en la *figura 4.2*. Aquí, puede verse la manera en que Azure RHOS utiliza los equilibradores de carga internos y públicos, que es donde se implementan la API y el enrutador en función de las opciones de visibilidad seleccionadas.

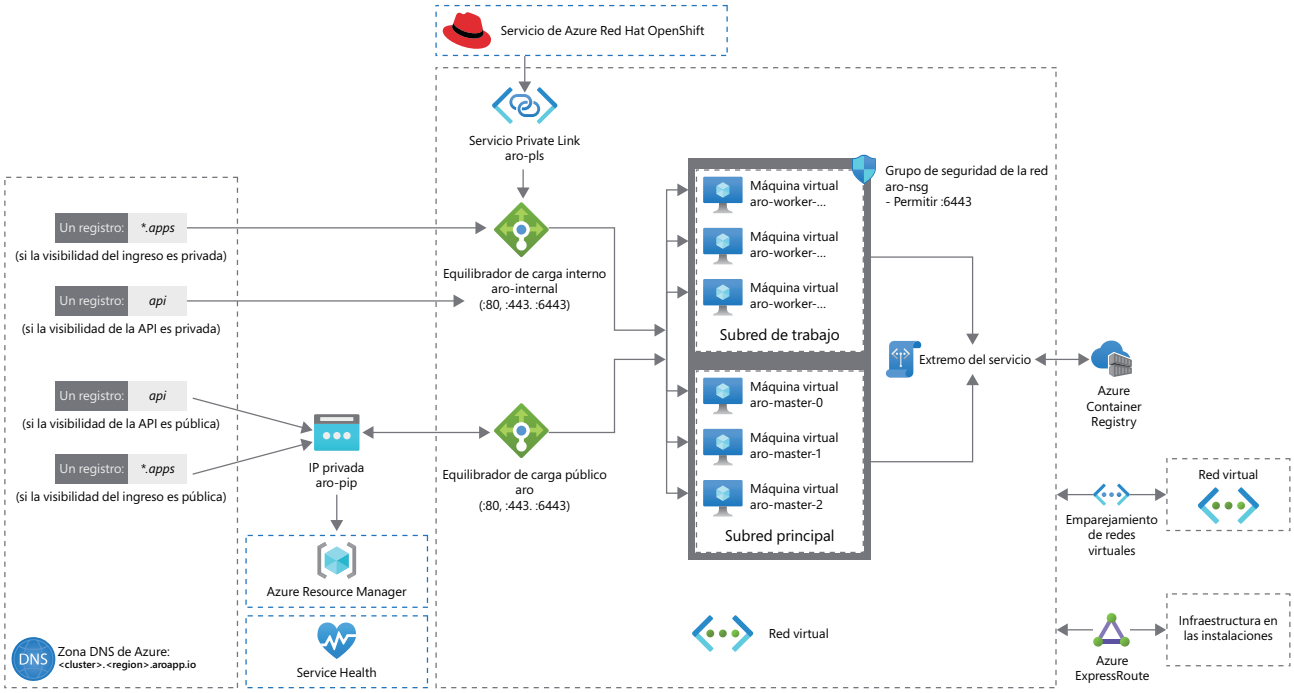


Figura 4.2: equilibradores de carga internos y públicos de Azure en Azure RHOS

A continuación, encontrará descripciones más detalladas de la visibilidad del servidor de la API y de la entrada.

Visibilidad del servidor de la API (nodo de control)

`--apiserver-visibility` puede ser **pública** o **privada**:

- Si es **privada**, no puede accederse desde el Internet público al plano de control de RHOS, donde se ejecuta la API de Kubernetes. Este plano de control, que históricamente se conoce como el "nodo maestro", está diseñado para que los desarrolladores y operadores controlen el clúster, y puede usarse para ajustar la capacidad de las aplicaciones y del clúster, así como implementarlos o eliminarlos. Las empresas que tienen conexiones de Azure ExpressRoute o utilizan una **red virtual privada (VPN)** para acceder a los recursos informáticos deberían elegir la visibilidad privada para la mayoría de los casos.
- Si es **pública**, sí puede accederse al clúster del plano de control desde el Internet público. Esto expone el clúster al riesgo de que cualquier persona de Internet lo ataque, incluso si el acceso está autenticado. A pesar de ello, la visibilidad `pública` es útil para los entornos de laboratorio o de pruebas en los que no puede controlarse la red de origen de los usuarios. Ahora, en el caso de los entornos en los que se almacenan los datos reales, o en cualquier entorno de producción, se recomienda configurar la visibilidad del servidor de la API como `privada`.

En la siguiente lista, se brindan ejemplos en los que se necesita conectividad para el servidor de la API. Se deben tener en cuenta al momento de elegir entre la visibilidad pública y la privada.

- Los desarrolladores que utilizan scripts y herramientas de su IDE (por ejemplo, `kubectl rollout`)
- Los operadores que inspeccionan el estado del clúster (por ejemplo, `kubectl get nodes`)
- Los servidores de CI/CD que necesitan inspeccionar o ajustar el estado de las implementaciones (por ejemplo, Azure DevOps)
- Las herramientas de seguridad del clúster que se conectan a este para examinar su estado
- Las herramientas de supervisión que dependen de la API de Kubernetes

En la mayoría de los casos, las redes se pueden configurar para funcionar con conexiones **privadas**, pero la lista que presentamos anteriormente puede incluir ejemplos adicionales del interior de su empresa, y es posible que encuentre casos en los que deba elegir la visibilidad **pública** para el servidor de la API.

Visibilidad de entrada (nodo de aplicaciones)

`--ingress-visibility` puede ser pública o privada:

- Si es **privada**, los servicios expuestos (que se relacionan con las aplicaciones que se ejecutan en el clúster) no permiten conexiones directas desde el Internet público. Aun así, es posible configurar el enrutamiento de red de manera que los usuarios pasen primero a través de Azure Firewall o del firewall de una aplicación web que podría ejecutarse en una red de Azure separada antes de conectarse a sus aplicaciones. La visibilidad **privada** también funciona bien si las aplicaciones en el clúster son solo para el uso interno dentro de su empresa, por ejemplo, el procesamiento de los pagos, el análisis de datos u otras aplicaciones web internas.
- Si es **pública**, se puede acceder a las aplicaciones en el clúster desde el Internet público. Sin embargo, si alojara un sitio web de comercio electrónico u otra aplicación pública en RHOS, por ejemplo, aún debería configurar un recurso de enrutamiento o una entrada para obtener acceso.

En ocasiones, se hace referencia al término entrada como el "enrutador" de OpenShift.

Recomendaciones para la etapa de producción

Se recomienda que haga lo siguiente:

- Acceda al clúster por medio de una conexión privada, no a través del Internet público. Azure ExpressRoute es la mejor opción cuando necesita que el clúster esté conectado permanentemente desde una red local o una oficina empresarial. Si ese no es el caso, una VPN también le ofrecerá una conexión privada. Puede encontrar más información sobre esto en la sección sobre **conectividad híbrida**.
- Configure la visibilidad del servidor de la API (plano de control) como `privada`. También puede restringir el acceso aún más con firewalls.
- Configure la visibilidad de entrada (plano de las aplicaciones) como `privada` para las aplicaciones que se ejecutan dentro de su empresa. Si tiene un caso de uso de aplicaciones en Azure RHOS que necesiten alojarse en el Internet público, configure un clúster separado: como mínimo, uno para las aplicaciones internas y otro para las externas que tenga visibilidad pública de entrada. Sin embargo, es posible combinar una configuración `pública` con una `privada` para la entrada dentro del mismo clúster.

Por supuesto que la mayoría de las empresas consultarán a sus equipos de redes y seguridad de Azure si es necesario aplicar controles adicionales. Por ejemplo, algunas empresas necesitan colocar un firewall para sus aplicaciones web. Este también es un patrón común para implementar aplicaciones en Azure RHOS.

Conectividad híbrida

La mayoría de las empresas que implementan Azure RHOS ejecutan aplicaciones que necesitan reconectarse a servicios de soporte de entornos locales. Esta reconexión a entornos locales se denomina, comúnmente, conectividad híbrida o arquitectura de nube híbrida. Existen varias maneras de brindar reconexión a los entornos locales. Las dos opciones más destacadas son las siguientes:

- La **VPN**: este método es adecuado para obtener conectividad de baja complejidad con Azure. Por lo general, las VPN se conectan por medio de Azure VPN Gateway. Para obtener más información, visite la página acerca de Azure VPN Gateway.
- **Circuitos de Azure ExpressRoute**: este método es adecuado para obtener una conexión con Azure que sea sólida, exclusiva y permanente. Para obtener más información, visite la página acerca de Azure ExpressRoute.

Independientemente del método que utilice, ambas soluciones brindan la conexión entre las aplicaciones locales y aquellas que se ejecutan en Azure RHOS.

Cuando se conecta desde un entorno local a Azure RHOS, es común hacerlo a través de una red virtual central. La *figura 4.3* es un diagrama explicativo acerca del funcionamiento de la conectividad con Azure ExpressRoute:

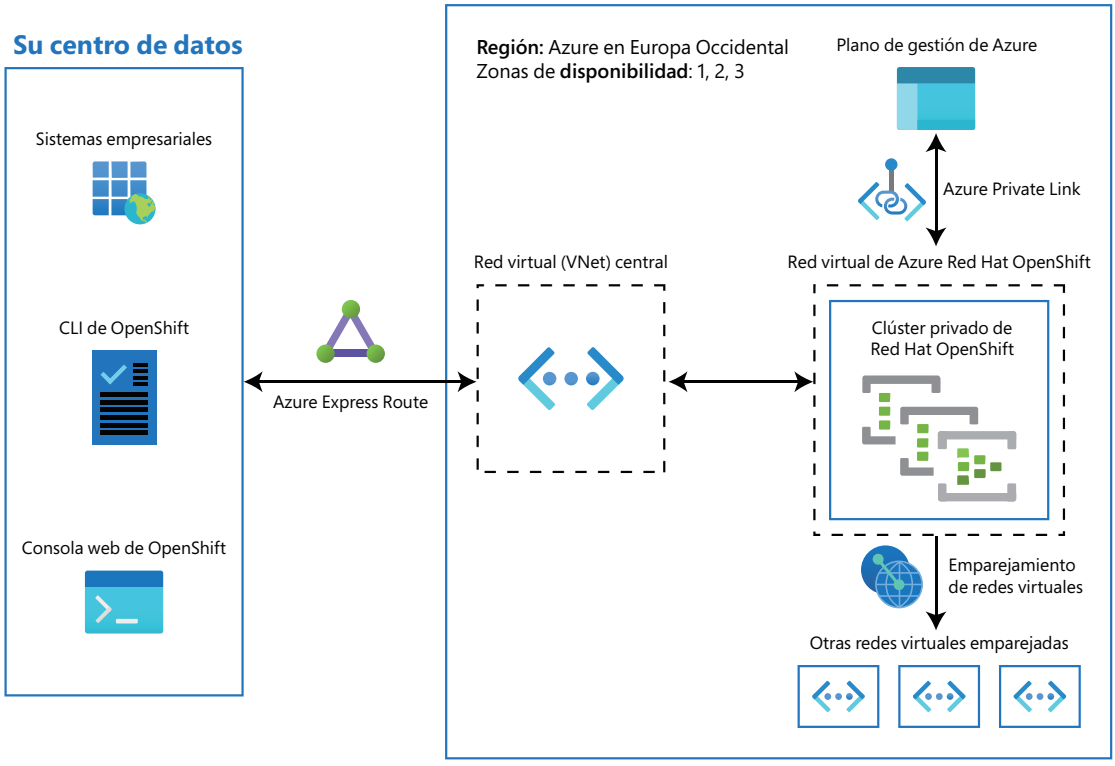


Figura 4.3: conectividad con Azure ExpressRoute

En el diagrama anterior, se muestra a grandes rasgos la arquitectura de Azure ExpressRoute y la forma en que se conecta con Azure RHOS. Si bien en el diagrama solo se muestra la conectividad con Azure ExpressRoute, la que se logra a través de una VPN es muy similar en términos conceptuales.

Aspectos importantes acerca de las aplicaciones que se ejecutan en arquitecturas híbridas

Cuando se ejecutan aplicaciones en Azure RHOS que se reconectan a entornos locales, deben considerarse algunos aspectos:

- **Latencia de la conexión:** mientras que Azure ExpressRoute ofrece una reconexión a los entornos locales con una latencia relativamente baja, las redes VPN que recorren el Internet público ofrecen una latencia significativamente mayor. Con ciertas aplicaciones, tales como los servidores web en los que la conexión sencillamente transmite el tráfico HTTP, es poco probable que esto provoque problemas. Sin embargo, si una aplicación del lado del servidor que se ejecuta en dicho servidor web necesita conectarse a una base de datos local, el rendimiento podría verse afectado negativamente.
- **Costo del tráfico entrante y saliente:** algunos tipos de conexión, a través de Azure o del proveedor de la conexión, poseen costos asociados al tráfico entrante y saliente. Una precaución sensata que le evitará sorpresas indeseables es, primero, medir los costos en un entorno de pruebas y, luego, proyectar el valor que tendrían con una carga pico.
- **Casos de error:** mientras que Azure ExpressRoute ofrece conexiones permanentes y sólidas, las conexiones a través de una VPN suelen sufrir interrupciones y desconexiones con frecuencia. Al ejecutarse a través del Internet público, la latencia y la calidad del servicio de una conexión VPN puede variar durante el transcurso del día. Es importante que ponga a prueba el rendimiento de las aplicaciones, tanto con un enlace híbrido en malas condiciones como en buenas. Debería también hacerse la siguiente pregunta: si la conexión se interrumpiera durante varias horas, ¿la aplicación en Azure RHOS podría continuar ejecutándose con un rendimiento menor?, ¿o su disponibilidad depende por completo de la conexión híbrida?

Es probable que su empresa necesite verificar varios aspectos más, pero los puntos anteriores deberían ser suficientes para comenzar a considerar si la arquitectura híbrida funcionaría bien en su empresa.

Resumen

En este capítulo, abordamos varias de las preguntas previas a la implementación que debería hacerse y sobre las que debería investigar antes de implementar Azure RHOS. En el capítulo siguiente, se ofrecen consejos útiles acerca de los recursos necesarios para implementar el clúster.

Capítulo 5

Implementación de un clúster de Azure Red Hat OpenShift

Resumamos lo que vimos hasta ahora. En este e-book, abordamos los siguientes temas:

- En el *Capítulo 2*, *Introducción a Red Hat OpenShift*, presentamos brevemente el servicio y explicamos las ventajas de elegirlo sobre Kubernetes solo.
- En el *Capítulo 3*, *Azure Red Hat OpenShift*, describimos los aspectos específicos del servicio de nube gestionado Azure Red Hat OpenShift y abordamos conceptos importantes como la arquitectura, la gestión, la autenticación, el soporte y los precios.
- En el *Capítulo 4*, *Etapa previa a la implementación: preguntas sobre la arquitectura empresarial*, abordamos las preguntas frecuentes que las empresas deberían resolver antes de implementar el servicio.

Ya podemos preparar e implementar un clúster. Encontrará las instrucciones oficiales para la implementación en el sitio de documentación. La preparación es parte de la implementación, ya que implica asegurarse de que toda la infraestructura de TI necesaria para la implementación esté lista.

Tutorial: Creación de un clúster de la versión 4 de Red Hat OpenShift en Azure

En este capítulo no versa sobre los comandos individuales que deberá escribir para crear el clúster, ya que estos pueden cambiar con el tiempo, y en la documentación ya se explica ese tema en detalle.

De todas maneras, el capítulo ofrece orientación acerca del proceso y lo que necesita considerar al momento de implementar el clúster.

Implementaciones manuales: expectativas en relación con el tiempo

Algunas empresas necesitan comprender la cantidad de tiempo que demanda la preparación de un clúster. Analicemos esto en detalle.

A grandes rasgos, los elementos que se requieren para realizar implementaciones son los siguientes:

- La línea de comandos az implementada en la estación de trabajo de un administrador del sistema
- Los proveedores de recursos registrados en su suscripción de Azure
- Un secreto de extracción de Red Hat (opcional)
- Un dominio para su clúster (opcional)
- Una red virtual con dos subredes vacías, una para los nodos de plano de control y otra para los de aplicación

En cuanto al tiempo necesario para configurar estos requisitos previos, es probable que un administrador experimentado de Azure y Red Hat OpenShift complete estas tareas en media hora la primera vez y en tan solo 10 minutos cuando las ejecute repetidamente como parte del proceso manual.

Una vez que se ha cumplido con los requisitos previos, el proceso automatizado de implementación suele tardar entre 25 y 40 minutos, en función de la actividad dentro de la región de Azure.

Automatización de la implementación

Como se sabe que el proceso de preparación de Azure RHOS está automatizado, es común que las empresas intenten usar herramientas para automatizar requisitos previos tales como la creación de redes, subredes, principios del servicio y más.

Existen muchas herramientas para automatizar estos pasos. Aquí recomendamos algunas:

- La herramienta de línea de comandos az: cuando se automatiza, usualmente se instala en un contenedor o algo similar como parte del proceso de CI/CD. Las herramientas que comúnmente se utilizan aquí son Jenkins, Azure DevOps y, probablemente, Ansible. Tenga en cuenta que esta herramienta, az, solo necesita implementarse una vez, pero es posible que otros clústeres deban configurar el ID de la suscripción de Azure para reflejar distintas partes de la empresa.
- Los proveedores de recursos registrados en su suscripción de Azure: como en el caso anterior, estos son parte de la configuración de la herramienta de línea de comandos az.
- Un secreto de extracción de Red Hat (opcional): Red Hat posee una API de REST compatible y documentada que permite obtener un secreto de extracción. Puede encontrar información relacionada en este **artículo**.
- Un dominio para su clúster (opcional): esto depende de cómo cree sus registros DNS, pero si utiliza Azure DNS, podría obtener uno a través de Terraform, Ansible u otra herramienta de automatización conocida.
- Una red virtual con dos subredes vacías, una para los nodos del plano de control (maestros) y otra para los nodos de aplicación (de trabajo): por lo general, esto se automatiza a través de herramientas conocidas de automatización de Azure como Terraform, Ansible y otras similares.

Cuando los requisitos previos están automatizados, el tiempo que tardaría con un proceso manual, entre 30 y 10 minutos, puede reducirse a un minuto o dos. Si bien no es posible agilizar la implementación del clúster (que siempre demora entre 25 y 40 minutos), este proceso ya es incluso más rápido que el local.

Además de agilizar el proceso, la automatización de la implementación ofrece otras ventajas. Por ejemplo, los clientes pueden diseñar un proceso sólido y repetible que se registre y audite con facilidad. También es muy común que diseñen elementos de catálogo de autoservicio en sus propios portales, lo que permite que los equipos implementen y remuevan clústeres de Azure RHOS sin necesidad de que el equipo de plataforma de nube los asista.

Acceder al clúster

En esta sección, se brindan referencias sencillas acerca del acceso a su clúster de Azure RHOS luego de su implementación.

A través de la interfaz de usuario web

Si instaló la CLI, puede recuperar la URL de inicio de sesión de su clúster desde un shell de Bash o desde la sesión de Azure Cloud Shell (Bash) en su portal de Azure. Solo debe ejecutar este comando:

```
az aro show -n $CLUSTER_NAME -g $RG_NAME --query "consoleProfile.url" -o tsv
```

El resultado debería ser similar a este: `openshift.xxxxxxxxxxxxxxxxxxx.eastus.aroapp.io`. La URL de inicio de sesión de su clúster será `https://` seguido del valor de `consoleProfile.url`, por ejemplo, `https://openshift.xxxxxxxxxxxxxxxxxxx.e.astus.aroapp.io`.

Abra esta URL en su navegador. Se le pedirá que inicie sesión como usuario de `kubeadmin`. Utilice el nombre de usuario y la contraseña que el instalador le brindó durante el proceso de instalación.

Ahora debería poder visualizar la consola web de Azure Red Hat OpenShift.

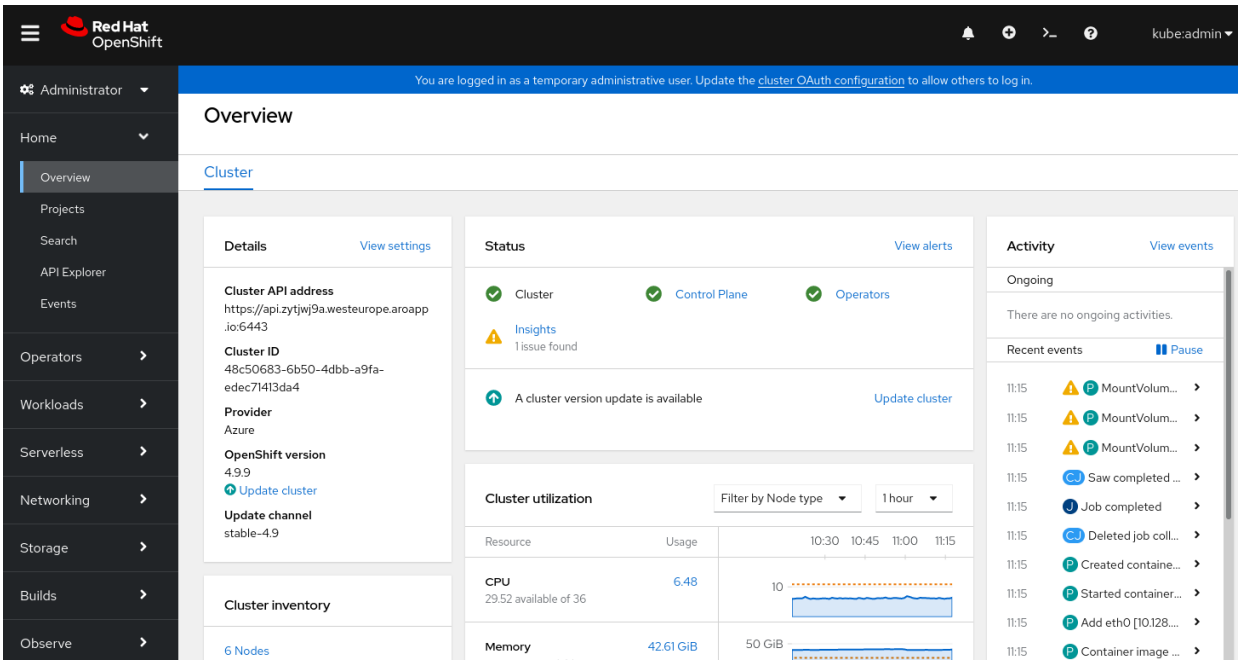


Figura 5.1: consola web de Azure Red Hat OpenShift

Tómese un tiempo para explorar la consola web. Observe si la versión de OpenShift en ejecución es reciente y si todos los elementos se encuentran en buen estado o están prontos a lograrlo luego de la instalación.

A través de la CLI de OpenShift (oc)

Deberá descargar la última versión de la CLI de OpenShift (oc). Para ello, solo debe iniciar sesión para ver la página en https://console.redhat.com/openshift/downloads.

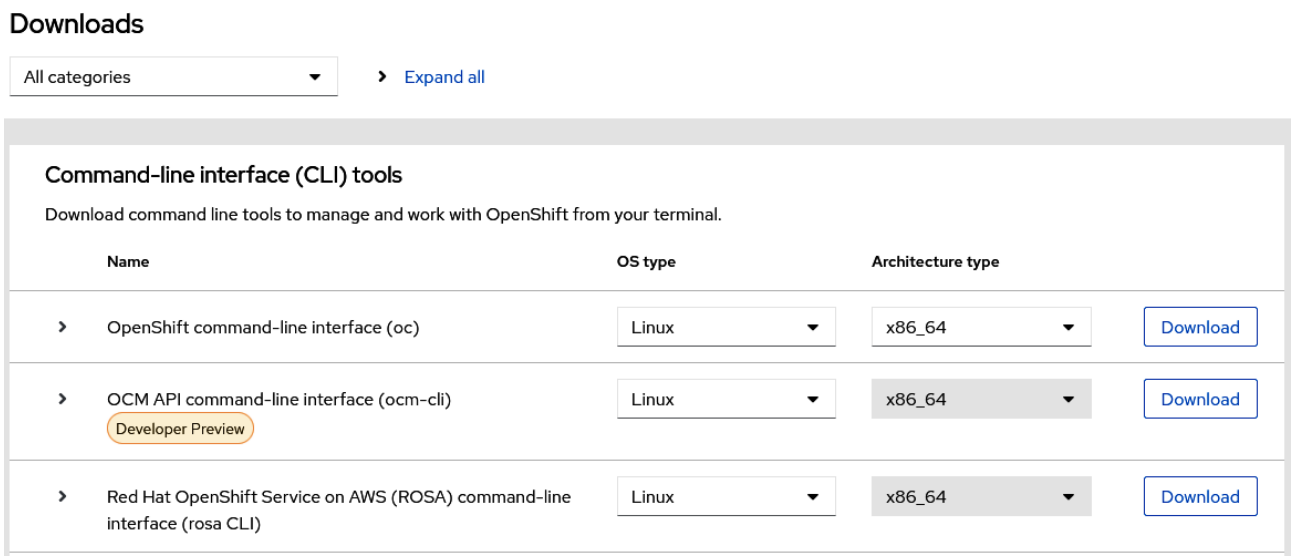


Figura 5.2: descarga de la interfaz de línea de comandos de OpenShift

Extraiga el archivo (`.tar.gz` o `.zip`) y guárdelo en su sistema. Luego, coloque el comando oc en algún lugar de la ruta. En Linux, usualmente se coloca el comando oc en `/usr/local/sbin/`.

Ejecutar la CLI de OpenShift e iniciar sesión en el clúster

Para autenticarse en su clúster desde la línea de comandos, deberá recuperar el comando de inicio de sesión y el token desde la consola web. Inicie sesión en la consola web desde su navegador, haga clic en el nombre de usuario en la esquina superior derecha y, luego, haga clic en **Copy login command** (copiar el comando de inicio de sesión).

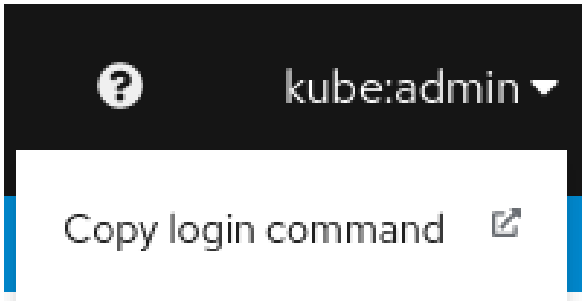


Figura 5.3: copia del comando de inicio de sesión

Se abrirá una nueva página similar a esta:

Figura 5.4: inicio de sesión con el token de la API

Ahora puede copiar este comando y pegarlo en una terminal para iniciar sesión en el clúster de Azure RHOS.

Por ejemplo, si utiliza la sesión de Azure Cloud Shell (Bash) en su portal de Azure, pegue el comando de inicio de sesión. Luego, el comando `oc status` debería verse de una manera similar a esta:

```
user@Azure: oc status
In project default on server https://api.cyki1k6g.westeurope.aroapp.io:6443

svc/openshift - kubernetes.default.svc.cluster.local
svc/kubernetes - 172.30.0.1:443 -> 6443

View details with 'oc describe <resource>/<name>' or list everything with 'oc get all'.
```

Ahora puede tomarse un tiempo para explorar mejor la línea de comandos oc y familiarizarse con su nuevo entorno de Azure RHOS. Si ya tiene experiencia con OpenShift 4, el servicio de nube de Azure RHOS debería resultarle muy familiar y funcionar exactamente igual que otros entornos de OpenShift 4 en los que haya trabajado anteriormente.

Resumen

Este capítulo fue muy breve, ya que las instrucciones oficiales para la implementación se encuentran actualizadas y deben considerarse la guía definitiva para ejecutar Azure RHOS en su suscripción de Azure. Notará que estas instrucciones son mucho más sencillas que las necesarias para implementar un entorno de OpenShift autogestionado. Esto se debe a que se ha dedicado mucho tiempo y esfuerzo a la ingeniería del servicio Azure RHOS, lo que brinda una integración estrecha con Azure y una arquitectura prescriptiva que se adapta a todos los casos, está extensivamente probada y tiene buena compatibilidad. A grandes rasgos, es un beneficio para las empresas, ya que, como se reduce el tiempo de preparación del servicio, puede enfocarse implementar aplicaciones.

Capítulo 6

Etapa posterior a la implementación: el día 2

Una vez implementado Azure RHOS, usualmente se necesitan llevar a cabo ciertas actividades de preparación para que el clúster pueda pasar a la etapa de producción. Este capítulo aborda muchas de las actividades comunes que deben realizarse después de la implementación, como las siguientes:

- Autenticación y Azure Active Directory: los temas incluyen cómo sincronizar los grupos de usuarios con un operador de comunidad
- Operadores: los temas incluyen OperatorHub
- Comprensión de los registros: los temas incluyen el reenvío de registros
- Comprensión de la supervisión: los temas incluyen la supervisión de los contenedores de OpenShift
- Actualizaciones y parches: los temas incluyen el ciclo de vida de la versión con soporte
- Ajuste de la capacidad del clúster: los temas incluyen el ajuste manual y automático de la capacidad del clúster
- Ajuste de la capacidad de las aplicaciones: se ofrece una nota breve acerca de este tema
- Configuración del secreto de extracción: se explica cómo registrarse en OpenShift Cluster Manager
- Rangos de límites: los temas incluyen las situaciones en las que pueden aplicarse
- Almacenamiento permanente: los temas incluyen las clases de almacenamiento disponibles y admitidas
- Seguridad y cumplimiento: se ofrece una nota acerca de los controles de seguridad

Estas secciones, en las que se detallan las tareas posteriores a la implementación, le permitirán comprender mejor lo que se necesita para que un clúster recién implementado esté preparado para las aplicaciones de producción.

Autenticación y Azure Active Directory

Azure RHOS es compatible con todos los proveedores de autenticación que integran la lista de documentación de OpenShift, la cual puede ver aquí. Sin embargo, es probable que la mayoría de los clientes utilice Azure Active Directory, que ya se encuentra disponible en Azure, para proporcionar autenticación e inicio de sesión único a Azure RHOS.

La integración con Azure Active Directory solo se puede configurar el "Día 2", ya que es posible que algunas empresas utilicen otro proveedor de configuración. El proceso de configuración e instalación toma alrededor de 15 minutos la primera vez, hasta que se familiarice con el proceso, cuando podrá llevarlo a cabo en tan solo cinco. Muchas empresas eligen automatizar parte de esta implementación con herramientas tales como plantillas de ARM o Ansible Playbooks.

- Configuración de la autenticación de Azure Active Directory para un clúster de Red Hat OpenShift en Azure (Portal)
- Configuración de la autenticación de Azure Active Directory para un clúster de Red Hat OpenShift en Azure (CLI)

Utilizar grupos de usuarios de Active Directory

Debido a la configuración de la autenticación de Azure Active Directory, solo los usuarios que ya cuenten con credenciales podrán iniciar sesión, pero los grupos con los que cuentan no se trasladarán a Red Hat OpenShift. Es muy común que una empresa desee importar grupos de usuarios desde Active Directory de modo de usarlos como ejemplo para configurar los permisos de usuarios desde OpenShift. No obstante, ya existe un operador con soporte de la comunidad que hace esto: Group Sync Operator.

- Group Sync Operator en GitHub

Tenga en cuenta que Group Sync Operator tiene soporte de la comunidad, es decir que, por el momento, ni Microsoft ni Red Hat ofrecen soporte oficial para este. Recomendamos seguir las instrucciones detalladas para la configuración del operador que se ofrecen en el archivo README del proyecto.

Operadores

Los operadores en OpenShift 4 son elementos fundamentales de diseño de la plataforma y brindan un valor muy alto a los clientes de Red Hat OpenShift. Los operadores se diseñan en código y ejecutan un servicio en un contenedor dentro del clúster. Algunos operadores se encargan de mantener, por ejemplo, las redes del clúster, la configuración de las máquinas y las actualizaciones del clúster. Suelen denominarse operadores del clúster. Puede ver una lista de ellos en la sección **Administration** (gestión) de la consola de OpenShift.

