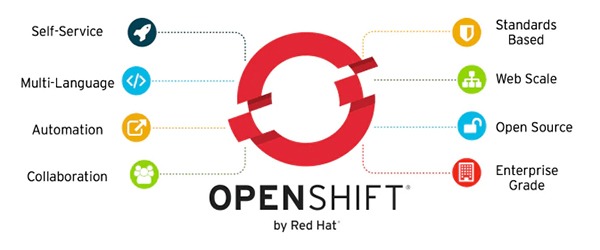
**Openshift : Deployment, Service, Networking/policy, VirtualService.**

*Trước khi đọc bài này cần tìm kiểu về Kubernetes trước.*



# **I. Tìm hiểu về Openshift:**

Openshift là nền tảng phát triển container.

* **OpenShift extends Kubernetes**

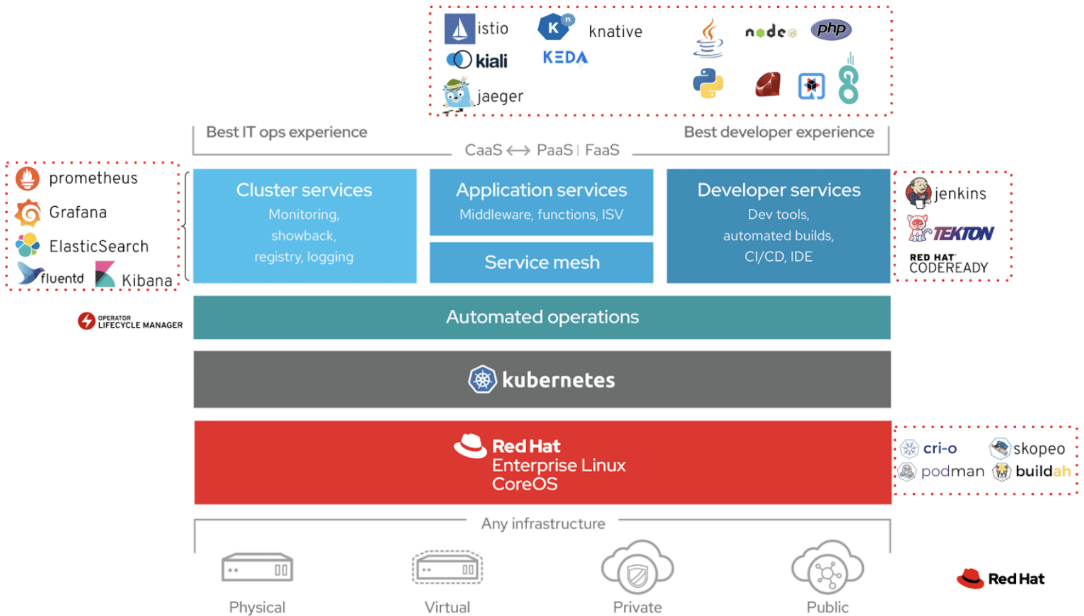
OpenShift dựa trên Kubernetes nhưng bổ sung và mở rộng thêm nhiều tính năng, công cụ để hỗ trợ việc triển khai, quản lý, và vận hành container trong môi trường doanh nghiệp.

## 1. **Kubernetes là lõi của OpenShift** :

=> OpenShift sử dụng Kubernetes làm nền tảng cơ bản, có các tiêu chuẩn như:

* **Orchestration**: Điều phối container.
* **Pods**: Nhóm container chạy trên các node.
* **Deployments**: Triển khai và quản lý ứng dụng.
* **Services & Ingress**: Kết nối và load balancing.

Kubernetes nguyên bản (upstream Kubernetes) thường yêu cầu cấu hình và vận hành thủ công nhiều thành phần. OpenShift giúp khắc phục điều này.



