



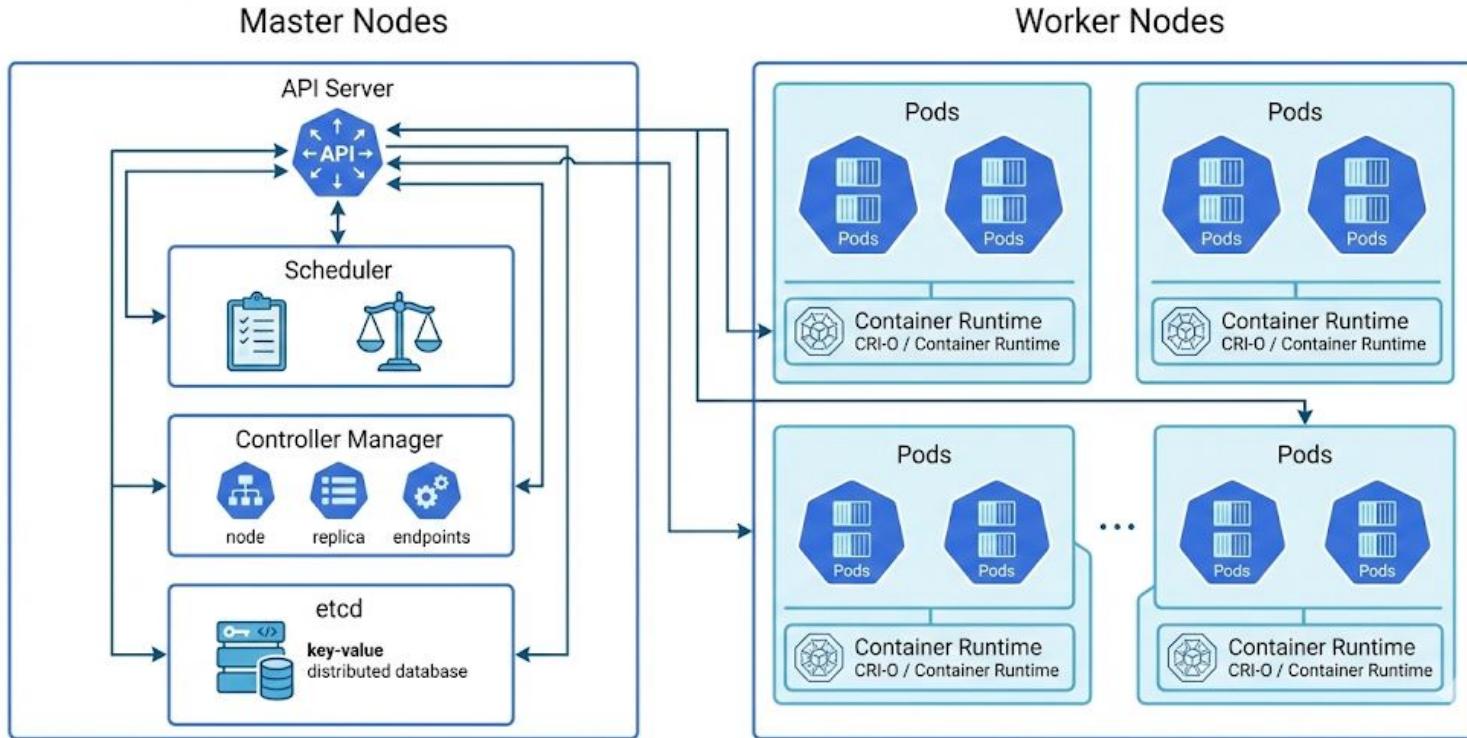
Welcome!

# Red Hat OpenShift Administration

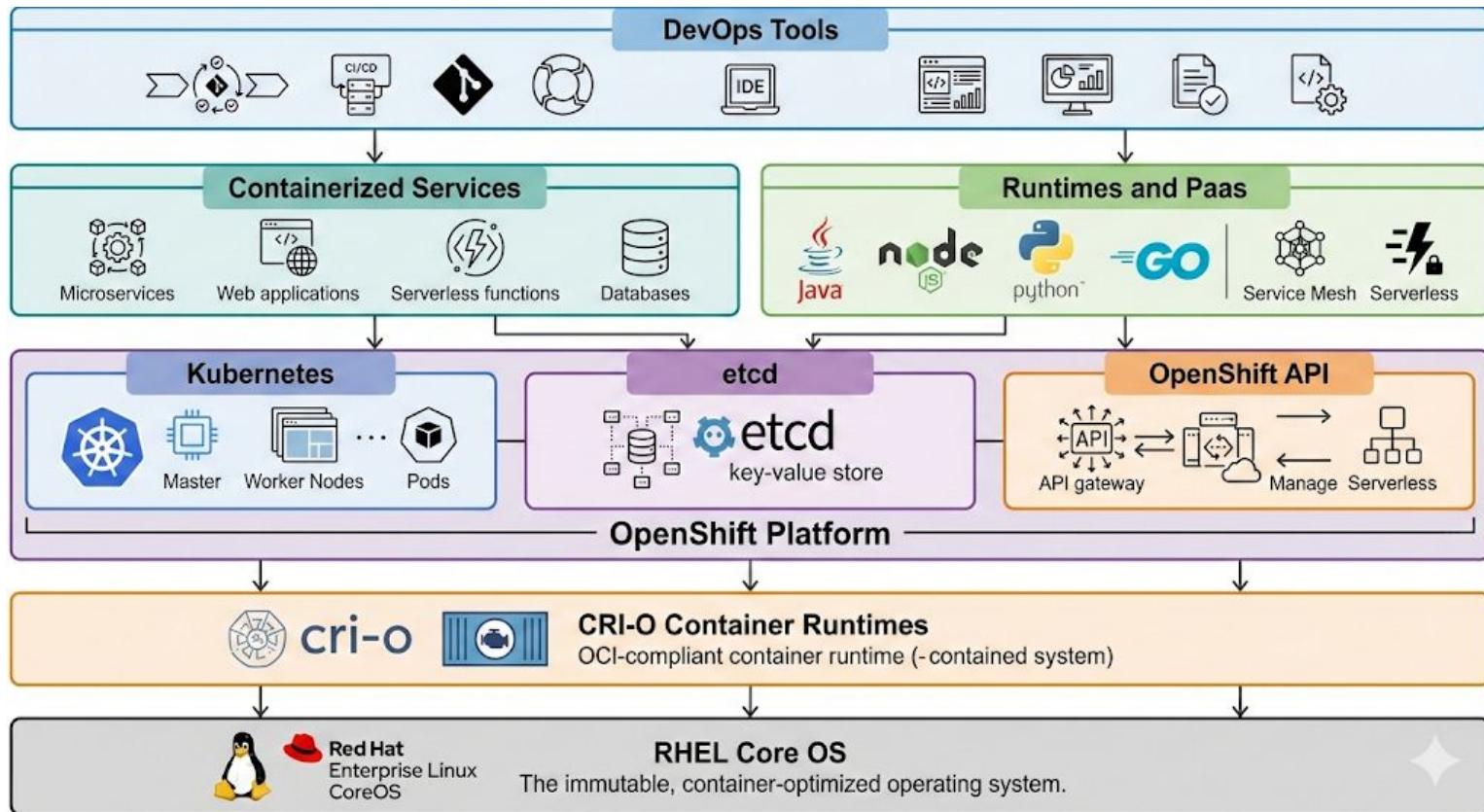


# OpenShift Quick recap

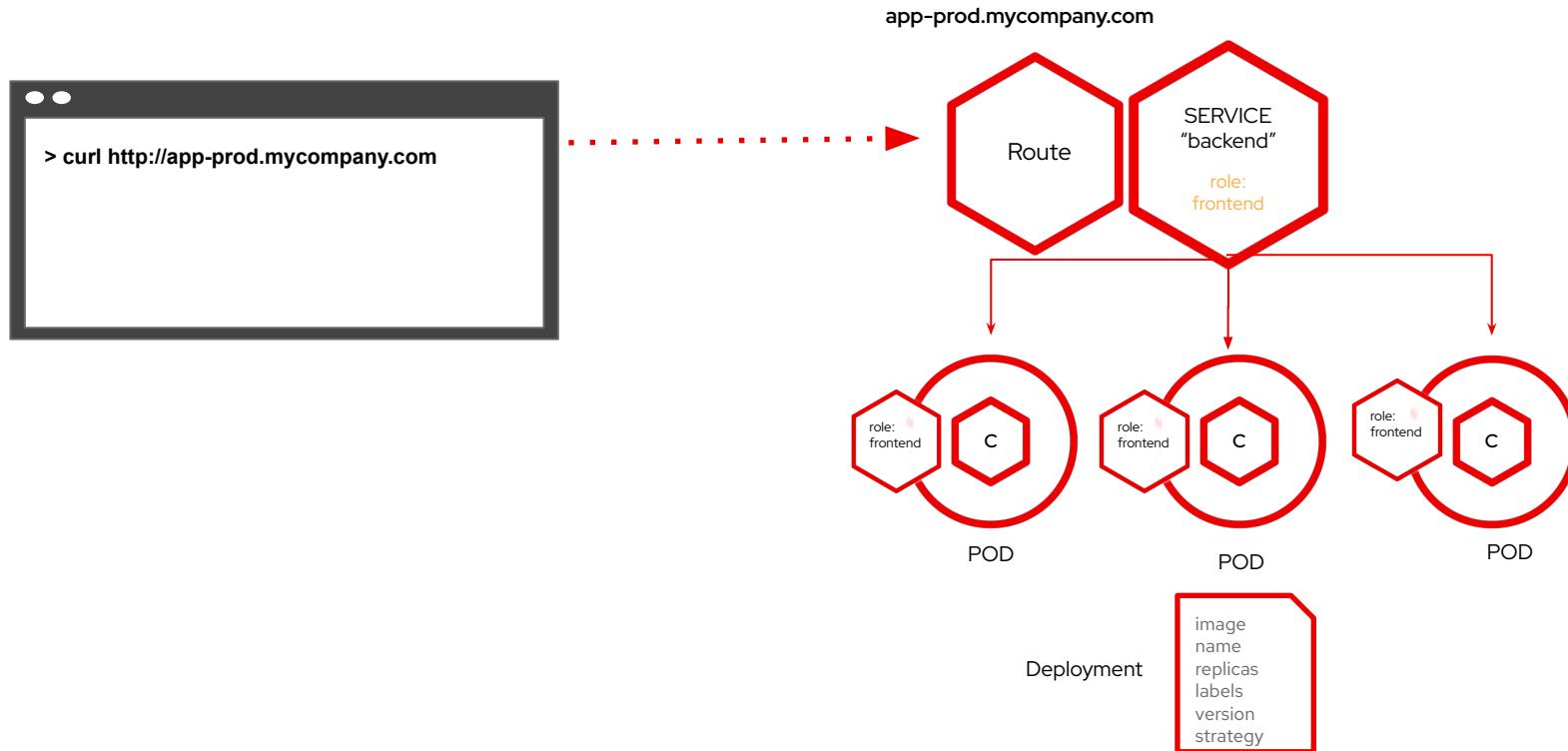
# Kubernetes: Architettura



# OpenShift: Architettura



# Core OpenShift Components





# Resource Management

## Capitolo 1

# Gestione Risorse OpenShift

## Gestione "Imperative"

- ▶ Linea di comando per creare, modificare o cancellare risorse
- ▶ Comandi eseguiti mediante il tool oc

```
oc create deployment myapp -image=nginx
```

