

OPENSHIFT



by Everton Nascimento Nogueira



**RED HAT®
OPENSHIFT**
Container Platform

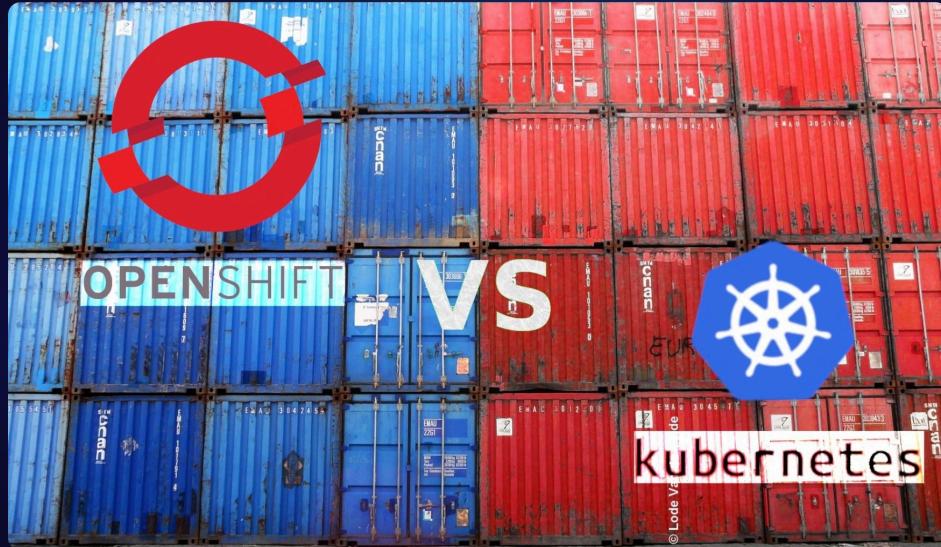


Introdução ao OpenShift:

O que é o OpenShift?

OpenShift é uma plataforma de contêineres Kubernetes empresarial desenvolvida pela Red Hat. Ela fornece um ambiente robusto para desenvolver, implantar e gerenciar aplicações em contêineres, oferecendo uma série de funcionalidades e ferramentas adicionais além do Kubernetes padrão.