### **1.1: Pod**

**=>** là thành phần cơ bản nhất để triển khai và chạy ứng dụng trên Kubernetes [1](https://kubernetes.io/vi/docs/tutorials/kubernetes-basics/explore/explore-intro/)[2](https://viblo.asia/p/kubernetes-series-bai-2-kubernetes-pod-thanh-phan-de-chay-container-YWOZr3QElQ0). Một Pod đại diện cho một nhóm gồm một hoặc nhiều container (ví dụ như Docker) cùng với tài nguyên được chia sẻ cho các container đó.

**Đặc điểm chính của Pod:**

* Bao gồm một hoặc nhiều container cùng với tài nguyên được chia sẻ như:
  + Lưu trữ chung dưới dạng Volumes
  + Kết nối mạng duy nhất
  + Thông tin về cách chạy từng container [1](https://kubernetes.io/vi/docs/tutorials/kubernetes-basics/explore/explore-intro/)
* Mô phỏng một "máy chủ logic" dành riêng cho ứng dụng [1](https://kubernetes.io/vi/docs/tutorials/kubernetes-basics/explore/explore-intro/)
* Các containers trong một Pod chia sẻ địa chỉ IP và không gian port [1](https://kubernetes.io/vi/docs/tutorials/kubernetes-basics/explore/explore-intro/)
* Là đơn vị nguyên tử trên nền tảng Kubernetes [1](https://kubernetes.io/vi/docs/tutorials/kubernetes-basics/explore/explore-intro/)

**Tại sao sử dụng Pod?**

* Cung cấp thêm chức năng quản lý và hỗ trợ cho container so với việc chạy trực tiếp [**2**](https://viblo.asia/p/kubernetes-series-bai-2-kubernetes-pod-thanh-phan-de-chay-container-YWOZr3QElQ0)
* Hỗ trợ kiểm tra trạng thái và restart container khi cần [**4**](https://devopsvn.tech/kubernetes/pod-la-gi)
* Cung cấp lifecycle để thêm hành động vào Pod khi khởi động/shutdown [**4**](https://devopsvn.tech/kubernetes/pod-la-gi)
* Giúp Kubernetes quản lý Pod thay vì quản lý container trực tiếp [**2**](https://viblo.asia/p/kubernetes-series-bai-2-kubernetes-pod-thanh-phan-de-chay-container-YWOZr3QElQ0)

**Cách hoạt động**

* Khi tạo Deployment, Kubernetes sẽ tạo ra Pods chứa các container bên trong [1](https://kubernetes.io/vi/docs/tutorials/kubernetes-basics/explore/explore-intro/)
* Mỗi Pod được gắn với Node nơi nó được lên lịch trình [1](https://kubernetes.io/vi/docs/tutorials/kubernetes-basics/explore/explore-intro/)
* Trong trường hợp lỗi Node, Pods tương tự sẽ được lên lịch trên Nodes khác [1](https://kubernetes.io/vi/docs/tutorials/kubernetes-basics/explore/explore-intro/)

### **1.2: Deployment**

**=>** là một object trong Kubernetes/OpenShift dùng để duy trì một số lượng replicas (các instance) của một Pod:

* Nó tự động tạo và loại bỏ Pods để duy trì số lượng replicas mong muốn.
* Có thể cập nhật image của Pods mà không ảnh hưởng đến uptime của ứng dụng.
* Cho phép rollback nếu update gây ra vấn đề.
* Có thể cấu hình liveness và readiness probes để kiểm tra trạng thái của Pods.

### **1.3: Service**

**=>** là một abstraction trên một group of Pods

* Cung cấp một stable network endpoint cho các Pod.
* Cho phép traffic được route đến đúng Pod.
* Có thể expose Service bằng LoadBalancer hoặc Ingress.
* Hỗ trợ load balancing và health checking.

### **1.4: Networking/PodDisruptionBudget Policy (PDB)**

=> là một resource để bảo đảm availability của application:

* Cung cấp một giới hạn về số lượng Pods có thể bị mất trong một khoảng thời gian.
* Có hai type: MaxUnavailable và MaxSurge.
* Thường được sử dụng cùng với Deployment.