## Gestione "Declarative"

- ▶ Usa YAML o JSON per definire lo stato di una Risorsa
- ▶ Applicato con: oc apply -f <file>.yaml



# Creazione dei Manifest Kubernetes

**Esistono diverse alternative:**

- ▶ Edit manuale ( anche mediante templates esistenti)
- ▶ Mediante comandi imperativi:

```
oc create deployment hello -o yaml --image nginx:v1.0 --save-config  
--dry-run=client > deployment.yaml
```

- ▶ Usando tools (Kustomize/Helm) per generare risorse da templates



## Stato desiderato di una Risorsa

- ▶ `oc apply` crea o modifica le risorse OpenShift/Kubernetes contenute in un manifest
- ▶ Usa un approccio “declarative” – si definisce lo stato richiesto e OpenShift sincronizza la risorsa con quello stato.

```
oc apply -f <filename.yaml>
```



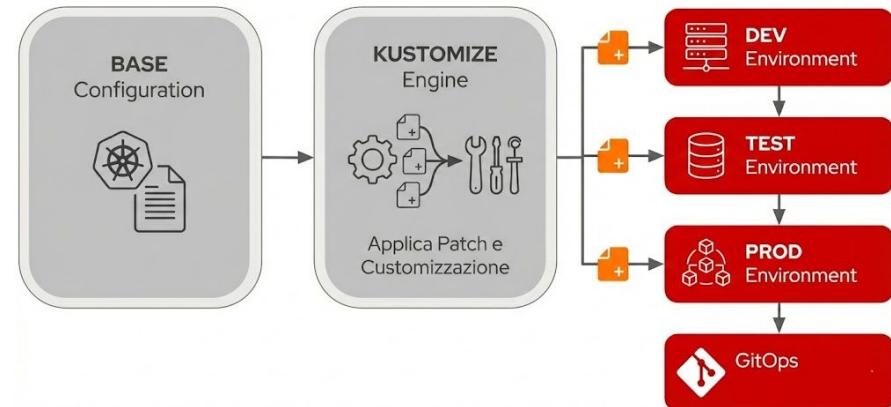
Come funziona?

- ▶ Se la risorsa non esiste viene creata.
- ▶ Se è già esistente, viene aggiornata in base alla definizione.
- ▶ Soltanto i campi presenti nel Manifest vengono aggiornati.

# Kustomize

## Cosa è Kustomize ?

- ▶ Tool open-source per semplificare la configurazione di Kubernetes
- ▶ Ideale per creare ambienti multipli (dev, test, prod) partendo da una base comune
- ▶ Consente di applicare patch e customizzazione per ogni ambiente
- ▶ Utilizzato per processi di GitOps



# Kustomize: struttura e Overlay

## Come si struttura un progetto con Kustomize ?

- ▶ La directory base contiene le risorse di partenza
- ▶ Ogni layer deve contenere il file **kustomization.yaml** per definire:
  - La risorsa base di riferimento
  - Le customizzazioni da applicare

```
my-app/
└── base
    └── base.yaml
    └── kustomization.yaml
└── overlays
    └── production
        └── kustomization.yaml
            └── patch.yaml
```

```
# base/kustomization.yaml

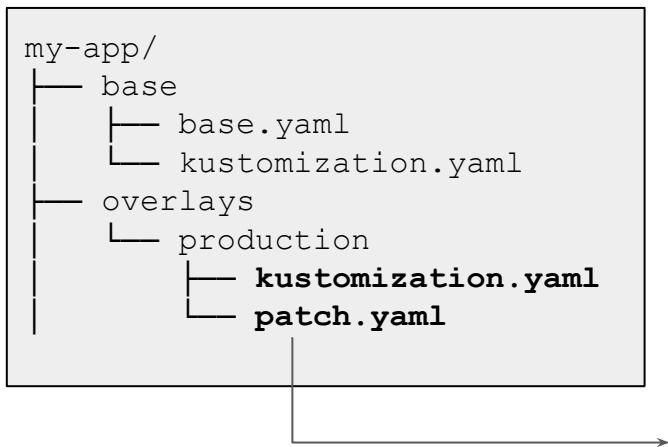
namespace: dev-environment
namePrefix: dev-

resources:
  - base.yaml
```

# Esempio: applicare l'Overlay di Production



- ▶ L'overlay Production importa quello base
- ▶ Aggiunge la sua configurazione specifica
- ▶ Esempio: aggiunge repliche dei Pods



```
resources:
  - ../../base

patches:
  - patch.yaml
```

```
apiVersion: apps/v1
kind: Pod
metadata:
  name: my-app
spec:
  replicas: 3
```



# Esempio: applicare l'Overlay di Production



Attenzione! La specifica del numero di repliche è gestita dal Deployment - non dal Pod !

```
my-app/
  └── base
      └── base.yaml
      └── kustomization.yaml
  └── overlays
      └── production
          └── kustomization.yaml
          └── patch.yaml
```

```
resources:
  - ../../base

patches:
  - patch.yaml
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-app
spec:
  replicas: 3
```





# Templates & Helm Charts

## Capitolo 2

# OpenShift Templates

Per la gestione di progetti più complessi esistono diverse opzioni disponibili in OpenShift. La soluzione built-in in OpenShift:

OpenShift Templates: Utilizzano blocchi JSON/YAML per definire risorse con Parametri.

- ▶  Disponibili out-of-the-box
- ▶  Semplici da usare e da condividere
- ▶  Gestiscono un numero variabile di risorse
- ▶  Customizzazione mediante Parametri

- ▶



# OpenShift Templates: Comandi

Verifica dei templates disponibili:

```
oc get templates -n openshift
```

Creazione di un Template:

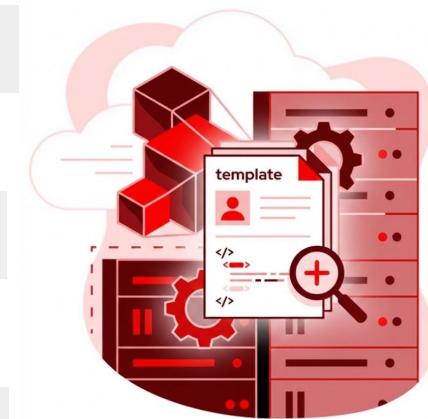
```
oc create -f <template-file.yaml> -n <namespace>
```

Process dei Parametri di un Template:

```
oc process <template-name> -p PARAM=value
```

Creazione di una applicazione da Template:

```
oc new-app --template=<template-name> -p PARAM=value
```





## Kubernetes Package manager



### Project Overview

- <https://helm.sh/>
- <https://github.com/helm/helm>



**CLOUD NATIVE**  
COMPUTING FOUNDATION

### Top level CNCF Project

- 2016 - Joined CNCF
- 2020 - Graduated status



### Active development community

- 13,000+ contributors
- 1,700+ contributing companies
- 9,500+ code commits



# Componenti di Helm



## CLI

Il tool *helm* consente di interagire con l'ecosistema Helm



## Templates

Definiscono in modo dinamico la creazione di Risorse Kubernetes



## Charts

I Packages che descrivono una applicazione Kubernetes



## Values

Gestisce la configurazione/variabili che saranno utilizzate nei Templates

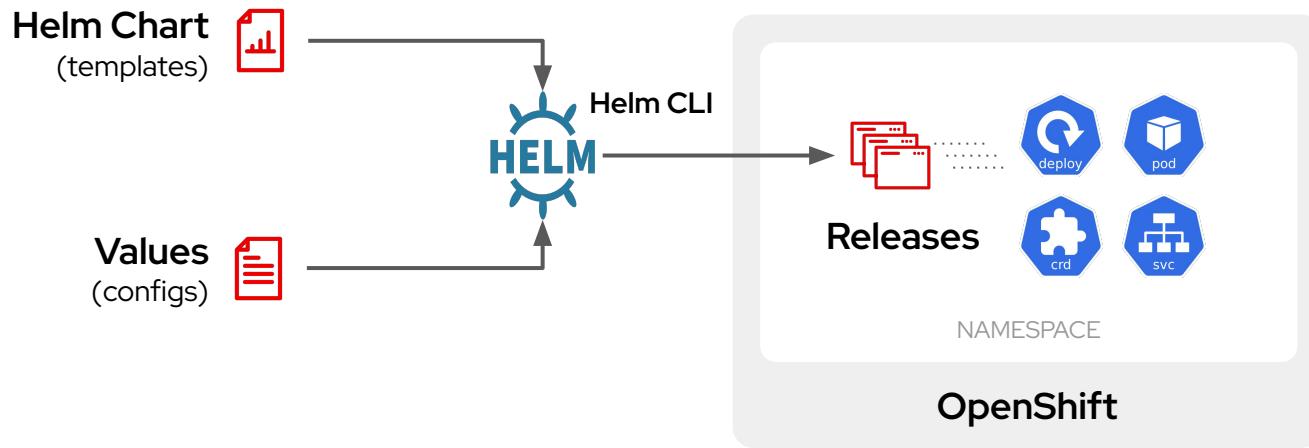


## Revisions

E' possibile avere più versioni (revisions) di un Chart Helm



# Helm Workflow

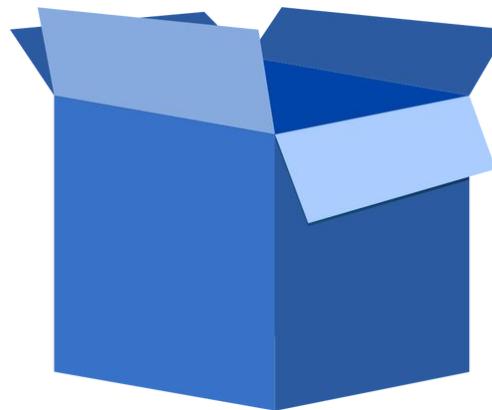


## Helm Chart



Elemento Fisico: I files che descrivono delle risorse Kubernetes

## Helm Release



Elemento logico: una istanza di un Chart dopo l'installazione /aggiornamento

# Creazione di un Helm Chart

- Bootstrap del Chart

```
$ helm create mychart
```

- Deploy del Chart “mychart”

```
$ helm install mychart
```

```
mychart/
Chart.yaml          # Information about the chart
LICENSE            # OPTIONAL: Chart license
README.md          # OPTIONAL: README file
values.yaml         # The default configuration values
values.schema.json # OPTIONAL: A JSON Schema for values
charts/             # Dependency charts
crds/               # Custom Resource Definitions
templates/          # Directory of templates
templates/NOTES.txt # OPTIONAL: Usage notes
```

