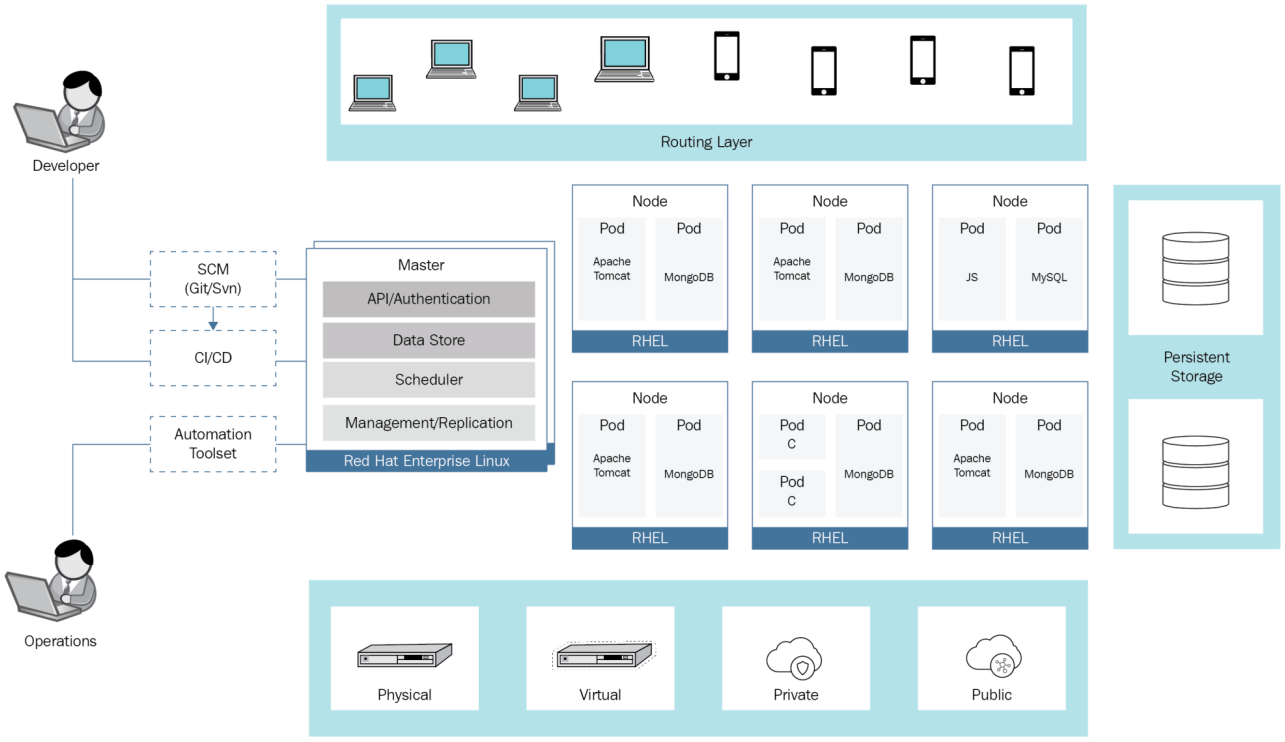
Workshop: DevOps con Red Hat Openshift Container Platform y Red Hat Ansible Automation