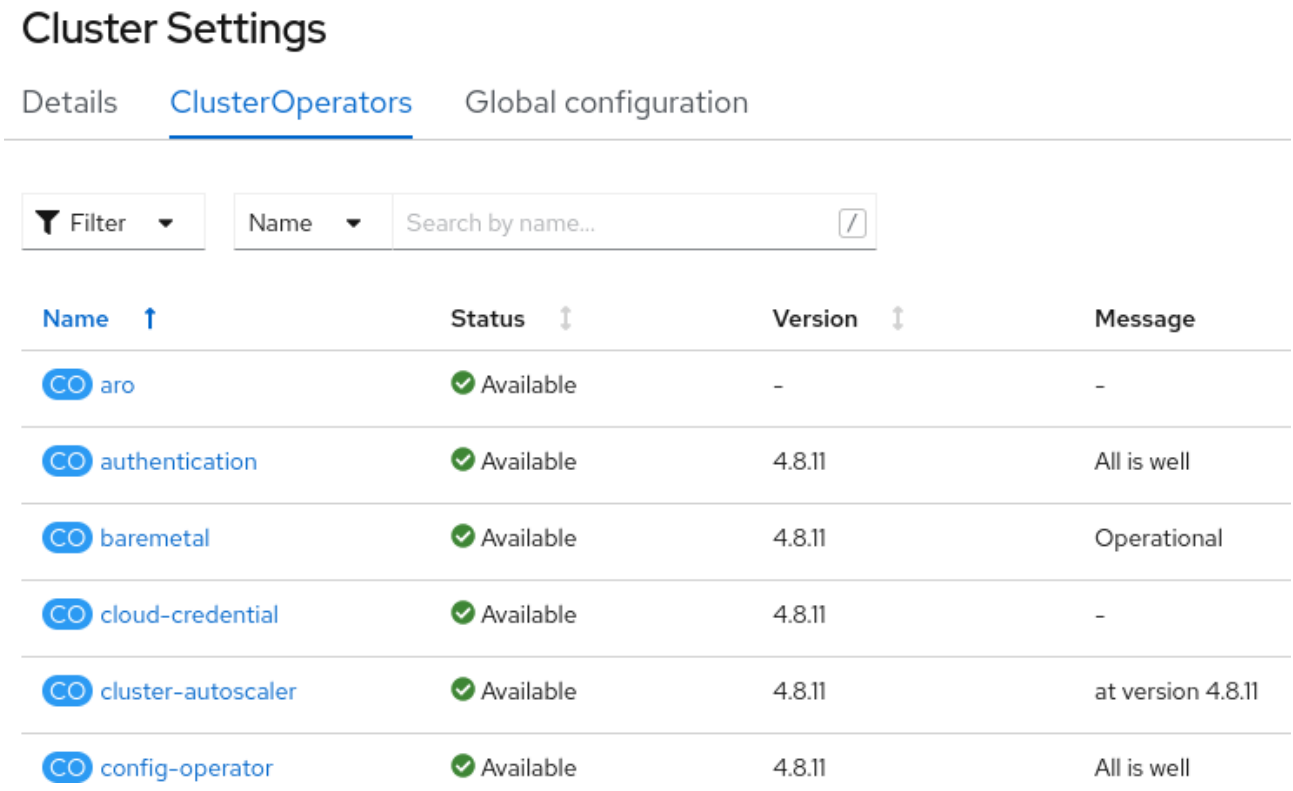


Figura 6.1: configuración del clúster

También puede ver que, en la imagen anterior, hay un operador exclusivo para Azure RHOS, el cual mantiene diversas partes del servicio y garantiza que el clúster esté en un estado de configuración admitido.

Los operadores pueden encargarse del mantenimiento de varias funciones, como el estado del servicio, las actualizaciones, los parches, el ajuste de la capacidad y muchas más.

OperatorHub

Gracias a OperatorHub, los administradores tienen acceso a muchos operadores, no solo a los estándares que se ejecutan en segundo plano en cada clúster. OperatorHub facilita el hallazgo de operadores conocidos, sean empresariales o de la comunidad, de modo que los administradores puedan utilizarlos durante la instalación de Red Hat OpenShift si lo desean. Puede encontrar OperatorHub en la barra lateral de cada clúster de OpenShift.

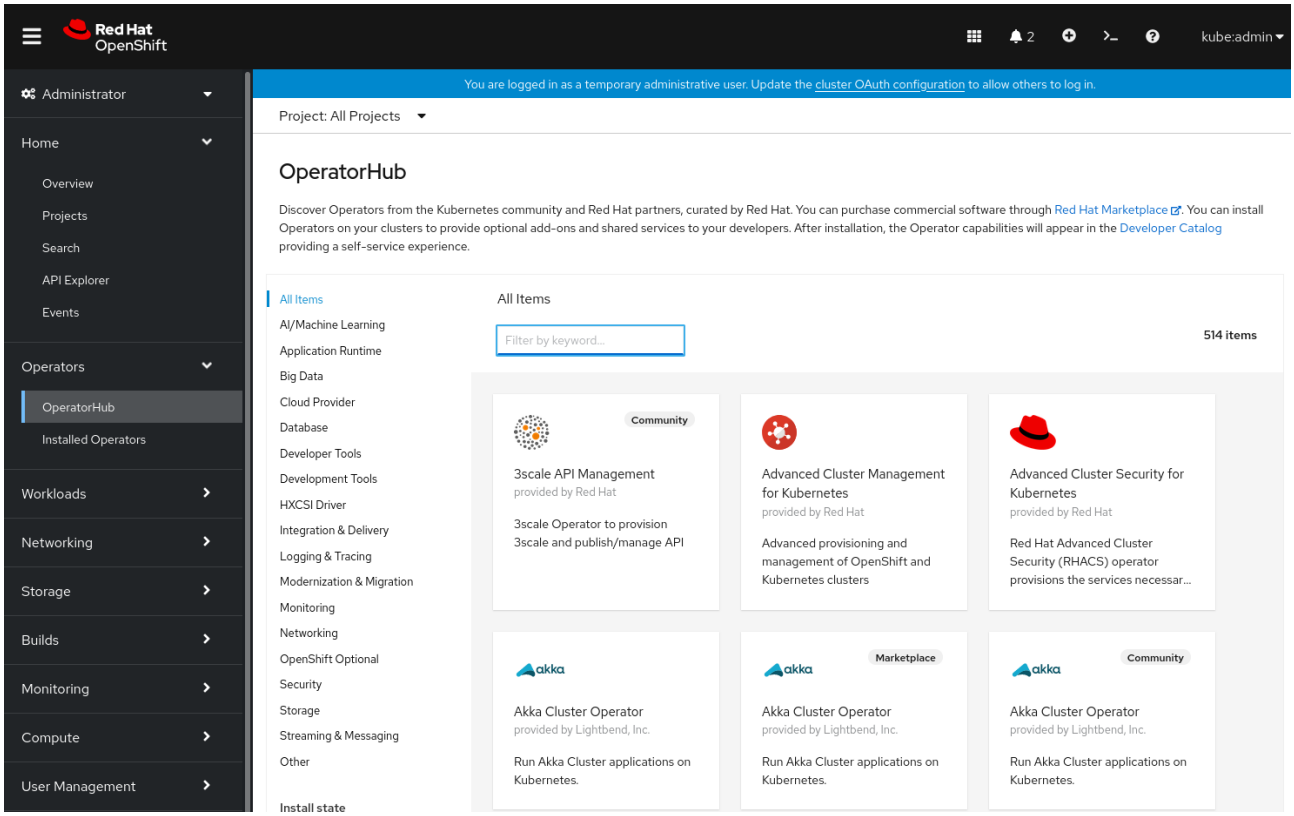


Figura 6.2: OperatorHub

Si los administradores usan los operadores, la implementación y el mantenimiento del software conocido serán tareas sencillas y veloces para ellos, ya que no deberán encargarse de la configuración ni del código de Kubernetes. Actualmente, varios productos de Red Hat se empaquetan como operadores, como Advanced Cluster Management, Red Hat OpenShiftService Mesh y Red Hat OpenShift Pipelines. Sin embargo, también existe una biblioteca inmensa de operadores para software que no proviene de Red Hat, como la base de datos MariaDB, Couchbase o los controladores de almacenamiento en bloques de IBM.

Para obtener más información acerca de los operadores, le recomendamos consultar los siguientes enlaces:

- Operatorhub.io
- Operadores en OpenShift

Comprender los registros

Azure RHOS utiliza la misma arquitectura de registros y supervisión que Red Hat OpenShift. Ambos utilizan los mismos operadores para los registros.

Los registros se dividen en tres categorías:

- **De aplicación**: son los registros de contenedor que generan las aplicaciones que se ejecutan en el clúster, excepto las de contenedores de infraestructura.
- **De infraestructura**: son los registros que generan los elementos de infraestructura que se ejecutan en el clúster y en los nodos de OpenShift Container Platform, como los de los diarios. Los elementos de infraestructura son pods que se ejecutan en proyectos de `openshift*`, `kube*` o los predeterminados.
- **De auditoría**: son los registros que genera auditd, el sistema de auditoría de nodos, que se almacenan en el archivo de registros `/var/log/audit/audit.log`, y los registros de auditoría del servidor de la API de Kubernetes y OpenShift.

Para obtener una descripción más detallada de los registros en OpenShift, consulte Understanding Red Hat OpenShift Logging (El concepto de los registros de Red Hat OpenShift) en la documentación del producto.

Utilizar el reenvío de registros del clúster a Azure Monitor

Una integración común es el envío de registros de Azure RHOS al servicio Container Insights de Azure Monitor. En ocasiones, se denomina Azure Log Analytics. Esto permite conservar los registros de forma permanente y a un bajo costo en Azure.

La arquitectura de registros de OpenShift ejecuta Fluent Bit en cada nodo. En principio, es probable que un administrador desee ajustar la configuración del servicio Fluent Bit para enviar registros de manera directa. Sin embargo, la política de soporte no cubre esta acción. OpenShift cuenta con un mecanismo integrado para reenviar registros sin perder el soporte: `ClusterLogForwarder`.

Mediante el uso de `ClusterLogForwarder`, es posible reenviar registros a Elasticsearch, Fluentd y Syslog. Sin embargo, Azure Monitor no admite estos protocolos de forma directa. Una solución alternativa para esto es disponer de una instancia intermedia adicional de Fluent Bit, la cual reciba registros de OpenShift y los reenvíe a Azure Monitor. En la actualidad, este enfoque se describe en la documentación de Microsoft y en la documentación de la comunidad.

El concepto de la supervisión

Como se trata de un servicio gestionado, no debería ser necesario implementar una supervisión personalizada compleja en Azure RHOS, ya que esta se brinda como parte del servicio que paga.

De todas maneras, es muy común que desee supervisar los contenedores de OpenShift por medio de Azure Container Insights. Esta función se encuentra, por el momento, en versión de prueba disponible para el público y puede encontrar información sobre ella en esta documentación.

Actualizaciones y parches

Cuando implemente un clúster de Azure RHOS, no se seleccionará un canal de actualización. Esto significa que su clúster no comenzará a recibir actualizaciones de manera predeterminada.

Para ver su canal de actualizaciones, diríjase a **Administration** (`administración`)→ **Cluster Settings** (Configuración del clúster) desde la barra lateral de navegación. Aquí podemos ver la configuración del clúster poco después de su implementación:

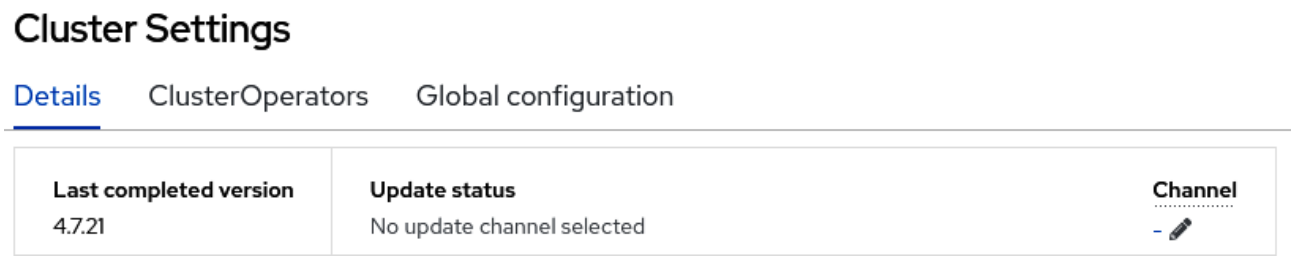


Figura 6.3: configuración del clúster luego de su implementación

Para seleccionar un canal de actualización, seleccione el enlace "-" y vea las opciones disponibles:

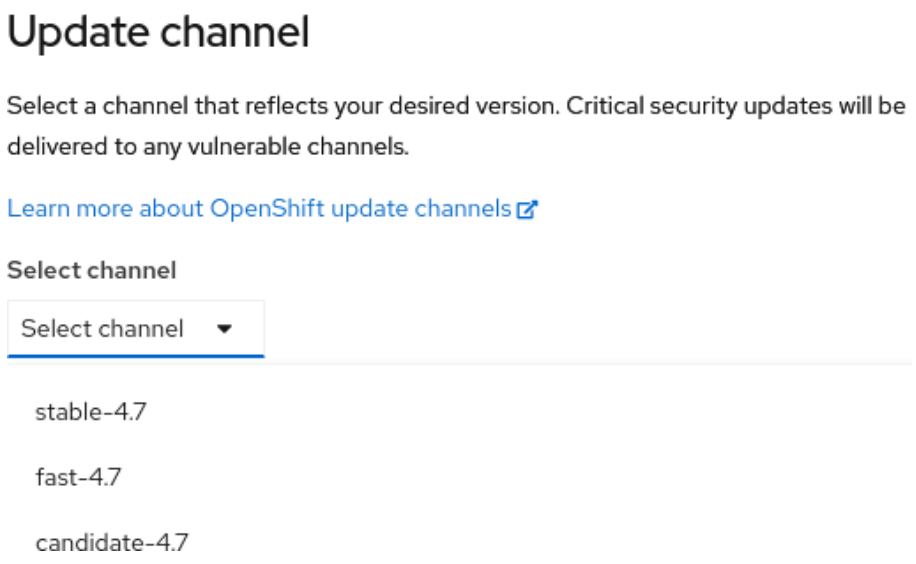


Figura 6.4: selección de un canal de actualización

Para comprender las diferencias entre `stable` (estable), `fast` (rápido) y `candidate` (candidato), vea la página de la documentación acerca de los [canales de actualización y las versiones](#).

Se recomienda que todos los clústeres en la producción se ejecuten en un canal **stable**.

Ciclo de vida de la versión con soporte

Es importante que sepa cuáles son las versiones de Azure RHOS que tienen soporte para planificar la implementación en la producción. La página oficial del ciclo de vida contiene información para que las empresas sepan cuáles versiones cuentan con soporte y cuáles no.

[Página de soporte del ciclo de vida de Azure RHOS](#)

A continuación, se ofrece un extracto simplificado de la información que contiene la página:

Azure RHOS ofrece soporte para dos versiones secundarias **disponibles de modo general (GA)** de Red Hat OpenShift Container Platform:

- La versión secundaria con disponibilidad general de lanzamiento más reciente en Azure RHOS, la cual denominaremos N
- Una versión secundaria anterior, la cual denominaremos N-1

Red Hat OpenShift Container Platform utiliza versionado semántico. En el versionado semántico, los distintos niveles de números de versión indican distintos niveles de versionado. En la siguiente tabla, se ilustran las distintas partes de un número semántico de versión. En este caso, se utiliza como ejemplo el número de versión 4.9.3:

Versión principal (x)	Versión secundaria (y)	Parche (z)
4	9	3

Cada número de la versión indica la compatibilidad general con la versión anterior:

- **Versión principal:** por el momento, no se planifica el lanzamiento de ninguna versión principal. Las versiones principales se actualizan cuando los cambios en la API o las versiones anteriores son incompatibles.
- **Versión secundaria:** estas se lanzan cada tres meses aproximadamente. En las actualizaciones de versiones secundarias, se pueden agregar funciones, mejorarlas, discontinuarlas o eliminarlas, como así también corregir los errores y mejorar la seguridad.
- **Parches:** generalmente, se lanzan cada semana o según se necesiten. En las actualizaciones de parches, se puede incluir la corrección de errores y las mejoras en la seguridad.

Lea la página sobre el ciclo de vida de Azure Red Hat OpenShift para comprender mejor las versiones con soporte.

Ajuste de la capacidad del clúster

Red Hat OpenShift Container Platform y Azure RHOS se diseñaron pensando en una arquitectura cuya capacidad pueda ajustarse. En OpenShift, si los administradores y los propietarios de las aplicaciones planean ajustar la capacidad, deberán considerar a los clústeres y a las aplicaciones de manera individual.

En esta sección, se aborda el ajuste de la capacidad del clúster y, en una sección separada, encontrará una nota breve acerca de la capacidad de ajuste de las aplicaciones.

Cantidad máxima admitida

El proceso de ajustar la capacidad de un clúster consiste en agregarle nodos de trabajo adicionales, lo que, a su vez, ofrece una mayor capacidad informática para las aplicaciones que se ejecutan en él. Desde luego, es normal preguntarnos cuándo será necesario ajustar también la capacidad del plano de control. Azure RHOS ya ofrece tres nodos de plano de control. En principio, estos cuentan con la capacidad necesaria para aumentar los nodos de trabajo a la cantidad máxima admitida en un clúster de Azure RHOS, la cual es de 60 por el momento.

En la actualidad, se admiten tres tamaños de instancias de nodos de plano de control: `Standard_D8s_v3`, `Standard_D16s_v3` y `Standard_D16s_v3`.

En cambio, para los nodos de trabajo, se admiten más: Compute-optimized (F series), Memory-optimized (E series) y General-Purpose (D series).

Puede encontrar una lista de los tipos de instancias de Azure que se admiten en la actualidad en la **página acerca de la política de soporte**.

Implementación mínima y ajuste de la capacidad a cero

En ocasiones, es posible que las empresas quieran implementar clústeres más pequeños, ya sea porque desean realizar pruebas y desarrollos o porque los casos de uso no requieren una disponibilidad muy amplia. En la actualidad, un clúster admite, como mínimo, tres nodos de plano de control y tres de aplicación, sin excepciones.

Ajuste manual de la capacidad del clúster

Si consultan el portal de Azure, los operadores podrían sentirse tentados a crear una máquina virtual para agregar más nodos de trabajo al clúster de Azure RHOS. Sin embargo, esto no es posible, ya que el grupo de recursos que contiene Azure RHOS está, por decirlo de algún modo, bloqueado para los administradores de Azure. Esto sucede independientemente de los permisos que posean: ni siquiera los usuarios con permiso de **propietario** pueden modificar el contenido de un grupo de recursos de Azure RHOS. Por este motivo, recibirán mensajes de error de permiso denegado si intentan crear máquinas virtuales de forma manual dentro del grupo de recursos de Azure RHOS.

El proceso de ajuste de la capacidad de Azure RHOS está pensado para funcionar con la característica `MachineSets` de OpenShift, mediante la cual se implementará un nodo de aplicación nuevo cuando se lo indiquen. Los usuarios con privilegio de cluster-admin encontrarán **MachineSets** en **Compute**.

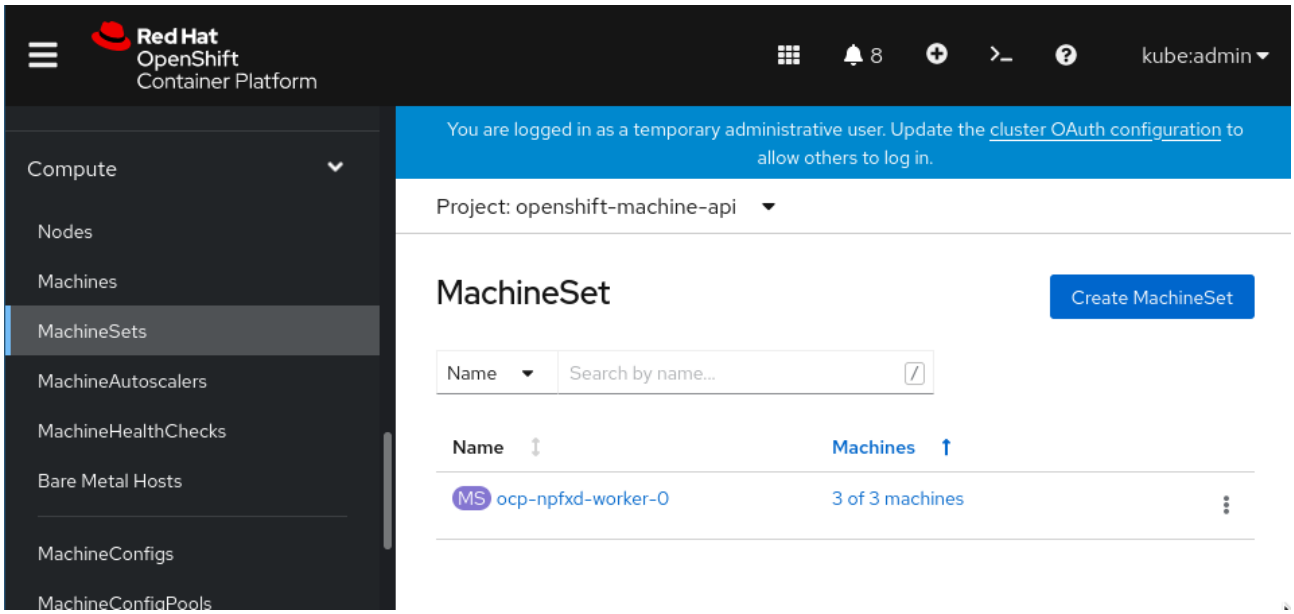


Figura 6.5: acceso de administrador a MachineSets

Si hace clic en el menú "edit" (editar) de un nodo de aplicación de `MachineSet`, verá que puede implementar una máquina virtual nueva para el nodo de aplicación con tan solo seleccionar **Edit Machine Count** (editar cantidad de máquinas).

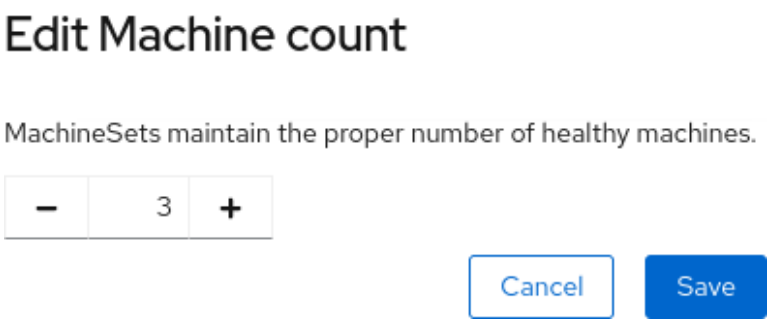


Figura 6.6: edición de la cantidad de máquinas

Una vez que actualizó la cantidad de máquinas, el administrador puede dirigirse al portal de Azure o a la vista **Compute** (informática) → **Machines** (máquinas) para observar la implementación de la máquina virtual nueva para el nodo de aplicación.

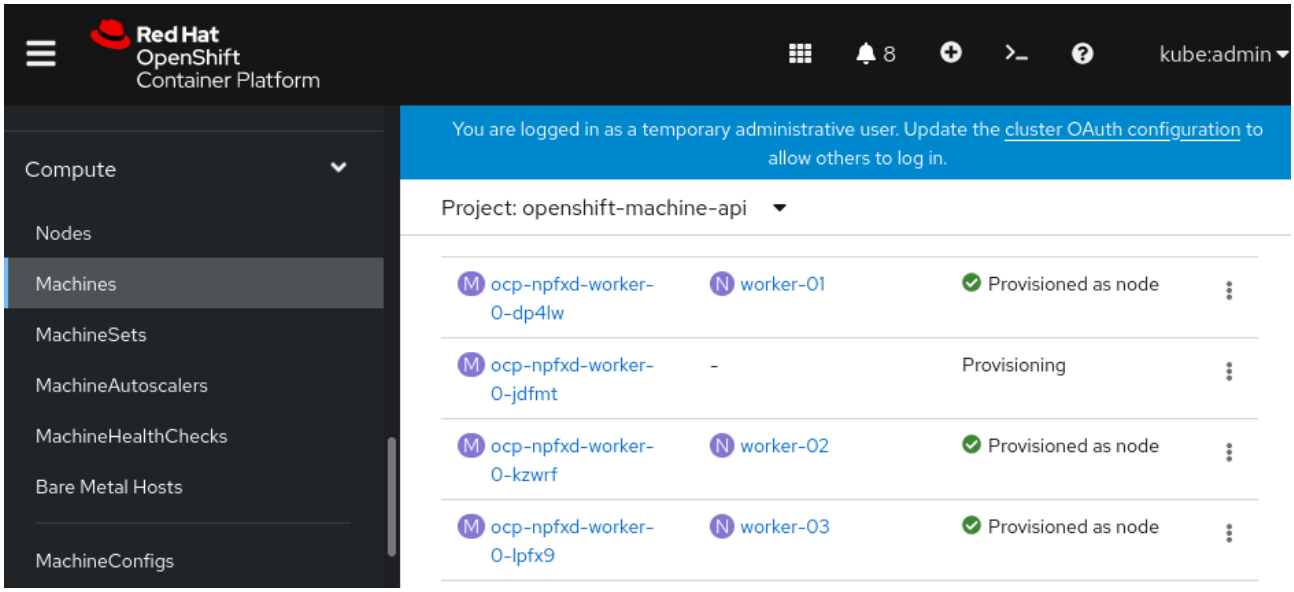


Figura 6.7: la vista de máquinas

En la mayoría de las regiones de Azure, implementar una nueva máquina virtual suele tomar alrededor de cinco minutos.

No es necesario que los administradores realicen ninguna tarea luego de la implementación para que la nueva capacidad esté en línea. OpenShift la pondrá a disponibilidad del clúster automáticamente y las aplicaciones podrán utilizarla cuando la necesiten.

Ajuste automático del clúster

El ajuste automático no está activado de manera predeterminada en Azure RHOS, pero activarlo es sencillo. Solo debe crear un recurso de `MachineAutoscaler`, el cual puede encontrar en el menú de la barra lateral **Compute**.

Los recursos de `MachineAutoscaler` se ejecutan en `MachineSet`, el cual, a su vez, crea o elimina máquinas virtuales para el nodo de aplicación para ajustar la capacidad según sea necesario. En el siguiente ejemplo, se muestra que es posible mantener una cantidad mínima o máxima de máquinas en un `MachineSet`:

```
apiVersion: autoscaling.openshift.io/v1beta1
kind: MachineAutoscaler
metadata:
    name: worker-us-east-1a
    namespace: openshift-machine-api
spec:
    minReplicas: 1
    maxReplicas: 12
    scaleTargetRef:
        apiVersion: machine.openshift.io/v1beta1
        kind: MachineSet
        name: worker
```

OpenShift también posee funciones para ajustar la capacidad de los clústeres según parámetros tales como la cantidad de RAM o CPU disponible. La elección de `MachineAutoscaler` o `ClusterAutoscaler` depende de la manera en que desee agregar o restar capacidad en su clúster.

Documentación de Red Hat OpenShift acerca de MachineAutoscalers y ClusterAutoscalers

Ajuste de la capacidad de las aplicaciones

Ajustar la capacidad de las aplicaciones de microservicios es un tema complejo que sobrepasa el alcance de este e-book. Sin embargo, a continuación, encontrará algunas páginas útiles para comenzar a explorar el tema.

- HorizontalPodAutoscaler: especifique la cantidad mínima y máxima de pods que quiere ejecutar y el uso de CPU o memoria al que estos deberían apuntar.
- Informática sin servidor y ajuste a cero con Knative.

El clúster supervisará los contenedores que se ejecuten y la capacidad restante. En el caso de que Knative o `HorizontalPodAutoscaler` soliciten más recursos de los que estén libres en el clúster, deberá utilizar `ClusterAutoscaler` o `MachineSet` para ajustar la capacidad y poder agregar al clúster los recursos necesarios.

Con este enfoque, las aplicaciones y el propio clúster podrán ajustarse de forma conjunta y aumentar o disminuir la capacidad en función de la demanda.

Configuración de un secreto de extracción (registrarse en OpenShift Cluster Manager)

Una implementación reciente de Azure RHOS no posee un secreto de extracción configurado para `cloud.redhat.com`. Esto implica que sus clústeres no aparecerán de manera predeterminada en la consola de la nube híbrida de Red Hat (http://console.redhat.com): esto se llama OpenShift Cluster Manager.

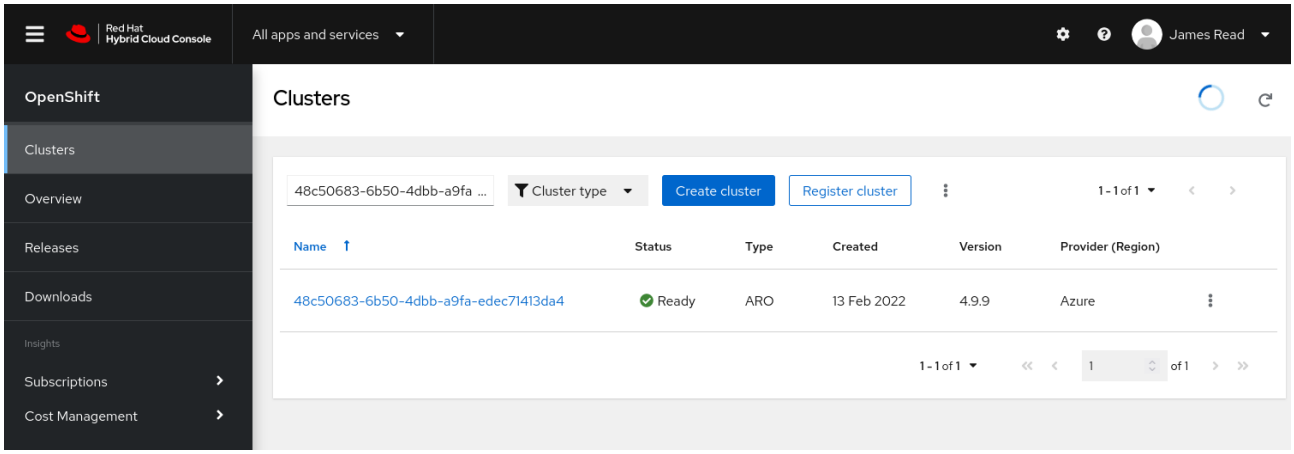


Figura 6.8: un clúster de Azure RHOS en OpenShift Cluster Manager

La configuración de un secreto de extracción para `cloud.redhat.com` es bastante sencilla y, además de mostrar sus clústeres en el portal de OpenShift Cluster Manager en http://console.redhat.com, le permitirá enviar solicitudes de soporte a Red Hat de manera directa a través del sistema estándar de solicitudes de soporte.

Puede encontrar instrucciones para configurar un secreto de extracción aquí:

- Agregar o actualizar un secreto de extracción

Otro de los beneficios de configurar un secreto de extracción es que los clientes podrán solicitar soporte a Red Hat directamente. El secreto de extracción otorga un derecho a su cuenta de Red Hat que permite que el equipo de soporte pueda ver el clúster. El proceso de solicitud de soporte, en ese caso, es igual que para cualquier otro producto de Red Hat.

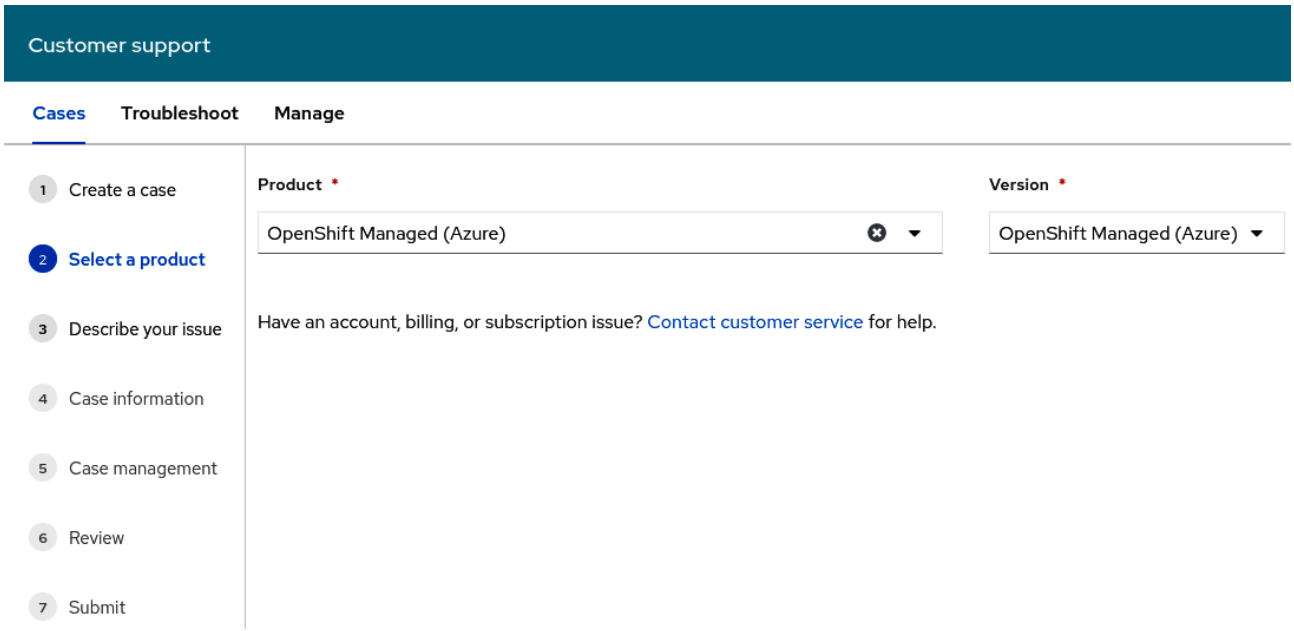


Figura 6.9: creación de un caso de soporte

Rangos de límites

Si un equipo de implementación o de aplicaciones implementa aplicaciones organizadas en contenedores, solo bastará que pase cierto tiempo para que una aplicación exhiba un comportamiento extraño y comience a consumir recursos del clúster innecesariamente. Un ejemplo de esto puede ser una aplicación defectuosa con fugas de memoria, o bien una que se haya configurado para ajustarse luego de consumir solo el 10 % de CPU en lugar del 100 %. Para evitar situaciones en las que las aplicaciones se salgan de control y consuman demasiados recursos, se recomienda utilizar `LimitRange`.