Helm Chart Directory Structure

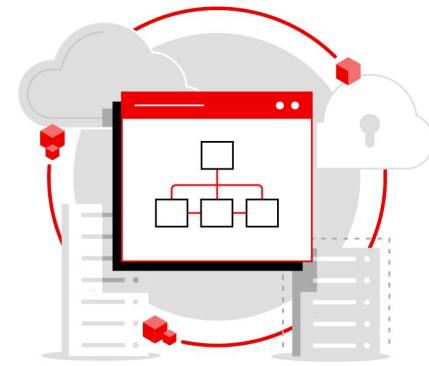
# Chart.yaml

- ▶ Definisce i metadati del chart Helm
- ▶ Dichiara le dipendenze dell'applicazione
- ▶ Permette di impacchettare e versionare il chart per la distribuzione
- ▶ Garantisce deployment riproducibili bloccando versioni specifiche
- ▶ Funziona come punto di ingresso per installazione o aggiornamento

```
apiVersion: v2
name: mychart
version: 0.1.0
description: A Helm chart for Kubernetes
type: application

dependencies:
  - name: jenkins
    version: 2.5.0

appVersion: "1.16.0"
```



# Helm Charts - comandi principali

## Installazione di un Helm Chart

```
$ helm install -f <values_file> ./mychart
```

- Elenco dei Charts

```
$ helm list
```

- Upgrade di una release

```
$ helm upgrade <release_name> <chart>  
--set version=1.1
```

- Rollback di una release

```
$ helm rollback <release> <revision>
```

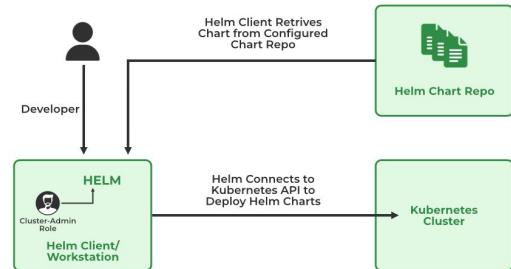
- Uninstall di un Chart

```
$ helm uninstall <release>
```

# Utilizzo dei Repositories negli Helm Charts

Helm semplifica il rilascio di applicazioni su Kubernetes tramite l'utilizzo di Repositories esterni. Il flusso tipico prevede:

- Aggiungere un repository da cui scaricare i charts,
- Installare un'applicazione specifica scegliendo il chart e i valori di configurazione.



```
# Aggiunge un nuovo repository Helm locale, da cui scaricare i chart.  
$ helm repo add do280-repo http://helm.ocp4.example.com/charts  
  
# Installa l'applicazione Etherpad dal repository, usando la configurazione  
$ helm install example-app do280-repo/etherpad -f values.yaml
```

# Artifact Hub

Artifact HUB  ?

STATS SIGN UP SIGN IN Show: 20

1 - 20 of 8476 results

FILTERS

Official   
 Verified publishers

KIND

Helm charts (7586)  
 Krew kubectl plugins (371)  
 OLM operators (252)  
 Tekton tasks (135)  
 Containers images (38)  
 Helm plugins (30)  
 Falco rules (23)  
 Keptn integrations (21)  
 Tinkerbell actions (11)  
 OPA policies (5)

kube-prometheus-stack ★ 273 Helm chart   
ORG: Prometheus REPO: prometheus-community Updated 2 days ago  
VERSION: 34.6.0 APP VERSION: 0.55.0  
kube-prometheus-stack collects Kubernetes manifests, Grafana dashboards, and Prometheus rules combined with docum...  
 Official In Production Verified Publisher Images Security Rating

ingress-nginx ★ 241 Helm chart   
ORG: Kubernetes REPO: ingress-nginx Updated a month ago  
VERSION: 4.0.18 APP VERSION: 1.1.2  
Ingress controller for Kubernetes using NGINX as a reverse proxy and load balancer  
 In Production Images Security Rating

Artifact Hub: Discover & launch great Kubernetes-ready apps

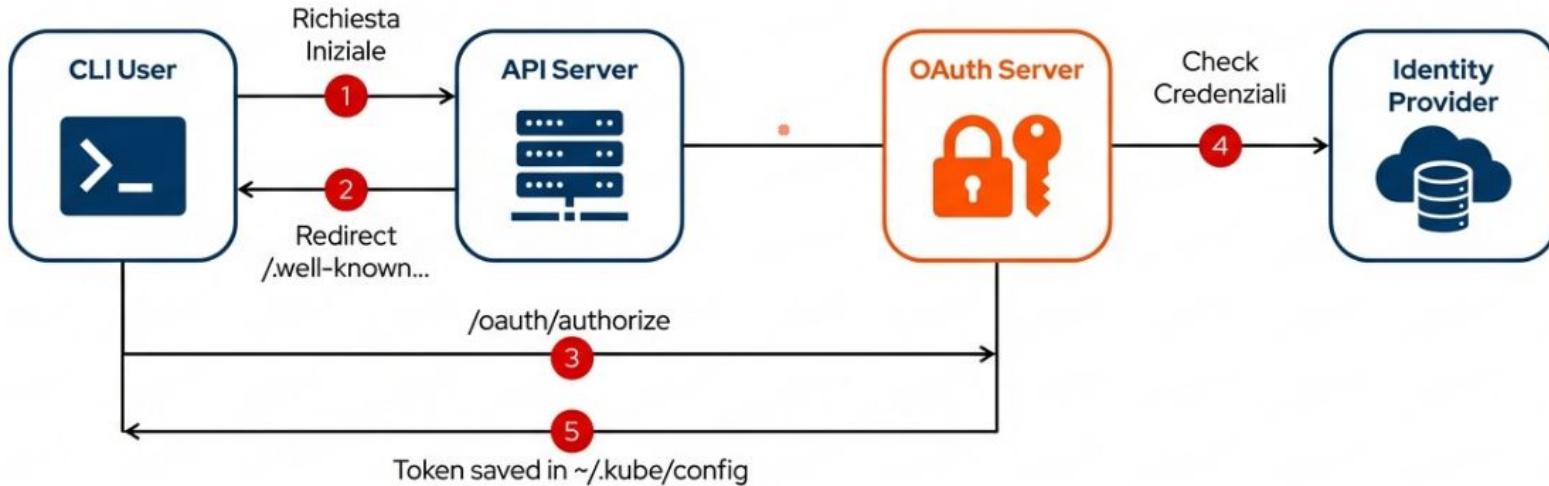


# Openshift Authentication & Authorization

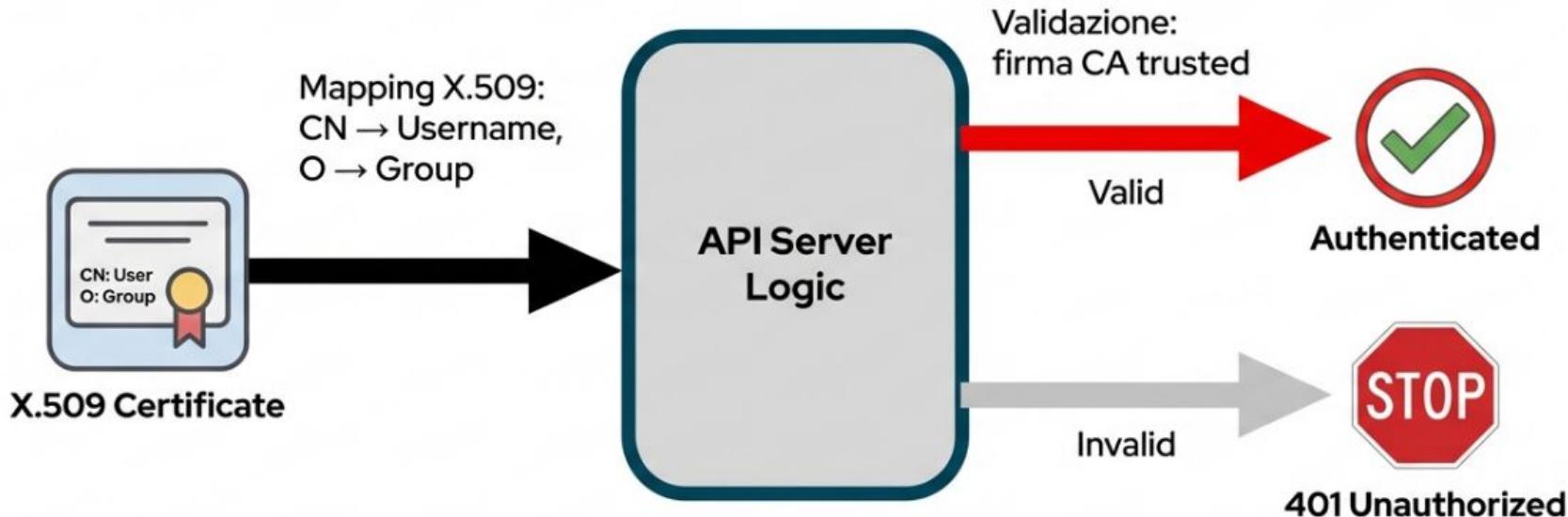
## Capitolo 3

# OpenShift Authentication - oc login

```
oc login -u user -p password https://api.ocp4.example.com:6443
```

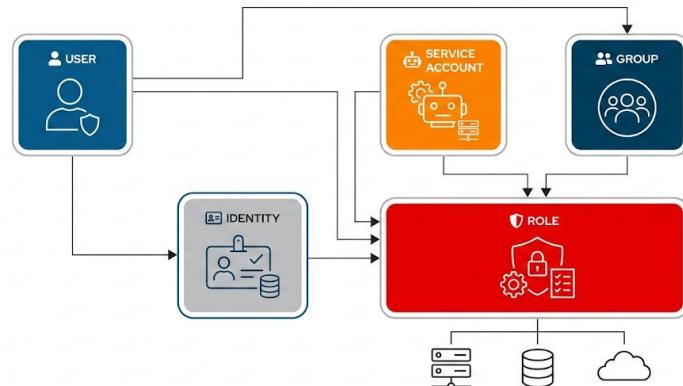


## OpenShift Authentication - Client Certificate



# Utenti e Gruppi in OpenShift

- ▶ **User:** Rappresenta un attore che interagisce con l'API; i permessi sono assegnati direttamente o tramite appartenenza a gruppi.
- ▶ **Identity:** Registra i dettagli delle autenticazioni utente riuscite, incluso l'Identity Provider
- ▶ **Service Account:** Consente alle applicazioni di accedere all'API senza usare credenziali personali
- ▶ **Group:** Insieme di utenti per semplificare l'assegnazione dei permessi.
- ▶ **Role:** Specifica le operazioni consentite sulle risorse ed è assegnata a utenti, gruppi o service accounts.



# Service Account



**Identità per l'Automazione**  
Account dedicati esclusivamente a pod, applicazioni e pipeline, non a utenti umani.



**Interazione API Server**  
Consente alle applicazioni di dialogare programmaticamente con il cuore del cluster in modo sicuro.

## Caratteristiche e vantaggi



**Autenticazione basata su Token**



**Isolamento per Namespace**



**Ciclo di Vita Indipendente**

Utente vs Service Account		
Caratteristica	Account Utente	Service Account
<b>Destinatario</b>	Sviluppatori / Admin	Pod / Pipeline / App
<b>Autenticazione</b>	Credenziali umane	Token crittografici
<b>Ambito</b>	Livello Cluster	Livello Namespace

# OpenShift Identity Providers

- ▶ **HTPasswd**: valida username/password tramite un secret generato con htpasswd.
- ▶ **OpenID Connect**: integra un identity provider OIDC tramite Authorization Code Flow.
- ▶ **Keystone**: consente autenticazione condivisa con OpenStack Keystone v3.
- ▶ **LDAP**: valida username/password con un server LDAPv3.
- ▶ **GitHub**: usa GitHub/GitHub Enterprise come identity provider OAuth.