### **1.5: VirtualService**

=> là một object trong Istio Service Mesh để route traffic đến các Services

* Cho phép định nghĩa rules để route traffic dựa trên các tiêu chí như header, cookie, etc.
* Có thể được sử dụng để can thiệp vào traffic giữa các microservices.

## **2.** **Mở rộng Kubernetes với tính năng bổ sung**

OpenShift không chỉ sử dụng Kubernetes mà còn tích hợp thêm các tính năng giúp dễ sử dụng và an toàn hơn.

* Tính năng quản lý nâng cao:

 **Source-to-Image (S2I)**: Tự động hóa việc tạo container từ mã nguồn (không cần Dockerfile).

 **Integrated CI/CD**: Hỗ trợ pipeline tích hợp với Jenkins hoặc GitOps.

 **User Interface (UI)**: Dashboard trực quan cho việc quản lý cluster và ứng dụng

* Bảo mật vượt trội:

 **Role-Based Access Control (RBAC)**: Kiểm soát truy cập dựa trên vai trò, được cài đặt mặc định.

 **SELinux**: Bảo mật ở cấp hệ điều hành, tích hợp chặt chẽ trong OpenShift.

 **Image Registry**: OpenShift đi kèm với một registry container nội bộ (Quay.io), giúp quản lý và bảo mật hình ảnh container.

* Quản lý vận hành:

 **Cluster Operators**: OpenShift tích hợp các **Operators** (một khái niệm từ Kubernetes) để tự động hóa việc quản lý thành phần của cluster (như update, sửa lỗi).

 **Networking**: OpenShift cung cấp SDN (Software-Defined Networking) tích hợp, giúp việc giao tiếp giữa các pod đơn giản hơn mà không cần cấu hình nhiều.

# **II. Các thành phần trong giao diện quản lý Openshift.**

**Openshift** giúp quản lý tập trung dự án gồm những service.

A screenshot of a computer

Description automatically generated

**Administrator**: Quản lý cơ sở hạ tầng OpenShift và các tài nguyên trên cụm

**Developer:** Triển khai và quản lý ứng dụng trên OpenShift

Menu chức năng :

**Administrator Developer**

A screenshot of a computer

Description automatically generatedA screenshot of a black screen

Description automatically generated

1. **Administrator**

**A screenshot of a computer

Description automatically generated1.1 Home**

 **Projects**:  
Hiển thị danh sách các dự án (projects) trong OpenShift. Dự án là một namespace logic, nơi chứa các tài nguyên như pods, services, config maps, và các ứng dụng liên quan.

 **Search**:  
Công cụ tìm kiếm tài nguyên (resources) trên cụm, ví dụ: pods, services, deployments, config maps. Giúp nhanh chóng tìm kiếm và truy cập tài nguyên cụ thể.

 **API Explorer**:  
Giao diện cho phép người dùng khám phá các API mà OpenShift hỗ trợ. Hữu ích khi cần tích hợp OpenShift với các công cụ khác hoặc thực hiện tự động hóa.

 **Events**:  
Hiển thị các sự kiện (events) liên quan đến hoạt động của cụm, chẳng hạn như trạng thái pod, lỗi triển khai, hoặc các thay đổi cấu hình. Công cụ quan trọng để theo dõi và khắc phục sự cố.

**1.2: Operators**

* **Installed Operators:**Hiển thị danh sách các operator đã được cài đặt trên cụm. Operators là các tiện ích mở rộng dùng để tự động hóa các tác vụ quản trị, như quản lý cơ sở dữ liệu, dịch vụ mạng, hoặc ứng dụng.