`LimitRange` permite que restrinja el consumo de recursos para ciertos objetos del proyecto. `LimitRange` puede aplicarse en lo siguiente:

- Pods y contenedores: puede configurar los requisitos de uso mínimo y máximo de CPU y memoria para los pods y sus contenedores.
- Flujos de imágenes: puede establecer límites para la cantidad de imágenes y etiquetas en un objeto de `ImageStream`.
- Imágenes: puede limitar el tamaño de las imágenes que se envían a un registro interno.
- **Reclamaciones de volumen permanente (PVC)**: puede restringir el tamaño de las PVC que pueden realizarse.

A continuación, se muestra un ejemplo de un rango de límites que se aplica a contenedores y que restringe la cantidad mínima, máxima y predeterminada de solicitudes para la CPU y la memoria:

```
apiVersion: "v1"
kind: "LimitRange"
metadata:
  name: "resource-limits"
spec:
  limits:
    - type: "Container"
      max:
        cpu: "2"
        memory: "1Gi"
      min:
        cpu: "100m"
        memory: "4Mi"
      default:
        cpu: "300m"
        memory: "200Mi"
      defaultRequest:
        cpu: "200m"
        memory: "100Mi"
      maxLimitRequestRatio:
        cpu: "10"
```

Este fragmento de código de `LimitRange` puede aplicarse copiando y pegando el YAML directamente en el editor de la consola de RHOS, o bien desde un archivo con `oc apply -f limitrange.yaml`.

Documentación acerca de rangos de límites

Almacenamiento permanente

Las máquinas virtuales que se implementan con Azure RHOS vienen con discos conectados para instalar Red Hat CoreOS y ejecutar el servicio. Los discos son exclusivamente para uso del clúster de OpenShift, no de las aplicaciones.

Las aplicaciones que requieren almacenamiento permanente deberían utilizar la característica `PersistentVolume` de Kubernetes. Hay una página excelente en la documentación de OpenShift acerca de este tema en la que se describe en detalle este concepto. Si bien Azure RHOS, como instalación autogestionada de OpenShift, teóricamente es compatible con todos los proveedores de `PersistentVolume`, en Azure los más utilizados son los siguientes:

Nombre	Tipo	Modos de acceso	Clase de almacenamiento
Archivos de Azure	Sistema de archivos, no compatible con POSIX	ReadWriteOnce	Archivo de Azure
Disco de Azure	Bloque	ReadWriteOnce	Disco de Azure
Red Hat OpenShift Data Foundation	Sistema de archivos, bloque, objeto	(Varios)	Documentos OCS/ODF

Seguridad y cumplimiento

En la actualidad, la documentación de RHOS incluye una cantidad considerable de contenido relevante para comprender la manera en que se implementan varios controles de seguridad y se mantiene el cumplimiento.

Documentación acerca de la seguridad y el cumplimiento de OpenShift

En esta sección de la documentación, se incluye:

- Una descripción detallada del funcionamiento de la seguridad de los contenedores de OpenShift. OpenShift ofrece varios controles de seguridad listos para usar que no se encuentran en otros servicios basados en Kubernetes y que protegen a su empresa y a sus aplicaciones.
- Análisis de los problemas de seguridad en los pods.
- Acceso a los registros de auditoría.
- Configuración de los certificados.

También incluye varios operadores útiles para la seguridad, por ejemplo:

- **Compliance Operator:** permite que se ejecuten análisis y proporciona formas de corregir los problemas comunes de seguridad.
- **File Integrity Operator:** verifica continuamente que los archivos permanezcan intactos, especialmente aquellos confidenciales relacionados a la configuración de seguridad.

Resumen

En este capítulo, abordamos la mayoría de las tareas y los temas que las empresas suelen considerar al momento de acondicionar un clúster recientemente implementado para las aplicaciones de producción. Aunque no sea necesario que revise cada uno de los temas en cada implementación subsiguiente, cuando implemente su primer clúster, debe tener en cuenta el almacenamiento permanente, los límites, la autenticación y todos los demás temas que se mencionan en el capítulo.

Por lo general, las empresas querrán automatizar estas tareas posteriores a la implementación tanto como sea posible. Por ejemplo, Azure Active Directory puede configurarse casi en su totalidad si utiliza un script de PowerShell o algunas plantillas de ARM, lo que reducirá el tiempo que consumen las tareas repetitivas cuando se implementan varios clústeres.

En el capítulo siguiente, avanzaremos hacia la implementación de una aplicación de ejemplo basada en su clúster de Azure RHOS, que ya estará listo para la producción.

Capítulo 7

Implementación de una aplicación de ejemplo

Uno de los objetivos principales de esta guía es ayudar a que sus destinatarios, es decir, quienes ejercen funciones de desarrollo y operación, comprendan lo que necesitan para pasar a la etapa de producción con Azure Red Hat OpenShift. Se da por sentado que el lector dispone de conocimientos básicos acerca de RHOS, ya que buena parte de su arquitectura, los portales de gestión y la experiencia del usuario son idénticos a los de Azure RHOS.

Este capítulo es un pantallazo general en el que se explica la implementación de una aplicación sencilla de ejemplo llamada "Fruit Smoothies" (licuados frutales). Con la aplicación, pensada como guía de inicio rápido y recordatorio de lo aprendido, debería poder comprender mejor la forma en que se utiliza Azure RHOS. Tenga en cuenta que esta aplicación de ejemplo para Azure Kubernetes Service se implementa con base en Azure RHOS para probar que OpenShift es completamente compatible con Kubernetes.

El contenido de este capítulo se basa ampliamente en http://aroworkshop.io con el agregado de descripciones y contexto adicionales.

Descripción general de la aplicación de calificaciones

La arquitectura de la aplicación es sencilla:

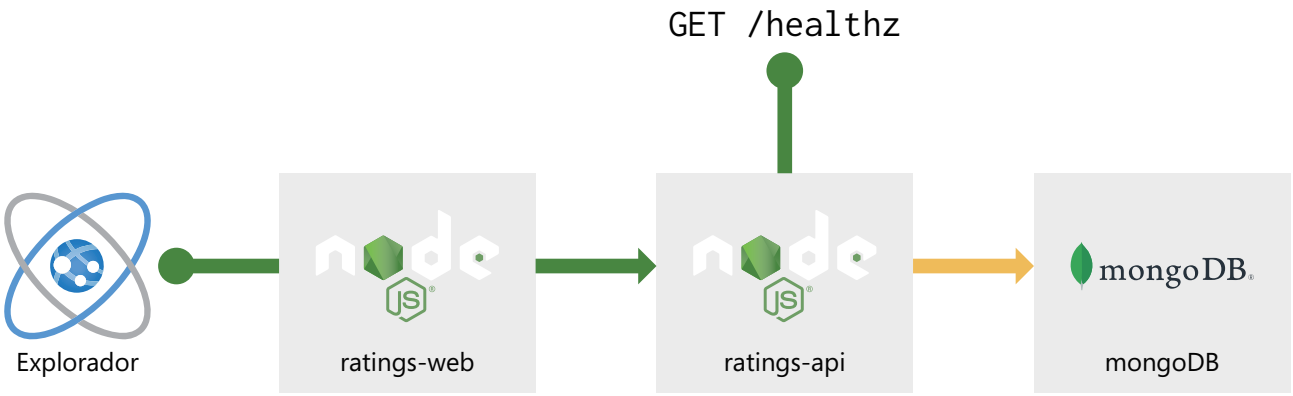


Figura 7.1: la arquitectura de la aplicación Fruit Smoothies

En el diagrama anterior, puede ver que la aplicación consta de tres servicios y dos extremos públicos que se muestran en la siguiente tabla. El primer extremo, que está a la izquierda del diagrama, representa el navegador web de un usuario que visualiza la aplicación web pública HTML. El segundo extremo, `healthz`, es un comprobador de estado en el servicio `ratings-api`. En la tabla siguiente, encontrará descripciones de cada uno de los servicios y enlaces a sus repositorios.

Elemento	Descripción	Repositorio de GitHub
`rating-web`	Frontend de la web dirigida al público: el sitio web.	Repositorio de GitHub
`rating-api`	Este servicio recibe ingresos de datos de la interfaz de usuario web y los almacena en la base de datos. También ofrece resultados de la base de datos y los envía a la aplicación web a través del puerto 3000.	Repositorio de GitHub
`mongodb`	Una base de datos NoSQL con datos cargados previamente.	Datos

Puede consultar los enlaces a los repositorios de GitHub para conocer las aplicaciones y comprenderlas en mayor profundidad. Con las instrucciones que le brindaremos a continuación, lo guiaremos paso a paso en la implementación de dichas aplicaciones.

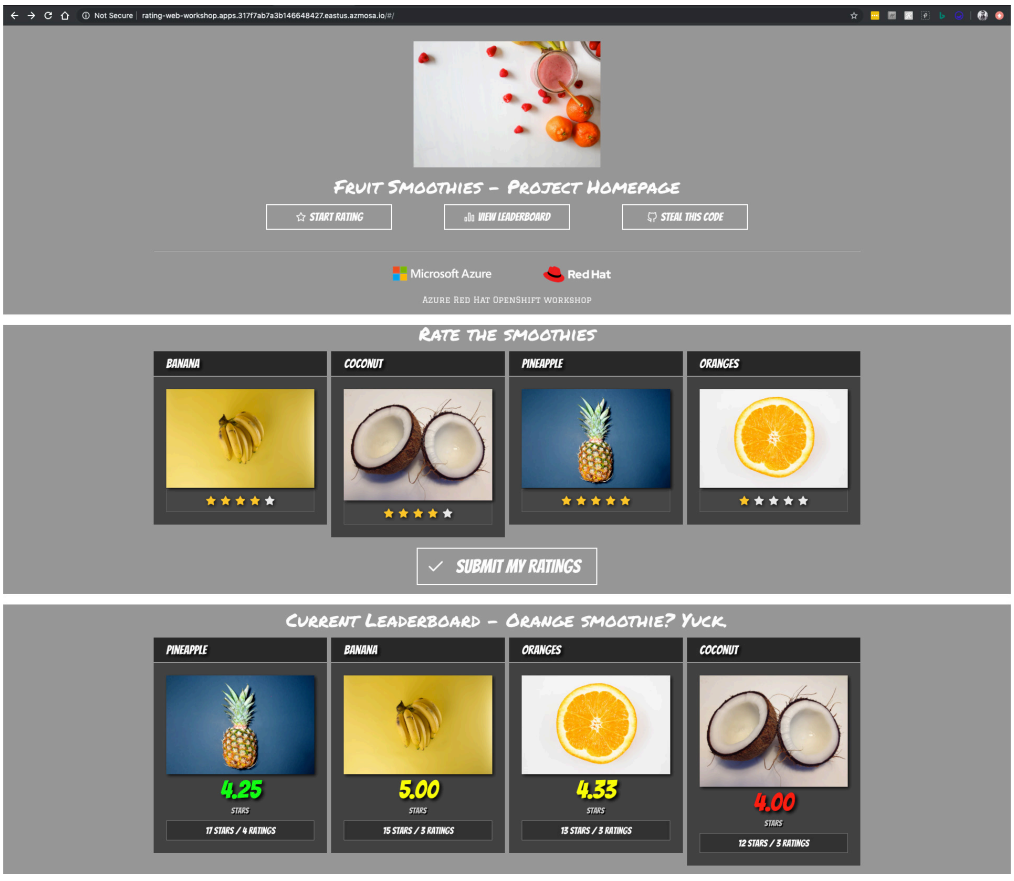


Figura 7.2: capturas de pantalla de la aplicación Fruit Smoothies

Una vez que haya terminado, tendrá una aplicación web en marcha que luce similar a las capturas previas. También entenderá mejor la manera en que los equipos de desarrollo y operaciones implementan aplicaciones en Azure Red Hat OpenShift, lo cual debería ayudar en el proceso de toma de decisiones cuando realice esta tarea por su cuenta.

Creacióndel clúster y conexión a él

En el resto de este capítulo, se da por sentado que usted tiene un entorno de Azure Red Hat OpenShift en ejecución con el cual trabajar. No hay requisitos especiales para esta aplicación de calificaciones, y se implementará en un entorno nuevo de esta plataforma. Si aún no tiene experiencia preparando un clúster, revise los capítulos siguientes:

- *Capítulo 4, Etapa previa a la implementación: preguntas sobre la arquitectura empresarial*
- *Capítulo 5, Implementación de un clúster de Azure Red Hat OpenShift*

La sección *Acceso al clúster* en el *Capítulo 5, Implementación de un clúster de Azure Red Hat OpenShift*, es un recordatorio útil sobre la manera de acceder a un clúster que quizás haya implementado antes.

Iniciode sesión en la consola web

Cada clúster de Azure Red Hat OpenShift tiene una dirección DNS para la consola web. Puede usar el comando `az aro list` para enumerar los clústeres de su suscripción de Azure actual:

```
az aro list -o table
```

Se mencionará la URL de la consola web del clúster. Abra el enlace en una nueva pestaña del explorador e inicie sesión con el usuario `kubeadmin` o con otra cuenta que tenga permiso para crear proyectos.

Ahora debería poder visualizar la consola web de Azure Red Hat OpenShift.

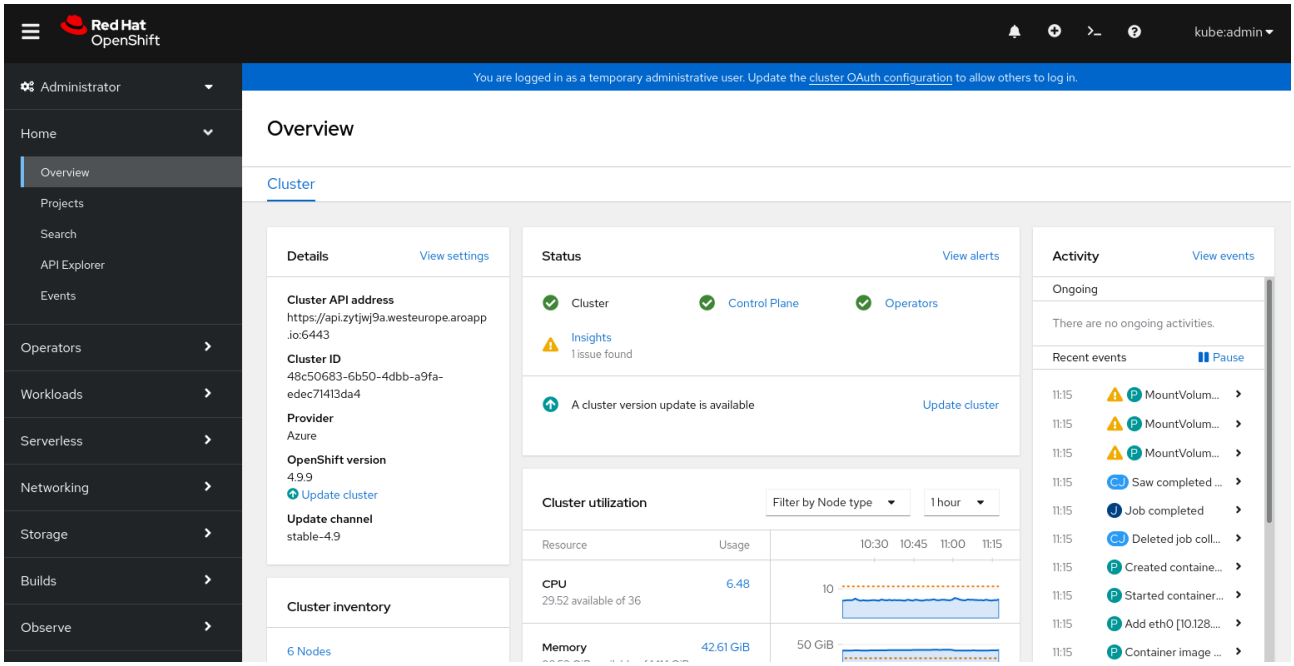


Figura 7.3: consola web de Azure Red Hat OpenShift

Instaleel cliente de OpenShift

Abra Azure Cloud Shell o use su terminal de Linux local e instale el cliente de la línea de comandos. Esta acción es necesaria para acceder al clúster desde la línea de comandos. Las instrucciones para hacerlo son las siguientes:

```
cd ~
curl https://mirror.openshift.com/pub/openshift-v4/clients/ocp/latest/openshift-client-linux.tar.gz > openshift-client-linux.tar.gz

mkdir openshift

tar -zxvf openshift-client-linux.tar.gz -C openshift

echo 'export PATH=$PATH:~/openshift' >> ~/.bashrc && source ~/.bashrc
```

O bien, para Windows o Mac, los enlaces de descarga son los siguientes:

- https://mirror.openshift.com/pub/openshift-v4/clients/ocp/latest/openshift-client-windows.zip
- https://mirror.openshift.com/pub/openshift-v4/clients/ocp/latest/openshift-client-mac.tar.gz

Debería poder ejecutar el comando oc desde todos esos paquetes descargados.

Recuperacióndel comando y el token de inicio de sesión

Cuando haya terminado de instalar el cliente, debe obtener un token para acceder al clúster. Inicie sesión en la consola web de OpenShift, haga clic en el nombre de usuario arriba a la derecha y luego en **Copy Login Command** (Copiar comando de inicio de sesión).

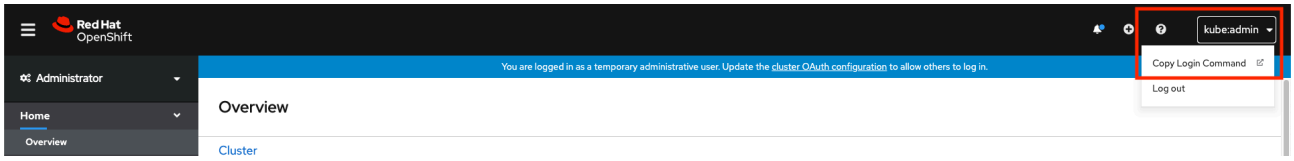


Figura 7.4: copia del comando de inicio de sesión para acceder al clúster

Pegue el comando de inicio de sesión en su shell (terminal de Linux local o Azure Cloud Shell). Debería poder conectarse al clúster.

Figura 7.5: uso del comando de inicio de sesión para conectarse al clúster

Una vez que se haya conectado, podemos crear el proyecto.

Creación de un proyecto

En Red Hat OpenShift, un proyecto es como una carpeta lógica donde deben estar los contenedores y las aplicaciones, como las de calificaciones. Puede usarlos para separar estos elementos o incluso los departamentos. Con las siguientes instrucciones, aprenderá a crear uno.

Si bien lo puede crear desde la interfaz web, en este caso utilizamos la línea de comandos.

```
oc new-project workshop
```

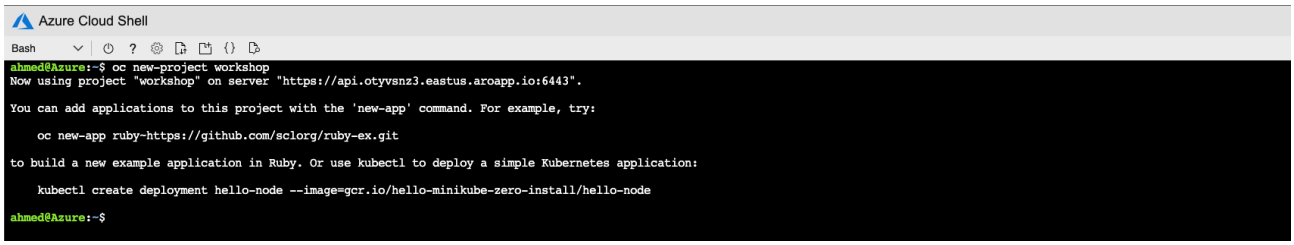


Figura 7.6: creación de un taller en Azure Cloud Shell

Una vez que se haya creado el proyecto, puede trasladarse a este mediante `oc project workshop`. El paso siguiente será implementar en el proyecto que acabamos de crear el primero de los tres microservicios que se presentaron al principio de este capítulo: MongoDB.

Recursos

- Documentación de Azure Red Hat OpenShift – Getting started with the CLI
- Documentación de Azure Red Hat OpenShift – Projects

Implementación de MongoDB

Azure Red Hat OpenShift proporciona una plantilla y una imagen en contenedores para facilitar la creación de un servicio nuevo de base de datos de MongoDB. La plantilla ofrece campos de parámetros para definir todas las variables obligatorias del entorno (usuario, contraseña, nombre de base de datos, etc.) con valores predeterminados, entre los que se incluye la generación automática de contraseñas. También definirá una configuración de implementación y un servicio.

La instancia de MongoDB se implementará directamente como una imagen de contenedores desde Docker Hub. OpenShift le permite hacer todo esto con la consola web. Cambie a la perspectiva del desarrollador desde la parte superior del menú, ingrese a la página **Add** (Agregar) y seleccione **Container images** (Imágenes en contenedores).

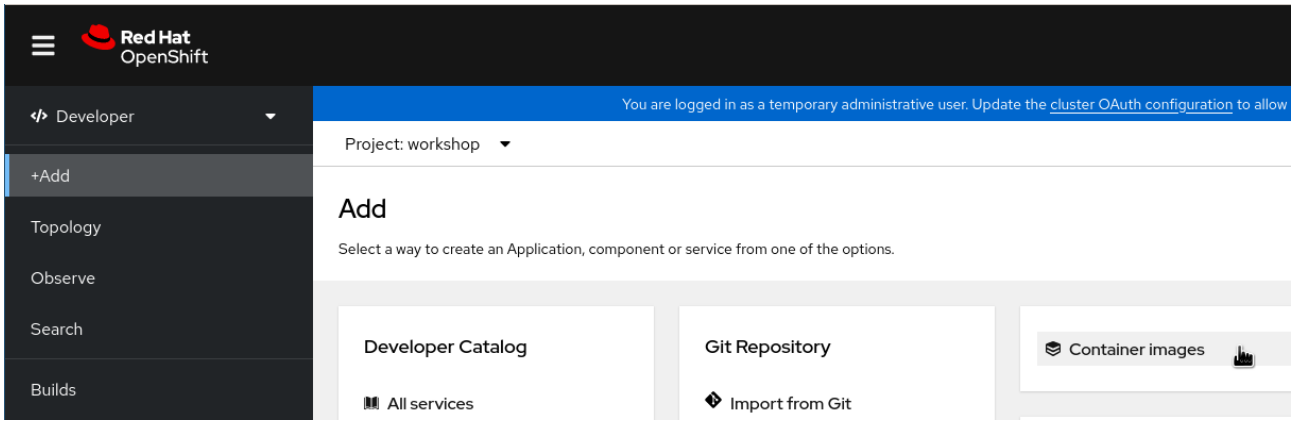


Figura 7.7: adición de una imagen en contenedores

Esto lo dirigirá a la página **Deploy Image** (Implementar imagen). Llene el formulario de esta manera:

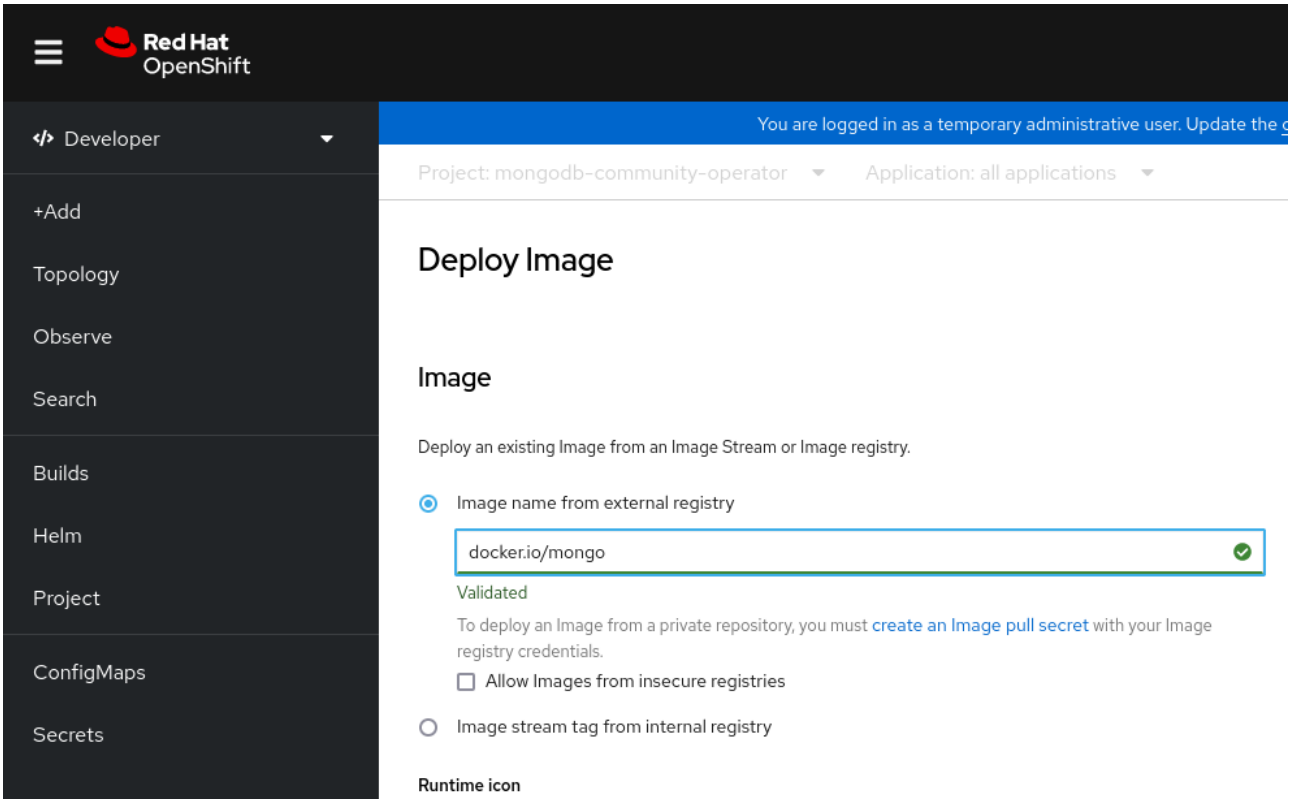


Figura 7.8: compleción del formulario para implementar una imagen

Asegúrese de establecer los valores de la siguiente forma:

Campo	Valor
Nombre de la imagen del registro externo	`docker.io/mongo`
Ícono del tiempo de ejecución	`mongodb`
Nombre de la aplicación	`ratings`
Nombre	`mongodb`
Crear la ruta a la aplicación	Sin marcar. Una ruta brindaría acceso externo a la base de datos, lo cual no se requiere para esta aplicación.
Tipo de recurso	Implementación

Cuando llegue al final del formulario, seleccione el enlace Deployment (Implementación) para expandirlo, de modo que pueda establecer las variables del entorno.

La siguiente variable actúa como un valor predeterminado para iniciar la base de datos por primera vez:

Nombre	Valor
MONGO_INITDB_DATABASE	ratingsdb

No se establece nombre de usuario ni contraseña para la base de datos de MongoDB. Esta es la configuración predeterminada y se desactivará la autenticación.

Controle la variable del entorno una vez más y luego haga clic en el botón **Create** (Crear) para continuar e implementar el contenedor de MongoDB.

Después de un momento, la instancia debería funcionar en el espacio de trabajo del proyecto. Puede ver esta implementación si cambia a la vista **Topology** (Topología).

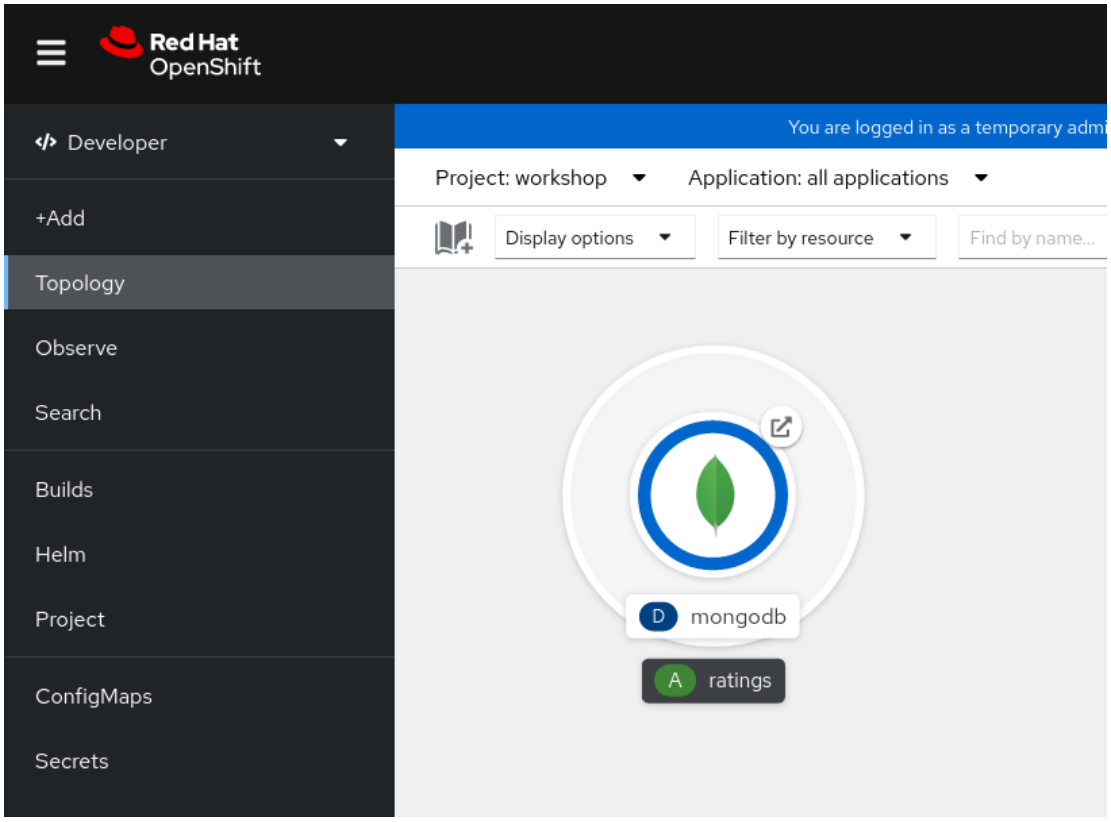


Figura 7.9: la instancia de MongoDB en ejecución

Ejecute el comando oc get all para ver el estado de la nueva aplicación y comprobar si la plantilla de MongoDB se implementó correctamente. El resultado de ejemplo de oc get all es el siguiente:

```
user@host: oc get all
NAME                            READY   STATUS    RESTARTS   AGE
pod/mongo-6c6fcb45b8-8wpdm      1/1     Running   0          29s

NAME              TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)     AGE
service/mongo     ClusterIP   172.30.88.119    <none>        27017/TCP   30s

NAME                      READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/mongo     1/1     1            1           30s

NAME                                 DESIRED   CURRENT   READY   AGE
replicaset.apps/mongo-6c6fcb45b8     1         1         1       30s

NAME                                        IMAGE REPOSITORY
TAGS      UPDATED
imagestream.image.openshift.io/mongo    image-registry.openshift-image-registry.svc:5000/workshop/mongo
latest    30 seconds ago
```

Si todo funciona bien, la columna STATUS debería mostrar que el contenedor se está ejecutando.

Recuperación del nombre del host del servicio de MongoDB

Una vez que se completa la implementación, debemos encontrar el servicio que se creó para que nos permita acceder a la base de datos desde el clúster. svc es la abreviatura de servicios:

```
user@host: oc get svc mongodb
NAME      TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)     AGE
mongo     ClusterIP   172.30.88.119    <none>        27017/TCP   77s
```

Se podrá acceder al servicio con el siguiente nombre de DNS, mongodb.workshop.svc.cluster.local, que está compuesto de [nombre del servicio].[nombre del proyecto].svc.cluster.local. Esto se resuelve únicamente dentro del clúster.

Implementación de la API de calificaciones

Es momento de implementar la segunda aplicación, `rating-api`, la cual es de Node.js y se conecta a una instancia de MongoDB para recuperar y calificar elementos. A continuación, encontrará algunos detalles que necesitará para completar esta tarea:

- `rating-api` en GitHub.
- El contenedor expone el puerto `3000`.
- Se configura una conexión de MongoDB con una variable de entorno llamada `MONGODB_URI`.

Anote esta información, ya que la necesitará en las próximas secciones.