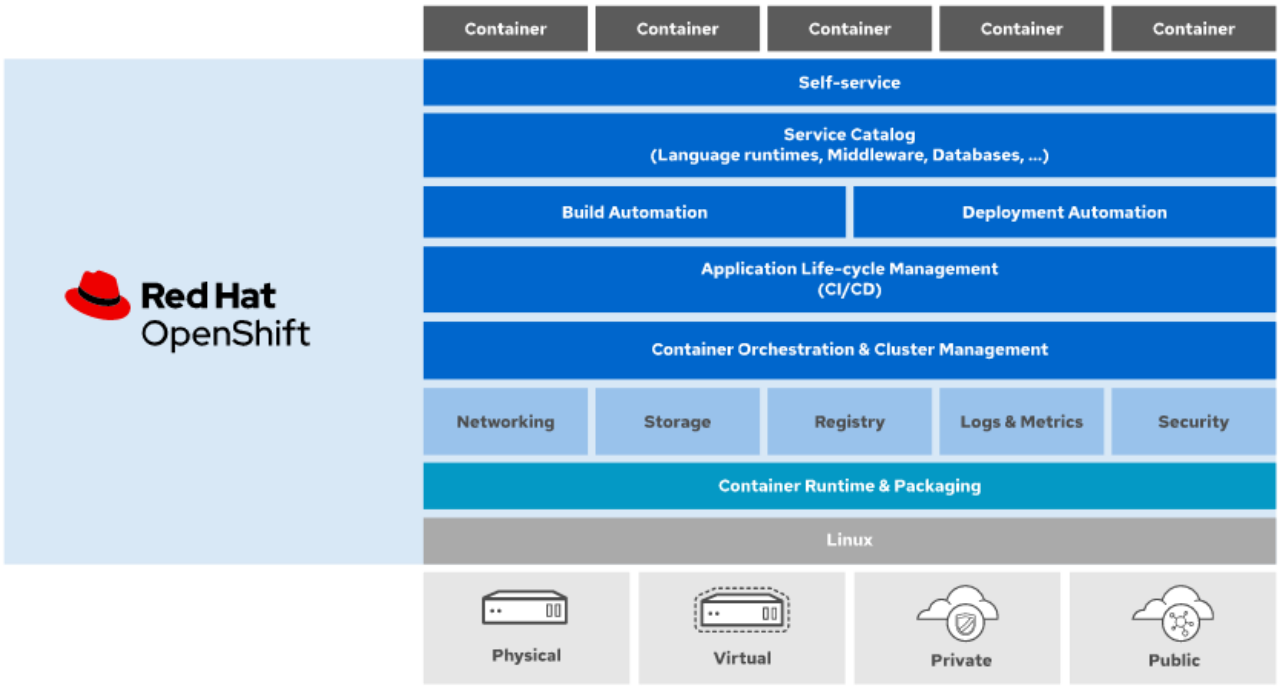
Ing. Mariano G. Barbero, Red Hat Certified Architect & Instructor

Crea tu ambiente

- Crear una cuenta en [Developer Sandbox for Red Hat OpenShift | Red Hat Developer]
- Crear una cuenta en github