# Httpd Provider Example

```

apiVersion: config.openshift.io/v1
kind: OAuth
metadata:
  name: cluster
spec:
  identityProviders:
    - name: my_htpasswd_provider
      mappingMethod: claim
      type: HTPasswd
      htpasswd:
        fileData:
          name: htpasswd-secret

```

 Apache HTTP Server Project

Apache > HTTP Server > Documentation > Version 2.4 > Programs

**htpasswd - Manage user files for basic authentication**

Available Languages: [en](#) | [fr](#) | [ko](#) | [tr](#)

htpasswd is used to create and update the flat-files used to store usernames and password for basic authentication of HTTP users. If htpasswd cannot access a file, such as not being able to write to the output file or not being able to read the file in order to update it, it returns an error status and makes no changes.

Resources available from the Apache HTTP server can be restricted to just the users listed in the files created by htpasswd. This program can only manage usernames and passwords stored in a flat-file. It can hash and display password information for use in other types of data stores, though. To use a DBM database see [dbmmanage](#) or [htdbm](#).

htpasswd hashes passwords using either bcrypt, a version of MD5 modified for Apache, SHA-1, or the system's crypt() routine. SHA-2-based hashes (SHA-256 and SHA-512) are supported for crypt(). Files managed by htpasswd may contain a mixture of different encoding types of passwords; some user records may have bcrypt or MD5-hashed passwords while others in the same file may have passwords hashed with crypt().

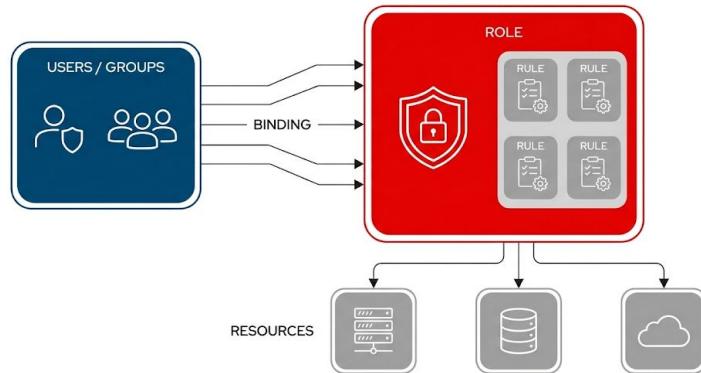
This manual page only lists the command line arguments. For details of the directives necessary to configure user authentication in [httpd](#) see the Apache manual, which is part of the Apache distribution or can be found at <http://httpd.apache.org/>.

# OpenShift RBAC

**Role-based access control (RBAC)**: controlla l'accesso alle risorse tramite:

- ▶ **Rule**: Azioni consentite.
- ▶ **Role**: Insieme di Regole.
- ▶ **Binding**: Associazione di utenti o gruppi a un ruolo

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



# OpenShift Roles

## Cluster Roles (Global Scope):

```
oc adm policy --help
```

- **cluster-admin** – Controllo totale del cluster.
- **cluster-reader** – Accesso read-only alle risorse del cluster.
- **admin** – Amministra / modifica un project (namespace).
- **edit** – Permesso di modifica/creazione di risorse ma non di gestire RBAC / Quotas.
- **view** – Accesso read-only alle risorse tranne secrets / RBAC.

## Namespace (Project) Roles

```
oc policy --help
```

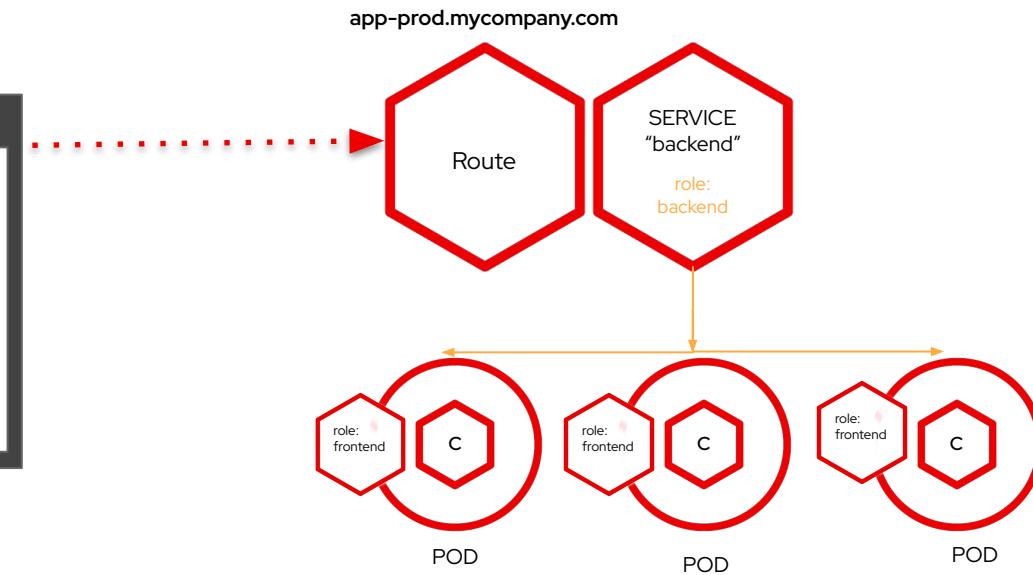
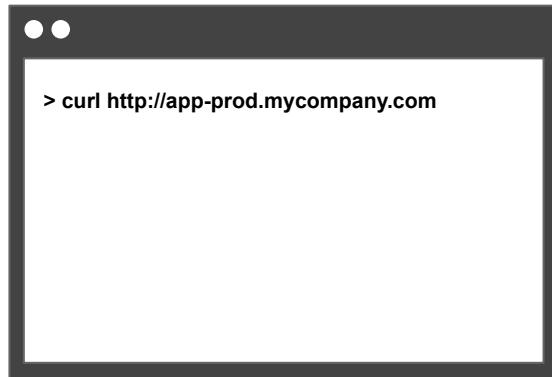
- **self-provisioner** – Consente la creazione di nuovi project.
- **registry-editor** – Gestisce immagini nel registro interno
- **registry-viewer** – Visualizza immagini nel registro interno
- **system:image-puller** – Può eseguire il pull delle immagini dal registro interno



# Securing OpenShift Routes

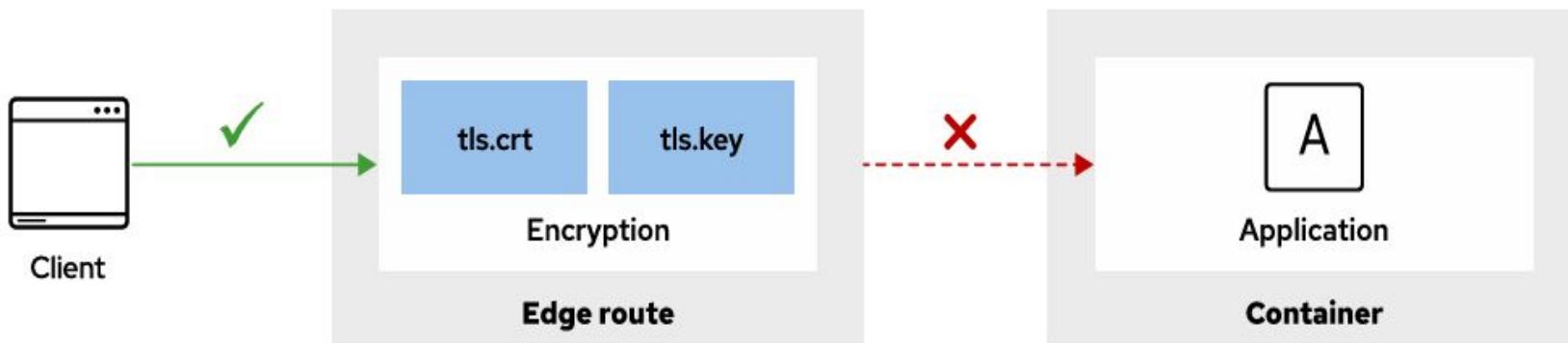
## Capitolo 4

# Le Routes permettono accesso esterno ai servizi del cluster via HTTP/HTTPS



# OpenShift Edge Routing 🔒

- ▶ Il Router OpenShift termina la connessione TLS direttamente all'ingresso del cluster.
- ▶ Il Certificato Pubblico è gestito dal Router.
- ▶ Adatto a scenari in cui l'applicazione non deve occuparsi della cifratura, riducendo complessità.



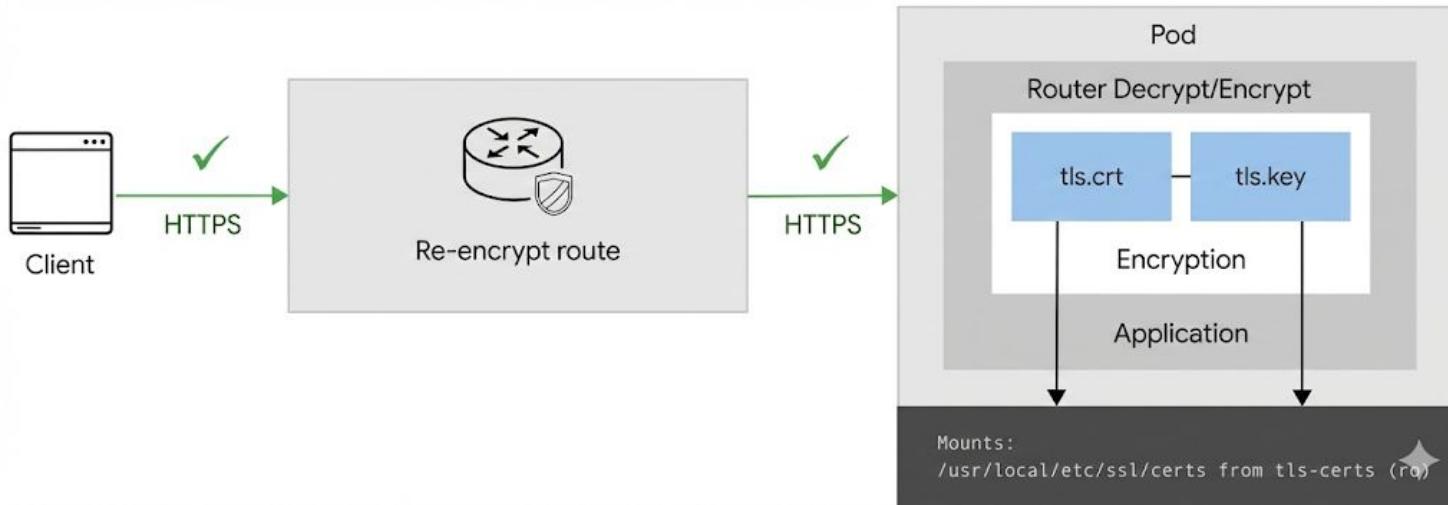
# OpenShift Passthrough Routing

- ▶ Il Router OpenShift inoltra direttamente il traffico TLS ai Pod senza terminare la connessione.
- ▶ Il workload deve avere a bordo i Certificati.
- ▶ Ideale quando si vuole preservare la cifratura end-to-end e delegare la sicurezza all'applicazione.