Bifurcación de la aplicación a su repositorio de GitHub

Para poder hacer cambios, como por ejemplo agregar webhooks de CI/CD, necesitará su propia copia del código `ratings-api`. En Git, esto se llama "bifurcación". Puede bifurcar la aplicación en su repositorio personal de GitHub. Diríjase al repositorio anterior y haga clic en el botón **Fork** (Bifurcación).

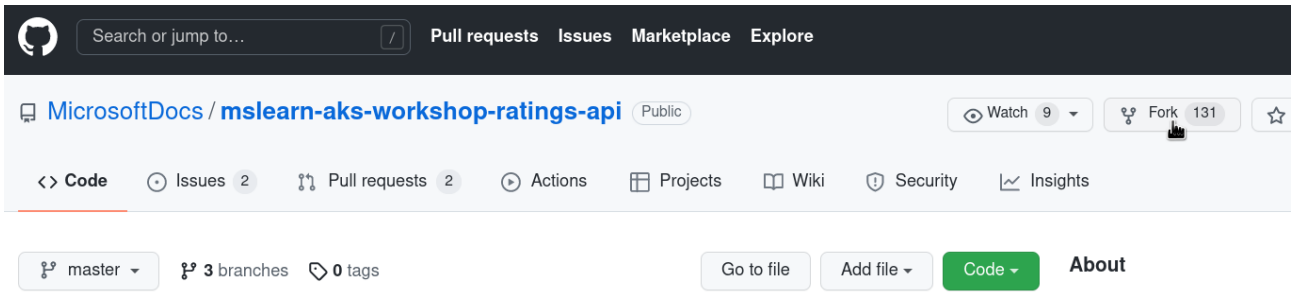


Figura 7.10: bifurcación de la aplicación en el repositorio de GitHub

Anote la nueva dirección del repositorio. Necesitará usarla en las instrucciones siguientes.

Usode la CLI de OpenShift para implementar rating-api

Con OpenShift se puede implementar un código directamente desde un repositorio de Git teniendo en cuenta el contenido y seleccionando una "imagen del compilador" basada en Java, PHP, Python o similares. En este caso, `rating-api` es una aplicación de JavaScript. La imagen del compilador descargará las dependencias de este lenguaje usando npm, lo cual resultará en una imagen en contenedores nueva almacenada en el registro interno de OpenShift. Esta estrategia se llama **Source 2 Image (S2I)** y se describe con más detalle en el glosario.

Puede iniciar una nueva compilación de S2I mediante oc new-app:

```
user@host: oc new-app https://github.com/<your GitHub username>/mslearn-aks-workshop-ratings-api
--strategy=source --name=rating-api

--> Found image 0aea15f (3 weeks old) in image stream "openshift/nodejs" under tag "14-ubi8" for
"nodejs"

    Node.js 14
    ----------
    Node.js 14 available as container is a base platform for building and running various Node.
js 14 applications and frameworks. Node.js is a platform built on Chrome's JavaScript runtime
for easily building fast, scalable network applications. Node.js uses an event-driven, non-
blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time
applications that run across distributed devices.

    Tags: builder, nodejs, nodejs14

    * The source repository appears to match: nodejs
    * A source build using source code from https://github.com/MicrosoftDocs/mslearn-aks-
workshop-ratings-api will be created
      * The resulting image will be pushed to image stream tag "rating-api:latest"
      * Use 'oc start-build' to trigger a new build
```

Cambie a la vista **Topology** (Topología) dentro de la consola web para poder ver que el compilador de la aplicación se inicia y que la implementación es exitosa luego de unos minutos.

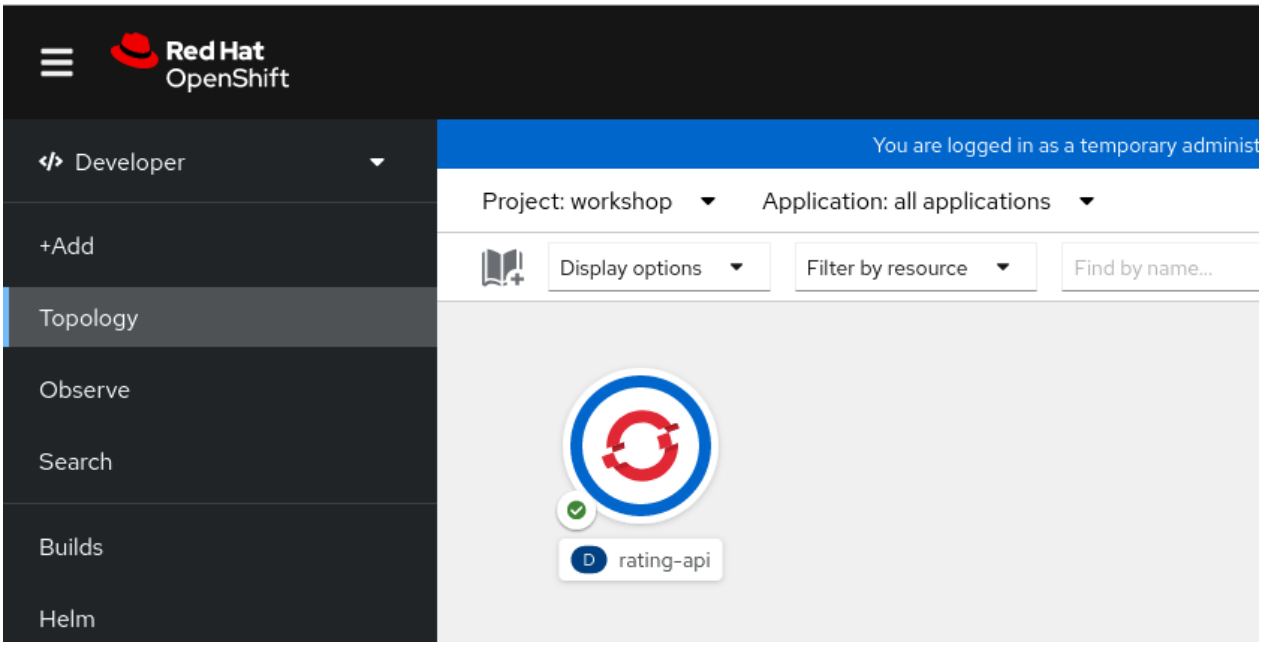


Figura 7.11: la vista Topology (Topología)

La compilación y el inicio del pod deberían tomar solo un minuto o dos.

Configuración de las variables de entorno requeridas

En este punto, ya se implementó la base de datos y la API de calificaciones, pero debemos indicarle a esta última cómo conectarse. La configuración de aplicaciones basadas en contenedores generalmente se logra usando variables del entorno.

Haga clic en la implementación y edítela. Cree la siguiente variable del entorno:

Nombre	Valor
MONGODB_URI	mongodb://mongodb.workshop.svc.cluster.local:27017/ratingsdb

Una vez que la haya guardado, esta nueva variable desencadenará una nueva implementación del servicio `ratings-api` para utilizarla.

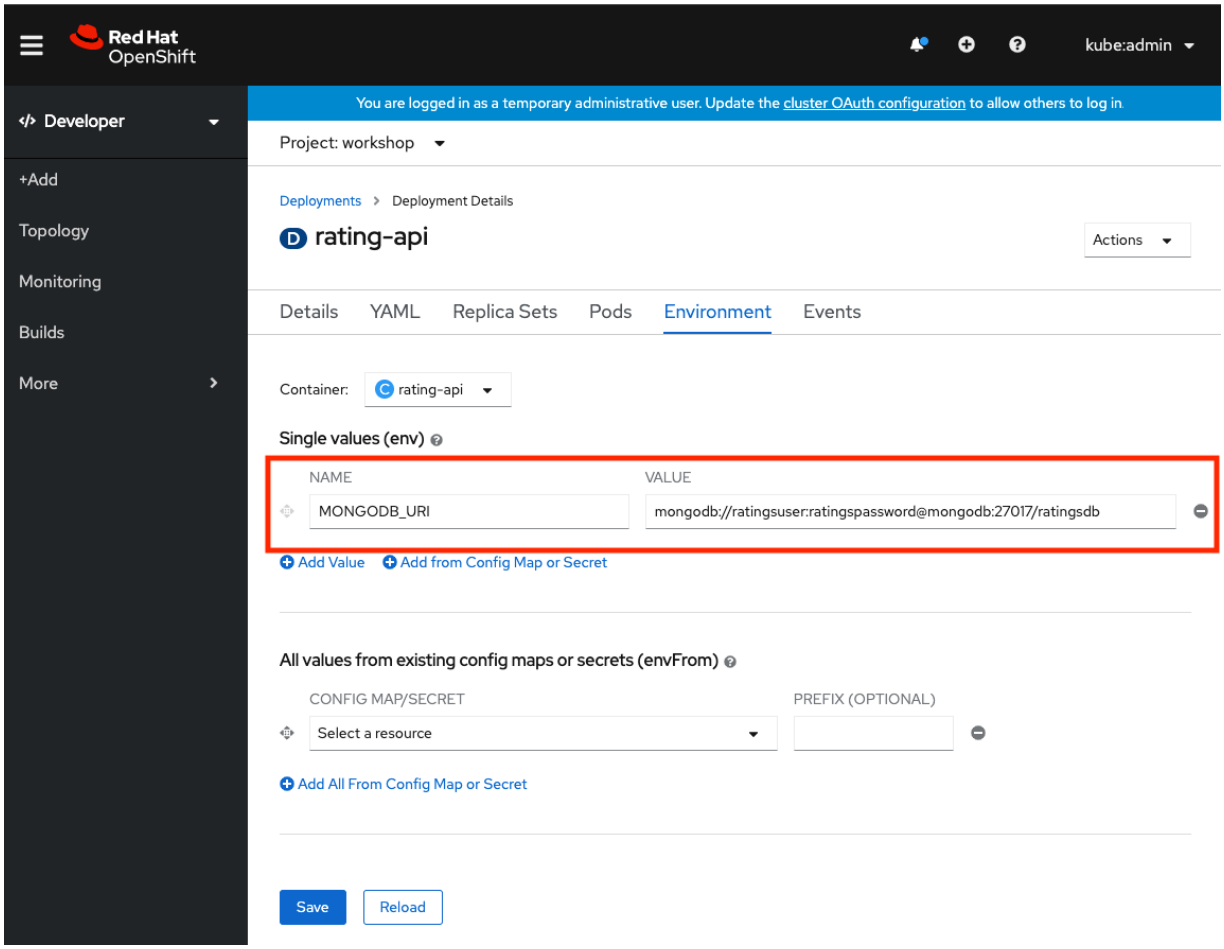


Figura 7.12: establecimiento de la variable del entorno MONGODB_URI mediante la consola web

También se puede hacer en la línea de comandos, de esta manera:

```
oc set env deploy/rating-api MONGODB_URI=mongodb://mongodb.workshop.svc.cluster.local:27017/ratingsdb
```

Independientemente del método que elija, OpenShift deberá reiniciar el contenedor para usar las nuevas variables de entorno.

Verifique que el servicio se esté ejecutando

Si se dirige a los registros de la implementación de `rating-api`, debería ver un mensaje que confirma que el código puede conectarse a MongoDB correctamente. Haga clic en la pestaña **Pods** de la pantalla en la que aparecen los detalles y, luego, en uno de los pods para hacerlo.

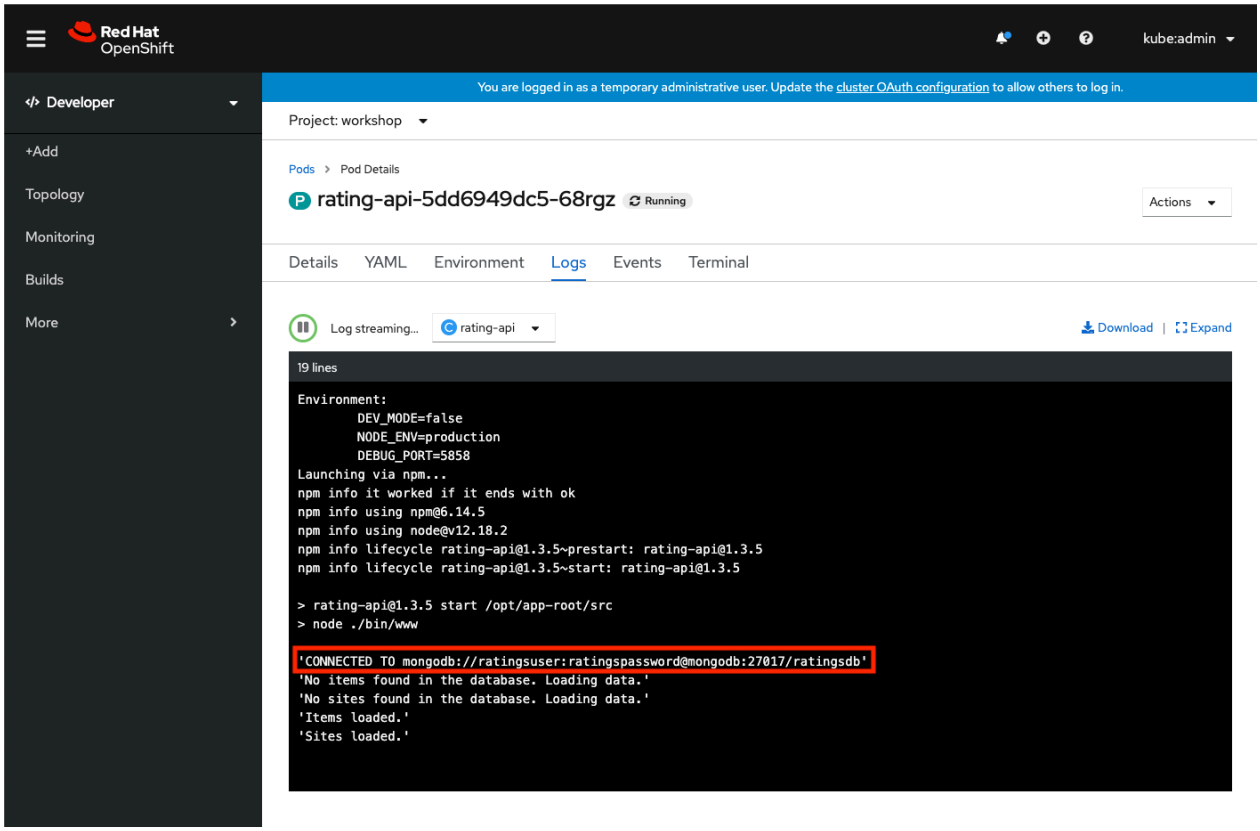


Figura 7.13: el mensaje del registro confirma que el código se puede conectar a MongoDB correctamente

Configure el puerto de servicio de rating-api

OpenShift creará un servicio usando el puerto `8080`. No obstante, debido a una actualización de la biblioteca, se ejecuta en el puerto `3000`, por lo cual es necesario editar el servicio predeterminado.

Diríjase al menú **Networking** → **Services** (Redes ▫ Servicios) y, desde la lista de servicios, edite `rating-api` de la siguiente manera (simplemente reemplace `8080` por `3000`):

```
  ports:
    - name: 3000-tcp
      protocol: TCP
      port: 3000
      targetPort: 3000
```

Reinicie el servicio para usar el puerto nuevo:

```
user@host: oc rollout restart deploy/rating-api
```

Ahora, el servicio estará conectado al puerto correcto: `3000` en lugar de `8080`.

Recupere el nombre del host de servicio de rating-api

Necesitaremos confirmar que haya un servicio de `rating-api`, ya que se utilizará en la sección siguiente, cuando la aplicación `rating-web` se implemente:

```
oc get service rating-api
```

Se podrá acceder al servicio con el nombre de DNS `rating-api.workshop.svc.cluster.local:3000`, en el puerto `3000`, que está compuesto de `[nombre del servicio].[nombre del proyecto].svc.cluster.local`. Esto se resuelve únicamente dentro del clúster.

Implementación de las calificaciones en frontend

`rating-web` es una aplicación de Node.js que se conecta a `rating-api`. A continuación, encontrará algunos detalles que necesitará saber para su implementación:

- `rating-web` en GitHub.
- El contenedor expone el puerto `8080`.
- La aplicación web se conecta a la API en el clúster interno de DNS usando un proxy con una variable del entorno llamada `API`.

Usode la CLI de OpenShift para implementar rating-web

Tal como sucede con `rating-api`, esta aplicación se puede implementar con S2I usando `oc new-app`. No obstante, esta vez se creará con una estrategia de Dockerfile que esté disponible en el repositorio de Git. Por lo tanto, no debe especificar un argumento `--strategy`:

```
user@host: oc new-app https://github.com/<your GitHub username>/mslearn-aks-workshop-ratings-web
--name rating-web

--> Found container image e1495e4 (2 years old) from Docker Hub for "node:13.5-alpine"

    * An image stream tag will be created as "node:13.5-alpine" that will track the source image
    * A Docker build using source code from https://github.com/MicrosoftDocs/mslearn-aks-
workshop-ratings-web will be created
      * The resulting image will be pushed to image stream tag "rating-web:latest"
      * Every time "node:13.5-alpine" changes a new build will be triggered

--> Creating resources ...
    imagestream.image.openshift.io "node" created
    imagestream.image.openshift.io "rating-web" created
    buildconfig.build.openshift.io "rating-web" created
    deployment.apps "rating-web" created
    service "rating-web" created
--> Success
```

El diseño de las dependencias en un contenedor llevará poco tiempo. Espere de 2 a 3 minutos a que termine antes de volver a la vista **Topology** (Topología) para que el servicio aparezca en línea.

Configuración de las variables del entorno requeridas

Cree la variable del entorno `API` para configurar la implementación de `rating-web`. El valor de esta variable será el nombre de host o el puerto del servicio `rating-api`.

En lugar de establecer la variable del entorno con la consola web de Azure RHOS, puede hacerlo con la CLI de OpenShift.

```
oc set env deploy rating-web API=http://rating-api:3000
```

Exposición del servicio rating-web usando una ruta

Si expone el servicio, los usuarios podrán acceder a él a través de una URL pública. En caso contrario, solo podrán lograrlo dentro del clúster.

```
user@host: oc expose svc/rating-web
route.route.openshift.io/rating-web exposed
```

Por último, obtenga la URL del servicio que expuso:

```
user@host: oc get route rating-web
NAME          HOST/PORT                                                PATH    SERVICES      PORT
TERMINATION   WILDCARD
rating-web    rating-web-workshop.apps.zytjwj9a.westeurope.aroapp.io          rating-web    8080-tcp
None
```

Tenga en cuenta que una parte del **nombre de dominio completo (FQDN)** incluye el nombre de la aplicación y el del proyecto de forma predeterminada. El resto es el subdominio de aplicaciones específicas del clúster de Azure Red Hat OpenShift.

Prueba del servicio

Abra el nombre del host en su explorador. Debería poder ver la página de la aplicación de calificaciones. Haga una prueba, emita algunos votos y vea la tabla de posiciones.

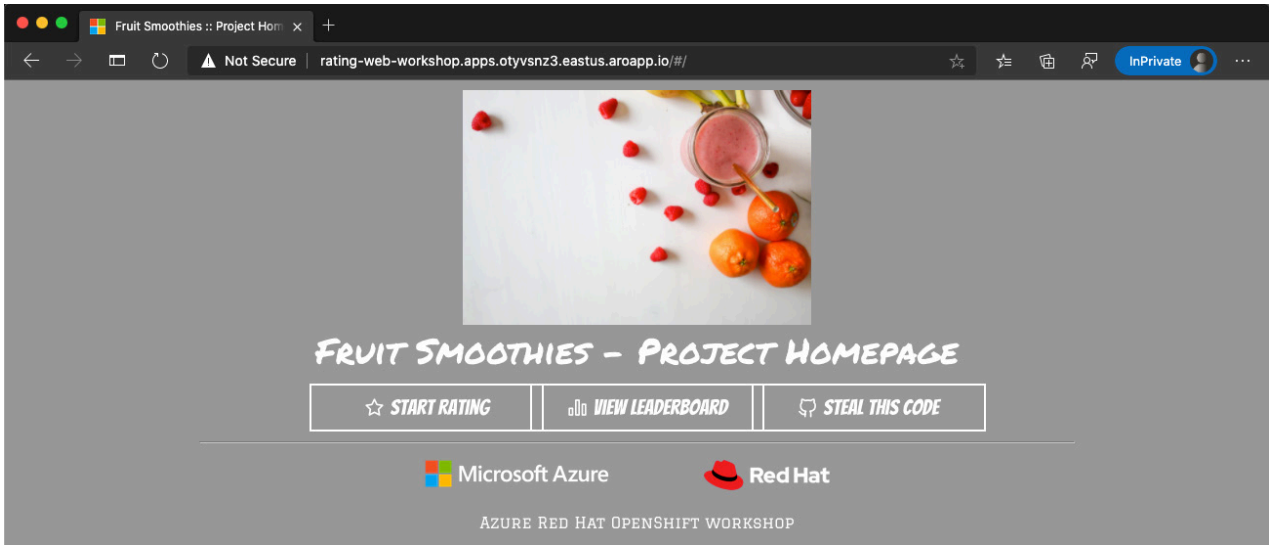


Figura 7.14: prueba del servicio de la aplicación de calificaciones

Con figuración del webhook de GitHub

Para desencadenar diseños de S2I cuando envía código a su repositorio de GitHub, necesitará configurar el webhook:

1. Recupere el secreto de activación del webhook de GitHub. Necesitará usarlo en la URL:

```
user@host: oc get bc/rating-web -o=jsonpath='{.spec.triggers..github.secret}'
3ffcc8d5-a243
```

   Anote la clave del secreto, ya que la necesitará en los próximos pasos.

2. Recupere la URL desde la configuración del compilador:

```
user@host: oc describe bc/rating-web
...
Webhook GitHub:
        URL:    https://api.quwhfg7o.westeurope.aroapp.io:6443/apis/build.openshift.io/v1/
namespaces/workshop/buildconfigs/rating-web/webhooks/<secret>/github
...
```

3. Reemplace el marcador de posición `<secret>` con el secreto que obtuvo en el primer paso. En este caso, es `3ffcc8d5-a243`, por lo que la URL consecuente es la siguiente:

```
https://api.quwhfg7o.westeurope.aroapp.io:6443/apis/build.openshift.io/v1/namespaces/workshop/
buildconfigs/rating-web/webhooks/3ffcc8d5-a243/github
```

   Usará esta URL para configurar el webhook en su repositorio de GitHub.

4. Diríjase al repositorio de GitHub. Seleccione **Add Webhook** (Agregar webhook) desde **Settings** → **Webhooks** (Configuración → Webhooks).
5. En el campo **Payload URL** (URL de carga útil), pegue su URL de GitHub con el valor `<secret>` modificado para usar su secreto.
6. En el campo **Content type** (Tipo de contenido), cambie el valor predeterminado `application/x-www-form-urlencoded` a `application/json`.

7. Haga clic en **Add webhook** (Agregar webhook).

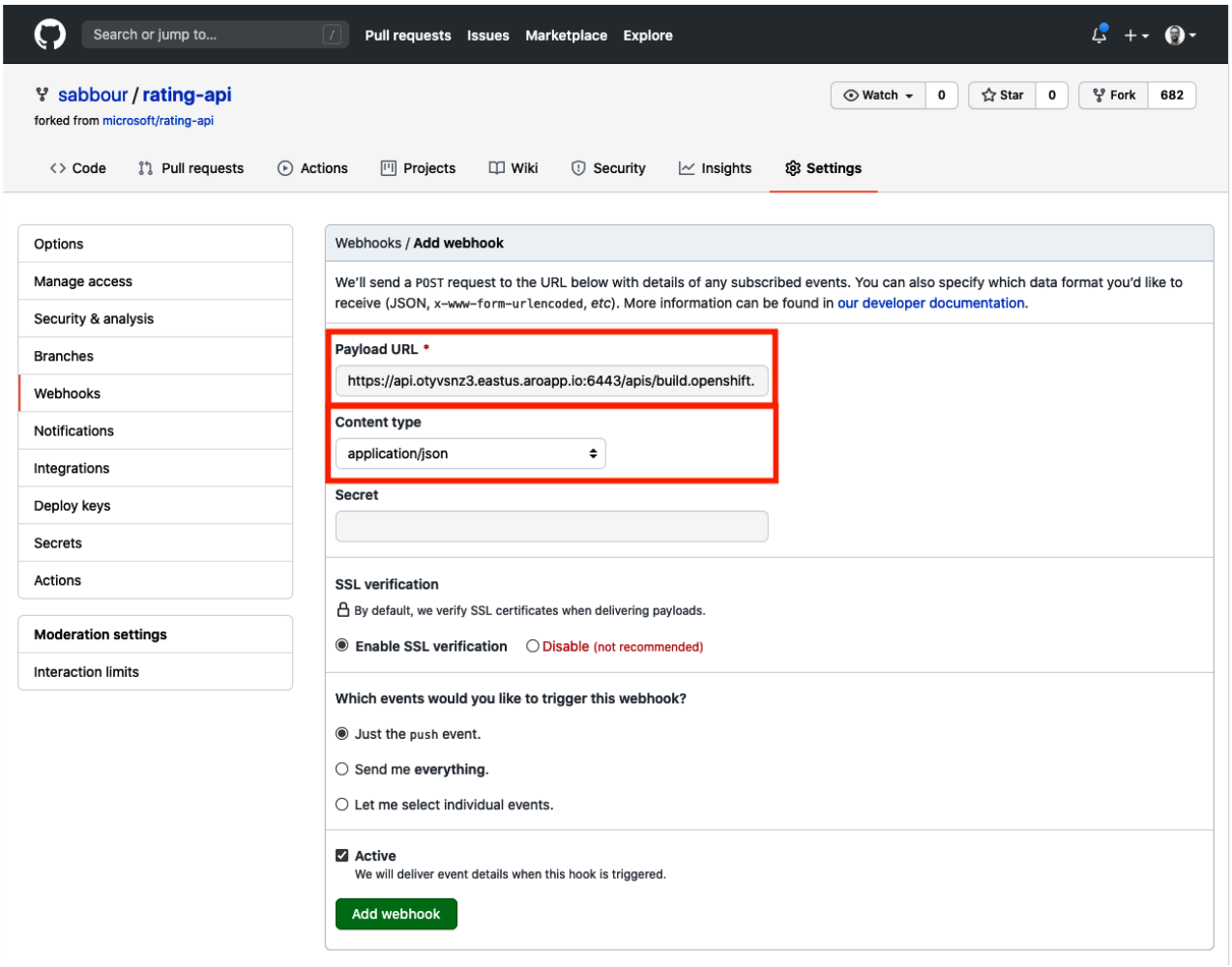


Figura 7.15: adición de un webhook

Debería ver un mensaje de GitHub que indique que su webhook se configuró correctamente.

Ahora, siempre que haga una modificación en su repositorio de GitHub, se iniciará de forma automática un nuevo compilador, el cual desencadenará una nueva implementación de ser exitoso.

Modificación de la aplicación del sitio web para ver la actualización

Diríjase al archivo `https://github.com/<your GitHub username>/rating-web/blob/master/src/App.vue` en su repositorio de GitHub.

Edite el archivo: cambie la línea `background-color: #999;` a `background-color: #0071c5`.

Confirme los cambios del archivo en la rama `master`.

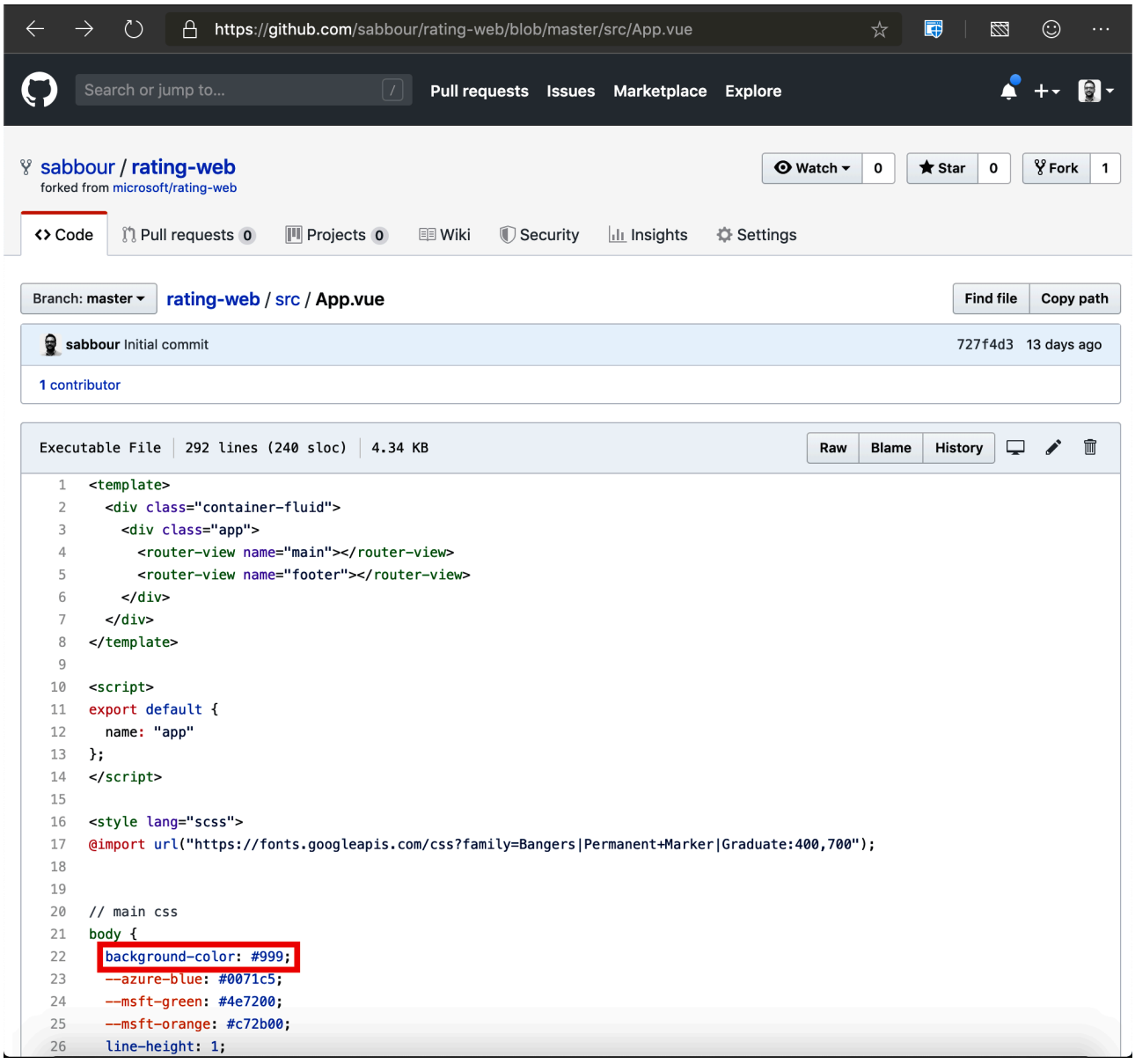


Figura 7.16: confirmación de los cambios en la rama master

A continuación, diríjase a la pestaña **Builds** (Diseños) en la consola web de OpenShift. Verá que el envío activó un nuevo diseño en la lista. Una vez listo, impulsará una nueva implementación y el sitio web debería mostrar el color actualizado.

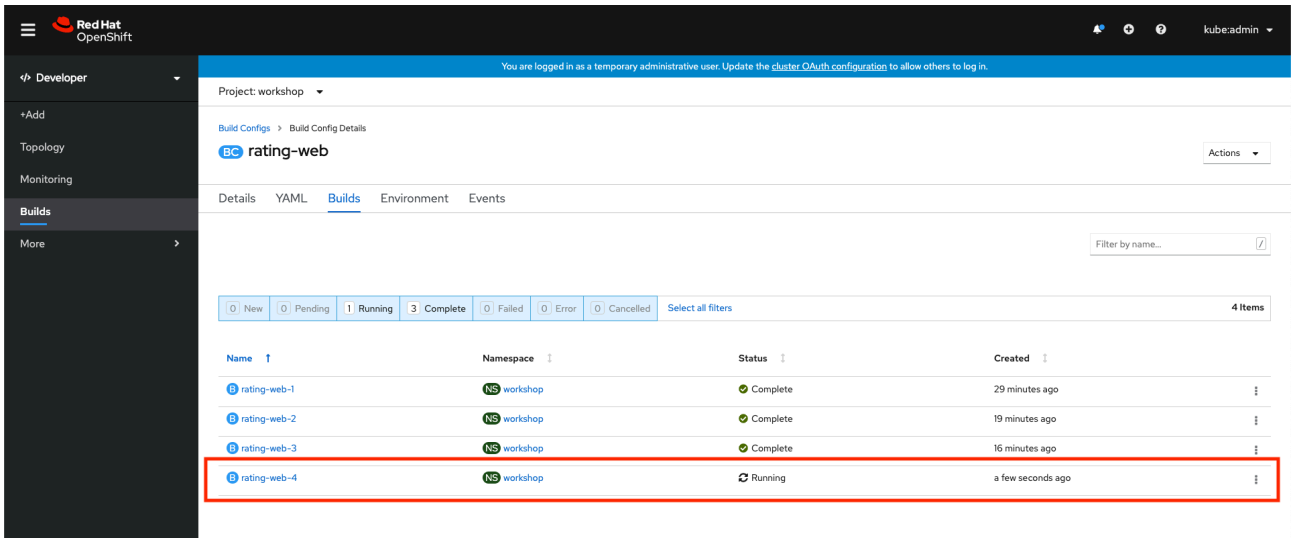


Figura 7.17: la pestaña Builds (Diseños) muestra el nuevo diseño en ejecución

Ahora, vuelva a la página `ratings-web`. Si todo funcionó de forma correcta, verá que cambió el color de fondo.