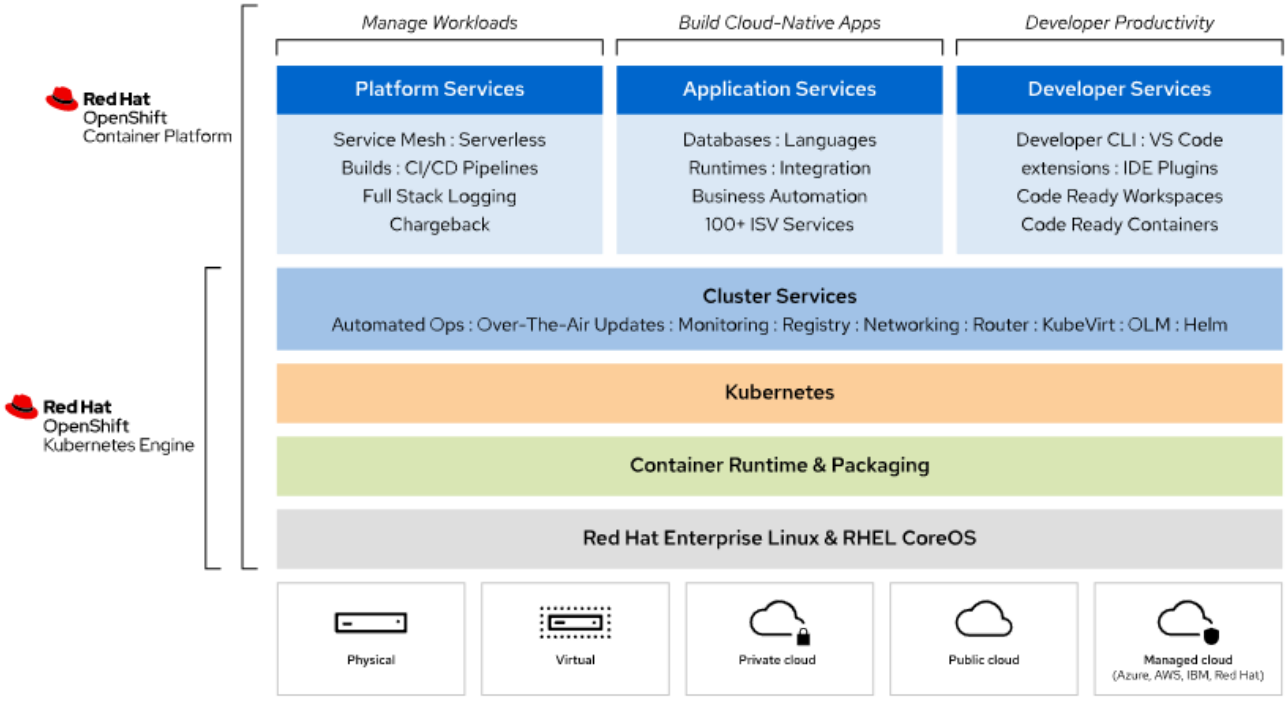
Arquitectura OCP4





La familia de productos de Red Hat OpenShift integra muchos componentes:

- El sistema operativo inmutable, optimizado para contenedores Red Hat Enterprise Linux CoreOS.
- El motor CRI-O, un motor de tiempo de ejecución de contenedor en conformidad con Open Container Initiative (OCI), de huella pequeña y con superficie de ataque reducida.
- Kubernetes, una plataforma de orquestación de contenedores de código abierto.
- Una consola web autoservicio.
- Una cantidad de servicios de aplicaciones preinstalados, como un registro interno de imágenes de contenedores, o un marco (framework) de monitoreo.
- Imágenes de contenedores certificadas para tiempos de ejecución, bases de datos y otros paquetes de software de varios lenguajes de programación.



Características de OCP 4

Alta disponibilidad
Kubernetes se ha diseñado pensando en la alta disponibilidad, tanto para los componentes internos como para las aplicaciones de usuarios. Un clúster de etcd de alta disponibilidad almacena el estado del clúster de OpenShift y sus aplicaciones.

Los recursos almacenados en etcd, como las opciones de configuración de implementación, proporcionan un reinicio automático de los contenedores para garantizar que su aplicación se esté ejecutando siempre y que se retiren los contenedores defectuosos. Esto no solo se aplica a sus aplicaciones, sino también a los servicios contenerizados que componen el clúster, como la consola web y el registro de imágenes interno.

Sistema operativo ligero
RHOCP emplea Red Hat Enterprise Linux CoreOS, el sistema operativo ligero de Red Hat que se centra en la agilidad, la portabilidad y la seguridad.

Red Hat Enterprise Linux CoreOS (RHEL CoreOS) es un sistema operativo inmutable optimizado para la ejecución de aplicaciones contenerizadas. Todo el sistema operativo se actualiza como una sola imagen en lugar de hacerlo paquete

por paquete, y las aplicaciones de usuarios y los componentes del sistema, como los servicios de red, se ejecutan como contenedores.

RHOCP controla las actualizaciones de RHEL CoreOS y sus opciones de configuración, por lo tanto, administrar un clúster de OpenShift supone administrar el sistema operativo en nodos de clústeres, lo que libera a los administradores de sistemas de estas tareas y reduce el riesgo de errores humanos.

Balanceo de carga
Los clústeres brindan tres tipos de balanceadores de carga: uno externo, que administra el acceso a la API de OpenShift; uno de HAProxy, para el acceso externo a las aplicaciones; y uno interno, que usa reglas de Netfilter para el acceso interno a aplicaciones y servicios.

Los recursos de ruta usan HAProxy para administrar el acceso externo al clúster. Los recursos de servicio usan reglas de Netfilter para administrar el tráfico que proviene desde dentro del clúster. La tecnología que usan los balanceadores de carga externos depende del proveedor de nube en el que ejecuta el cluster.

Automatización de escalamiento
Los clústeres de OpenShift se pueden adaptar al aumento del tráfico de aplicaciones en tiempo real mediante el inicio automático de nuevos contenedores y, luego, la finalización automática de los contenedores cuando se reduce la carga. Esta opción garantiza que el tiempo de acceso de su aplicación siga siendo óptimo, independientemente del número de conexiones o actividades simultáneas.