Openshift vs Kubernetes



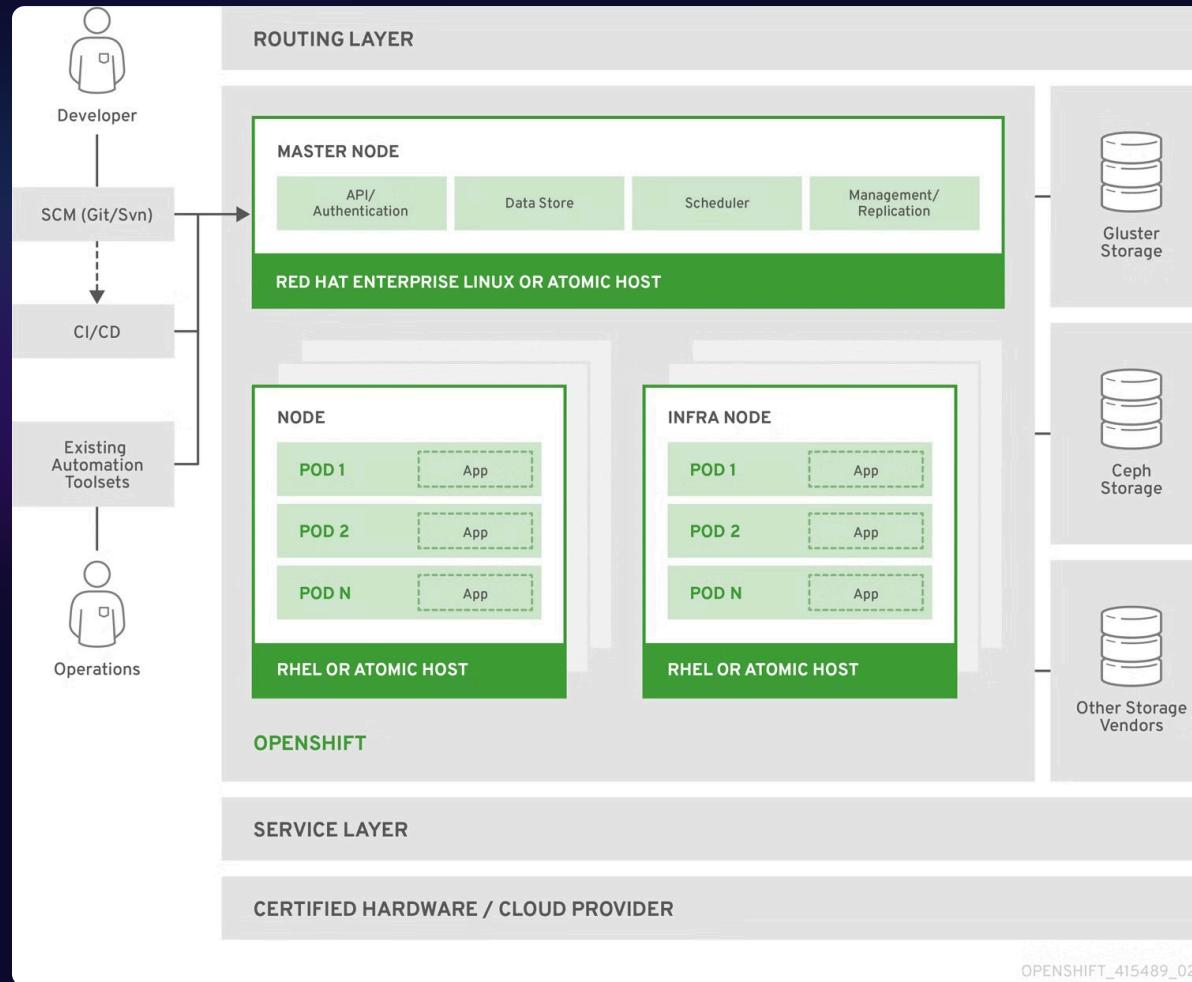
Contêiner

Um contêiner é uma unidade padronizada de software que empacota o código e todas as suas dependências, permitindo que a aplicação seja executada de maneira rápida e confiável em qualquer ambiente