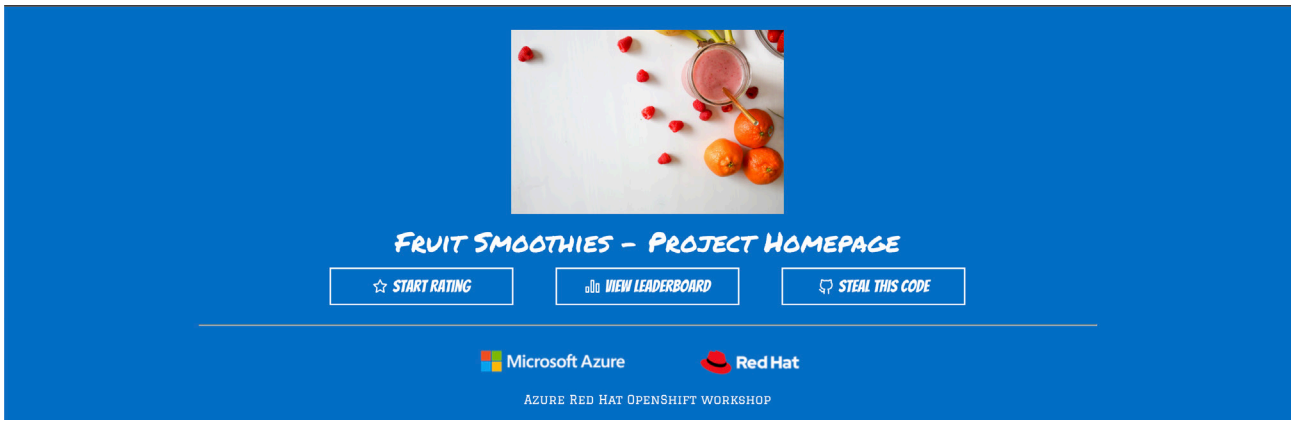


Figura 7.18: la página de inicio de Fruit Smoothies con el nuevo color

Resumen

En este capítulo, implementamos una aplicación básica compuesta de tres aplicaciones de microservicios más pequeñas: una base de datos de MongoDB, `ratings-api` y el código `ratings-web`. Si bien esta no se asemeja a una de producción, cumple la función de un recordatorio rápido de los conceptos al hacer este proceso en Azure Red Hat OpenShift. Puede usar estas instrucciones a modo de referencia.

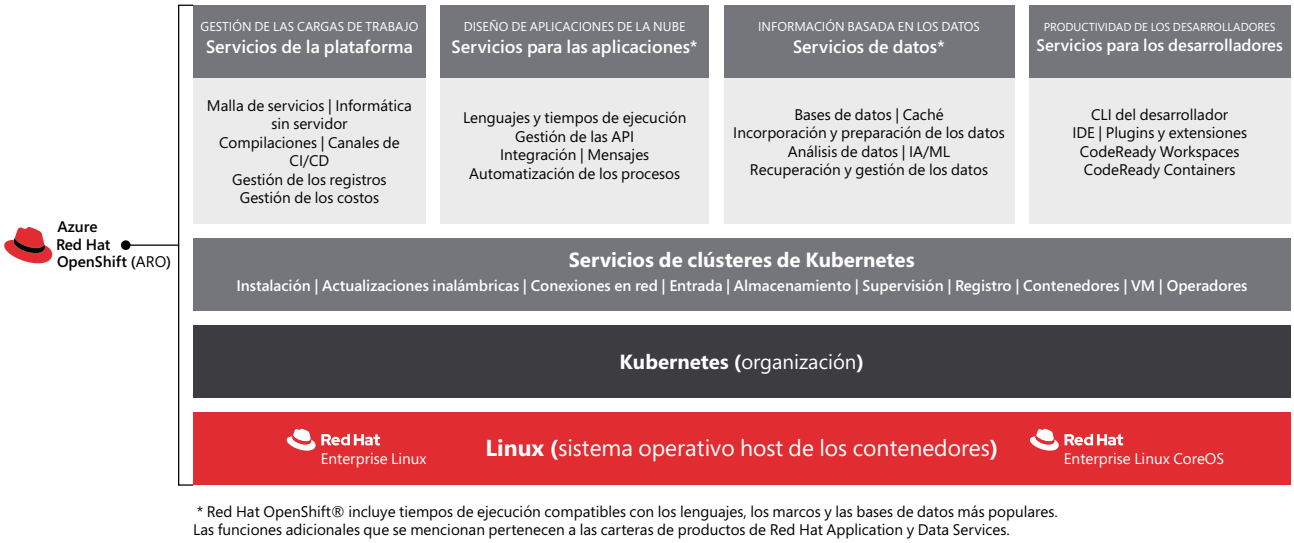
Red Hat OpenShift también es compatible con la implementación de aplicaciones desde la plataforma OperatorHub, los charts de Helm y los sistemas externos de CI/CD como Azure DevOps. La definición de la estrategia más adecuada para su empresa dependerá de las herramientas y tecnologías que usted usa a nivel interno.

En el próximo capítulo, analizaremos el valor adicional de Azure RHOS, el cual la distingue OpenShift como una plataforma de aplicaciones basada en Kubernetes. Obtendremos información detallada sobre las características y las funciones diseñadas para respaldar las necesidades complejas de las aplicaciones empresariales.

Capítulo 8

Análisis de la plataforma de la aplicación

En los capítulos anteriores, aprendimos que Red Hat OpenShift ofrece varios servicios basados en Kubernetes. Estos se combinan para crear una verdadera plataforma de aplicaciones y se pueden agrupar en una de cinco categorías: servicios de plataforma, de aplicaciones, de datos, de desarrollo y de clústeres de Kubernetes.



* Red Hat OpenShift® incluye tiempos de ejecución compatibles con los lenguajes, los marcos y las bases de datos más populares. Las funciones adicionales que se mencionan pertenecen a las carteras de productos de Red Hat Application y Data Services.

Figura 8.1: los servicios incluidos en Azure Red Hat OpenShift

Los elementos agrupados en **OpenShift Platform Plus (Multicluster Management**, **Cluster Security** y **Global Registry**) son productos adicionales que requieren una suscripción. Son compatibles con Azure Red Hat OpenShift, pero no se incluyen en la oferta.

En las secciones siguientes, analizaremos algunos de los beneficios clave que ofrecen esos servicios y proporcionaremos enlaces para que obtenga más información.

Servicios de clústeres: registro de contenedores integrado

Con Red Hat OpenShift se incluye un registro de contenedores interno e integrado que se aplica ni bien se implementa el clúster. Este se usa para servicios internos del clúster, como los operadores, y para los contenedores de aplicaciones de los clientes de forma predeterminada. Es estándar, no requiere una configuración adicional e incluso la mantiene un operador de la infraestructura.

Es posible que los clientes que implementan un servicio de contenedores de Kubernetes precisen su propio registro para que sus imágenes en contenedores se mantengan privadas. Con OpenShift, no serán necesarias la instalación y la configuración del día 2, ya que el registro de contenedores integrado está disponible dentro del clúster. Este es un ejemplo de una característica sencilla de OpenShift que ahorra tiempo.

Integrated OpenShift Container Platform Registry overview
Los administradores suelen exponer este registro de contenedores fuera del clúster para que los usuarios externos puedan insertar imágenes en él. Esto es totalmente compatible dentro de Azure Red Hat OpenShift. Puede obtener más información en la documentación estándar de OpenShift sobre la exposición del registro.

Servicios de plataforma: canales de OpenShift

Los clientes de Red Hat OpenShift cuentan con muchas maneras de diseñar sus aplicaciones y con muchas herramientas populares de CI/CD, como Jenkins, CircleCL y GitHub Actions, que tienen complementos compatibles. No obstante, también proporciona funciones en la plataforma con canales de contenedores en la nube mediante el operador de OpenShift Pipelines.

OpenShift Pipelines está basado en el proyecto de la comunidad llamado Tekton. Cada etapa del canal, como incorporar el código de un repositorio de Git, ejecutar un compilador de Java o preparar un paquete RPM, funciona dentro de un contenedor. Esto significa que los desarrolladores y los operadores pueden formar canales complejos y sofisticados, usando todas las ventajas que ofrecen los contenedores, para diseñar software.

OpenShift Pipelines se puede instalar con OperatorHub. Solo tiene que dirigirse a **OperatorHub** y seleccionar el operador para que se inicie el proceso. No se requiere ninguna configuración y la instalación se completará usualmente en menos de un minuto.

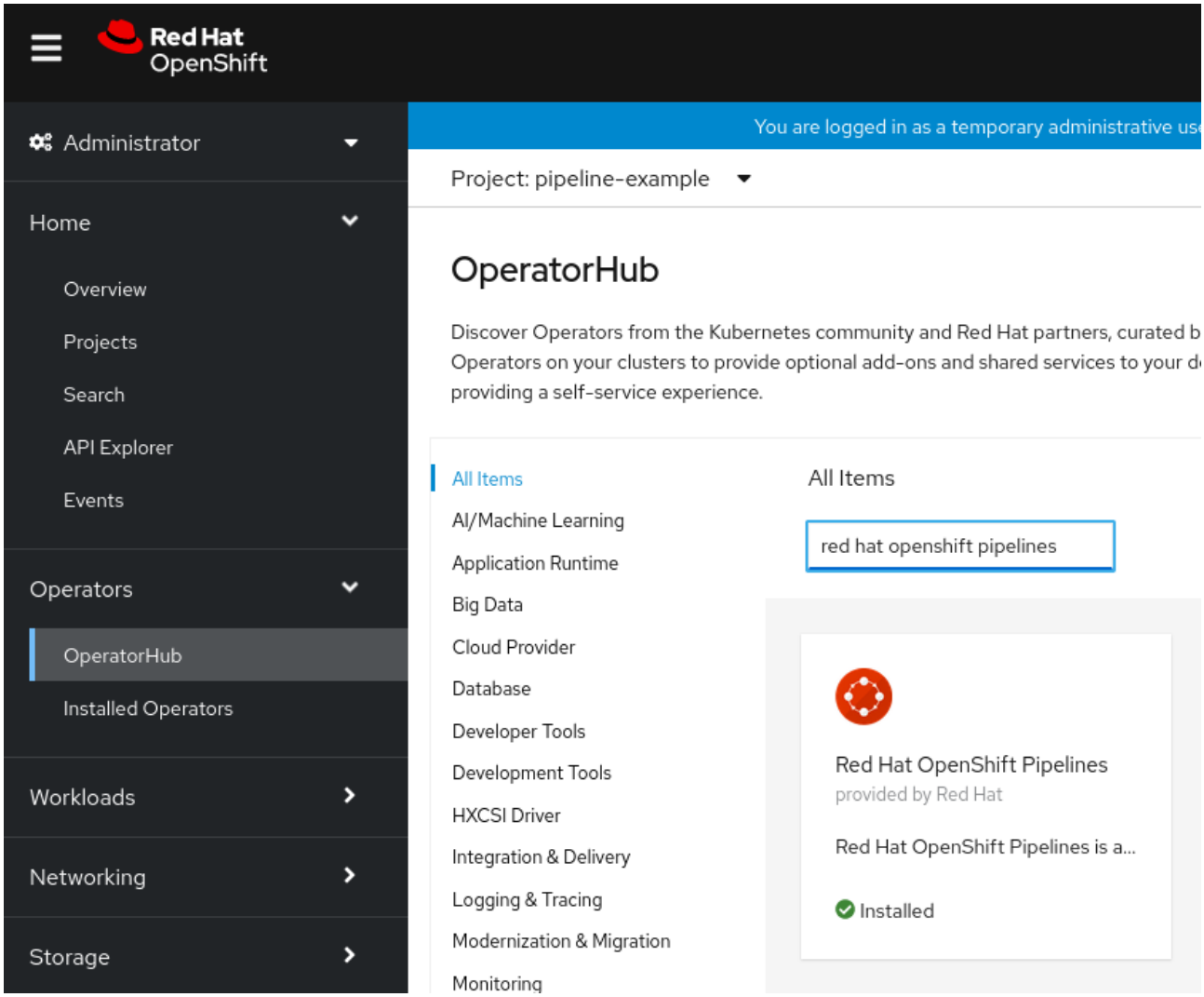


Figura 8.2: instalación de OpenShift Pipelines con OperatorHub

Cuando se instale el operador de OpenShift Pipelines, encontrará una nueva sección llamada **Pipelines** (Canales) en la barra lateral, así como la opción para agregar canales a elementos del catálogo, como al diseñar el ejemplo de Node.js.

Figura 8.3: adición de canales

Con OpenShift Pipelines también puede usar el generador visual para configurar y diseñar canales complejos de ramificación. A continuación, puede ver una captura de pantalla con un ejemplo.

Figura 8.4: un canal complejo

Si bien muchas empresas utilizarán una variedad de herramientas y tecnologías para diseñar el software, OpenShift Pipelines facilita la integración directamente dentro de la plataforma, ya que aprovecha todas las ventajas que ofrecen los contenedores. Brinda una solución de CI/CD uniforme y fácil de usar que requiere una configuración mínima, independientemente de la infraestructura subyacente que se emplee.

Más información

- Understanding OpenShift Pipelines in the OpenShift
- Sitio de la comunidad de Tekton

Servicios de plataforma: OpenShift Serverless

Hay un concepto erróneo que indica que los contenedores son solo una tecnología útil para los servicios de larga duración. En realidad, muchos trabajos pequeños y funciones sin servidor se ejecutan en contenedores de corta duración. Debido a las ventajas que estos ofrecen en términos de la velocidad de inicio, la uniformidad y la facilidad de cierre, también son útiles para las cargas de trabajo sin servidor. Desde luego que todos los servicios de este tipo requieren servidores subyacentes para ejecutar código y, por esta razón, a veces nos referimos a estos como "función como servicio".

Red Hat OpenShift habilita las cargas de trabajo sin servidor, o de función como servicio, mediante el operador de OpenShift Serverless, el cual está basado en el proyecto open source tan conocido llamado Knative.

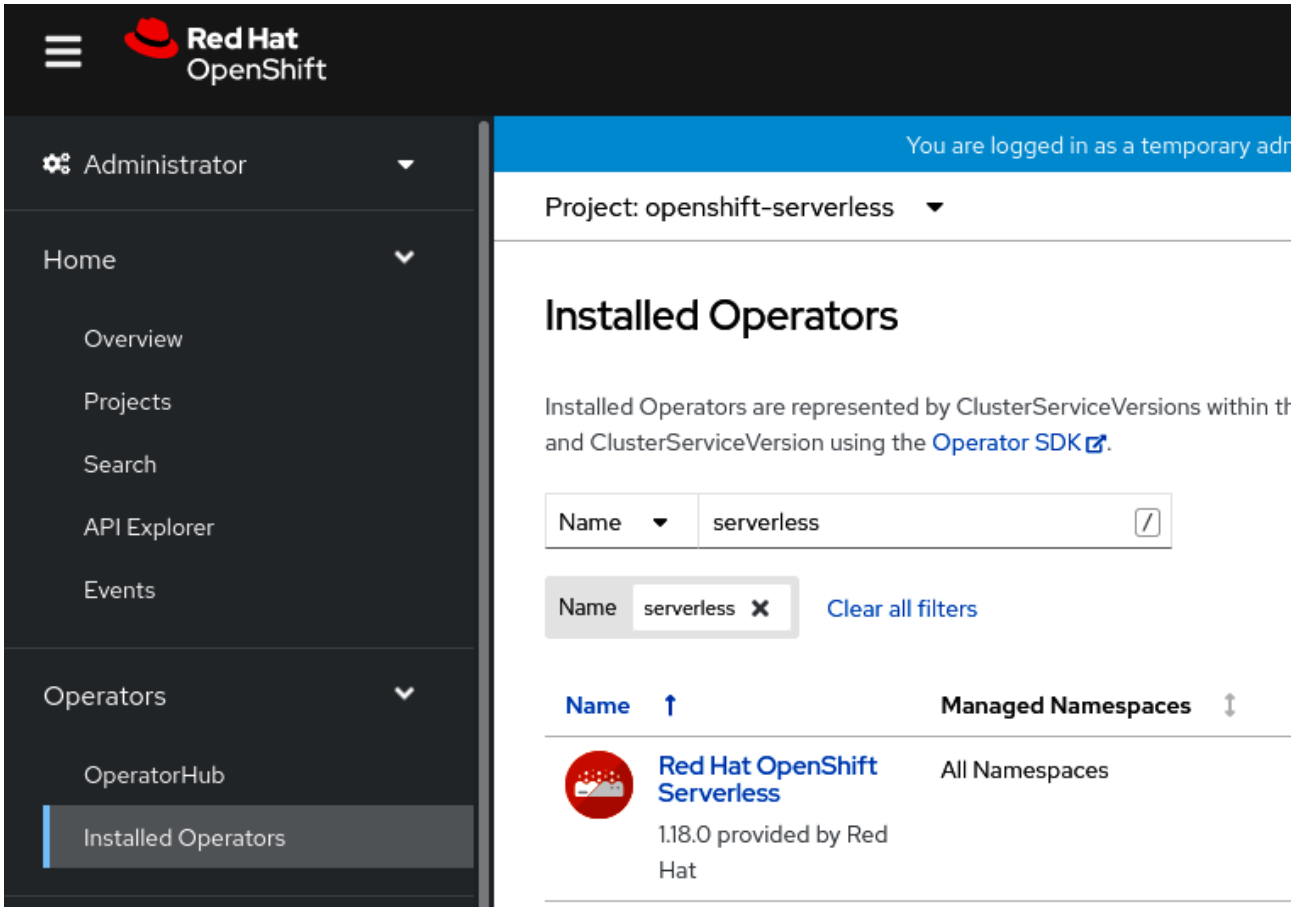


Figura 8.5: vista de los operadores instalados

Una vez que el operador se implementó en el clúster (lo cual, en general, toma uno o dos minutos), debemos configurarlo. Hay dos **definiciones de recursos personalizados (CRD)** particulares a las que debemos prestar atención: **Knative Serving** y **Knative Eventing**.

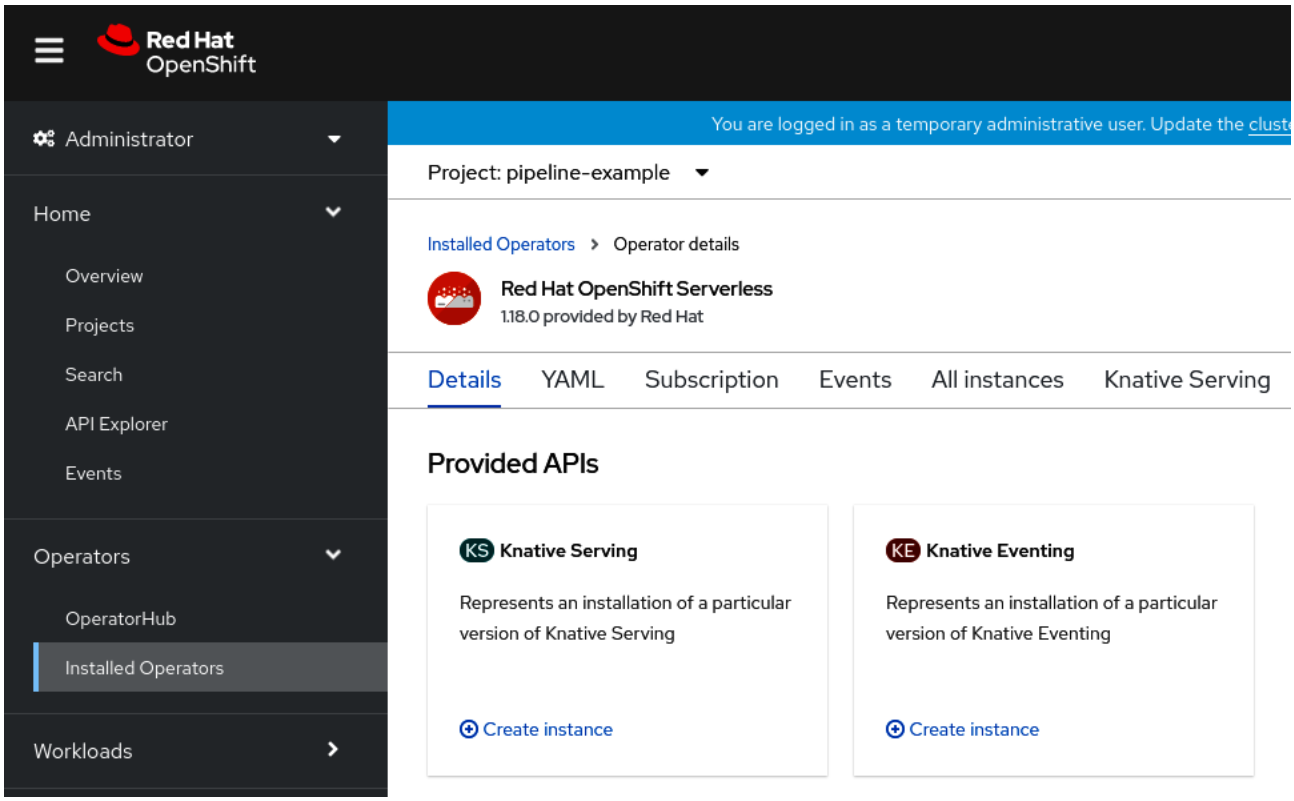


Figura 8.6: Knative Serving y Knative Eventing en el menú de operadores

- **Knative Serving** simplifica la implementación de la aplicación, ajusta su capacidad de forma dinámica en función del tráfico entrante y respalda las estrategias de implementación personalizada con división del tráfico. Un ejemplo de esto es una función que carga una imagen a un bucket de almacenamiento y registra el proceso en una base de datos NoSQL.
- **Knative Eventing** permite los "enlaces tardíos" de fuentes de eventos en el tiempo de ejecución de la aplicación (en oposición al tiempo de diseño). Un ejemplo de esto es una aplicación que responde a nuevas cargas de imágenes en un bucket de almacenamiento. Esta no necesita tener en cuenta el bucket al momento del diseño o de la implementación del evento, pero Knative Eventing facilita el "enlace" entre dicha fuente de eventos y la aplicación en el tiempo de ejecución.

Las dos secciones siguientes incluyen ejemplos para cada uno de estos tipos de aplicaciones sin servidor en OpenShift.

Servicios de plataforma: OpenShift Serverless: ejemplo de Knative Serving

Ahora, demostraremos la manera en que Knative Serving permite ajustar una aplicación de forma automática y, en particular, a cero cuando no hay solicitudes. Un ejemplo muy sencillo que podemos usar es una página web que presenta imágenes ASCII de mascotas:

- Repositorio de GitHub php-ascii-pets

Es muy sencillo agregar este repositorio de Git, ya que Red Hat OpenShift detecta de forma automática una imagen del compilador de PHP compatible.

También encuentra el modo de diseñar este proyecto automáticamente.

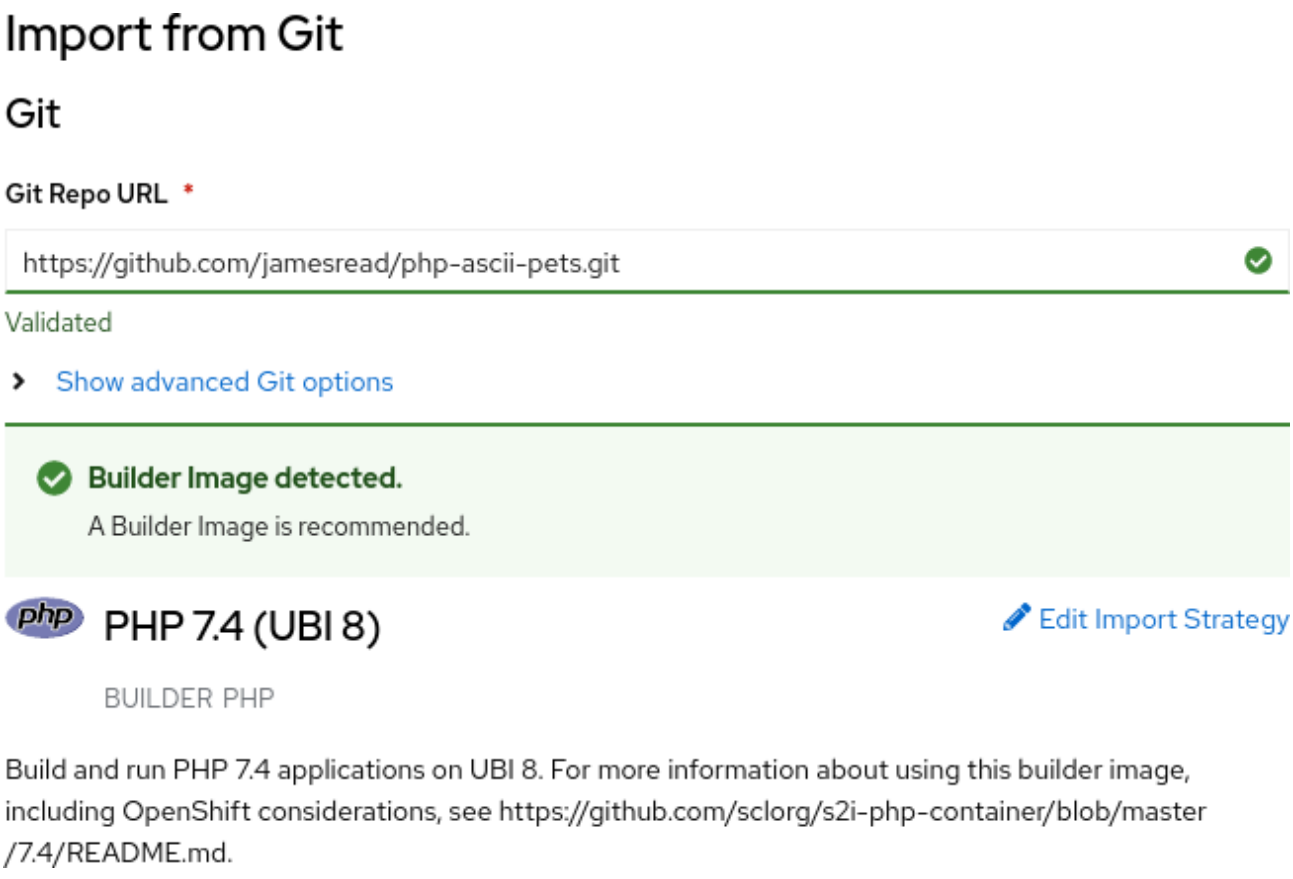


Figura 8.7: una imagen del compilador detectada y recomendada de forma automática

Al seleccionar el tipo de implementación, se determina que sea "sin servidor". Tenga en cuenta que, cuando se instala OpenShift Serverless, se habilita una implementación de este tipo.

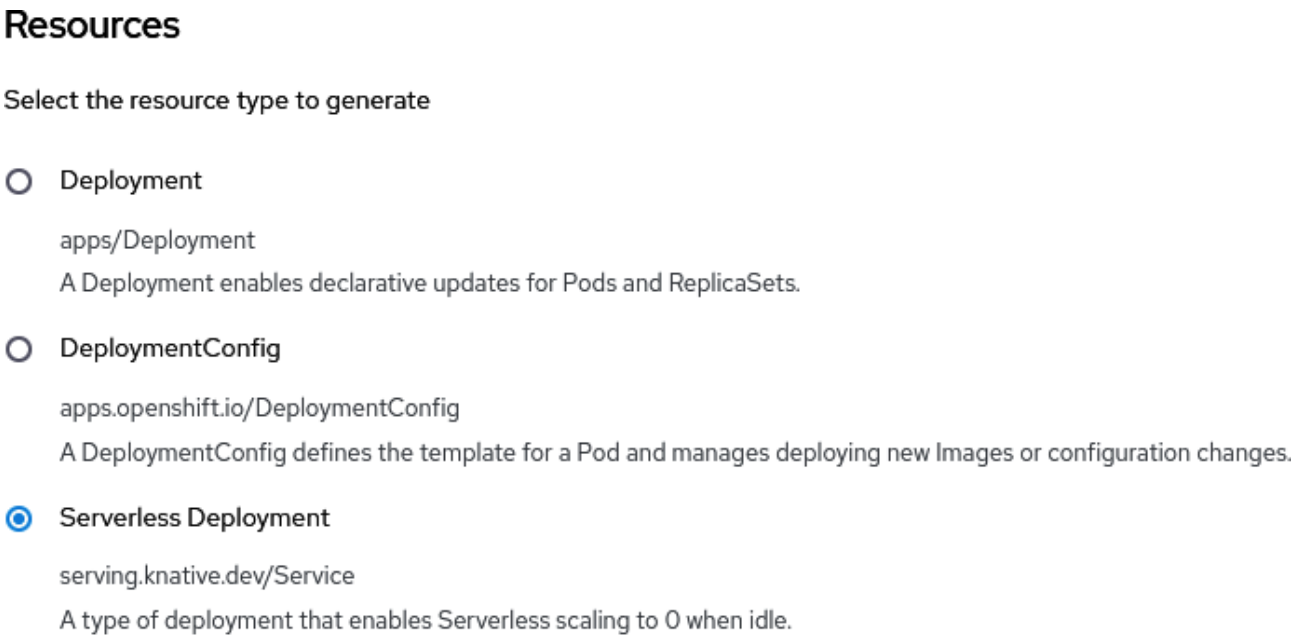


Figura 8.8: tipo de implementación sin servidor

El primer diseño tomará un momento en completarse. Una vez listo, debería haber un pod disponible. La vista de topología debería verse de esta manera:

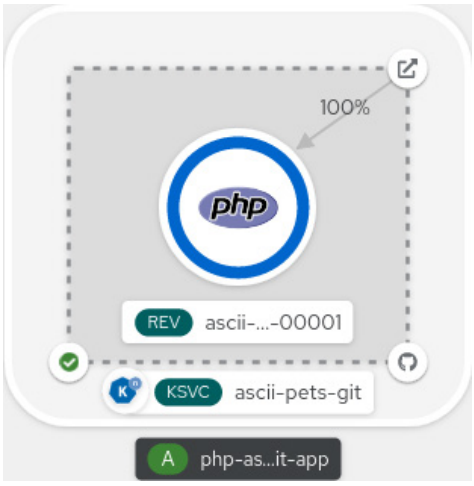


Figura 8.9: una aplicación de Knative Serving

La página de la aplicación se ve como la *Figura 8.9*. Es una aplicación muy sencilla, pero la "mascota", el arte en ASCII, está ligada al nombre de host del pod. En nuestra sencilla aplicación de demostración, cuando ponemos mucha carga adicional, Knative Serving genera más pods, por lo que usted podrá ver las diferentes "mascotas".

No obstante, si no tiene solicitudes durante un minuto, Knative Serving ajustará la aplicación a cero. Si observamos la vista de topología, notamos que ya no se está ejecutando la aplicación.

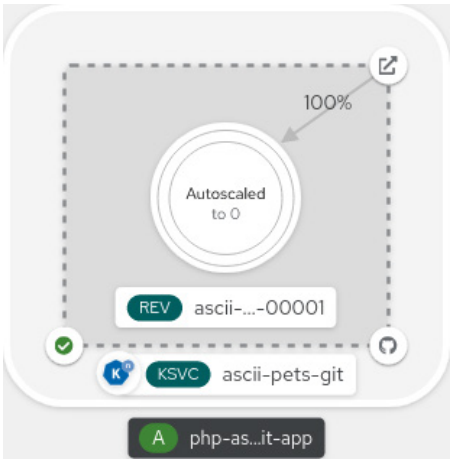


Figura 8.10: una implementación sin servidor, con Knative Serving, con una aplicación ajustada a cero

Finalmente, si alguien visita la página, la aplicación ajustará de manera automática a una réplica, dos, tres o más, en función de la cantidad de instancias que OpenShift considera necesarias para satisfacer la demanda.

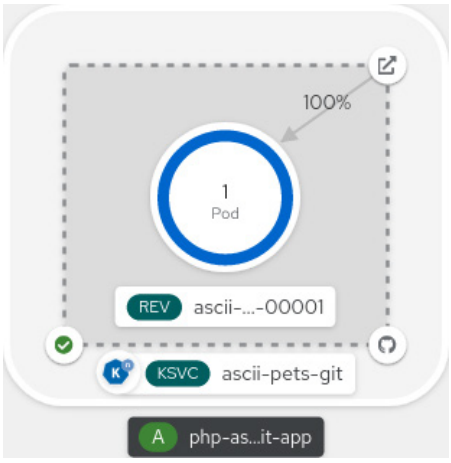


Figura 8.11: ajuste automático en función del tráfico

Red Hat OpenShift Serverless, con Knative Serving, permite que las empresas ajusten su capacidad de forma dinámica con muy poca configuración adicional y sin generar cambios en las aplicaciones. Esto puede ser sumamente útil para que las implementaciones se mantengan en un tamaño adecuado y para evitar que los recursos se usen y se paguen sin necesidad alguna.