Los clústeres de OpenShift también pueden agregar o quitar nodos trabajadores en el clúster de acuerdo con la carga combinada de muchas aplicaciones, para mantener la velocidad de respuesta y reducir los costos en nubes públicas y privadas.

Registro y monitoreo
RHOCP viene con una solución de monitoreo avanzado basada en Prometheus que recopila cientos de métricas de su clúster. Esta solución interactúa con un sistema de alerta que le permite obtener información detallada acerca de la actividad y del estado de su clúster.

RHOCP viene con una solución de registro combinado avanzado basada en Elasticsearch que permite conservar a largo plazo registros de los contenedores y nodos del clúster.

Detección de servicios

RHOCP ejecuta un servicio de DNS interno en el clúster y configura todos los contenedores a fin de que usen ese DNS interno para la resolución de nombres. Esto significa que las aplicaciones pueden usar nombres descriptivos para buscar otras aplicaciones y servicios, sin la sobrecarga de un catálogo de servicios externo.

Almacenamiento
Kubernetes agrega una capa de abstracción entre el back-end y el consumo de almacenamiento. Por ende, las aplicaciones pueden consumir almacenamiento de largo plazo, de corto plazo, de bloques y de archivos mediante definiciones de almacenamiento unificadas independientes del back-end de almacenamiento. De esta manera, las aplicaciones no dependen de API de almacenamiento de proveedores de nubes particulares.

RHOCP incorpora una serie de proveedores de almacenamiento que permiten el aprovisionamiento automático en proveedores de nubes y plataformas de virtualización populares. Por lo tanto, los administradores de clústeres no necesitan administrar en detalle como soluciones de almacenamiento propietario.

Administración de aplicaciones
RHOCP permite a los desarrolladores automatizar el desarrollo y la implementación de sus aplicaciones. Use la función de OpenShift Source-to-Image (S2I) (Origen a imagen [S2I]) para crear automáticamente contenedores basados en su código fuente y ejecutarlos en OpenShift. El registro interno almacena imágenes de contenedores de aplicaciones que se pueden reutilizar. Esto acelera la publicación de sus aplicaciones.

El catálogo del desarrollador, al que se puede acceder desde la consola web, es un sitio para publicar y acceder a las plantillas de aplicación. Soporta muchos lenguajes de tiempo de ejecución, como Python, Ruby, Java y Node.js, además de servidores de mensajería y bases de datos. Puede expandir el catálogo mediante la instalación de nuevos operadores, que son aplicaciones y servicios preempaquetados que incorporan inteligencia operativa para la implementación, la actualización y el monitoreo de las aplicaciones.

Ampliación de clústeres
RHOCP emplea los mecanismos de extensión estándares de Kubernetes, como las API de extensión y las definiciones de recursos personalizadas, para agregar funciones que, de lo contrario, no provienen de Kubernetes upstream. OpenShift empaqueta estas extensiones como operadores para facilitar la instalación, la actualización y la administración.

OpenShift también incluye Operator Lifecycle Manager (OLM), que simplifica la detección, la instalación y la actualización de aplicaciones y componentes de infraestructuras empaquetados como operadores.

Red Hat, en colaboración con AWS, Google Cloud y Microsoft, lanzó OperatorHub, disponible en https://operatorhub.io. Dicha plataforma es un repositorio y un mercado públicos para operadores compatibles con OpenShift y otras distribuciones de Kubernetes que incluyen OLM.

Red Hat Marketplace es una plataforma que permite el acceso a software certificado empaquetado como operadores de Kubernetes que se pueden implementar en un clúster de OpenShift. El software certificado incluye implantaciones automáticas y actualizaciones sin problemas para una experiencia integrada.

Ansible para administrar recursos de Openshift

Objetivos

Después de completar esta sección, debería poder automatizar la creación y modificación de recursos de OpenShift utilizando los módulos Ansible k8s.

Explicación de los módulos de Ansible para Kubernetes

Hay un puñado de módulos disponibles para ayudar a administrar e interactuar con Kubernetes. Esos mismos módulos también funcionan con OpenShift porque OpenShift es una distribución de Kubernetes. El nombre abreviado de Kubernetes, k8s, se usa para referirse a todos los módulos de Kubernetes y es el nombre del grupo que precede a todos los módulos de Kubernetes.