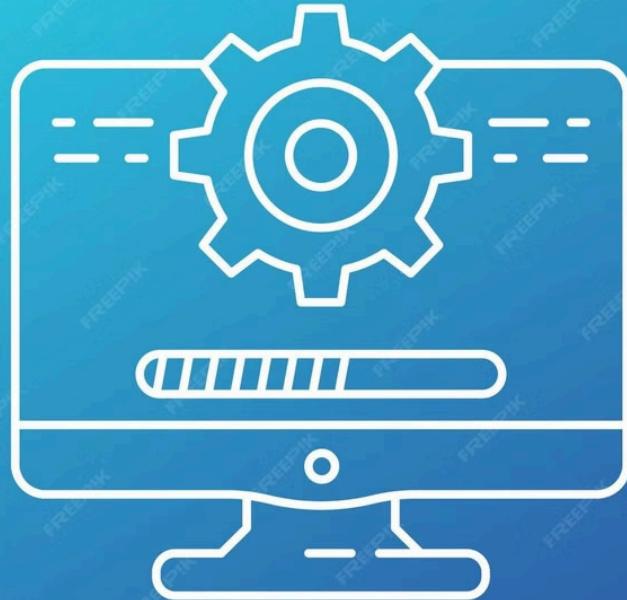
Arquitetura do OpenShift



Arquitetura do OpenShift



Visão geral do OpenShift Web Console



FIM do DIA 1

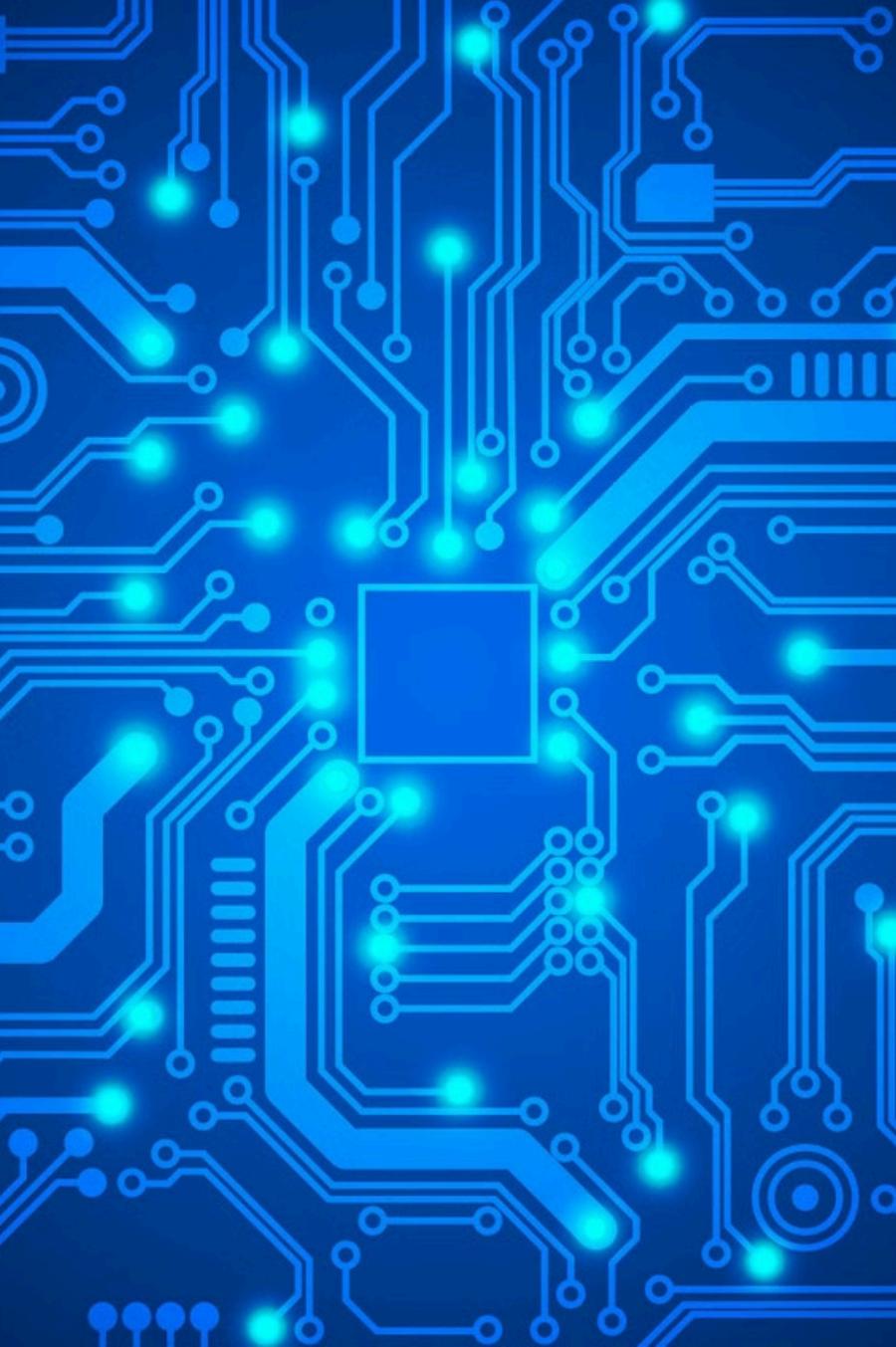
OPENSIFT

Parte 02

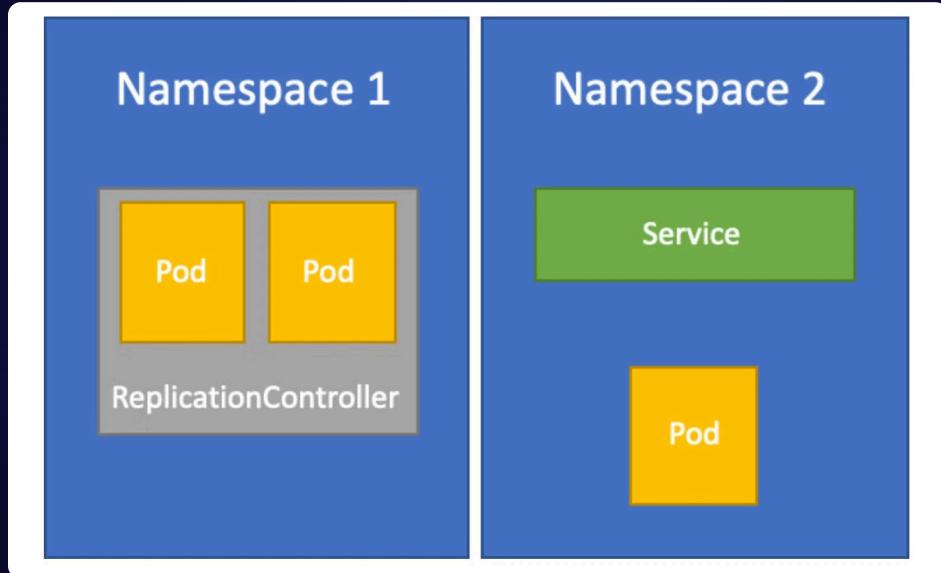


by Everton Nascimento Nogueira





Componentes básicos do OpenShift



Namespace/Project

Namespace é uma forma de dividir um único cluster em ambientes virtuais logicamente separados. Cada namespace tem seus próprios recursos, como Pods, Serviços, Deployments, etc. Isso permite que múltiplas equipes ou projetos compartilhem o mesmo cluster sem interferir umas nas outras, já que os recursos dentro de um namespace não podem acessar diretamente os recursos de outro namespace (a menos que configurado explicitamente).



Yaml

YAML é um formato de serialização de dados legível por humanos, comumente usado para configuração de arquivos e troca de dados entre linguagens de programação com diferentes tipos de dados. É projetado para ser fácil de ler e escrever, oferecendo uma sintaxe simples e minimalista.

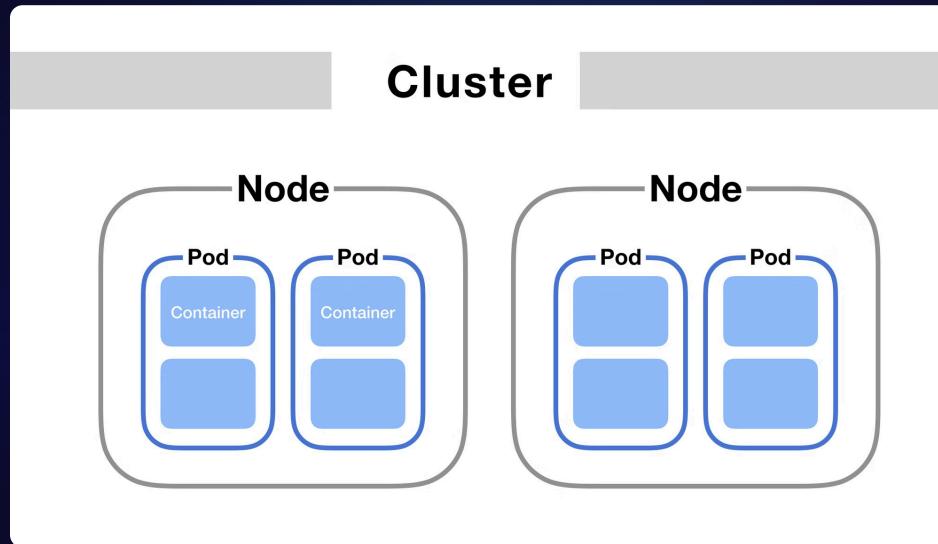
XML	JSON	YAML
<pre>1 <Servers> 2 <Server> 3 <name>Server1</name> 4 <owner>Prajwal</owner> 5 <status>active</status> 6 </Server> 7 <Server> 8 <name>Server2</name> 9 <owner>John</owner> 10 <status>inactive</status> 11 </Server> 12 </Servers></pre>	<pre>1 { 2 "Servers": [3 { 4 "name": "Server1", 5 "owner": "Prajwal", 6 "status": "active" 7 }, 8 { 9 "name": "Server2", 10 "owner": "John", 11 "status": "inactive" 12 } 13] 14 }</pre>	<pre>1 Servers: 2 - name: Server1 3 owner: Prajwal 4 status: active 5 - name: Server2 6 owner: John 7 status: inactive</pre>