Más información

- About OpenShift Serverless
- learn.openshift.com, que incluye un curso sobre OpenShift Serverless
- Sitio de la comunidad de Knative

Servicios de plataforma: OpenShift Serverless: ejemplo de Knative Eventing

Desarrollemos en mayor profundidad la información sobre OpenShift Serverless: este servicio también puede ajustar una aplicación en función de los indicadores, además del tráfico entrante. En especial en contextos tales como la arquitectura impulsada por eventos, es conveniente poder aumentar las instancias de una aplicación para procesar los eventos de una cola de mensajes, o bien despertarse con un temporizador.

Con la misma implementación, la consola de OpenShift ofrece una configuración sencilla para las diferentes fuentes de eventos.

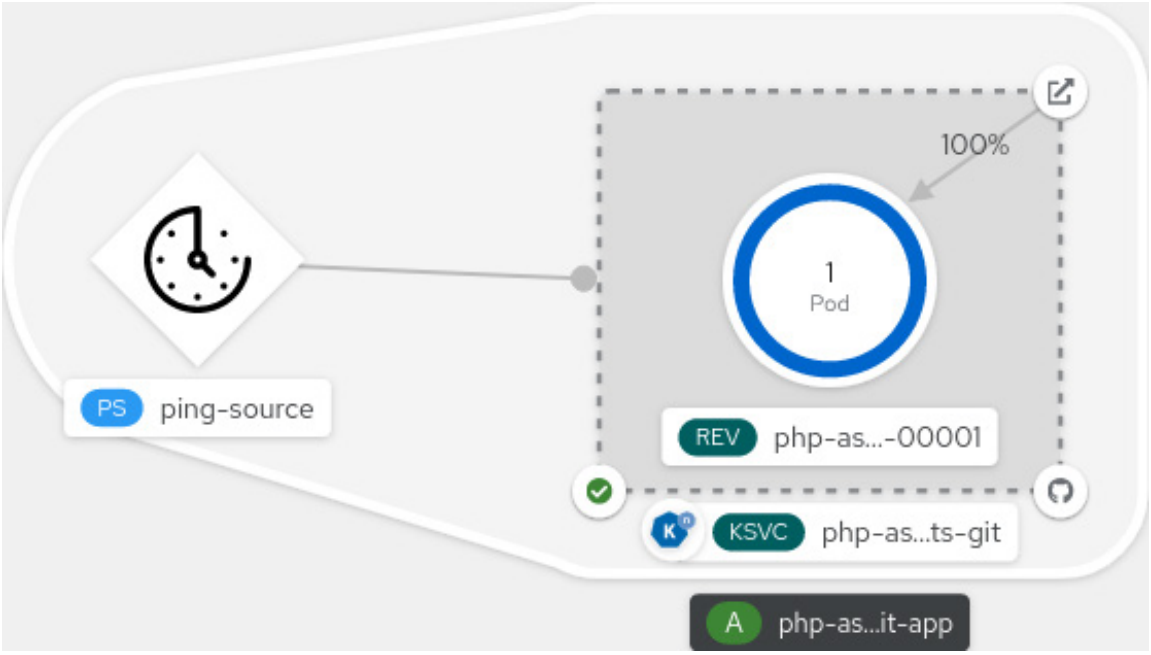


Figura 8.12: una fuente de eventos "ping", la cual notifica a la aplicación en función de un temporizador

Dos casos de uso sencillos para una fuente de eventos ping sería "despertar" ya sea un contenedor de trabajos de respaldo todos los días a la medianoche o un contenedor de supervisión que se ejecuta periódicamente.

Más información

- OpenShift Serverless Eventing explained in 5 minutes
- About OpenShift Serverless

Servicios de plataforma: OpenShift Service Mesh

La solución Red Hat OpenShift Service Mesh, la cual se basa en el proyecto open source Istio, agrega una capa transparente sobre las aplicaciones distribuidas actuales sin que sea necesario realizar cambios en el código del servicio. Puede agregar soporte de esta solución si implementa un proxy de sidecar especial en su entorno, el cual intercepta todas las comunicaciones de redes entre microservicios. Puede configurar y gestionar la malla de servicios con las funciones del plano de control.

Con OpenShift Service Mesh, se habilitan, entre otras, las siguientes funciones:

- El cifrado transparente para la comunicación entre servicios, con mTLS automática.
- La compatibilidad con varias versiones de un servicio y la posibilidad de realizar pruebas A/B, por ejemplo.
- La capacidad de observar la manera en que se comunican los microservicios con Kiali.
- El rastreo de llamadas servicio a servicio con Jaeger.

Tal como todas las demás funciones de la plataforma de OpenShift, Service Mesh se instala con un operador de Red Hat.

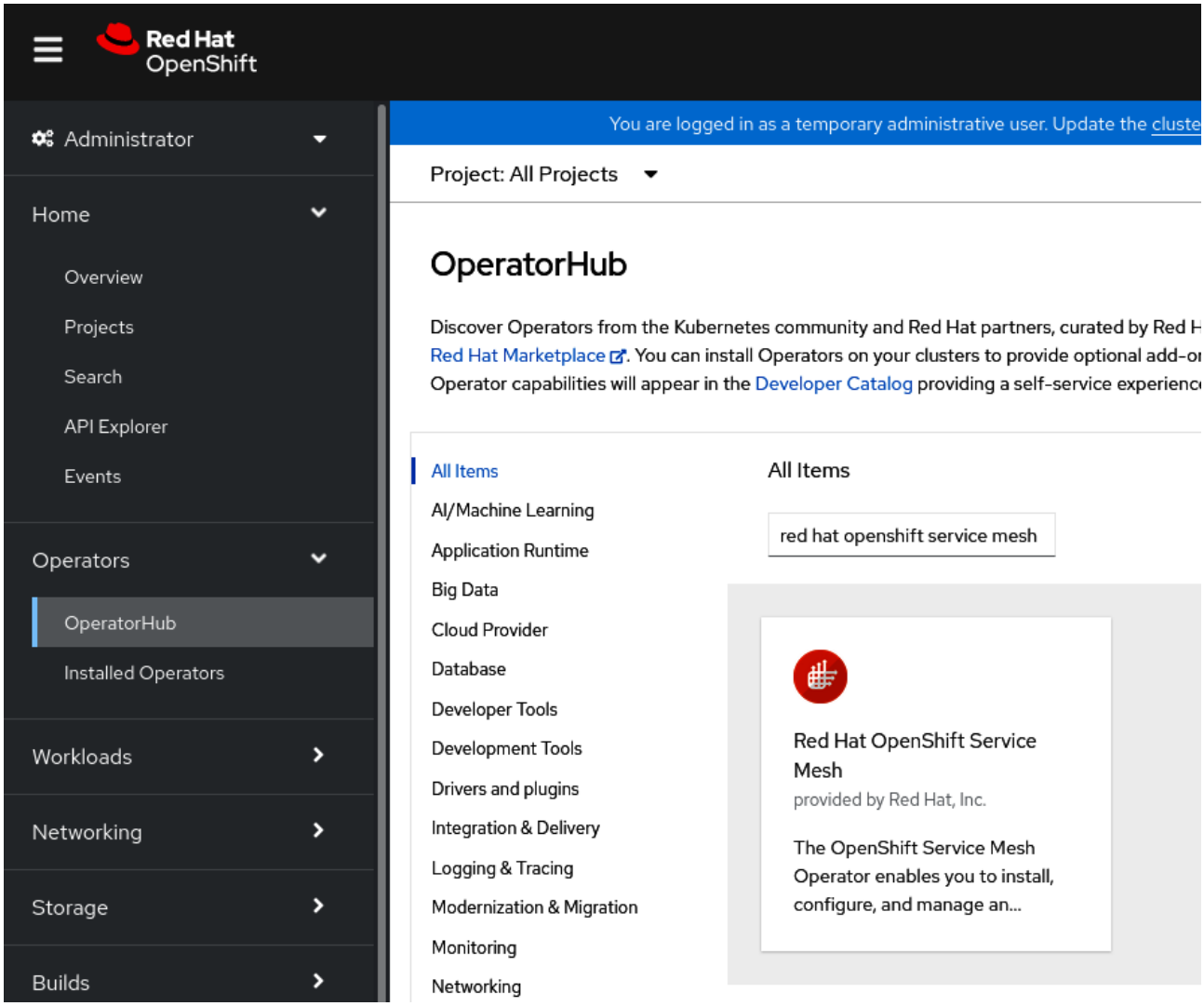


Figura 8.13: instalación de Service Mesh con OperatorHub

La documentación de OpenShift Service Mesh incluye una excelente aplicación de ejemplo llamada la demostración BookInfo. Es sencilla y emula una librería que se ejecuta en OpenShift.

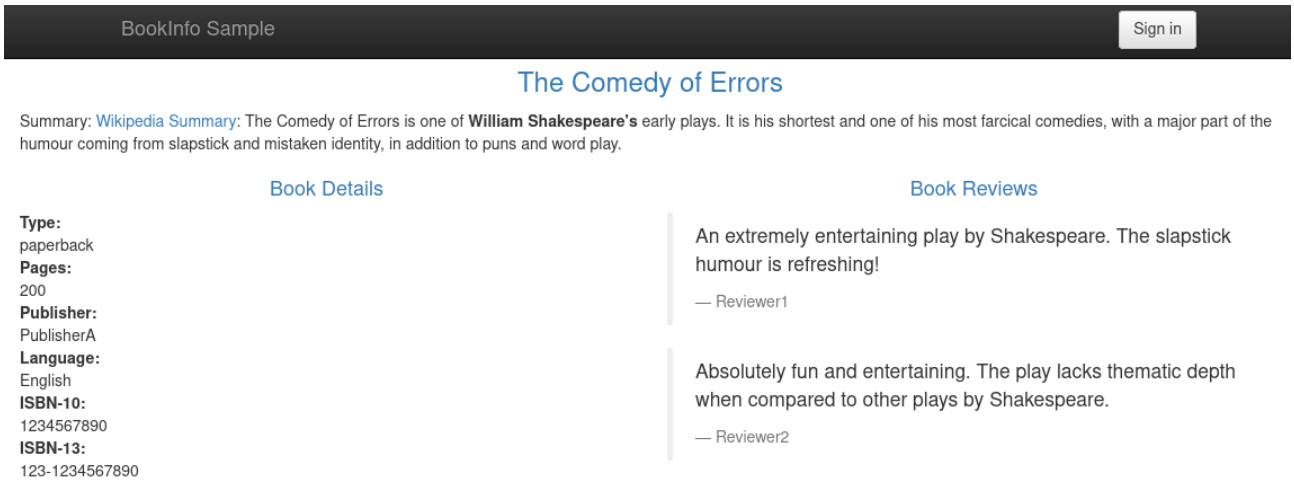


Figura 8.14: la aplicación BookInfo

Para que esta aplicación se ejecute con Service Mesh, debemos crear un plano de control. Podemos hacerlo fácilmente con la interfaz gráfica de OpenShift, como se muestra en la *Figura 8.15*:

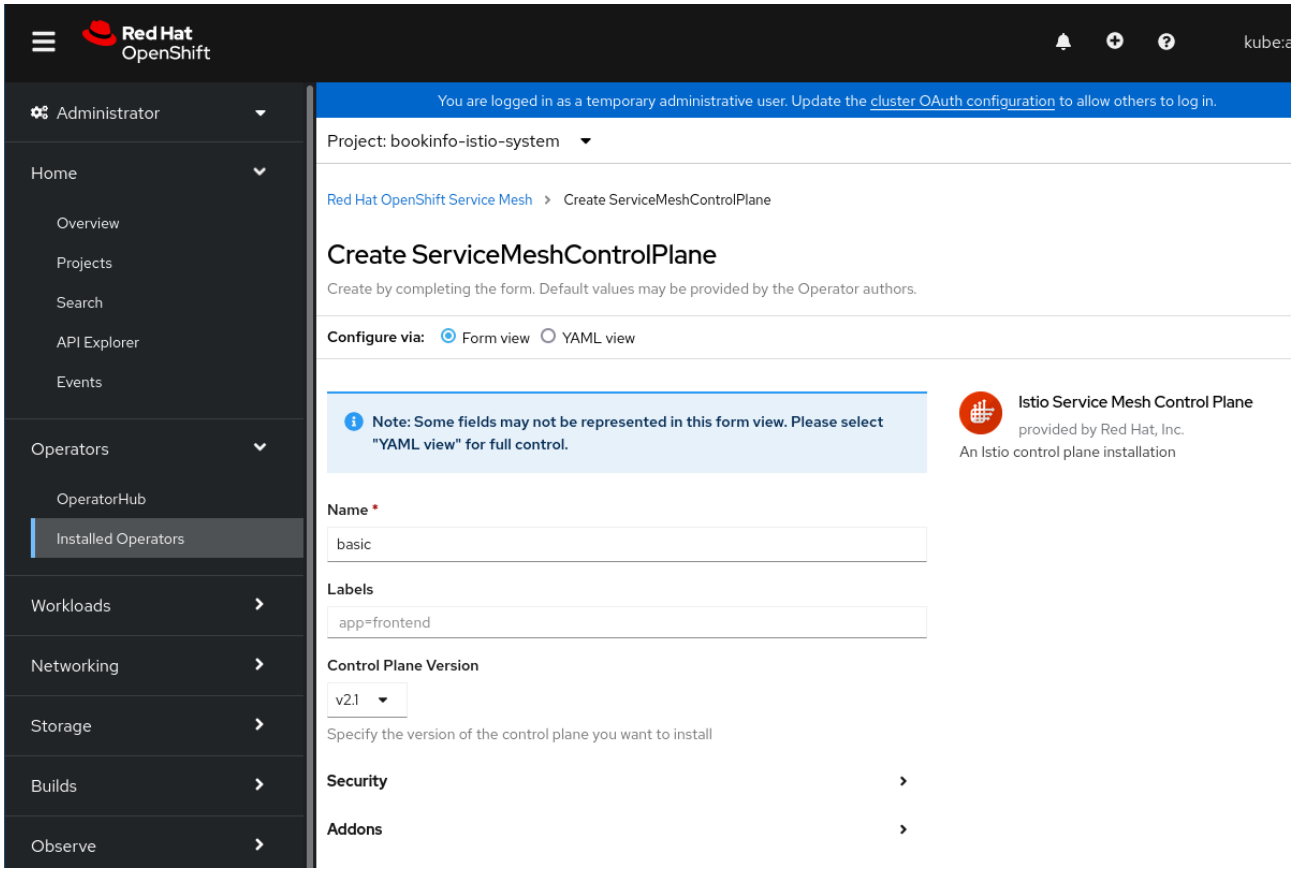


Figura 8.15: configuración del plano de control en el proyecto bookinfo-istio-system de BookInfo

El proyecto aceptará el tráfico entrante mediante el plano de control, para el cual se creó un proyecto aparte para alojarlo, llamado `bookinfo-istio-system`.

No es necesario separar el plano de control de Service Mesh del proyecto, e incluso es posible compartirlo entre varios proyectos.

En las próximas dos secciones, analizaremos dos casos de uso de Service Mesh en mayor detalle: capacidad de observación y rastreo distribuido.

Servicios de plataforma: OpenShift Service Mesh: capacidad de observación con Kiali

Teniendo en cuenta los pods subyacentes de esta aplicación, la vista Topology (Topología) está compuesta de seis microservicios.

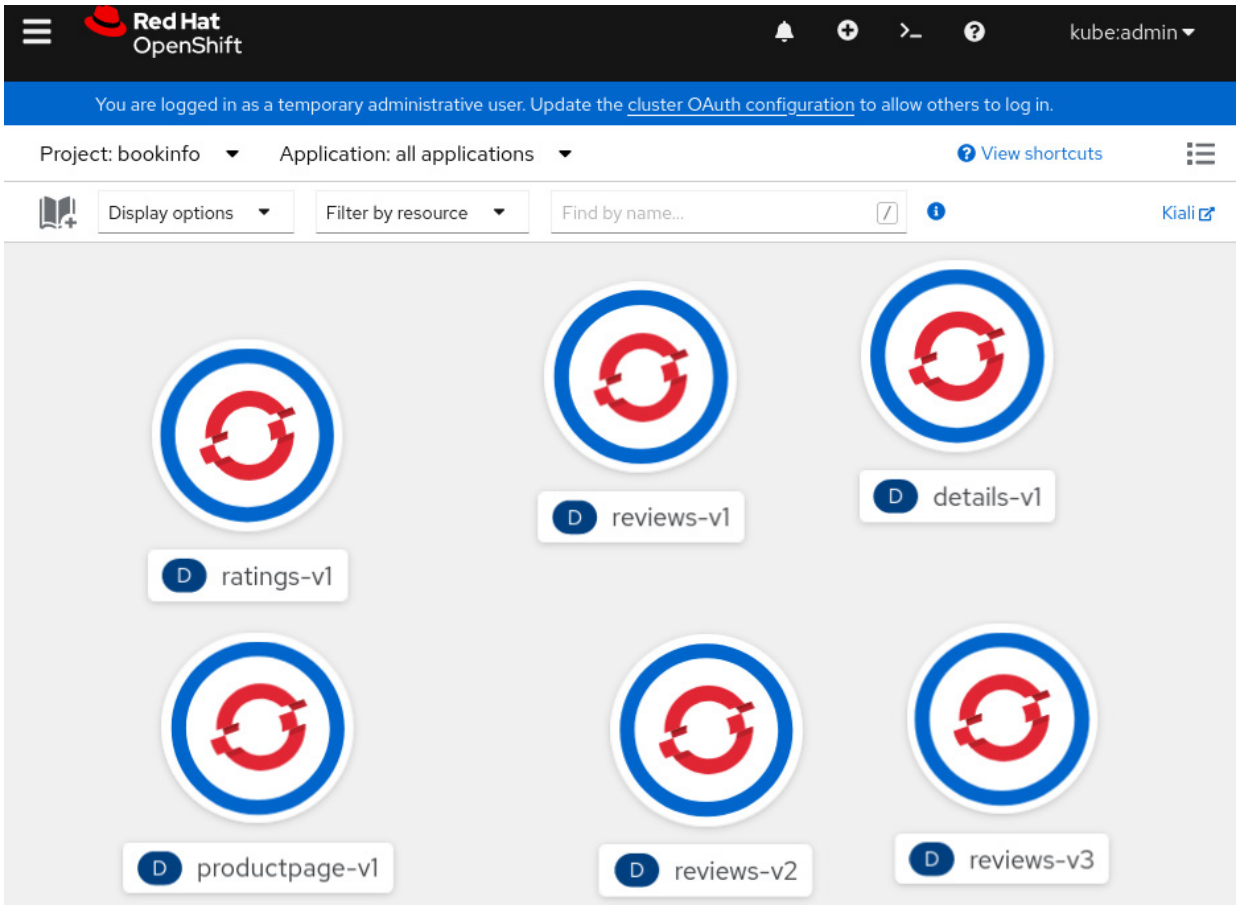


Figura 8.16: los seis servicios que constituyen la aplicación BookInfo

Si bien esta vista es útil, no demuestra la manera en que estos servicios se conectan. Ingrese la primera ventaja de Service Mesh: la capacidad de ajuste. Si abrimos una consola levemente diferente, llamada Kiali, podemos ver que es una representación arquitectónica más precisa de esos seis microservicios.

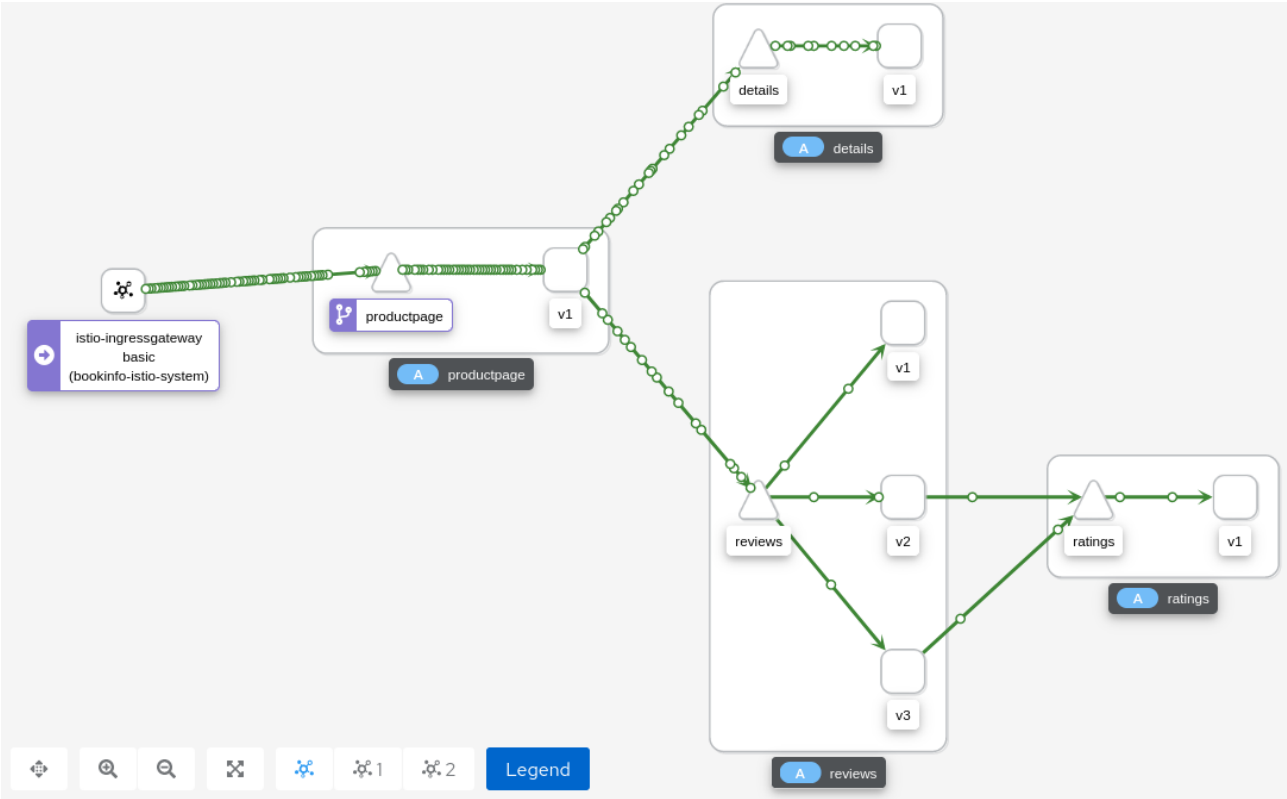


Figura 8.17: el gráfico de la arquitectura de la aplicación generado de forma automática, habilitado por Service Mesh

Esta vista muestra la conectividad real entre los servicios y un diagrama del tráfico en tiempo real. En este caso, la aplicación obtiene una gran cantidad de tráfico, y cada solicitud se representa con una animación circular en la conexión.

Si llevamos esta capacidad al siguiente nivel, un administrador puede hacer clic en una de las conexiones del servicio y ver el perfil del tráfico. En este caso, podemos ver que el estado general es **HTTP OK**.

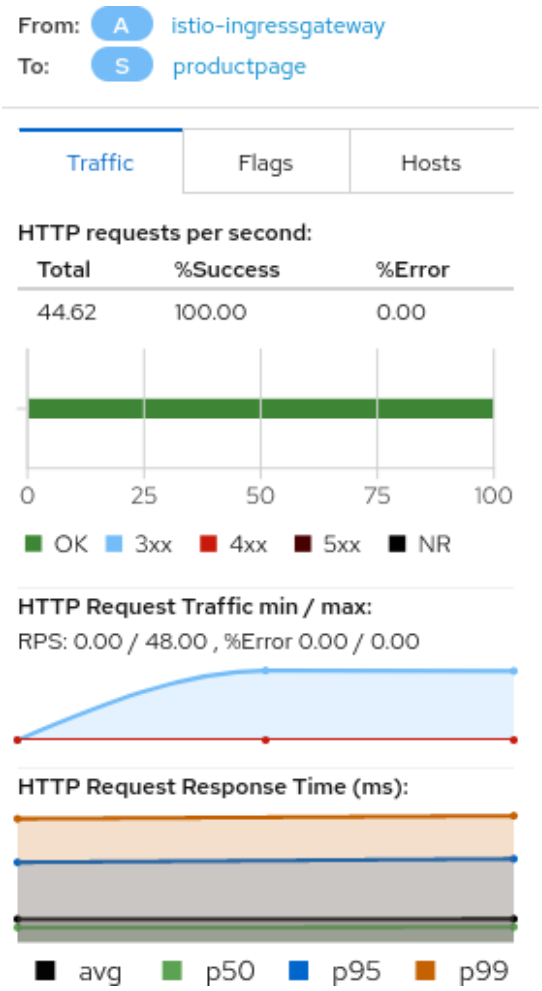


Figura 8.18: varios estados de HTTP

Podemos introducir un error artificial en esta aplicación si cerramos el microservicio `details`, ajustándolo a cero réplicas:

Figura 8.19: ajuste a cero réplicas del servicio details, para simular una falla

Ahora, si volvemos a la vista de Kiali, verá que se agregaron algunos elementos al diagrama. No obstante, lo que más llama la atención es que la conexión al servicio `details` está resaltada en rojo para indicar un error.

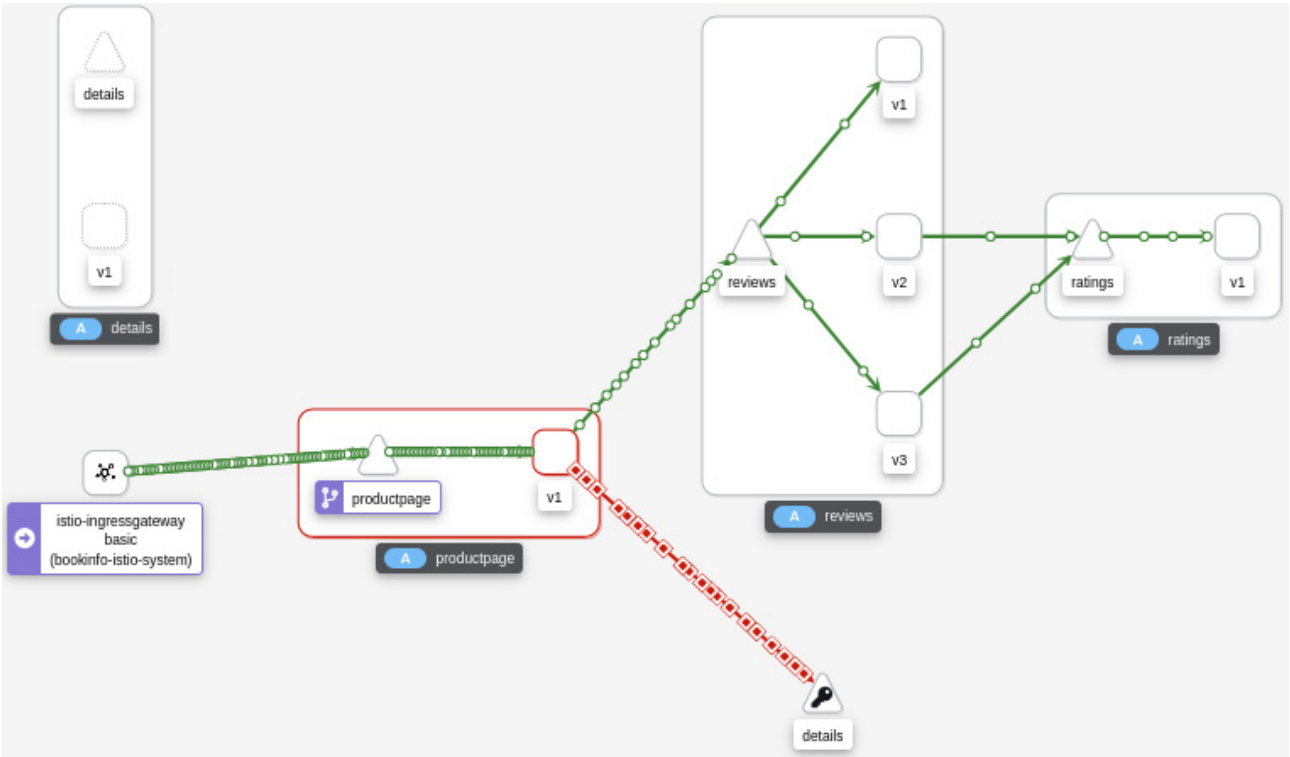


Figura 8.20: la conexión en rojo indica un problema con este servicio

Aprendimos que Kiali es de gran ayuda para la capacidad de observación. En aplicaciones más pequeñas como BookInfo, Service Mesh es útil. Sin embargo, a medida que estas se hacen más grandes y complejas e involucran 20 microservicios o más, así como muchos tipos diferentes de interacciones, las herramientas como Service Mesh se vuelven altamente valiosas e incluso esenciales.

Servicios de plataforma: Red Hat OpenShift: rastreo distribuido con Jaeger

Otro servicio muy valioso que proporciona Service Mesh es Jaeger, el cual permite el rastreo distribuido de aplicaciones de microservicios complejas. En la sección anterior, trabajamos con BookInfo y observamos que cuando se utilizaba el servicio `details` fuera de línea, las solicitudes comenzaban a fallar.

En este caso, la causa era fácil de identificar: `details` no estaba disponible y nosotros éramos los responsables. Sin embargo, si el motivo hubiese requerido una investigación más profunda, el rastreo distribuido de Jaeger habría ayudado mucho.

Jaeger rastrea las solicitudes del primer servicio usando conexiones posteriores en el sistema. Si bien se requiere código de la aplicación adicional para habilitarlo, las bibliotecas del cliente y los cambios mínimos de código facilitan esta tarea a los desarrolladores.

Si observamos el servicio `productpage` (que está escrito en Python), podemos identificar el código relevante que permite el rastreo distribuido de Jaeger en ese caso. Este código importa la biblioteca del cliente de Jaeger en `productpage`:

```
from jaeger_client import Tracer, ConstSampler
from jaeger_client.reporter import NullReporter
from jaeger_client.codecs import B3Codec
```

Una vez que se importe la biblioteca, la página del producto necesita instalar un nuevo rastreador. El código inicia un `Tracer` de Jaeger en el servicio `productpage`:

```
tracer = Tracer(
    one_span_per_rpc=True,
    service_name='productpage',
    reporter=NullReporter(),
    sampler=ConstSampler(decision=True),
    extra_codecs={Format.HTTP_HEADERS: B3Codec()}
)
```

Por último, mientras seguimos en `productpage`, le decimos a Jaeger que queremos crear un intervalo nuevo, es decir, una solicitud nueva a varios servicios:

```
span = tracer.start_span(
    operation_name='op', child_of=span_ctx, tags=rpc_tag
)
```

Jaeger seguirá el rastreador en varios servicios, los cuales deben adaptarse para trabajar con este software. No obstante, las bibliotecas de los clientes están disponibles para la mayoría de los lenguajes de programación conocidos.

El código fuente completo para el servicio `productpage` se puede encontrar en GitHub.