## OpenShift Re-encryption Routing

- ▶ Garantisce la sicurezza end-to-end: il traffico HTTPS viene terminato al router e ri-cifrato verso i Pod.
- ▶ Consente di usare certificati diversi: il router presenta un Certificato Pubblico, mentre l'applicazione usa il proprio Certificato Interno.



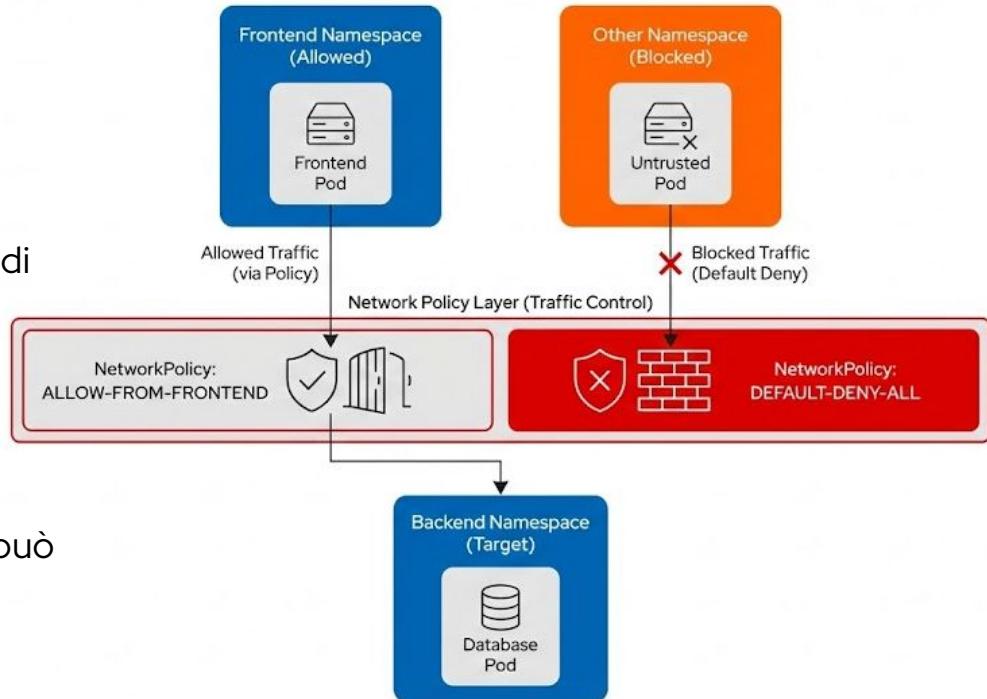


# Network Policies

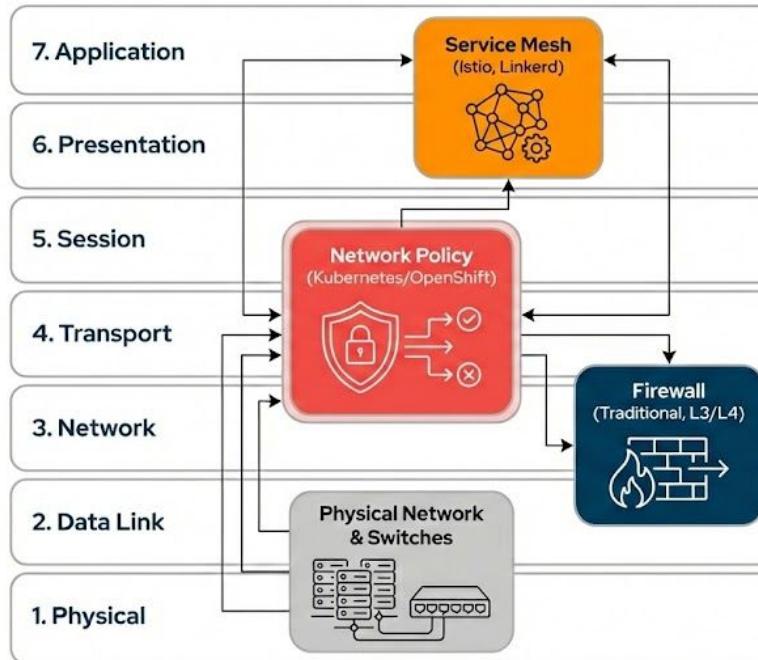
## Chapter 4

# Network Policies

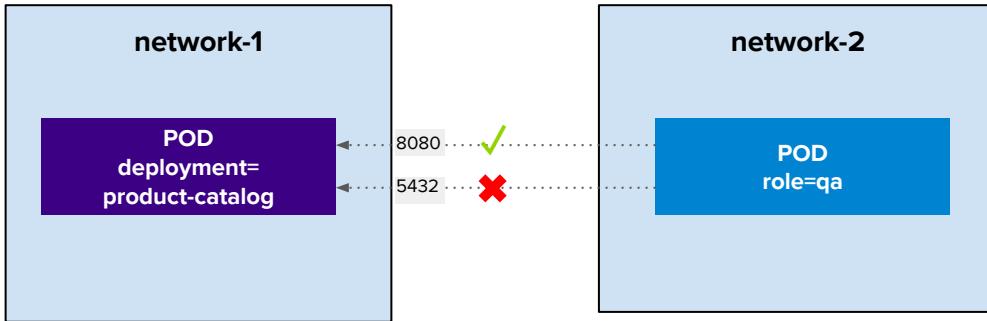
- ▶ Strumento chiave per isolare applicazioni all'interno del Cluster
- ▶ Basate su **label**: selezionano Pods e Namespaces tramite labelSelector.
- ▶ Controllo del traffico: definiscono regole di sicurezza a livello di rete.
  -  **Ingress** → specifica chi può connettersi al pod.
  -  **Egress** → definisce dove il pod può connettersi.



# Dove sono le Network Policies nello Stack OSI?



# Esempio di Network Policy



*Bookshelf*

## 4.3. Configure Network Policies

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: network-1-policy
  namespace: network-1
spec:
  podSelector:
    matchLabels:
      deployment: product-catalog
  ingress:
  - from:
    - namespaceSelector:
        matchLabels:
          network: network-2
    podSelector:
      matchLabels:
        role: qa
  ports:
  - port: 8080
    protocol: TCP
```

# Utilizzo di una Policy deny-all

## Senza una policy “deny-all”:

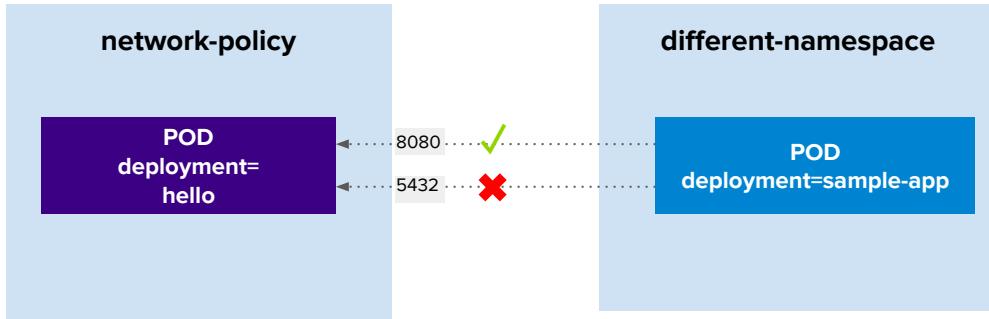
- ▶ I pod non selezionati da alcuna NetworkPolicy possono comunicare liberamente
- ▶ I pod selezionati da una NetworkPolicy possono comunicare solo se il traffico corrisponde alle regole definite.

```
kind: NetworkPolicy
apiVersion:
networking.k8s.io/v1
metadata:
name: default-deny
spec:
podSelector: {}
```

## Con una policy “deny-all”:

- ▶ Tutto il traffico viene bloccato per i pod target, a meno che non sia esplicitamente consentito da regole aggiuntive specifiche.

# Lab 4.4 network-policy



```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: deny-all
spec:
  podSelector: {}
```

Regole di accesso al Pod "hello"

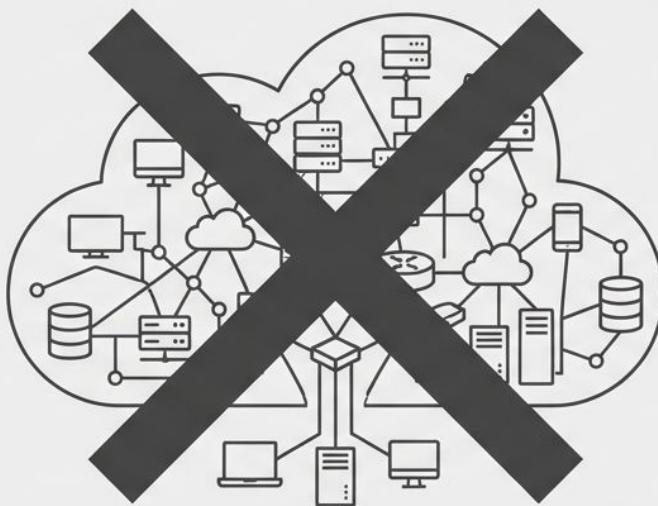
- ✓ Ingress traffic da "sample-app" nel namespace "different-namespace" e port 8080