XML | JSON | YAML

```
apiVersion: apps/v1
kind: Deployment
metadata:
  creationTimestamp: "2021-06-28T15:31:11Z"
  generation: 1
  labels:
    app: my-test-deployment
  name: my-test-deployment
  namespace: default
  resourceVersion: "213326"
  uid: ee6c68bc-d0e-474d-9ec9-44634412b137
spec:
  progressDeadlineSeconds: 600
  replicas: 3
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app: my-test-deployment
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
    type: RollingUpdate
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: my-test-deployment
    spec:
      containers:
        - image: nginx
          imagePullPolicy: Always
          name: nginx
          ports:
            - containerPort: 8000
              protocol: TCP
          resources: {}
          terminationMessagePath: /dev/termination-log
          terminationMessagePolicy: File
        dnsPolicy: ClusterFirst
        restartPolicy: Always
```

YAML

exemplo



POD

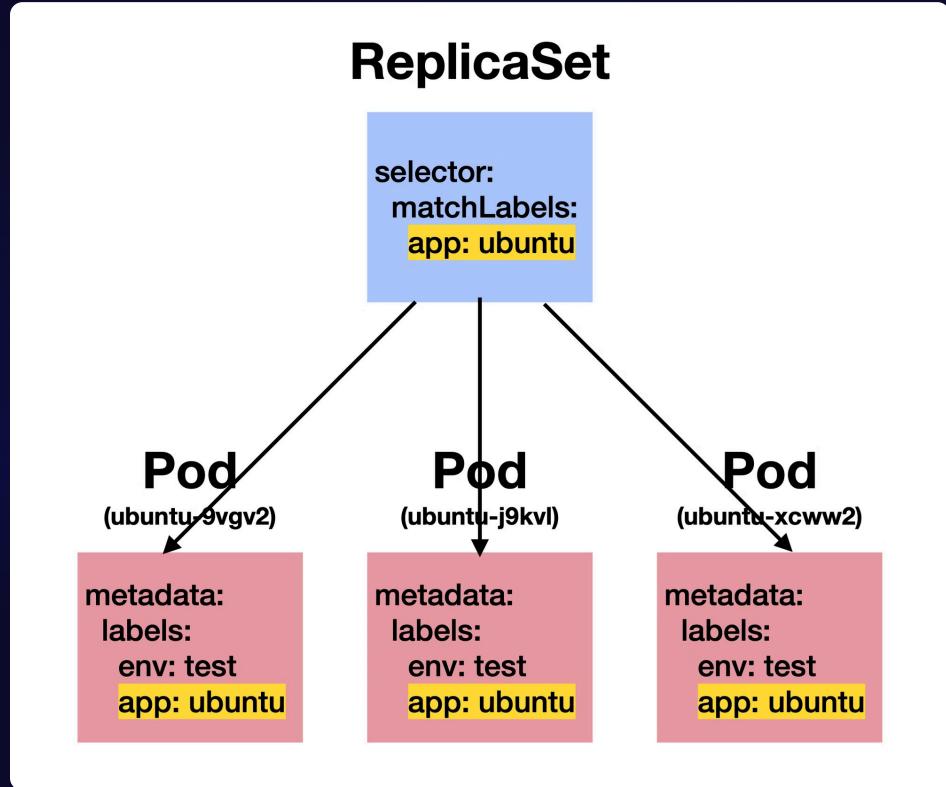
Pod é a menor unidade de implantação em Kubernetes. Ele encapsula um ou mais contêineres que compartilham o mesmo espaço de rede e armazenamento, permitindo a execução de aplicações de maneira conjunta.

```
apiVersion: v1
kind: Pod
metadata:
  name: meu-pod
  namespace: meu-namespace
  labels:
    app: minha-aplicacao
    tier: frontend
    env: producao
spec:
  containers:
    - name: meu-container
      image: nginx:latest
      ports:
        - containerPort: 80
      env:
        - name: AMBIENTE
          value: "producao"
      resources:
        limits:
          memory: "256Mi"
          cpu: "500m"
        requests:
          memory: "128Mi"
          cpu: "250m"
      volumeMounts:
        - mountPath: /usr/share/nginx/html
          name: meu-volume
      volumes:
        - name: meu-volume
          emptyDir: {}
  restartPolicy: Always
```

POD

Pod é a menor unidade de implantação em Kubernetes. Ele encapsula um ou mais contêineres que compartilham o mesmo espaço de rede e armazenamento, permitindo a execução de aplicações de maneira conjunta.