Hay dos opciones para equipar el código: mediante las bibliotecas de clientes de Jaeger, que hoy en día se consideran obsoletas, o con las opciones de estándares abiertos del proyecto OpenTelemetry, las cuales son las preferidas:

- Bibliotecas de OpenTelemetry

Una vez que la iniciativa de equipar la aplicación se complete, podrá ver rastros que lucen de esta manera:

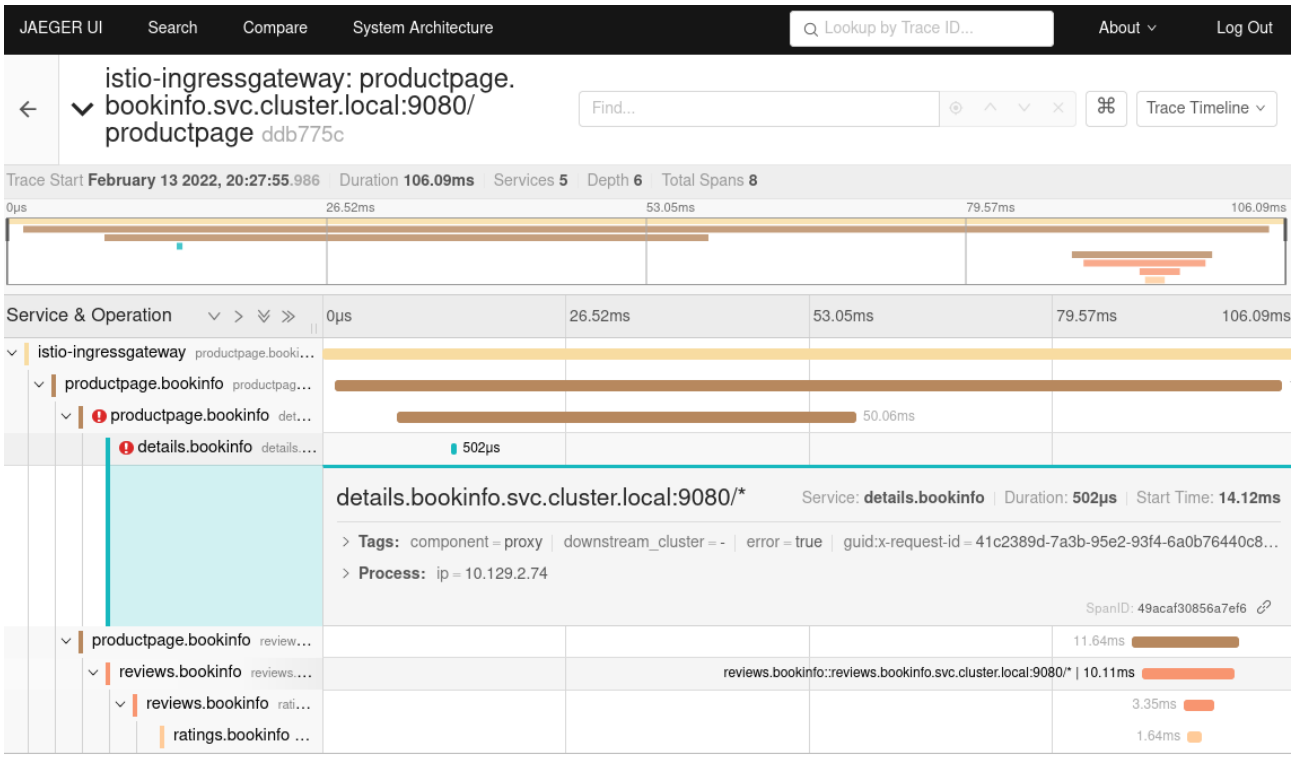


Figura 8.21: un rastro de llamada

Esta vista muestra un rastro de llamada de una manera similar a la de Kiali pero en una vista jerárquica. Cada una de las filas se puede expandir para mostrar detalles sobre la duración de la solicitud, el tiempo de inicio, la dirección IP fuente, entre otros. Este nivel de información se vuelve prácticamente fundamental cuando trabajamos con aplicaciones de microservicios complejas.

Si volvemos al error que tratábamos de diagnosticar, la vista del rastreo muestra claramente que el microservicio `details` tiene problemas. En este caso, el error es claro, pero si expandimos la vista, podemos ver que se están emitiendo códigos de error `HTTP 503`.

Figura 8.22: detalles del error

HTTP 503 es el código de error estándar para "servicio no disponible". En esta instancia, esto se debe a que el servicio se ajusta a cero réplicas. Un aumento restaurará el servicio.

Si combina las herramientas Jaeger y Kiali, obtendrá mucha potencia, lo cual es útil cuando las aplicaciones de microservicios se vuelven más complejas. Están incluidas en Azure Red Hat OpenShift y no suponen costos ni suscripciones adicionales.

Resumen

Este capítulo abordó algunos de los servicios clave con valor agregado que ofrece RHOS en comparación con solo un entorno de Kubernetes "vacío". Si bien es posible replicar un nivel similar de funcionalidad de OpenShift al instalar todos los proyectos open source upstream correspondientes, la integración, el ciclo de vida y el soporte son la clave de la complejidad que ofrece Azure RHOS.

Gracias a estos servicios de plataforma, de aplicaciones, de datos, de desarrollo y de clústeres, la productividad de los desarrolladores y operadores aumenta cuando trabajan con Kubernetes y cuando implementan varias aplicaciones empresariales complejas.

Capítulo 9

Integración con otros servicios

Es normal que una aplicación no pueda existir en su totalidad dentro de Red Hat OpenShift, ya que la mayoría depende de bases de datos o servicios en Azure de alguna forma u otra.

Azure Red Hat OpenShift no "rompe" la conectividad de ninguna manera en este caso. Si una aplicación depende de Azure CosmosDB, por ejemplo, aún debería poder conectarse fuera de RHOS a CosmosDB sin sufrir ningún cambio. Según la aplicación o la empresa, puede implementar algunas de estas dependencias externas por separado o al mismo tiempo.

Si piensa que obtendrá ventajas con esta tarea, entonces Azure Service Operator puede simplificar mucho el proceso. En lugar de usar la CLI de Azure, Azure Cloud Shell o plantillas de ARM, puede implementar estos recursos como si fueran de Kubernetes con Azure Service Operator.

Azure Service Operator

Azure Service Operator es otro operador que facilitará su vida, independientemente de que sea el desarrollador o el administrador que implementa aplicaciones en Azure Red Hat OpenShift. Una ventaja es que permite que se preparen los servicios de Azure de manera sencilla y sin la necesidad de salir de la consola de OpenShift, lo cual acelera y facilita el proceso, el cual puede incluir dependencias de servicios de Azure, como Azure CosmosDB.

En segundo lugar, permite que esas dependencias en servicios de Azure se almacenen junto con la definición de la aplicación de OpenShift, con un enfoque de infraestructura como código y usando archivos estándares de Kubernetes YALM.

Funciona de una manera muy sencilla: Azure Service Operator busca **CRD** nuevas, definidas en YALM, que coincidan con la definición de un recurso en Azure. Luego, este servicio traduce YALM en las llamadas a las API de Azure necesarias para crear lo que se pida. Con el operador ya instalado, los usuarios pueden explorar el catálogo del desarrollador de OpenShift y ver la pantalla de creación de muchos de los recursos comunes de Azure, como las direcciones IP públicas, las bases de datos SQL o Azure Firewall.

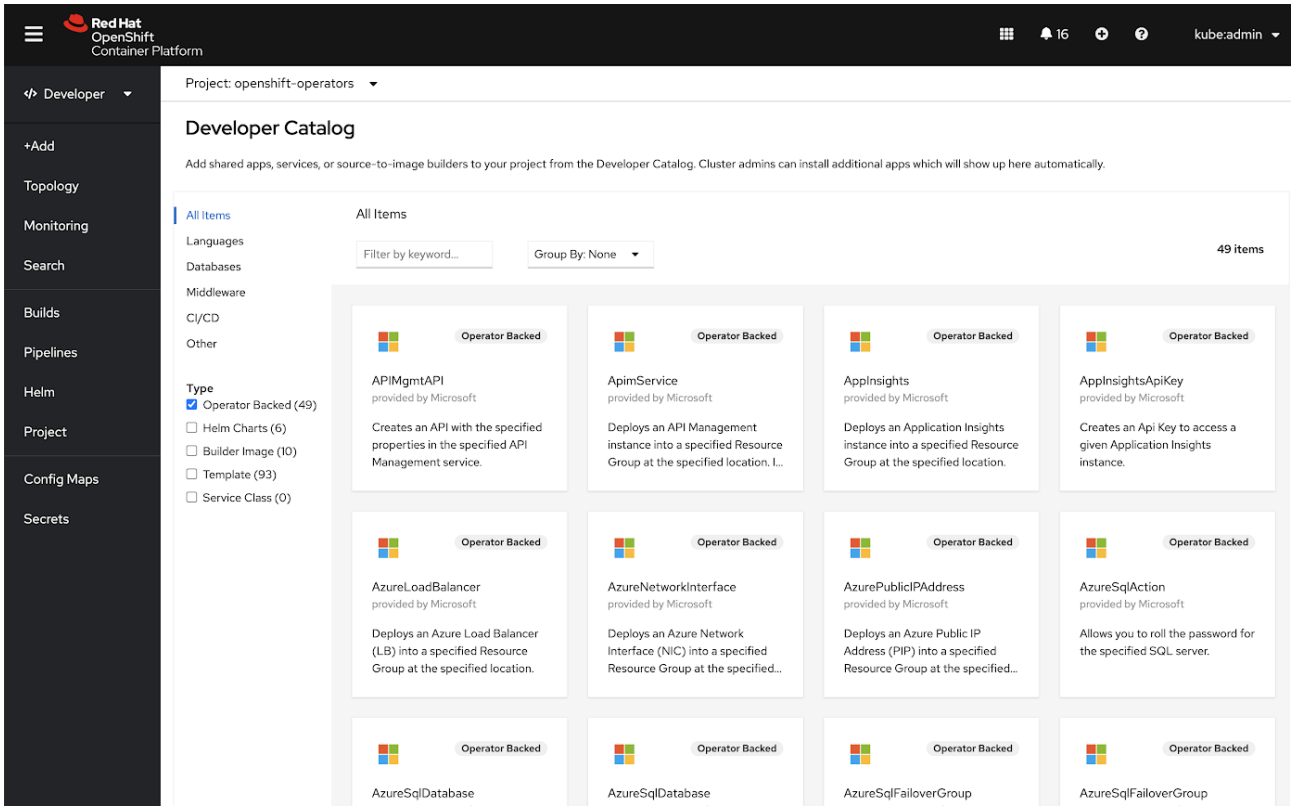


Figura 9.1: algunos de los servicios de Azure expuestos en Azure Service Operator

Azure Service Operator se puede instalar mediante OperatorHub y puede estar disponible para todos los usuarios en OpenShift.

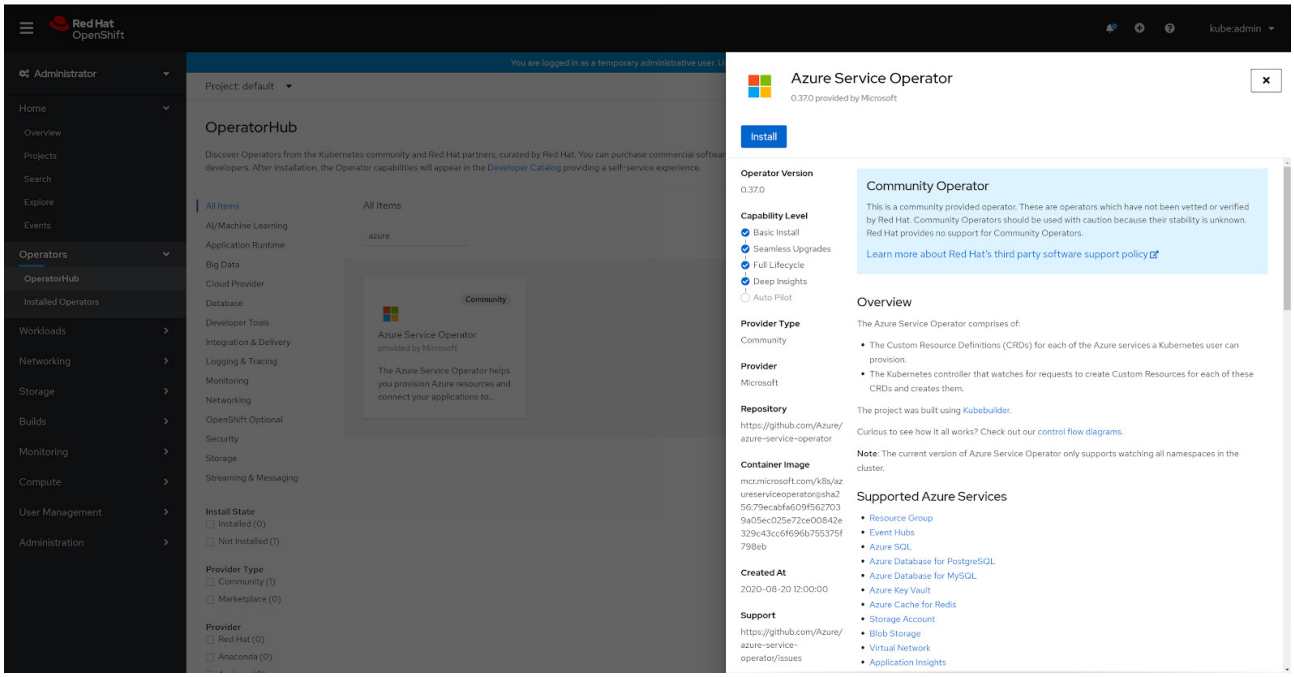


Figura 9.2: la pantalla de instalación de Azure Service Operator, con la galería de OperatorHub en el fondo

El uso de Azure Service Operator no es obligatorio para los servicios que se ejecutan en Azure, y es importante entender que los servicios subyacentes se implementan en esta plataforma y no en OpenShift. Sin embargo, Azure Service Operator acelera y facilita este proceso para aquellas aplicaciones que tienen dependencias del servicio de Azure.

Un caso de uso conocido es la implementación de bases de datos dependientes junto con la aplicación. Por ejemplo, para una aplicación web que usa una instancia de Azure CosmosDB, debe implementar Azure Service Operator como parte de esta tarea en OpenShift. Este producto incluye soporte para Azure SQL Database, Azure Database for MySQL, Azure PostgreSQL, entre otros.

Asumiendo que ya está instalado, el siguiente manifiesto de Kubernetes se puede almacenar con la aplicación de OpenShift y se puede usar para preparar la base de datos MySQL:

Fuente: tomado del repositorio de muestras de Azure Service Operator

```
apiVersion: azure.microsoft.com/v1alpha1
kind: MySQLServer
metadata:
  name: aso-wpdemo-mysqlserver
spec:
  location: eastus2
  resourceGroup: aso-wpdemo-rg
  serverVersion: "8.0"
  sslEnforcement: Disabled
  minimalTLSVersion: TLS10 # Possible values include: 'TLS10', 'TLS11', 'TLS12', 'Disabled'
  infrastructureEncryption: Enabled # Possible values include: Enabled, Disabled
  createMode: Default # Possible values include: Default, Replica, PointInTimeRestore (not
implemented), GeoRestore (not implemented)
  sku:
    name: GP_Gen5_4 # tier + family + cores eg. - B_Gen4_1, GP_Gen5_4
    tier: GeneralPurpose # possible values - 'Basic', 'GeneralPurpose', 'MemoryOptimized'
    family: Gen5
    size: "51200"
    capacity: 4
```

No sería común usar Azure Service Operator para servicios que comprenden dependencias complejas. Las herramientas tales como las plantillas de ARM de Azure o Bicep serían más adecuadas en estos casos.

Para obtener más información sobre Azure Service Operator, consulte las siguientes publicaciones de blogs:

- Publicación del blog de septiembre de 2020
- Azure Service Operator on OperatorHub.io
- Azure Service Operator on GitHub

Integración con Azure DevOps

Junto con OpenShift Pipelines, muchos usuarios querrán integrar Azure RHOS con Azure DevOps. Estas soluciones pueden existir juntas sin problemas, y algunos proyectos utilizarán una solución y otros, una distinta. Incluso, en algunos casos, podrían usar ambas. La elección de una solución u otra depende de algunos factores.

En términos generales, Azure DevOps brinda un nivel alto de integración con otras herramientas de Azure, y puede implementar con facilidad en OpenShift, así como en otros recursos informáticos.

Por otro lado, OpenShift Pipelines es una oferta bien integrada que viene con OpenShift y proporciona una experiencia multicloud uniforme.

Azure DevOps trata a OpenShift como cualquier otro clúster de Kubernetes. Por lo tanto, todas las interfaces y las API de Kubernetes comunes trabajarán como se espera que lo hagan.

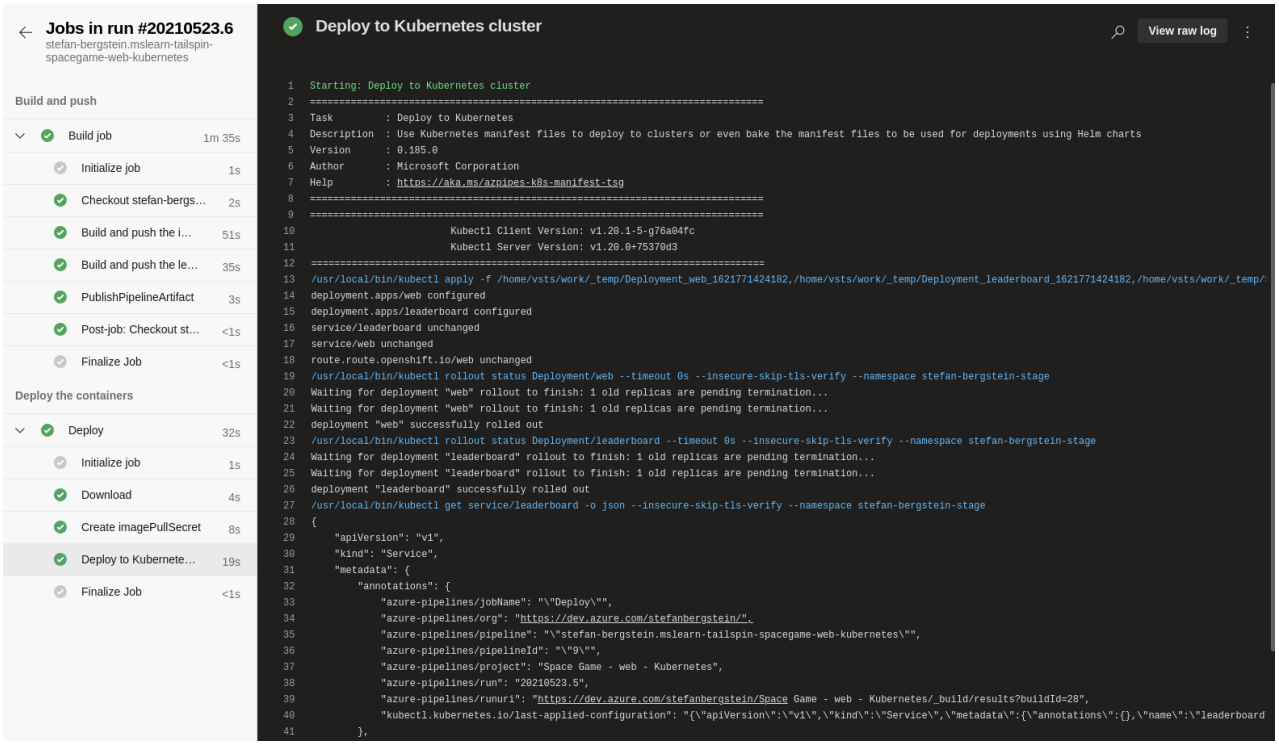


Figura 9.3: un canal de Azure DevOps enviando contenido a OpenShift

Más información

En la siguiente publicación del blog de la comunidad, encontrará un tutorial estupendo para comenzar con Azure Red Hat OpenShift y Azure DevOps:

- Publicación del blog: mayo de 2021

Integración con Visual Studio Code

Como parte de la valiosa integración para los desarrolladores que ofrece OpenShift, vienen plugins para muchos IDE conocidos, incluido Visual Studio Code. De esta forma, los desarrolladores y los administradores pueden acceder a los recursos de Kubernetes y OpenShift con facilidad y rapidez, sin tener que dejar de usar su IDE.

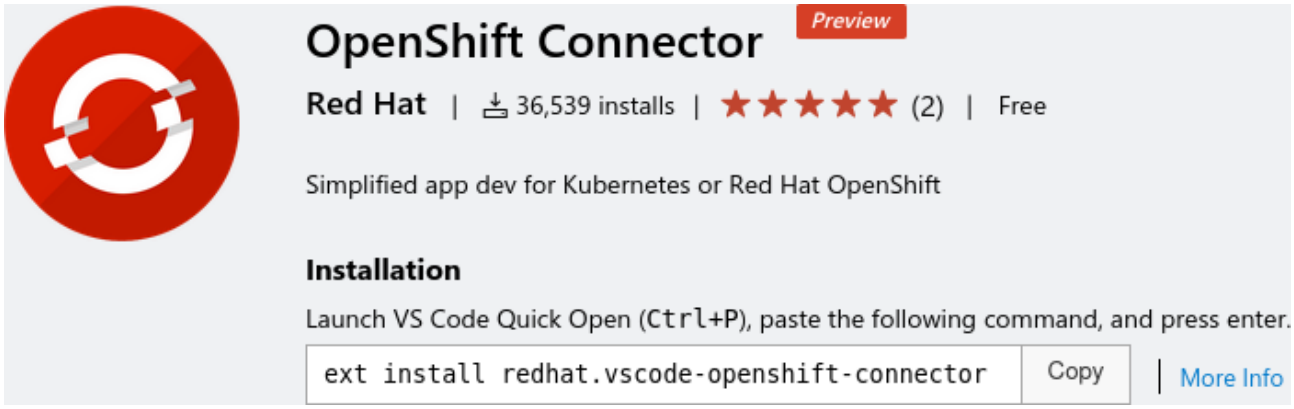


Figura 9.4: instalación de OpenShift Connector

Una vez que descargue el conector y lo instale en Visual Studio Code, deberá iniciar sesión en el clúster de OpenShift. A continuación, se detalla un primer proyecto sencillo del estilo "Hola, mundo", con una conexión a Azure Red Hat OpenShift:

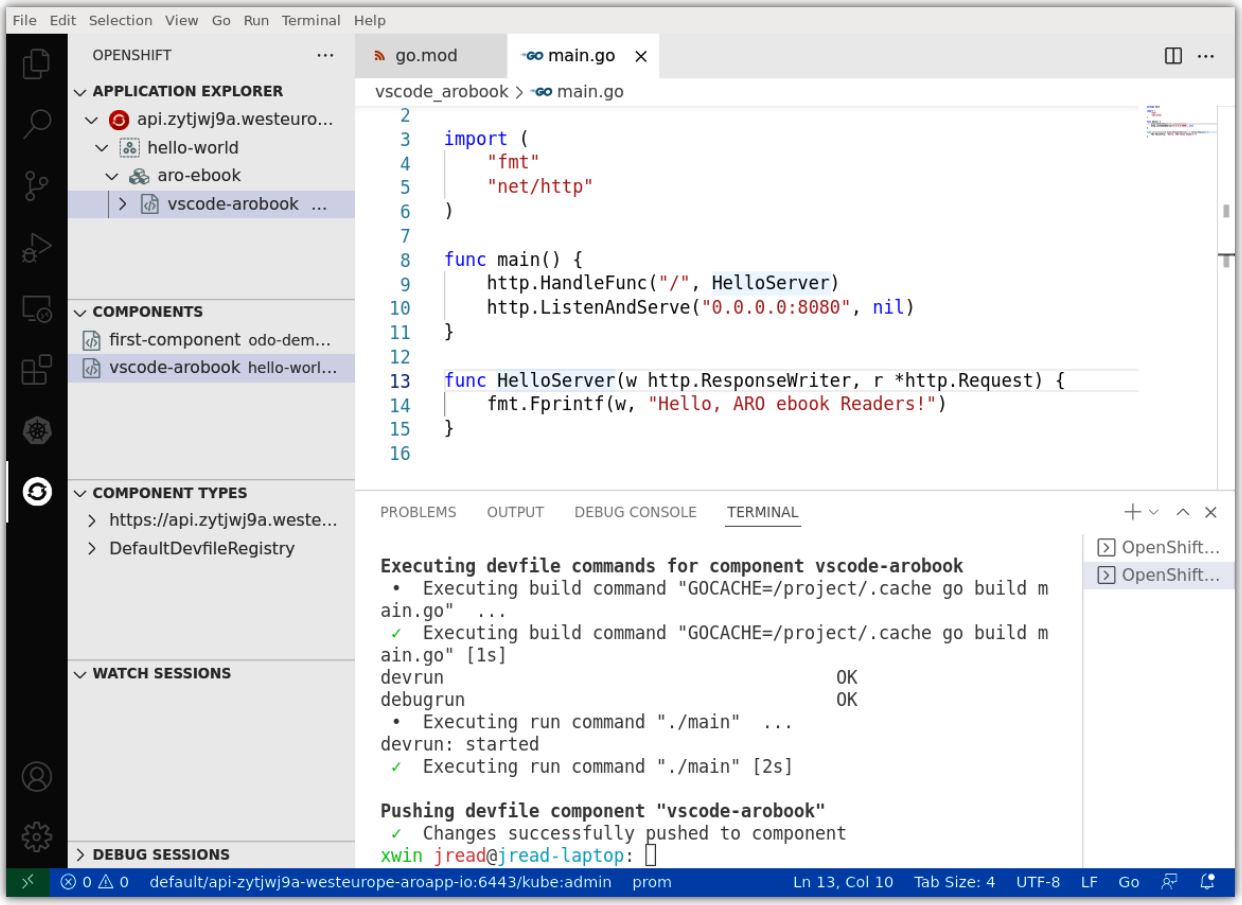


Figura 9.5: un proyecto Golang recién implementado con OpenShift Connector en Visual Studio Code

Cuando utiliza OpenShift Connector con Visual Studio Code, tiene como beneficio un frontend gráfico para la popular herramienta "OpenShift Do" (conocida comúnmente como "odo"). De esta forma, los desarrolladores pueden pensar en conceptos más complejos y no solo en los componentes de Kubernetes sin procesar. No necesitan preocuparse tanto en los controladores como Deployment o ReplicaSet y ese tipo de detalles. En la captura de pantalla anterior, observe un proyecto sencillo con un servicio de HTTP expuesto.

Además, el conector también incluye la posibilidad de enviar los cambios de código directamente a OpenShift, sin tener que primero utilizar el control de versiones sí o sí. Notará que es muy útil para desarrollar contenedores con mayor velocidad, ya que solo debe diseñarlos e implementarlos, sin tener que enviarlos al control de versiones.

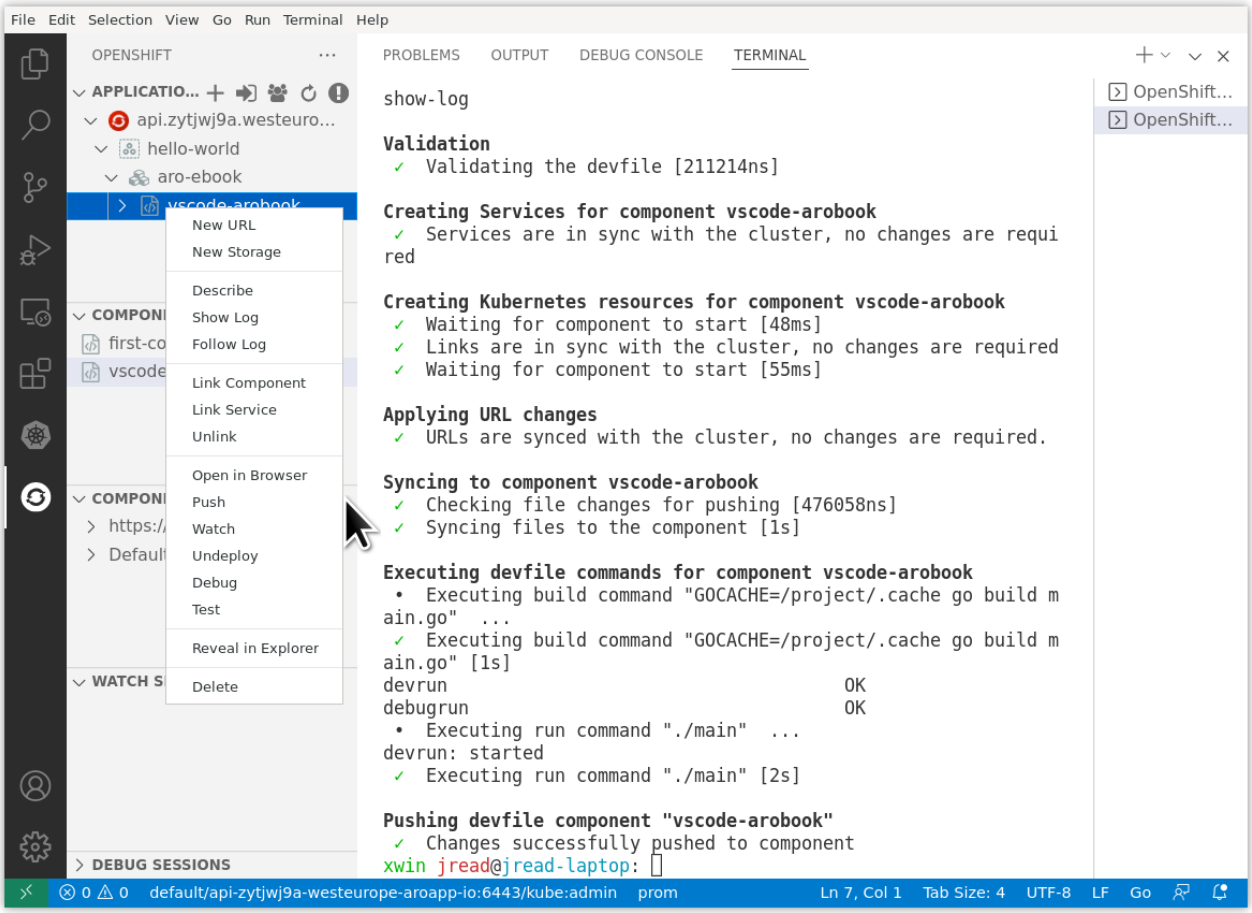


Figura 9.6: el menú para enviar cambios en un proyecto y un cambio reciente en la ventana de la terminal

Básicamente, este conector acelera los ciclos de desarrollo y prueba para los desarrolladores que prefieren utilizar Visual Studio Code.

Figura 9.7: la aplicación básica de Golang que se ejecuta en OpenShift

Obviamente, los desarrolladores no están obligados a usar solo Visual Studio Code. De hecho, muchos trabajan con Vim, Eclipse u otros editores. Sin embargo, este es muy popular entre los que prefieren una experiencia de "IDE ligero".

Más información

- Demostración en video: cómo usar Visual Studio Code con OpenShift
- OpenShift Connector en Visual Studio Code

Integración con GitHub Actions

Ahora puede utilizar GitHub Actions para realizar implementaciones en cualquier entorno de Red Hat OpenShift, incluido Azure Red Hat OpenShift. Desde cualquier repositorio de GitHub, diríjase a **Actions** (Acciones)→ **New Workflow** (Flujo de trabajo nuevo) y seleccione **OpenShift** de la lista de acciones disponibles.

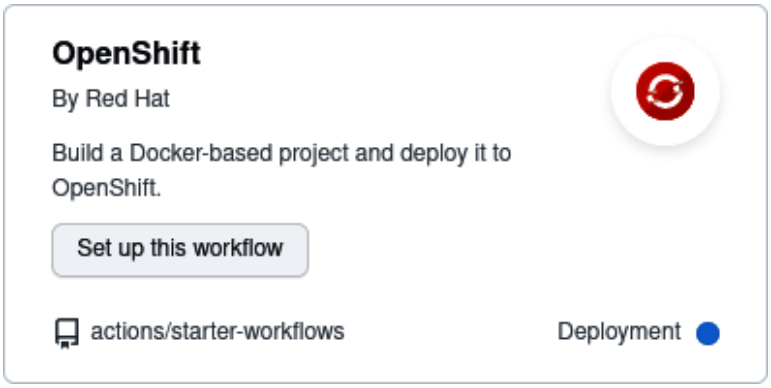


Figura 9.8: implementaciones en OpenShift desde GitHub

La plantilla predeterminada viene con muchas acciones de ejemplo muy buenas y solo necesita que se configuren algunas variables del entorno para permitir la conexión con un clúster de OpenShift:

```
name: OpenShift

env:
  # ✏ EDIT your repository secrets to log into your OpenShift cluster and set up the context.
  # See https://github.com/redhat-actions/oc-login#readme for how to retrieve these values.
  # To get a permanent token, refer to https://github.com/redhat-actions/oc-login/wiki/Using-a-
Service-Account-for-GitHub-Actions
  OPENSHIFT_SERVER: ${{ secrets.OPENSHIFT_SERVER }}
  OPENSHIFT_TOKEN: ${{ secrets.OPENSHIFT_TOKEN }}
  # ✏ EDIT to set the kube context's namespace after login. Leave blank to use your user's
default namespace.
  OPENSHIFT_NAMESPACE: ""
  ...
```

A continuación, brindamos dos recursos útiles: una publicación de blog muy buena que describe cómo realizar estas implementaciones, así como una demostración en video.

- Publicación de blog
- Demostración en video de GitHub Actions con OpenShift

Resumen

En este capítulo, abordamos muchas de las integraciones comunes que hemos visto con los clientes que usan Azure Red Hat OpenShift. Como se detalló en la introducción del capítulo, con las API estándares de OpenShift, puede integrar muchos más servicios que no se mencionan aquí. Lo mismo sucede si utiliza los operadores enumerados en OperatorHub.

En el próximo capítulo, compartiremos algunas prácticas recomendadas y sugerencias sobre el modo de incorporar los equipos de aplicaciones y ganar terreno en la adopción del servicio dentro de su empresa.

Capítulo 10

Incorporación de las cargas de trabajo y los equipos

Una vez que haya realizado los pasos *previos a la implementación*, haya implementado Azure Red Hat OpenShift y haya terminado las tareas posteriores necesarias, todo lo que queda es asistir a los equipos de desarrolladores y de aplicaciones e incorporarlos al clúster. La forma en que lo lleve a cabo dependerá de la cultura de su empresa básicamente: a algunos equipos les gusta "lanzarse al vacío" y dar los primeros pasos, mientras que otros prefieren recibir cierta orientación.