**1.3: Workloads**

* **Pods:**Danh sách các pods đang chạy trong cụm. Pods là đơn vị triển khai nhỏ nhất trong Kubernetes, chứa các container chạy trong cụm.
* **Deployments:**Một loại cấu hình đặc biệt của OpenShift để triển khai ứng dụng, thường được sử dụng khi cần tích hợp các tính năng độc quyền của OpenShift.
* **StatefulSets**:  
  Quản lý các ứng dụng yêu cầu trạng thái (stateful), chẳng hạn như cơ sở dữ liệu, nơi mỗi pod cần có một danh tính duy nhất.
* **Secrets**:  
  Quản lý các thông tin nhạy cảm như mật khẩu, token, và khóa mã hóa. Secrets giúp bảo mật thông tin nhạy cảm trong cụm.
*  **ConfigMaps**:  
  Lưu trữ các cấu hình không nhạy cảm cho ứng dụng, chẳng hạn như biến môi trường hoặc tệp cấu hình.
* **CronJobs**:  
  Định nghĩa và quản lý các tác vụ tự động hóa chạy theo lịch trình, ví dụ: sao lưu dữ liệu hoặc gửi email.
* **Jobs**:  
  Chạy các tác vụ chỉ thực hiện một lần, chẳng hạn như xử lý dữ liệu hoặc tính toán ngắn hạn.
* **DaemonSets**:  
  Đảm bảo mỗi node trong cụm có một bản sao của pod, thường dùng cho các ứng dụng như log collectors hoặc monitoring agents.
* **ReplicaSets**:  
  Đảm bảo số lượng pods luôn được duy trì ổn định theo cấu hình, hỗ trợ tính sẵn sàng cao.
* **ReplicationControllers**:  
  Cũng giống như ReplicaSets, nhưng là tính năng cũ hơn, ít được sử dụng trong các phiên bản Kubernetes hiện đại.
* **HorizontalPodAutoscalers**:  
  Tự động điều chỉnh số lượng pods dựa trên tài nguyên sử dụng (CPU, RAM), giúp tối ưu hóa tài nguyên.
* **PodDisruptionBudgets**:  
  Đảm bảo số lượng tối thiểu pods luôn sẵn sàng ngay cả khi có sự gián đoạn (ví dụ: cập nhật hoặc bảo trì).

**1.4: Serverless**

* **Serving**:  
  Quản lý các ứng dụng **serverless** (không máy chủ), nơi tài nguyên chỉ được sử dụng khi có yêu cầu. Thích hợp cho các ứng dụng như API endpoint hoặc các chức năng chạy tạm thời.
* **Eventing**:  
  Quản lý sự kiện trong hệ thống, dùng để kích hoạt các ứng dụng hoặc chức năng dựa trên các sự kiện cụ thể, chẳng hạn như tin nhắn từ hệ thống bên ngoài hoặc trạng thái cụm thay đổi.

**1.5:** **Networking**:

* Quản lý các tài nguyên liên quan đến mạng như:
  + **Services**: Quản lý kết nối giữa các pod.
  + **Routes**: Cung cấp truy cập bên ngoài cụm OpenShift.
  + **Ingresses**: Quản lý kết nối HTTP/HTTPS.
  + **NetworkPolicies**: Quản lý quyền truy cập mạng giữa các pod.

**1.6: Storage**:

* Quản lý các tài nguyên lưu trữ:
  + **PersistentVolumeClaims (PVCs)**: Yêu cầu lưu trữ lâu dài.
  + **StorageClasses**: Định nghĩa các loại lưu trữ.
  + **VolumeSnapshots**: Chụp nhanh dữ liệu trong PVC.
  + **VolumeSnapshotClasses**: Định nghĩa kiểu snapshot.
  + **Object Storage**: Lưu trữ đối tượng (object-based storage).

**1.7:** **Builds**:

* Quản lý quá trình xây dựng ứng dụng:
  + **BuildConfigs**: Cấu hình quá trình xây dựng.
  + **Builds**: Danh sách các bản dựng.
  + **ImageStreams**: Lưu trữ hình ảnh container để triển khai.

**1.8**: **Pipelines**:

* Tích hợp và triển khai liên tục (CI/CD) sử dụng các pipeline để tự động hóa quá trình phát triển và triển khai phần mềm.