ReplicaSet

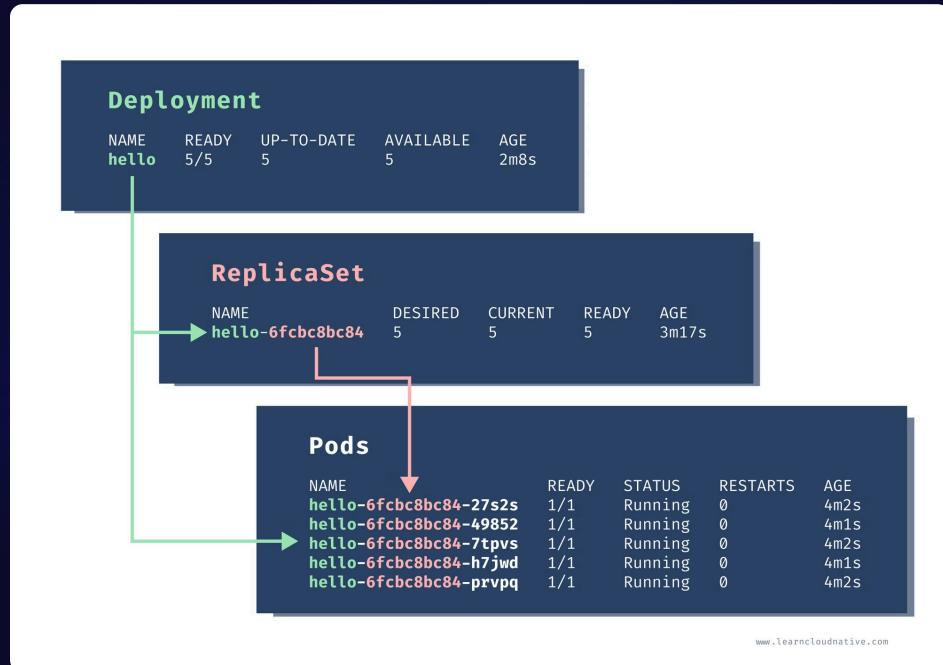
Um ReplicaSet é **um processo que executa várias instâncias de um pod e mantém o número especificado de pods constante**. Ele garante que um conjunto estável de pods réplica esteja em execução a qualquer momento, o que garante um número especificado disponível de pods idênticos.

```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: meu-replicaset
  namespace: meu-namespace
  labels:
    app: minha-aplicacao
spec:
  replicas: 3
  selector:
    matchLabels:
      app: minha-aplicacao
  template:
    metadata:
      labels:
        app: minha-aplicacao
    spec:
      containers:
        - name: meu-container
          image: nginx:latest
          ports:
            - containerPort: 80
          resources:
            limits:
              memory: "256Mi"
              cpu: "500m"
            requests:
              memory: "128Mi"
              cpu: "250m"
```

ReplicaSet

Um ReplicaSet é um processo que executa várias instâncias de um pod e mantém o número especificado de pods constante. Ele garante que um conjunto estável de pods réplica esteja em execução a qualquer momento, o que garante um número especificado disponível de pods idênticos.