Este capítulo incluye un conjunto de listas de verificación que brindan una buena base y sirven para asegurarse de que sus equipos puedan comenzar a trabajar más rápido cuando usen Azure Red Hat OpenShift.

Tenga en cuenta que puede ampliar estas listas de verificación según la estructura y la cultura de su empresa.

Lista de verificación: pasos previos a la incorporación

Antes de incorporar una carga de trabajo nueva, es importante transmitirle al equipo de aplicaciones, o a los propietarios de esa carga, que Azure Red Hat OpenShift se implementó y diseñó de forma tal que ya cumple con las prácticas recomendadas de su empresa:

- Se implementó en una zona de aterrizaje de Azure o en otro de sus entornos aprobado para las aplicaciones de la empresa.
- El clúster tiene toda la configuración del "día 2" necesaria, como las clases de almacenamiento y la autenticación (se describen en el *Capítulo 6, Etapa posterior a la implementación: el día 2*).
- Su equipo de plataformas configura el clúster por completo y le proporciona soporte. Además, cuenta con el respaldo del soporte integrado de Microsoft y Red Hat.
- El clúster se actualiza a la versión estable más reciente y se le aplican los parches adecuados, lo cual se seguirá haciendo en adelante.

- El clúster de Azure Red Hat OpenShift tiene conectividad con los recursos esenciales necesarios:
  - Con el entorno en las instalaciones, a través de Azure ExpressRoute o una VPN
  - Con un repositorio de artefactos empresariales (como uno de Maven de Java)
  - Con el Internet público, para descargar las dependencias durante el diseño de una aplicación

Incorporación de las cargas de trabajo piloto

Incorpore al menos dos cargas de trabajo piloto como patrón común para lanzar Azure Red Hat OpenShift en una empresa:

- **La mínima carga de trabajo importante**: lo ideal como equipo de SRE es que la primera carga piloto sea una aplicación pequeña y conocida con requisitos técnicos muy sencillos. Sería una aplicación un poco más compleja que aquella de estilo "Hola, mundo", ya que normalmente debería tener un propietario comercial real dentro de la empresa. Sin embargo, con la primera carga, se procura revisar que el clúster de Azure Red Hat OpenShift satisfaga las necesidades empresariales al nivel del clúster: la conectividad de Azure, el inicio de sesión de Azure Active Directory y la identificación de los problemas más sencillos (temas comunes que probablemente se encuentran con cualquier aplicación).
- **Una carga de trabajo importante de referencia**: una vez que se implementa una mínima carga de trabajo sencilla, realmente se puede concluir su adopción si la segunda carga es lo suficientemente compleja e importante o, incluso, esencial para el negocio. Esta carga ayudará a identificar y resolver problemas tales como la conectividad con las bases de datos importantes, las pruebas de rendimiento y los registros/indicadores según sea necesario. Más importante aún, este tipo de carga de trabajo se puede utilizar como **referencia interna** dentro de su empresa. De esta forma, los futuros equipos de desarrollo y aplicaciones podrán adquirir confianza en la plataforma de Azure Red Hat OpenShift.

Obviamente, hay otros patrones para incorporar las primeras cargas de trabajo que podrían funcionar en su empresa. Quizá sea necesario apuntar a las aplicaciones de un centro de datos que se eliminará dentro de poco o a las que se ejecutan en un servidor Java antiguo.

Lista de verificación: reunión de incorporación con otros equipos

A la hora de incorporar un nuevo equipo de desarrollo o de aplicaciones en el clúster, una buena práctica es realizar una reunión de incorporación:

- Cree uno o varios espacios de nombre para que use el equipo.
- Comparta con el equipo una breve demostración sobre Red Hat OpenShift y destaque la funcionalidad clave de realizar implementaciones desde GitHub (o equivalente).
- Compruebe que el clúster tenga suficiente capacidad para implementar la carga de trabajo objetivo (CPU, RAM, almacenamiento, etc.).
- Verifique que los miembros del equipo puedan iniciar sesión en el clúster: la conectividad con Azure Active Directory facilita esta tarea.
- Comunique los detalles de contacto del equipo de SRE (o similar) que gestiona el clúster.
- Proporcione una lista con los enlaces para comenzar a usar OpenShift:
    - http://learn.openshift.com
    - https://github.com/openshift-labs/starter-guides
    - http://docs.openshift.com

Lista de verificación: llamadas regulares para la comprobación de estado

Una vez que el equipo de desarrollo y de aplicaciones haya comenzado la implementación en el clúster, organice reuniones regulares para comprobaciones de estado. Pueden ser parte del encuentro diario o pueden ser 30 minutos semanales, con eso alcanza. Aquí tiene una lista de los temas que quizá quisiera abordar:

- ¿Cómo le ha ido al equipo en general? ¿Se implementó alguna aplicación?
- ¿Ha habido algún problema reciente en el uso del clúster, ya sea de conectividad o de conocimiento de Red Hat OpenShift?
- ¿Ha habido alguna interrupción en Azure o con el clúster que haya perjudicado la experiencia?
- Haga un recordatorio sobre la disponibilidad y los detalles de contacto del equipo de SRE para responder las preguntas.

Puede pensar en algún otro punto que ya haya abordado en llamadas regulares de comprobación de estado para proyectos similares. Agregue los ítems que usted crea que pueden funcionar en esta lista de verificación.

Patrones que se deben evitar: entornos de prueba (sandbox)/sitios de prueba de los desarrolladores

Si quiere fomentar la adopción entre los desarrolladores de su empresa, debe evitar el patrón común de ofrecerles un entorno de prueba (sandbox) y esperar que los equipos de desarrollo y de aplicaciones comiencen a adoptar Azure Red Hat OpenShift. Este entorno de prueba también es conocido como "sitio de prueba de los desarrolladores". Sin no se cuenta con ningún otro recurso ni soporte de seguimiento, Azure Red Hat OpenShift puede ser un entorno intimidante con mucha complejidad para asimilar. En la mayoría de los casos, adoptar un patrón de "sandbox" puede generar un gasto desaprovechado en el cloud computing y clústeres vacíos.

Sin embargo, si su empresa tiene experiencia en este tipo de entornos con los equipos de desarrollo, aquí le brindamos algunos consejos para probar este entorno y lograr el éxito:

- Ofrezca, al menos, una breve demostración de lo que se puede hacer con Azure Red Hat OpenShift.
- Comparta algunos documentos y los enlaces para comenzar que se mencionan arriba.
- Organice un evento estilo "hackatón" y desafíe a los desarrolladores a diseñar algo con OpenShift en una pequeña competencia.
- Ofrezca la opción de brindar un soporte ampliado, una introducción al equipo de SRE y una incorporación con mayor orientación.

Si sigue una lista de verificación de patrones en este proceso guiado, como se detalla anteriormente, es muy probable que se concluya la adopción de la plataforma.

Azure Red Hat OpenShift Developer Workshop

Azure Red Hat OpenShift Developer Workshop (https://aroworkshop.io) es un recurso útil creado previamente que puede utilizar dentro su empresa para brindar una experiencia guiada y práctica de Azure Red Hat OpenShift. El taller ofrece dos experiencias de laboratorio, diseñadas para que un desarrollador o un operador que no conoce OpenShift tenga un tutorial guiado. Ambas experiencias destacan la funcionalidad clave de OpenShift y demuestran cómo se puede usar. La experiencia de laboratorio nro. 1 es una introducción al diseño moderno de microservicios y la nro. 2 abarca los aspectos internos del servicio.

Use el contenido de aroworkshop.io en su clúster interno de Azure Red Hat OpenShift si quiere dar a conocer esta herramienta dentro de su empresa. Antes del taller, describa la manera en que se ha implementado Azure Red Hat OpenShift dentro de la suscripción actual de su empresa. Además, explique que el uso de ese servicio dentro del taller podría asemejarse a la experiencia de utilizar Azure Red Hat OpenShift para alojar las aplicaciones de producción.

Resumen

En este capítulo, se le brindaron algunas listas de verificación útiles y cierta orientación para incorporar con éxito las cargas de trabajo y los equipos en Azure Red Hat OpenShift. Se describió un patrón común que se debe evitar: los clústeres de "sandbox" sin soporte. No espere que una guía incluya una lista completa de los recursos y los ítems de listas de verificación que se pueda aplicar a todas las empresas. Por eso, es importante que adapte los contenidos de este capítulo a sus propias necesidades.

La nube ofrece una serie atractiva de servicios. Sin embargo, muchos pasan inadvertidos o se usan mal, porque las empresas se centran mucho en la tecnología y su implementación. Si bien es importante comprender esos temas, solo comprenden una pequeña parte de la ecuación cuando se trata de usar ese servicio en la producción y adoptarlo con eficiencia. Este capítulo tuvo como objetivo destacar la importancia de las capacitaciones, las reuniones regulares y las actividades similares si desea adoptar Azure Red Hat OpenShift con éxito.

Capítulo 11

Conclusión

Los equipos de desarrollo y operaciones pueden destinar la mayor parte de su día laboral a la preparación, la configuración, el mantenimiento y la supervisión de los clústeres y el canal de CI/CD. Si es así, no pueden dedicar su valioso tiempo a lo que mejor hacen: mantener las aplicaciones a la vanguardia.

Como hemos aprendido en esta guía, Azure Red Hat OpenShift le permite implementar los clústeres de Red Hat OpenShift completamente gestionados sin tener que preocuparse por diseñar o administrar la infraestructura que se necesita para ejecutarlos. Hemos visto que ejecutar Kubernetes solo tiene algunas limitaciones, principalmente en relación con la parte práctica extra que se necesita con las tareas que podrían automatizarse con Azure Red Hat OpenShift.

A la hora de elegir la estrategia adecuada para la gestión de clústeres en su empresa, tenga en cuenta las ventajas y las desventajas que presenta una plataforma del tipo de Kubernetes en comparación con una como Azure Red Hat OpenShift. Esta última está diseñada en el marco de Kubernetes y le ofrece más beneficios.

Para obtener más información sobre Azure Red Hat OpenShift, visite la página del producto o revise la sección de los documentos. También puede considerar hacer un taller práctico y registrarse para ver el webinar cuando más le convenga. Lo más importante es que acuda a Microsoft y Red Hat para que lo asistan a la hora de evaluar Azure Red Hat OpenShift.

Autores y versiones

James Read <james@redhat.com>
Arquitecto principal de Soluciones en Red Hat,
cubriendo Microsoft, versión actualizada y revisada de AROv4

Ahmed Sabbour <asabbour@microsoft.com>
Gerente principal de Marketing de productos en Microsoft,
cubriendo Azure Red Hat OpenShift, versión inicial de AROv3

Autores de ediciones anteriores

Oren Kashi <okashi@redhat.com>
Gerente principal de Marketing de productos técnicos en Red Hat

Gracias a Brooke Jackson, Nermina Miller, Jose Moreno, Ahmed Sabbour, Aditya Datar, Vince Power, Alex Patterson y otras personas que amablemente ofrecieron su tiempo y sus comentarios para mejorar esta guía.

Capítulo 12

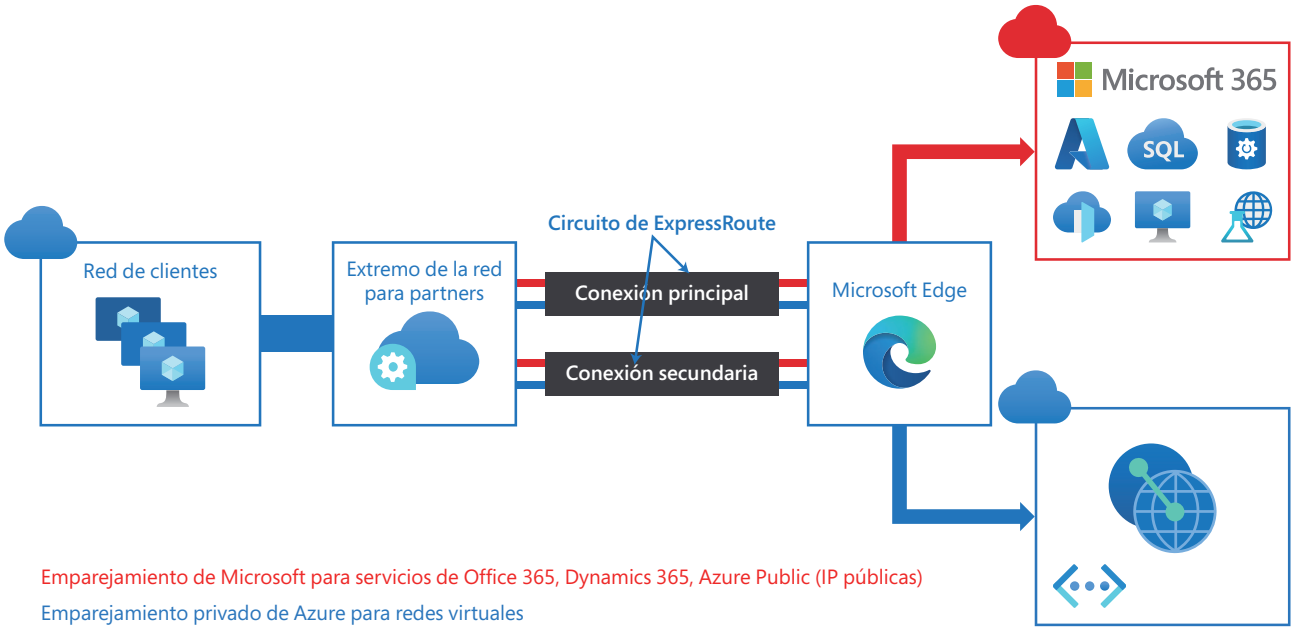
Glosario

Este glosario sirve como referencia rápida y útil para comprender algunos de los términos que se usan en esta guía y en el ecosistema de Azure Red Hat OpenShift. Los títulos se ordenan de forma alfabética.

Azure ExpressRoute

ExpressRoute le permite ampliar sus redes en las instalaciones a la nube de Microsoft a través de una conexión privada y con la ayuda de un proveedor de conectividad. Con esta tecnología, puede establecer conexiones con los servicios de nube de Microsoft, como Microsoft Azure y Microsoft 365.

A continuación, podrá ver un diagrama de red de ExpressRoute, extraído de la documentación de Microsoft Azure:



Para obtener más información sobre ExpressRoute, diríjase a la página sobre ExpressRoute en la documentación de Microsoft Azure.

Comprenda cómo funciona ExpressRoute con Azure en el *Capítulo 4: Etapa previa a la implementación: preguntas sobre la arquitectura empresarial.*

Compilaciones y flujos de imágenes

La compilación consiste en transformar los parámetros de entrada en el objeto resultante. Generalmente, se utiliza para convertir estos parámetros o el código fuente en una imagen ejecutable. El objeto BuildConfig es la definición del proceso de compilación completo.

Para usar Kubernetes, Azure Red Hat OpenShift crea contenedores formateados de Docker a partir de las imágenes de la compilación y los envía a un registro de imágenes de contenedores.

Los objetos de la compilación tienen varias características en común: las entradas para la compilación, la necesidad de finalizar el proceso de creación, el registro de este proceso, la publicación de los recursos de las que funcionen correctamente y la publicación del estado final de ellas. Las compilaciones aprovechan las restricciones de los recursos y delimitan el uso de la CPU y la memoria, así como del tiempo de ejecución de los pods y de sí mismas.

El sistema de compilación de Azure Red Hat OpenShift ofrece compatibilidad ampliable para las estrategias relativas a este proceso, las cuales se basan en los tipos elegibles que se indican en su API: Hay tres principales disponibles:

- **Compilación de Docker:** utiliza un Dockerfile, el cual se puede cargar en un repositorio fuente o extraer de él.
- **Compilación Source-to-Image (S2I):** aprovecha un repositorio fuente, como Git, y define el proceso para diseñar la aplicación a partir de archivos de compilación conocidos, como los `.pom` de Marven para los proyectos Java.
- **Compilación personalizada**

Las compilaciones de Docker y S2I son compatibles de forma predeterminada.

Según el compilador que se haya usado, variará el tipo de objeto que se cree. En el caso de las compilaciones de Docker y S2I, serán imágenes ejecutables, y en el de las personalizadas, será el resultado de lo que haya indicado quien creó la imagen del compilador.

Contenedor

Las unidades básicas de las aplicaciones de Azure Red Hat OpenShift se denominan contenedores. Estas tecnologías de Linux son mecanismos ligeros que permiten aislar los procesos en ejecución para que se comuniquen únicamente con los recursos que se les han asignado.

Es decir, aún si se ejecutan muchas instancias de aplicaciones en los contenedores de un único host, no compartirán información sobre sus procesos, archivos, redes y otros elementos necesariamente. Aunque los contenedores puedan utilizarse para cargas de trabajo arbitrarias, normalmente cada uno de ellos ofrece un solo servicio, que suele denominarse microservicio, como un servidor web o una base de datos.

Imágenes de los contenedores

Los contenedores en Azure Red Hat OpenShift se basan en las imágenes de los contenedores formateados de Docker. Una imagen es un código binario que incluye todos los requisitos para ejecutar un solo contenedor, así como los metadatos que indican sus necesidades y funciones.

Imagínelo como una tecnología empaquetada. Los contenedores solo tienen acceso a los recursos definidos en la imagen, a menos que especifique otros adicionales cuando los crea. Al implementar la misma imagen en varios contenedores en diversos hosts y equilibrar la carga entre ellos, Azure Red Hat OpenShift puede proporcionar redundancia y capacidad de ampliación a través de la incorporación de equipos para un servicio empaquetado en una imagen.

Implementaciones y sus configuraciones

Azure Red Hat OpenShift se basa en los controladores de replicaciones, por lo que amplía la compatibilidad al ciclo de vida de desarrollo y de instalación del software con el concepto de las implementaciones. Uno de los ejemplos más sencillos es una implementación que crea un ReplicationController nuevo y permite que inicie los pods. Sin embargo, las implementaciones de Azure Red Hat OpenShift también permiten trasladarse de la instalación actual de una imagen a otra nueva y establecen si los enlaces se ejecutarán antes o después de crear el ReplicationController.

El objeto DeploymentConfig de Azure Red Hat OpenShift define estos detalles en una implementación:

1. Los elementos de una definición del ReplicationController
2. Los activadores para crear una nueva implementación de forma automática

3. La estrategia para llevar a cabo la transición entre las implementaciones
4. Los enlaces del ciclo de vida

Cada vez que se activa una implementación, ya sea de forma manual o automática, un pod del implementador la gestiona, incluso reduce la capacidad del ReplicationController antiguo, aumenta la del nuevo y ejecuta los enlaces. Luego de que finaliza la implementación, el pod perdura por tiempo indefinido para conservar los registros sobre ella. Cuando otra la reemplaza, se conserva el ReplicationController anterior para que sea sencillo restaurarla, en caso de ser necesario.

Si desea obtener instrucciones más detalladas sobre la creación de las implementaciones y la interacción con ellas, visite la página sobre las implementaciones y DeploymentConfigs.

Plantillas

La plantilla describe un conjunto de objetos que sirven como parámetro y se pueden procesar para generar la lista de objetos que creará Azure Red Hat OpenShift. La puede utilizar para crear cualquier elemento para el que cuente con permiso en el proyecto, como los servicios y las configuraciones de compilación y de implementación. La plantilla también puede determinar el conjunto de etiquetas que se aplicarán a cada objeto definido en ella.

Puede crear la lista de objetos desde la plantilla mediante la CLI o, si se encuentra cargada en el proyecto o en la biblioteca global de plantillas, mediante la consola web. Para obtener información sobre el grupo de plantillas adaptadas, consulte las bibliotecas de plantillas y los flujos de imagen de OpenShift.

Pods y servicios

Azure Red Hat OpenShift se basa en el concepto de Kubernetes sobre los pods, el cual explica que se trata de uno o más contenedores que se implementan juntos en un host, y de la unidad informática más pequeña que se puede definir, implementar y gestionar.

Los pods son equiparables a las instancias de una máquina (física o virtual) para un contenedor. A cada uno de ellos se les asigna su propia dirección IP interna, de forma tal que cuentan con la totalidad del espacio para el puerto, y los contenedores que se encuentran dentro suyo pueden compartir el almacenamiento local y las redes.

Los pods tienen un ciclo de vida: se los define, se les asigna un nodo para que se ejecuten en él, y lo hacen hasta que el contenedor sale o se lo elimina por algún otro motivo. Luego de que el pod sale, se lo eliminará o se lo conservará para acceder a sus registros sobre los contenedores, según su política y código de salida.

En Azure Red Hat OpenShift se considera que los pods son ampliamente inmutables; es decir que no se puede modificar su definición mientras están en ejecución. Para implementar cambios, el sistema borra el pod actual y lo recrea con una configuración distinta, con una imagen de base o con ambos. Como sus elementos del tiempo de ejecución se consideran desechables y se los crea nuevamente desde la imagen del contenedor definida, los usuarios no deberían gestionarlo directamente, sino que los controladores de nivel superior deberían encargarse de ello.

Proyectos y usuarios

Un proyecto es un espacio de nombre de Kubernetes con anotaciones adicionales y, además, el instrumento central desde el cual se gestiona el acceso de los usuarios frecuentes a los recursos. Permite que una comunidad de usuarios organice y administre su contenido de forma individual. Los administradores deben otorgarles acceso a los proyectos y, si tienen permitido crearlos, dispondrán de uno propio automáticamente. Los proyectos pueden tener un nombre, un `displayName` y una descripción distintos.

El nombre obligatorio es un identificador único para el proyecto que no puede superar los 63 caracteres y tiene mayor visibilidad cuando se utilizan las herramientas de la CLI o la API. El `displayName` opcional es la forma en que aparece el proyecto en la consola web (es `name`, de forma predeterminada). La descripción opcional puede contener una explicación más detallada sobre el proyecto, y también se puede ver en la consola web.

Los desarrolladores y los administradores pueden interactuar con el proyecto por medio de la CLI o la consola web.

Registro de los contenedores

Azure Red Hat OpenShift ofrece **OpenShift Container Registry (OCR)**, un registro integrado para las imágenes de los contenedores que incorpora la capacidad de implementar automáticamente nuevos repositorios de imágenes según se soliciten. Es una ubicación incorporada hacia donde las compilaciones de las aplicaciones que crean los usuarios pueden llevar las imágenes resultantes.

Cada vez que se envía una nueva imagen a OCR, el registro le informa a Azure Red Hat OpenShift sobre ella y le proporciona toda la información pertinente, como el espacio de nombre, el nombre y sus metadatos. Distintas partes del servicio gestionado aprovechan las imágenes recientes y crean compilaciones e implementaciones nuevas.

Azure Red Hat OpenShift también puede usar cualquier servidor que implemente la API del registro para las imágenes de los contenedores como una fuente de datos, incluso Docker Hub y Azure Container Registry.

ReplicaSets

Un ReplicaSet es parecido a un ReplicationController y se encarga de que determinada cantidad de réplicas del pod se ejecuten en un momento dado. Sin embargo, la diferencia entre ellos es que el ReplicaSet es compatible con los requisitos del selector basado en conjuntos, mientras que el ReplicationController solo admite los que se basan en la igualdad.

```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
    name: frontend-1
    labels:
        tier: frontend
spec:
    replicas: 3
    selector:
        matchLabels:
            tier: frontend
        matchExpressions:
        - {key: tier, operator: In, values: [frontend]}
    template:
        metadata:
            labels:
                tier: frontend
        spec:
            containers:
                - image: openshift/hello-openshift
                  name : helloworld
                  ports:
                  - containerPort: 8080
                    protocol: TCP
            restartPolicy: Always
```

En el código anterior puede observar:

- Una consulta de etiqueta sobre un conjunto de recursos. Los resultados de `matchLabels` y `matchExpressions` unidos de manera lógica.
- Un selector basado en la igualdad que especifica los recursos con etiquetas que coincidan con él.
- Un selector basado en grupos que filtra las claves, el cual selecciona todos los recursos cuya `key` sea igual a `tier` y cuyo `value` sea igual a `frontend`.

ReplicationController

Un ReplicationController se encarga de que una cantidad específica de réplicas de un pod se ejecuten en todo momento. Si los pods salen o se los elimina, creará nuevas instancias hasta alcanzar la cantidad estipulada. De igual manera, si hay una cantidad mayor que la esperada en ejecución, eliminará tantos como sea necesario para coincidir con el número especificado.

La configuración del ReplicationController está compuesta por estos elementos:

- La cantidad de réplicas esperadas (se puede configurar durante el tiempo de ejecución)
- La definición del pod que se utiliza cuando se lo replica
- El selector para identificar los pods gestionados
- El selector consiste en un conjunto de etiquetas asignadas a un pod y gestionadas por el ReplicationController. Están incluidas en la definición del pod sobre el cual el ReplicationController crea instancias, tarea en la que utiliza el selector para determinar la cantidad que ya están en ejecución y, de esta manera, llevar a cabo el reajuste necesario.

Dado que el ReplicationController no hace un seguimiento de la carga ni del tráfico, no puede llevar a cabo el ajuste automático basado en estos parámetros, sino que necesita una función externa que se encargue de adaptar la cantidad de réplicas.

El ReplicationController es uno de los objetos principales de Kubernetes, y a continuación se ejemplifica la definición de uno de ellos:

```
apiVersion: v1
kind: ReplicationController
metadata:
    name: frontend-1
spec:
    replicas: 1
    selector:
        name: frontend
    template:
        metadata:
            labels:
                name: frontend
        spec:
            containers:
                - image: openshift/hello-openshift
                  name: helloworld
                  ports:
                - containerPort: 8080
                  protocol: TCP
            restartPolicy: Always
```

En el código anterior puede observar:

- La cantidad de copias del pod que se debe ejecutar
- El selector de etiquetas del pod que se debe ejecutar
- Una plantilla para el pod que crea el controlador
- Las etiquetas del pod deben incluir las del selector
- El nombre no puede superar los 63 caracteres después de expandir los parámetros

Rutas y entradas

Azure Red Hat OpenShift es compatible tanto con las rutas como con las entradas. Ambas se utilizan para exponer un servicio con un nombre de DNS, como www.example.com, para que los clientes externos puedan encontrarlo.

Las rutas se diseñaron originalmente en Red Hat OpenShift 3, y, a partir de ese momento, Red Hat trabajó con la comunidad de Kubernetes para estandarizarlas en lo que actualmente conocemos como **Ingress**.

El recurso Ingress de Kubernetes en OpenShift Container Platform implementa el controlador de entrada con un servicio de enrutador compartido que se ejecuta como un pod en el clúster. El uso de este controlador es la forma más común de gestionar el tráfico entrante. Puede adaptar y replicar este pod tal como lo haría con cualquier otro. Este servicio de enrutador se basa en HAProxy, una solución open source de equilibrador de carga.

La ruta de OpenShift Container Platform envía tráfico entrante a los servicios que se encuentran en el clúster. Además, ofrece funciones avanzadas que posiblemente no sean compatibles con los controladores de entrada estándar de Kubernetes, como el nuevo cifrado y el acceso directo de la TLS y la división del tráfico para las implementaciones azul-verde.

El tráfico entrante accede a los servicios en el clúster a través de la ruta. Las rutas y las entradas son los recursos principales para gestionarlo. Ambas ofrecen funciones similares, como la admisión de las solicitudes externas y su delegación según la ruta. Sin embargo, las entradas solo admiten ciertos tipos de conexión, como HTTP/2, HTTPS y el **indicador del nombre del servidor (SNI)** y la TLS con certificado. En OpenShift Container Platform, las rutas se generan de forma tal que cumplan con las condiciones que se especifican en el recurso de entrada.

Source-to-Image (S2I)

S2I es un kit de herramientas y un flujo de trabajo para diseñar imágenes de contenedores reproducibles a partir del código fuente. Para generar las imágenes listas para ejecutarse, lo introduce en un contenedor, y este lo prepara para su ejecución. Dado que se crean imágenes del compilador que se autoorganizan, puede versionar y controlar los entornos de compilación tal como utiliza las imágenes del contenedor para versionar los del tiempo de ejecución.

En los lenguajes dinámicos como Ruby, los entornos de tiempo de ejecución y de compilación suelen ser iguales. S2I parte de una imagen del compilador que describe el entorno con Ruby, Bundler, Rake, Apache, GCC y otros paquetes necesarios para configurar y ejecutar una aplicación instalada de Ruby, y luego sigue estos pasos:

1. Inicia un contenedor desde la imagen del compilador con el código fuente de la aplicación incorporado en un directorio conocido.
2. El proceso del contenedor transforma este código en la configuración ejecutable adecuada. Particularmente en este caso, instala las dependencias con Bundler y lo traslada a un directorio donde Apache está preconfigurado para buscar el archivo `config.ru` de Ruby.
3. Confirma el contenedor nuevo y establece que el punto de entrada de la imagen sea un script (que proporciona la imagen del compilador), el cual iniciará Apache para que aloje a la aplicación de Ruby.

Para los lenguajes compilados, como C, G o Java, es posible que las dependencias que se necesitan para la compilación superen el tamaño de los equipos del tiempo de ejecución reales en gran medida. S2I habilita procesos de compilación de múltiples pasos para que las imágenes del tiempo de ejecución se mantengan compactas. Consisten en la creación de un equipo binario, como un ejecutable o un archivo WAR de Java, en la primera imagen del compilador, el cual se extrae posteriormente y se incorpora en la segunda imagen del tiempo de ejecución, que sencillamente lo ubica en el lugar correcto para que se ejecute.

Estos son los pasos que se deben seguir, a modo de ejemplo, para crear un canal reproducible de compilación para Tomcat (el conocido servidor web de Java):

1. Cree una imagen del compilador que contenga OpenJDK y Tomcat y prevea la incorporación de un archivo WAR.
2. Diseñe una segunda imagen que ubique a Maven y a otras dependencias estándar por encima de la primera imagen y prevea la incorporación del proyecto Maven en ella.
3. Solicite a S2I con el código fuente de la aplicación Java y la imagen de Maven para crear la aplicación WAR deseada.
4. Vuelva a solicitarlo con el archivo WAR del paso anterior y la imagen de Tomcat inicial para crear una imagen del tiempo de ejecución.

Si agregamos nuestra lógica de compilación en las imágenes y las combinamos en varios pasos, podemos mantener cerca los entornos de compilación y de tiempo de ejecución (el mismo JDK y los mismos JAR de Tomcat) y evitamos la implementación en producción de las herramientas de diseño.

Estas son las ventajas que obtiene al utilizar la estrategia de compilación S2I:

- **Replicación**: permite que los entornos de compilación se versionen de forma directa al concentrarlos en una imagen de contenedor y definir una interfaz sencilla (código fuente incorporado) para los autores de las llamadas. Las compilaciones replicables son uno de los requisitos clave para que las actualizaciones de seguridad y la integración continua sean posibles en las infraestructuras organizadas en contenedores. Además, las imágenes de los compiladores garantizan la replicación y la capacidad de intercambiar los tiempos de ejecución.
- **Flexibilidad**: todos los sistemas de compilación actuales que se pueden ejecutar en Linux, también lo pueden hacer dentro de un contenedor, y cada compilador individual también puede ser parte de un canal más grande. Los scripts que procesan el código fuente de la aplicación también se pueden incorporar en la imagen del compilador, por lo que las imágenes actuales se pueden adaptar después de su creación para que gestionen la fuente.
- **Velocidad**: S2I fomenta la representación de las aplicaciones en una capa de imagen única, en lugar de compilar varias capas en un solo Dockerfile. Esto permite ahorrar tiempo en las etapas de creación e implementación y tener un mayor control sobre el resultado de la imagen final.

- **Seguridad**: las compilaciones que utilizan Dockerfiles se ejecutan sin varios de los controles de operaciones normales, generalmente como superusuario y con acceso a la red de los contenedores. Dado que se lanzan en un solo contenedor, se puede usar S2I para manejar los permisos y los privilegios que tiene disponible la imagen del compilador. Este, en conjunto con plataformas como OpenShift, permite que los administradores controlen estrictamente los privilegios que tienen los desarrolladores al momento de hacer la compilación.

Trabajos

La finalidad de los trabajos es similar a la de los ReplicationController: ambos deben crear pods por motivos concretos. Sin embargo, los primeros están diseñados para los pods que se usarán una sola vez, mientras que los segundos son para los que se ejecutarán de forma permanente. El trabajo hace un seguimiento de las finalizaciones exitosas y, cuando se alcanza la cantidad especificada de estas, se completa.

Consulte el tema *Trabajos* para obtener más información acerca de su uso.

Zonas de aterrizaje de Azure

Las zonas de aterrizaje de Azure son un patrón de implementación común entre las empresas que planifican usar Azure para realizar una implementación a gran escala, teniendo en cuenta los ajustes, la seguridad, las redes y la identidad.

Introducción a las zonas de aterrizaje de Azure

En la actualidad, hay un proyecto comunitario de GitHub en desarrollo que ofrece algunas recomendaciones sobre la implementación de Azure Red Hat OpenShift en una arquitectura de zona de aterrizaje de Azure: https://github.com/Azure/Enterprise-Scale/tree/main/workloads/ARO