```
spec:
  podSelector:
    matchLabels:
      deployment: hello
  ingress:
    - from:
        - namespaceSelector:
            matchLabels:
              network: different-namespace
        podSelector:
          matchLabels:
            deployment: sample-app
  ports:
    - port: 8080
      protocol: TCP
```



# Protect Internal Traffic

## Capitolo 4

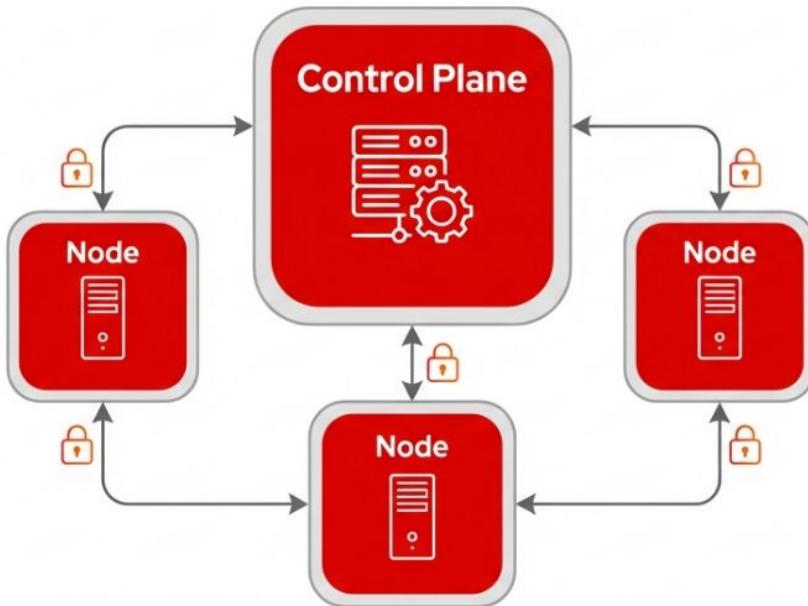


## Zero-trust environments



- ▶ Tutto il traffico è considerato potenzialmente insicuro.
- ▶ Accesso solo se esplicitamente consentito.
- ▶ Crittografia obbligatoria.
- ▶ I Client devono verificare l'autenticità del server.

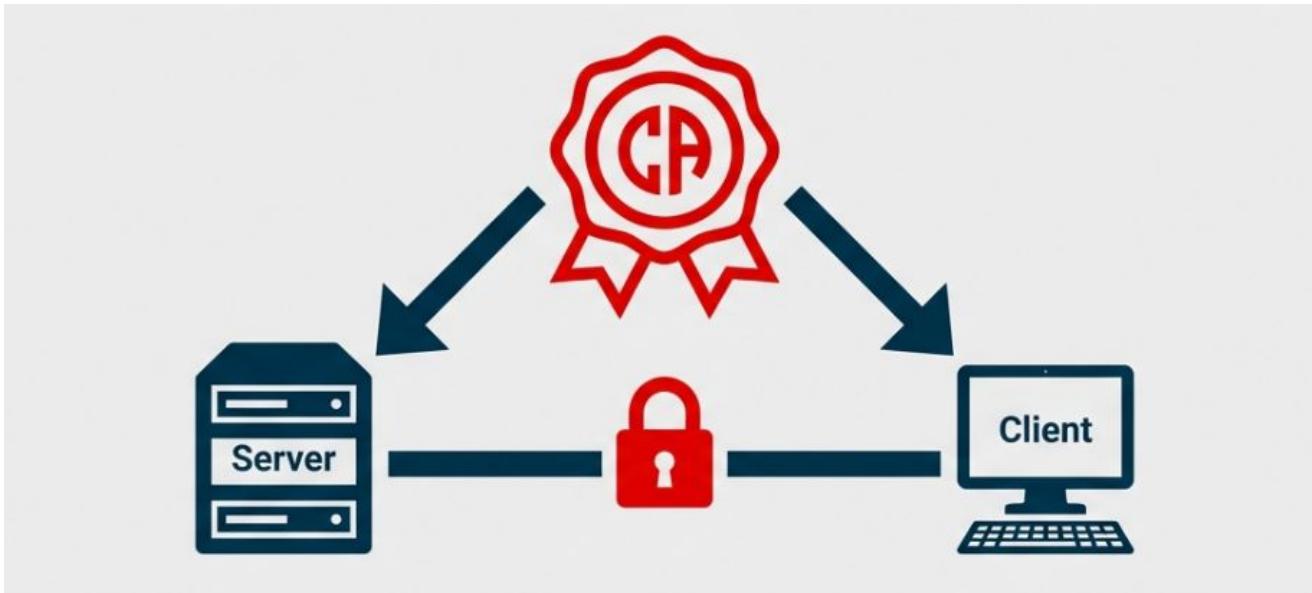
# Sicurezza del Control Plane



OpenShift Cifra il traffico di rete tra i nodi e il control plane

- ▶ 🚫 Impedisce l'accesso esterno al traffico interno
- ▶ 🛡️ Sicurezza più forte rispetto a Kubernetes (che non ha cifratura predefinita)
- ▶ ⚙️ La verifica e la cifratura a livello applicativo non sono applicate di default

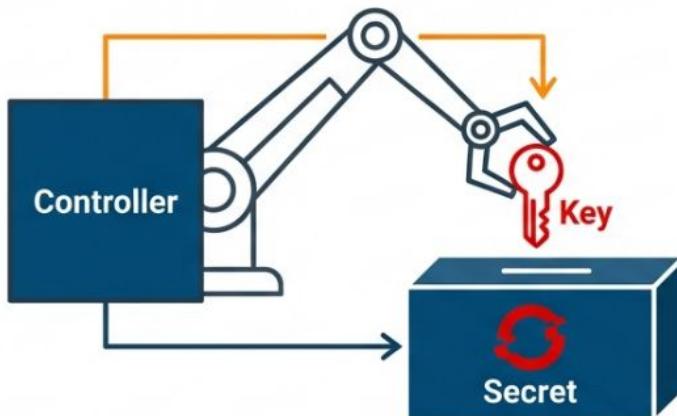
## Ruolo della Certificate Authority



- ▶ Una CA fidata deve firmare i certificati per la cifratura
- ▶ Le applicazioni possono verificare l'autenticità usando certificati firmati dalla CA

# Strategie per cifrare il traffico interno

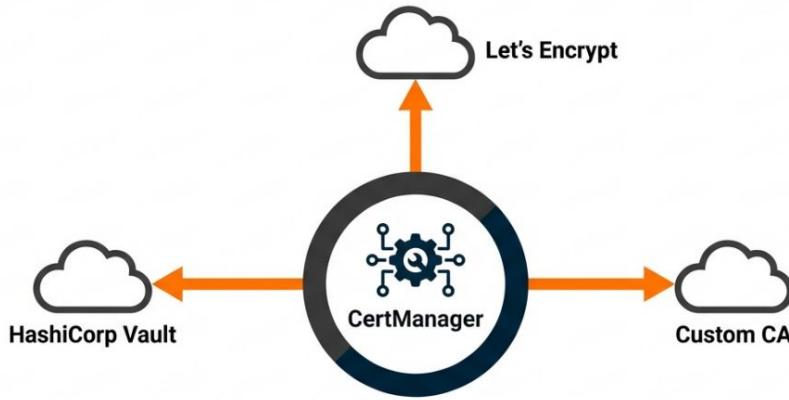
## Server Certificates (service-ca controller)



- ▶ Genera automaticamente certificati firmati per i servizi interni
- ▶ Memorizza i certificati nei secrets, che le applicazioni possono montare
- ▶ I client devono fidarsi della CA del service-ca controller per l'autenticazione

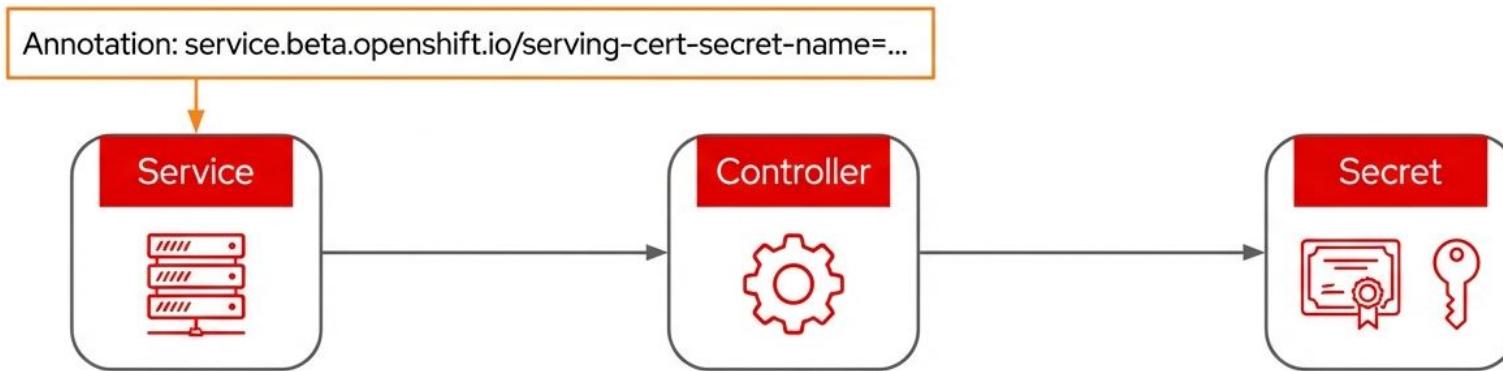
# Strategie per cifrare il traffico interno

## CertManager Operator



- ▶ 🔑 Automatizza l'emissione e il rinnovo dei certificati usando CA interne o esterne
- ▶ 🌐 Supporta Let's Encrypt, HashiCorp Vault e CA personalizzate
- ▶ 🔗 Funziona con Kubernetes Ingress, Routes e TLS tra servizi

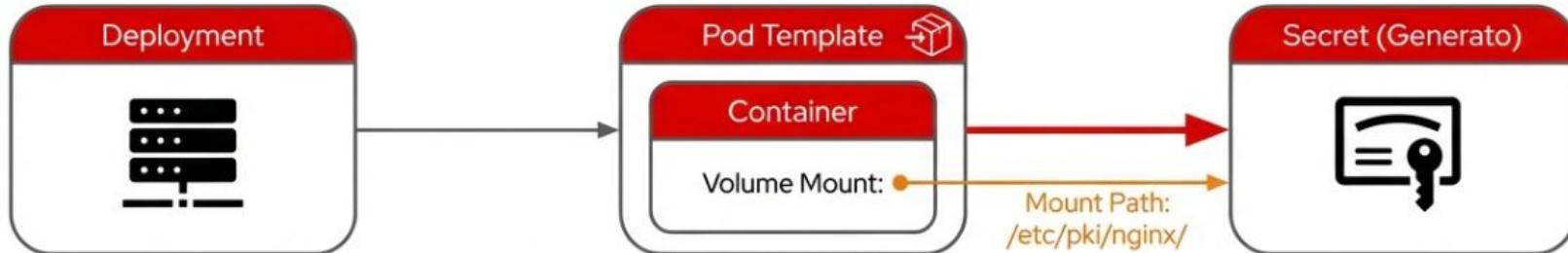
## Workflow dei Server Certificates: Fase 1



### Service-CA Controller

- ▶ Genera e firma certificati di servizio per il traffico interno
- ▶ Crea un secret contenente certificato firmato e chiave

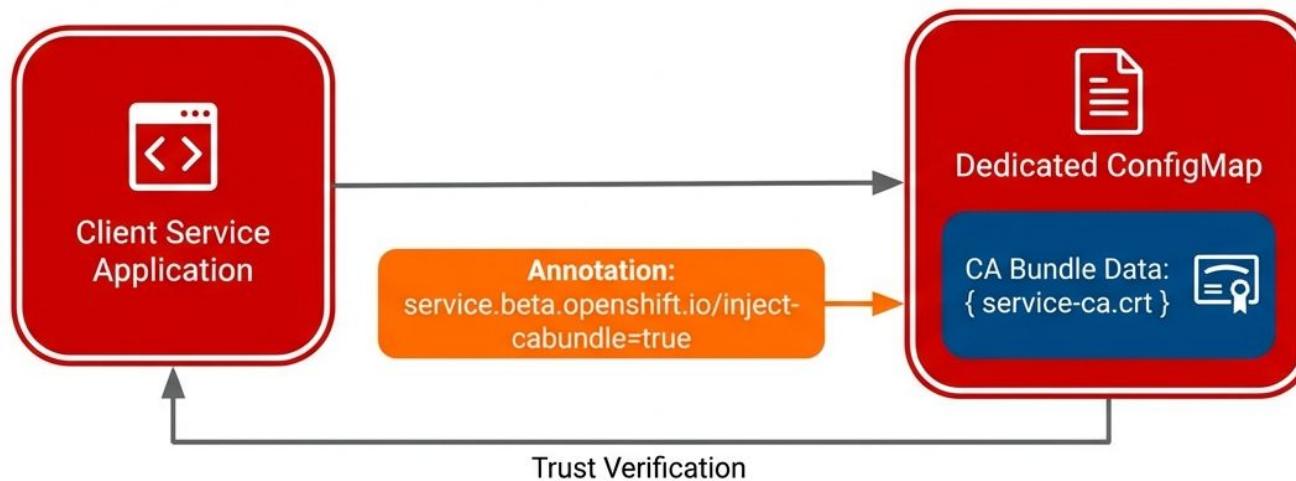
## Workflow dei Server Certificates: Fase 2



### Utilizzo del Certificato

- ▶ I Deployment possono montare il secret come volume
- ▶ I servizi usano il certificato firmato per comunicazioni sicure

## Workflow dei Server Certificates: Fase 3



- ▶ Le applicazioni client devono fidarsi della CA del service-ca controller
- ▶ Garantisce comunicazioni sicure e verificate tra i servizi

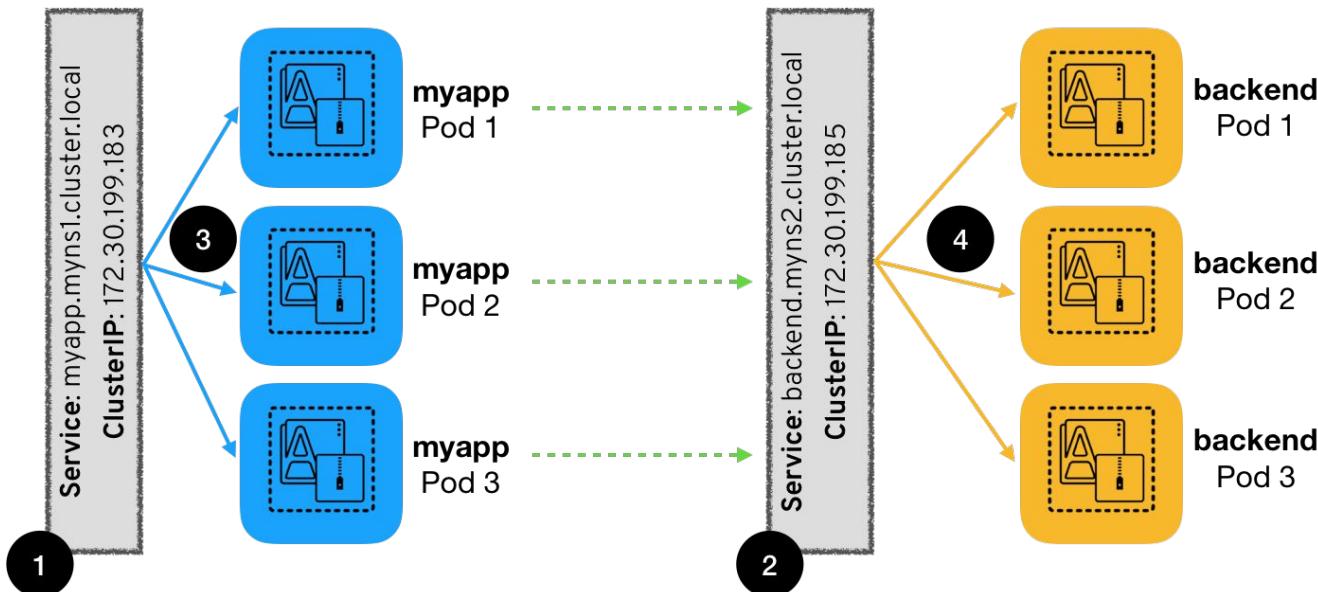


# Exposing non-HTTP Applications

## Chapter 5

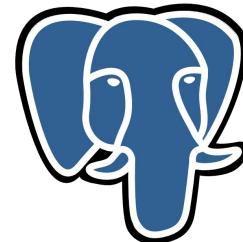
# Service Type

- Di default i Service di OpenShift sono esposti usando un Cluster IP
- Pertanto, sono visibili solo all'interno del Cluster



## Casi d'uso comuni per NON-HTTP Services

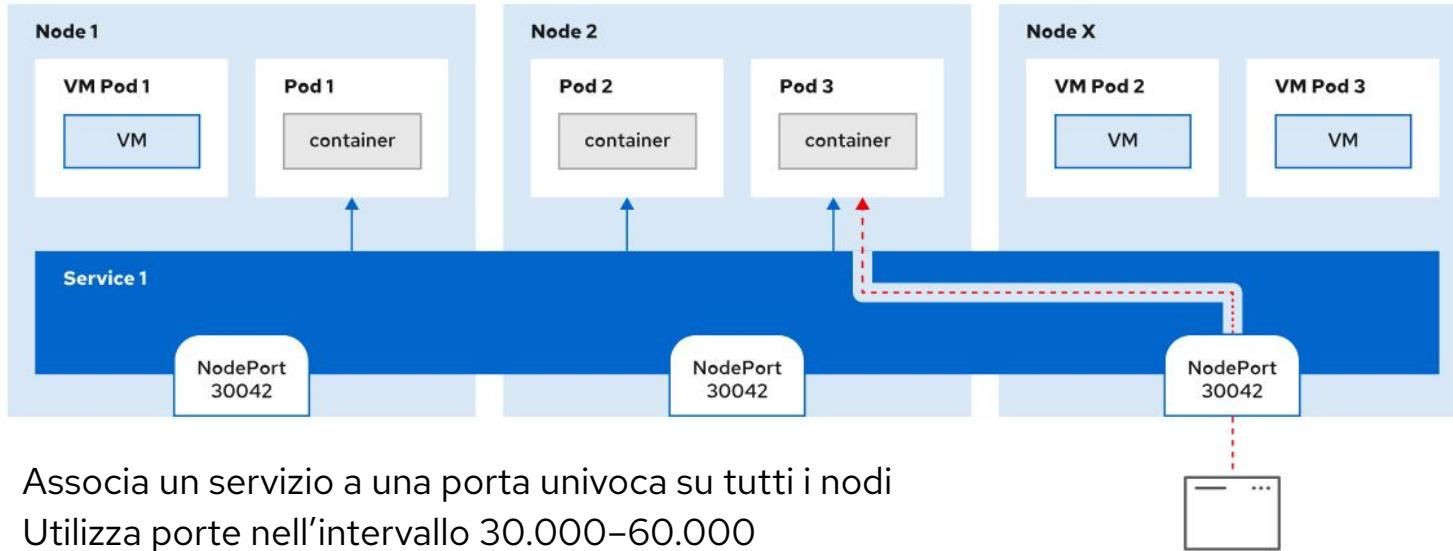
- Databases (PostgreSQL/MySQL)
- Broker MQTT, Kafka, ActiveMQ
- Custom TCP/UDP Services



Postgre<sup>SQL</sup>



# Esposizione Service mediante NodePort

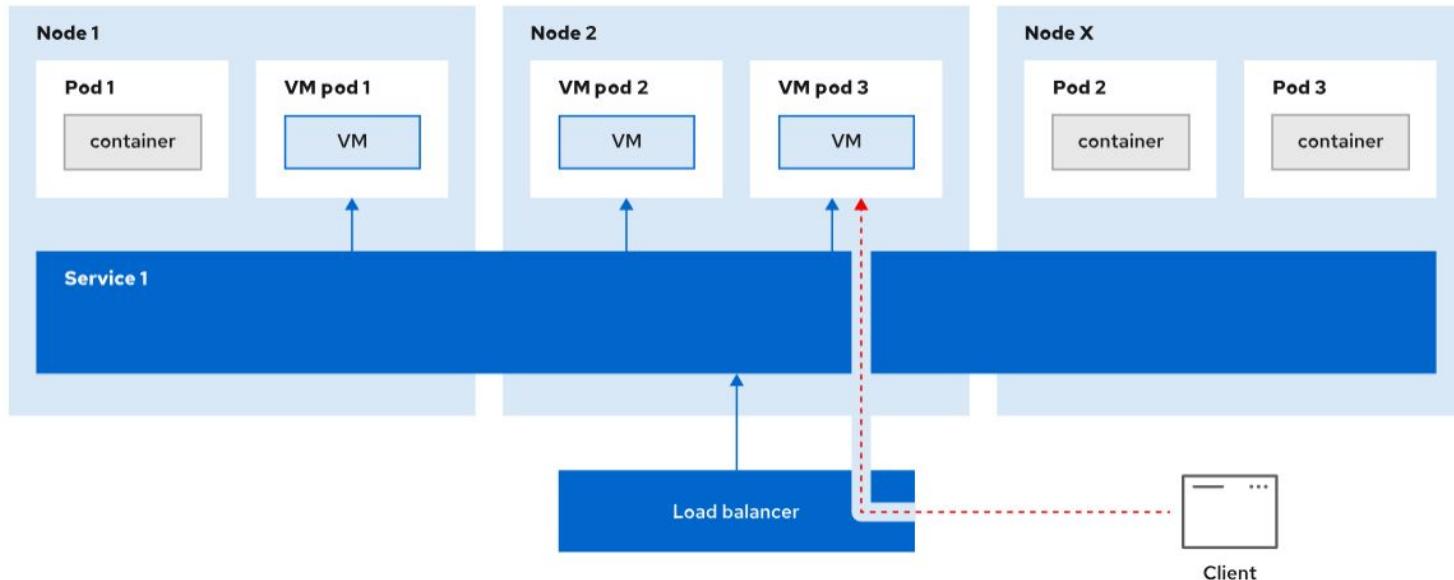


- Associa un servizio a una porta univoca su tutti i nodi
- Utilizza porte nell'intervallo 30.000–60.000
- 🔥 Le regole del firewall devono consentire il traffico sulla porta specifica<sup>client</sup>

✓ Vantaggi: funziona senza bisogno di load balancer esterni

✗ Svantaggi: richiede di conoscere gli IP dei nodi e configurare il firewall

# Esposizione Service mediante LoadBalancer



- Richiedono la disponibilità di un Servizio di LoadBalancing
- I cloud provider forniscono normalmente i propri servizi di load balancer
- MetalLB è un componente che offre load balancing ai cluster che non girano su un cloud provider

# MetalLB Operator



- Può essere utilizzato per bilanciare il traffico verso OpenShift sia *on-premises* che in cloud
- Ha due modalità per annunciare le informazioni di raggiungibilità degli indirizzi IP del load balancer:
  - Layer 2
  - BGP
- Due componenti principali:
  - Controller → uno per cluster
  - Speaker → uno per nodo (DaemonSet)