Deployment

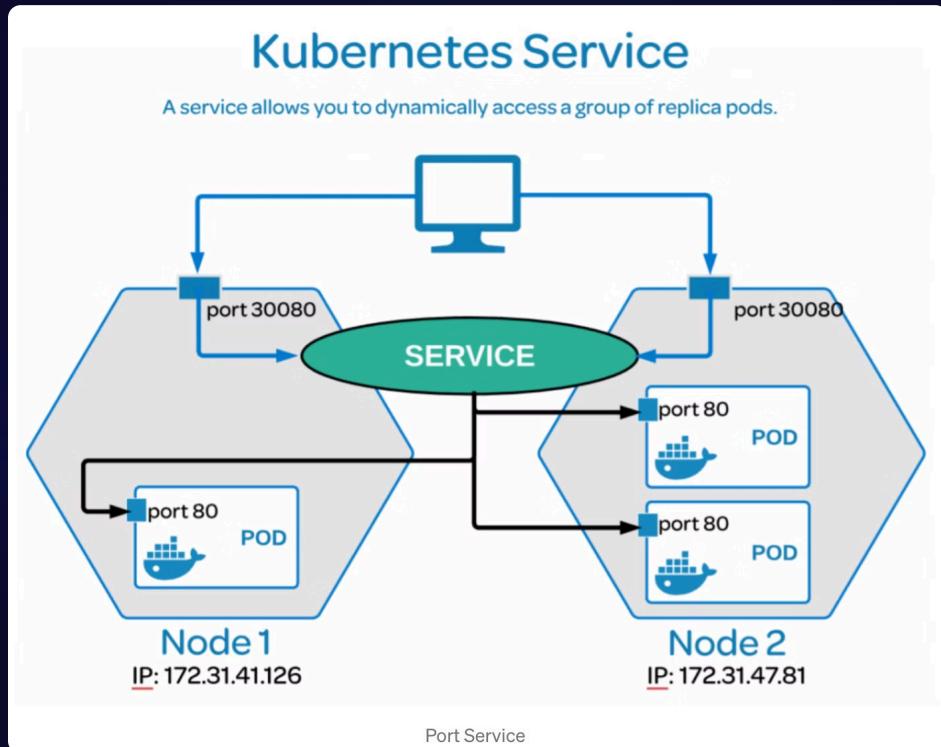
Um **Deployment** no Kubernetes/OpenShift é um recurso de alto nível que gerencia a criação, atualização e escalabilidade de aplicações containerizadas. Ele fornece uma maneira declarativa de especificar o estado desejado de aplicações e o Kubernetes/OpenShift se encarrega de garantir que o estado atual da aplicação corresponda ao estado desejado.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: meu-deployment
  namespace: meu-namespace
  labels:
    app: minha-aplicacao
spec:
  replicas: 3
  selector:
    matchLabels:
      app: minha-aplicacao
  template:
    metadata:
      labels:
        app: minha-aplicacao
    spec:
      containers:
        - name: meu-container
          image: nginx:latest
          ports:
            - containerPort: 80
          env:
            - name: AMBIENTE
              value: "producao"
          resources:
            limits:
              memory: "256Mi"
              cpu: "500m"
            requests:
              memory: "128Mi"
              cpu: "250m"
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 1
```

Deployment

Um **Deployment** no Kubernetes é um recurso de alto nível que gerencia a criação, atualização e escalabilidade de aplicações containerizadas. Ele fornece uma maneira declarativa de especificar o estado desejado de aplicações e o Kubernetes se encarrega de garantir que o estado atual da aplicação corresponda ao estado desejado.





Service

Um **Service** no Kubernetes é um recurso que permite expor uma aplicação em execução em um grupo de Pods como um serviço de rede estável e confiável. Como os Pods são efêmeros, ou seja, podem ser criados ou destruídos a qualquer momento, o Service atua como um ponto de acesso consistente para eles. Isso significa que, independentemente de quantas vezes os Pods mudem ou sejam recriados, o Service garante que o tráfego de rede seja direcionado corretamente, mantendo a continuidade e estabilidade da aplicação.

Hostname de serviço

Exemplo:

panda-nifi.panda-nifi.svc.cluster.local

- panda-nifi: Nome do serviço dentro do namespace.
- panda-nifi Nome do namespace
- svc: Indica que é um serviço (Service) dentro do Kubernetes. É uma parte fixa que identifica o recurso como um serviço.
- cluster.local: Este é o domínio DNS padrão usado dentro do cluster Kubernetes. Ele designa que o serviço está acessível apenas dentro do cluster, e não externamente, a menos que você configure isso explicitamente (como através de uma Route)

Service

```
apiVersion: v1
kind: Service
metadata:
  name: meu-service
  namespace: meu-namespace
  labels:
    app: minha-aplicacao
spec:
  selector:
    app: minha-aplicacao
  ports:
    - protocol: TCP
      port: 80
      targetPort: 8080
  type: ClusterIP
```

Um **Service** no Kubernetes é um recurso que permite expor uma aplicação em execução em um grupo de Pods como um serviço de rede estável e confiável. Como os Pods são efêmeros, ou seja, podem ser criados ou destruídos a qualquer momento, o Service atua como um ponto de acesso consistente para eles. Isso significa que, independentemente de quantas vezes os Pods mudem ou sejam recriados, o Service garante que o tráfego de rede seja direcionado corretamente, mantendo a continuidade e estabilidade da aplicação.

Route



Route é um recurso que permite expor uma aplicação rodando dentro do cluster para o mundo externo. As Routes são essenciais para permitir que usuários externos accessem serviços que estão sendo executados nos Pods dentro do cluster.

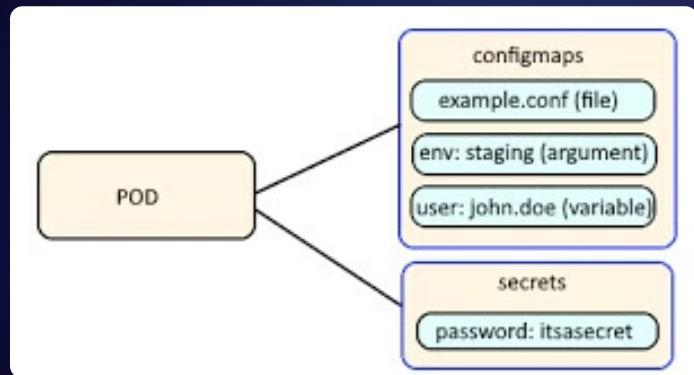
```
apiVersion: route.openshift.io/v1
kind: Route
metadata:
  name: minha-aplicacao
  namespace: meu-namespace
  labels:
    app: minha-aplicacao
spec:
  host: minha-aplicacao.meudominio.com
  to:
    kind: Service
    name: minha-aplicacao-service
    weight: 100
  port:
    targetPort: 8080
  tls:
    termination: edge
    insecureEdgeTerminationPolicy: Redirect
  wildcardPolicy: None
```

Route

Route é um recurso que permite expor uma aplicação rodando dentro do cluster para o mundo externo. As Routes são essenciais para permitir que usuários externos accessem serviços que estão sendo executados nos Pods dentro do cluster.



ConfigMaps



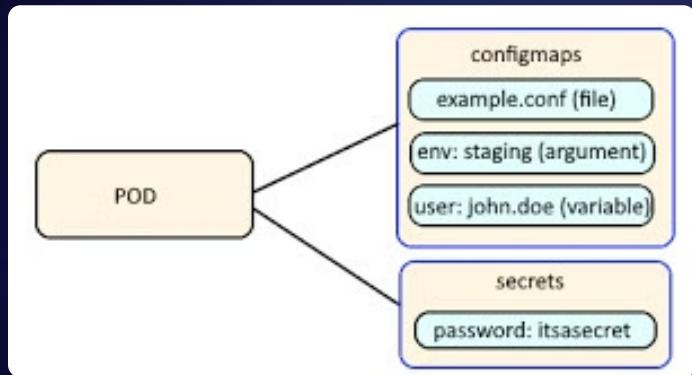
ConfigMaps no Kubernetes e Openshift são objetos usados para armazenar dados de configuração em pares chave-valor. Esses dados podem ser usados por Pods para configurar a aplicação que eles estão executando, sem a necessidade de embutir essa configuração diretamente na imagem do contêiner.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: meu-configmap
  namespace: meu-namespace
  labels:
    app: minha-aplicacao
data:
  database_url: jdbc:mysql://db.example.com:3306/meubanco
  max_connections: "100"
  feature_flag_enabled: "true"
```