Módulos k8s en Ansible 2.9

Nombre de módulo	**Descripción**
`k8s`	Administra objetos de Kubernetes. Este módulo tiene acceso a todas las API de Kubernetes, lo que le permite crear, actualizar o eliminar objetos de Kubernetes.
`k8s_auth`	Se autentica en clústeres de Kubernetes que requieren un inicio de sesión explícito, como OpenShift. Este módulo devuelve una api_key que los otros módulos pueden usar para la autenticación.
`k8s_info`	Recupera información sobre objetos de Kubernetes.
`k8s_scale`	Establece un nuevo tamaño para una implementación, un conjunto de réplicas, un controlador de replicación o un trabajo.
`k8s_service`	Administra servicios en Kubernetes.

La mayoría de estos módulos tienen parámetros comunes como api_key, context y namespace. Los parámetros compartidos se pueden establecer en valores predeterminados mediante el atributo module_defaults.

Ejemplo module_defaults

```
- hosts: localhost
  module_defaults:
    group/k8s:
      api_key: "{{ auth_token }}"
      host: https://api.example.com/
      ca_cert: ca.pem
```

Ejemplo de Playbook

```
- name: Log in to OpenShift
  hosts: localhost
  tasks:
    - name: Log in (obtain access token)
      k8s_auth:
        host: https://api.ocp4.example.com:6443
        username: admin
        password: redhat
        validate_certs: false
      register: k8s_auth_results

- name: Demonstrate k8s modules
  hosts: localhost
  vars:
    namespace: dev
  module_defaults:
    group/k8s:
      namespace: "{{ namespace }}"
      api_key: "{{
hostvars['localhost']['k8s_auth_results']['k8s_auth']['api_key'] }}"
      host: https://api.ocp4.example.com:6443
      validate_certs: false
  tasks:
    - name: Create objects from the manifest
      k8s:
        state: present
        src: "{{ playbook_dir + '/files/manifest.yml' }}"

    - name: Get a info about Pods that are web apps in dev or test
      k8s_info:
        kind: Pod
        label_selectors:
          - app = web
          - tier in (dev, test)

    - name: Scale deployment up
      k8s_scale:
        kind: Deployment
        name: example_deployment
        replicas: 3
```

```
    - name: Expose https port with ClusterIP
      k8s_service:
        state: present
        name: example_service
        ports:
          - port: 443
            protocol: TCP
        selector:
          key: value
```

Nota

Es posible que la apiVersion en el manifiesto deba ser más explícita para el paquete python de openshift. En lugar de poner v1, use el punto final completo, como autorización.openshift.io/v1.

Importante

La implementación actual de los módulos k8s no respeta los estándares de roles. Cuando utilice los parámetros src o resource_definition, debe proporcionar la ruta completa. Los módulos no verificarán los directorios de plantillas o archivos estándar en un rol automáticamente.

Describiendo las dependencias de los módulos K8s

Los paquetes de requisitos previos de Python para los módulos k8s, k8s_info y k8s_scale son:

- `openshift` >= 0.6
- `PyYAML` >= 3.11

El módulo k8s_service no requiere PyYAML y requiere openshift >= 0.6.2.

El módulo k8s_auth requiere:

- `urllib3`
- `requests`
- `requests-oauthlib`

Explicación de las opciones de autenticación del módulo

Autentíquese con la API mediante un archivo kubeconfig, HTTP Basic Auth o tokens de autenticación.

Si no se proporcionan opciones de conexión, el cliente de openshift intenta cargar el archivo de configuración predeterminado desde ~/.kube/config. El parámetro del módulo kubeconfig especifica una ruta a un archivo de configuración de Kubernetes

existente. El parámetro del módulo de contexto elige un contexto del archivo kubeconfig.

Para autenticarse a través de HTTP Basic Auth, proporcione los parámetros del módulo de host, nombre de usuario y contraseña. El parámetro del módulo de proxy opcional se conecta a través de un proxy HTTP.

El módulo k8s_auth proporciona un token de autenticación que se puede usar con el parámetro del módulo api_key para autenticarse con la API.

Verifique los certificados SSL para el servidor API utilizando los parámetros del módulo ca_cert, client_cert y client_key. Alterne la validación de certificados con el parámetro del módulo validate_certs.