```
apiVersion: v1
kind: Service
metadata:
  name: mylb
spec:
  ports:
  - name: http
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: myapp
type: LoadBalancer
```



# Multus Networking

## Capitolo 5

# Advanced Networking



## Prestazioni

Necessità di alte prestazioni separati dal traffico di gestione



## Isolamento

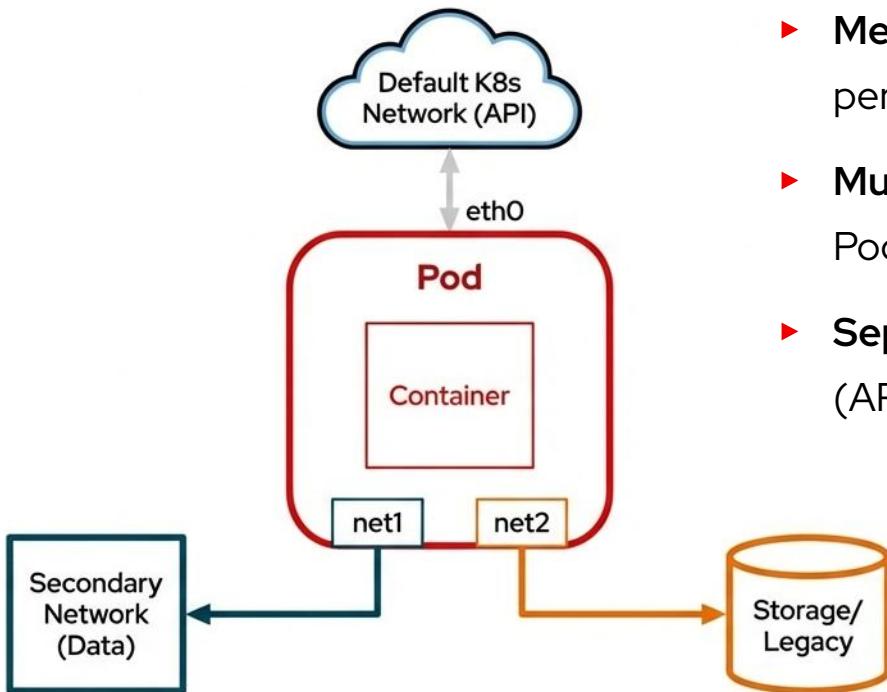
Segregazione del traffico sensibile di alcuni Nodi



## Legacy

Connettività a Network esterne senza usare NAT

## Il motore dell'architettura: Multus CNI



- ▶ **Meta-Plugin:** Openshift utilizza Multus CNI per orchestrare plugin multipli
- ▶ **Multi-Homing:** Permette di collegare un Pod a più interfacce di rete
- ▶ **Separazione:** Divide il traffico di controllo (API K8s) dal traffico dati intensivo

## Pre-requisiti: Preparazione del Nodo



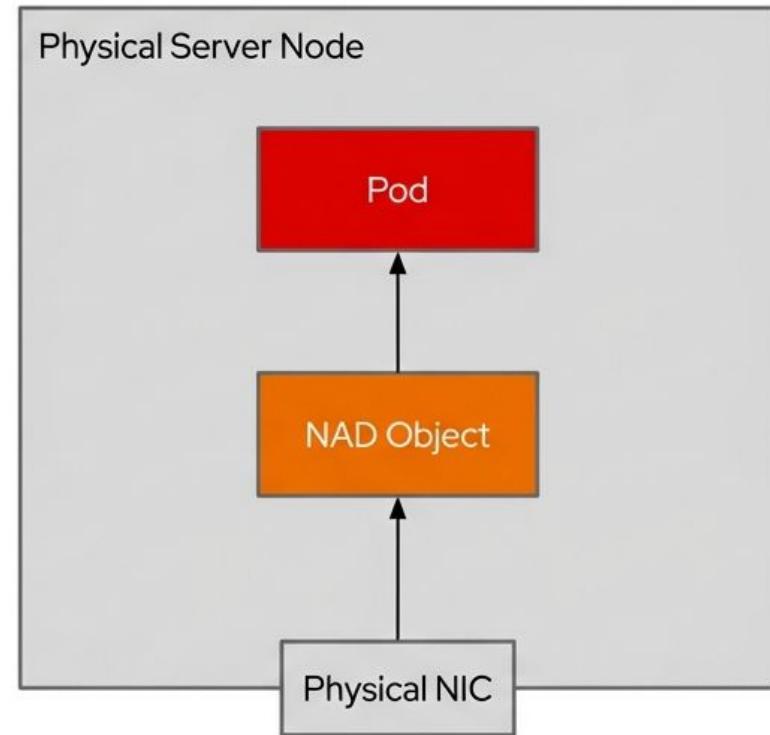
- ▶ **NMState:** Un Operator che consente la gestione dichiarativa delle configurazioni di rete dei nodi del cluster.
- ▶ **Scopo:** Configurare la rete dei nodi fisici o virtuali nell'infrastruttura OpenShift.
- ▶ **Dove opera?** A livello di nodo, configurando NIC, VLAN, bonding, bridge e altre impostazioni di rete del sistema operativo host.

**TO BE  
CONTINUED... ➡**

DO316

# Network Attachment Definition

- La Network Attachment Definition consente di collegare i Workloads a reti esterne
- Modalità:
  - Host device
  - Bridge
  - IPVLAN
  - MACVLAN



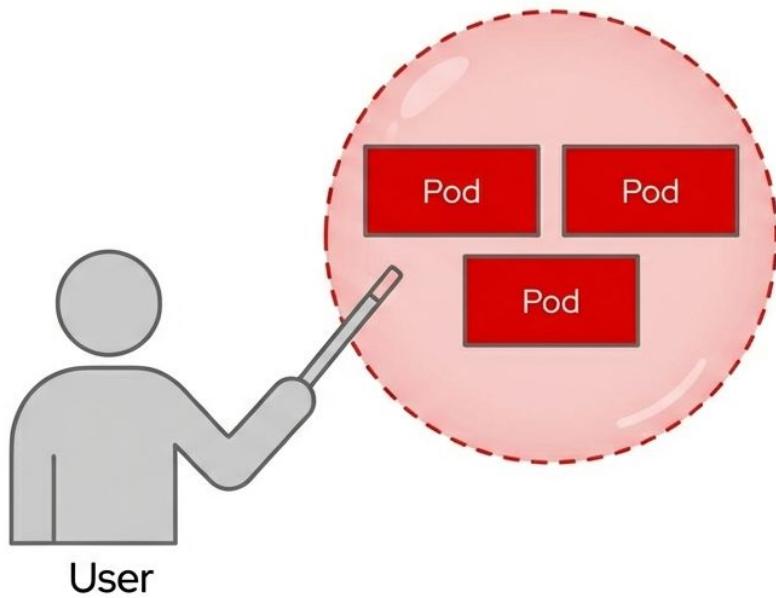
## Configurazione NAD di esempio (Host Device)