ConfigMaps

ConfigMaps no Kubernetes são objetos usados para armazenar dados de configuração em pares chave-valor. Esses dados podem ser usados por Pods para configurar a aplicação que eles estão executando, sem a necessidade de embutir essa configuração diretamente na imagem do contêiner.

Secrets



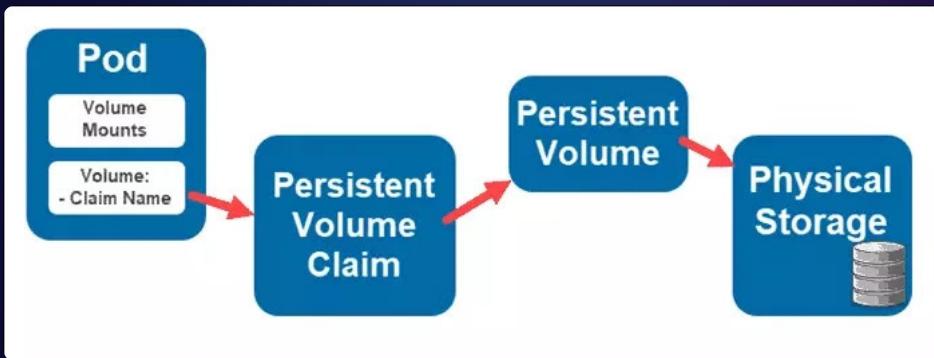
Secrets no Kubernetes são objetos usados para armazenar e gerenciar informações sensíveis, como senhas, tokens, chaves SSH, ou certificados. Eles são semelhantes aos ConfigMaps, mas são projetados especificamente para dados que precisam ser mantidos em segredo.

```
apiVersion: v1
kind: Secret
metadata:
  name: meu-secret
  namespace: meu-namespace
  labels:
    app: minha-aplicacao
type: Opaque
data:
  username: dXNlcm5hbWU=
  password: cGFzc3dvcmQ=
```

Secrets

Secrets no Kubernetes são objetos usados para armazenar e gerenciar informações sensíveis, como senhas, tokens, chaves SSH, ou certificados. Eles são semelhantes aos ConfigMaps, mas são projetados especificamente para dados que precisam ser mantidos em segredo.

PV



Persistent Volume (PV)

Um **Persistent Volume (PV)** é uma parte do armazenamento provisionada por um administrador. Ele é uma abstração do armazenamento físico que pode ser oferecido por diversos provedores de armazenamento (como NFS, iSCSI, cloud providers, etc.)

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: meu-pv
  labels:
    type: local
spec:
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  storageClassName: manual
```

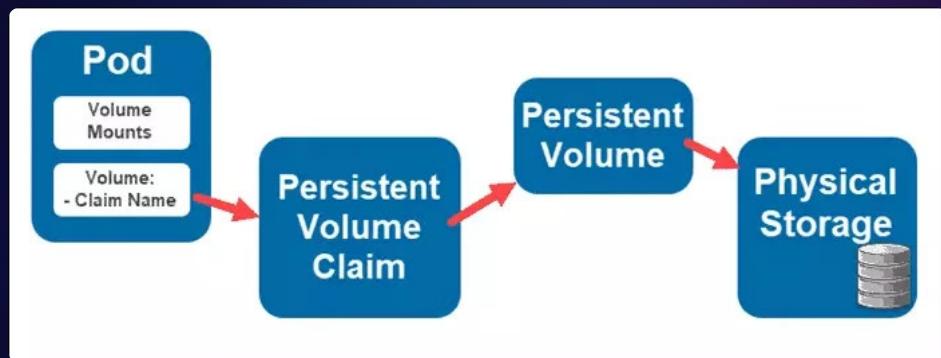
PV

Persistent Volume (PV)

Um **Persistent Volume (PV)** é um recurso no Kubernetes e Openshift que representa uma peça de armazenamento persistente no cluster. Ele abstrai o armazenamento físico e oferece uma interface para que os contêineres possam acessar volumes de forma persistente, independentemente de onde estejam rodando no cluster.



PVC



Persistent Volume Claim (PVC)

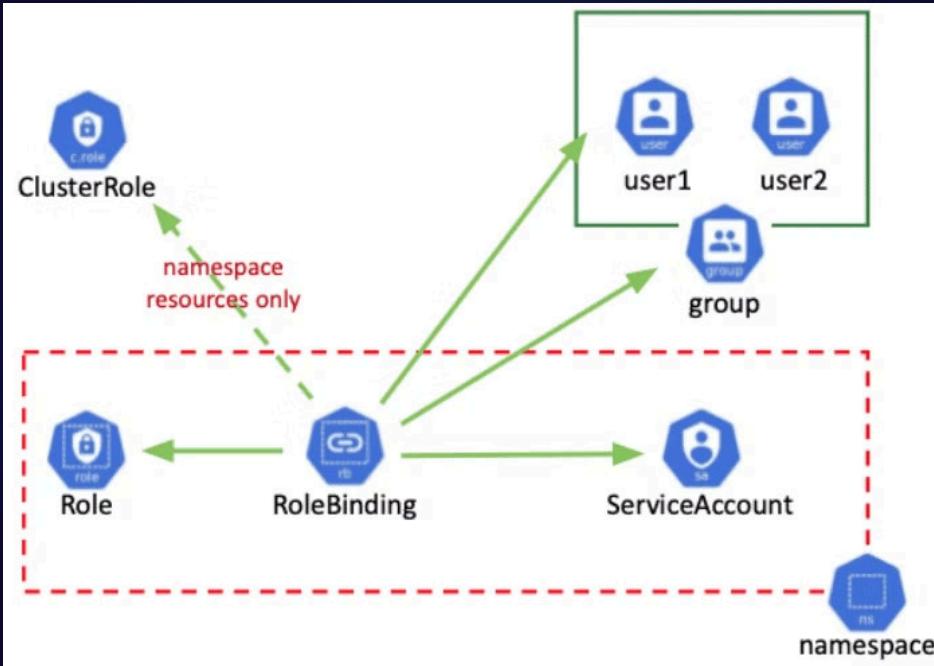
Uma **Persistent Volume Claim (PVC)** é uma requisição de armazenamento feita por um usuário ou aplicação. Ela descreve a quantidade de armazenamento e o modo de acesso desejado (por exemplo, leitura/escrita única ou múltipla). Quando uma PVC é criada, o Kubernetes tenta encontrar um PV que satisfaça a requisição especificada. Se encontrar, o PV é associado à PVC, e o armazenamento é montado no Pod que solicita a PVC.

PVC

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: example-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 5Gi
  volumeName: example-pv
```

Persistent Volume Claim (PVC)

Uma **Persistent Volume Claim (PVC)** é uma requisição de armazenamento feita por um usuário. Ela descreve a quantidade de armazenamento e o modo de acesso desejado (por exemplo, leitura/escrita única ou múltipla). Quando uma PVC é criada, o Kubernetes tenta encontrar um PV que satisfaça a requisição especificada. Se encontrar, o PV é associado à PVC, e o armazenamento é montado no Pod que solicita a PVC.



RBAC (Role-Based Access Control)

OpenShift RBAC (Role-Based Access Control) é o sistema utilizado para gerenciar permissões e acesso a recursos dentro de um cluster OpenShift, de forma granular e segura. O RBAC permite que você controle quem pode fazer o quê dentro do cluster, associando permissões a **funções** (roles), que são, então, atribuídas a **usuários** ou **grupos**.

RBAC (Role-Based Access Control)

Default cluster role

- **admin**
- **basic-user**
- **cluster-admin**
- **cluster-reader**
- **self-provisioner**
- **view**

Role

```
kind: Role  
  
apiVersion: rbac.authorization.k8s.io/v1  
  
metadata:  
  
namespace: my-namespace  
  
name: pod-manager  
  
rules:  
  
- apiGroups: [""]  
  
resources: ["pods"]  
  
verbs: ["get", "list", "watch", "create", "delete"]
```

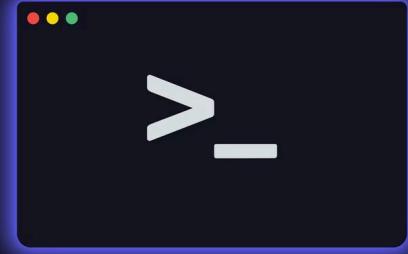
RoleBinding

```
kind: RoleBinding  
  
apiVersion: rbac.authorization.k8s.io/v1  
  
metadata:  
  
name: bind-pod-manager  
  
namespace: my-namespace  
  
subjects:  
  
- kind: User  
  
name: john-doe  
  
apiGroup: rbac.authorization.k8s.io  
  
roleRef:  
  
kind: Role  
  
name: pod-manager
```

CLI (Command Line Interface)

OC e KUBECTL

oc e kubectl são ferramentas de linha de comando usadas para interagir com clusters Kubernetes, mas cada uma é específica para diferentes plataformas.



Hands-On: Implantando uma Aplicação Simples



FIM do DIA 2