```
kind: NetworkAttachmentDefinition
metadata:
  name: example
spec:
  config: |-  
    {  
      "cniVersion": "0.3.1",  
      "type": "host-device",  
      "device": "ens4",  
      "ipam": { "type": "dhcp" }  
    }"
```

Driver CNI

Interfaccia Master

# User Defined Networks (UDN)

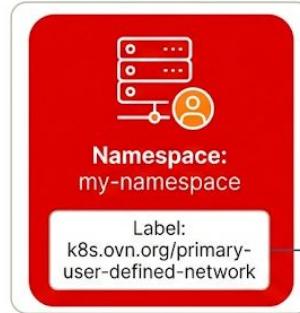


- ▶ Novità di OpenShift 4.18: Gestita da OVN-Kubernetes
- ▶ Non-Admin: Permette agli utenti di creare reti senza privilegi di cluster-admin
- ▶ Isolamento Tenant: Garantisce che il traffico rimanga confinato nel namespace
- ▶ Due tipologie: Primary UDN e Secondary UDN

# Primary UDN

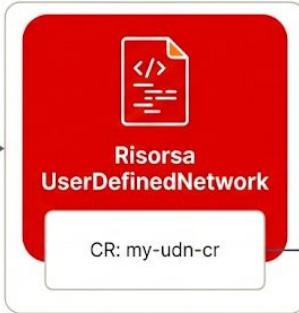
Sostituisce la cluster pod network e tutti i pod del namespace usano quella rete come interfaccia primaria

## 1. Etichetta il Namespace (Obbligatorio)



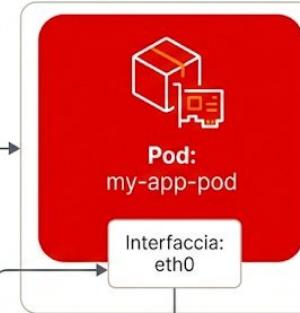
Nota: Se manca questa label alla nascita del namespace, la Primary UDN non funzionerà.

## 2. Definisci la UserDefinedNetwork CR



Crei la risorsa nel namespace appena etichettato.

## 3. Creazione Pod (Automatica)

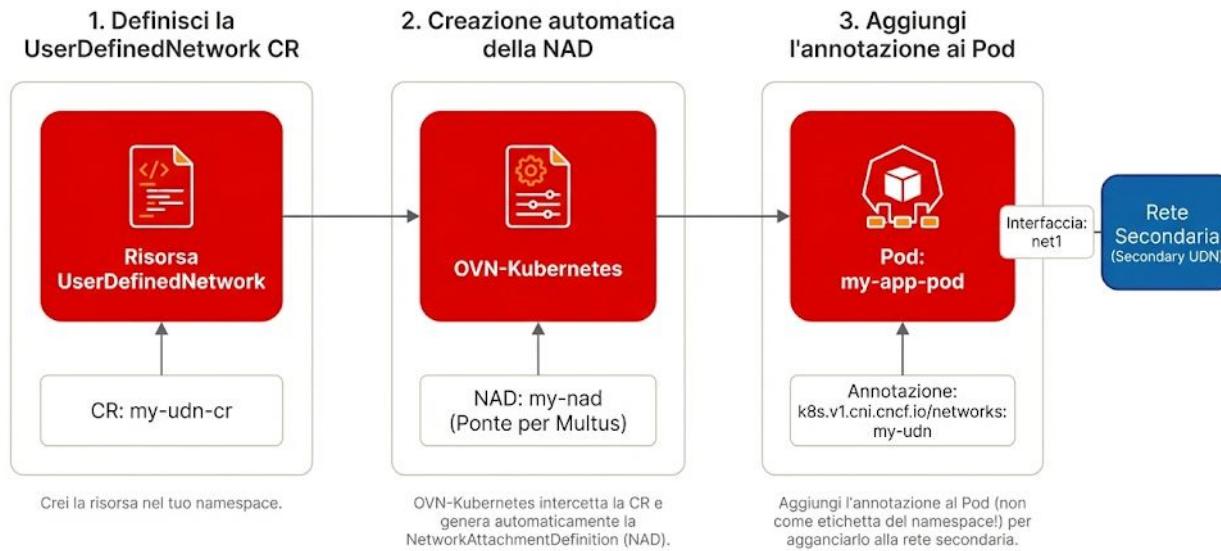


Nessuna annotazione necessaria. Tutti i Pod in questo namespace verranno automaticamente agganciati a questa nuova rete isolata sulla loro Interfaccia primaria eth0.

Rete Isolata  
(Primary UDN)

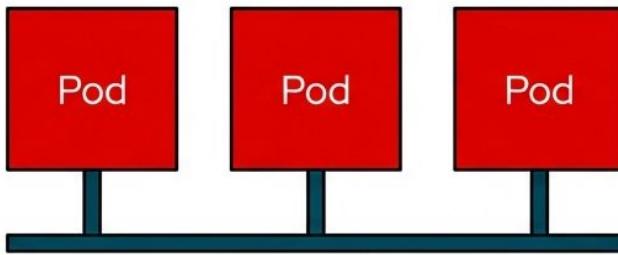
# Secondary UDN

Aggiunge un'interfaccia secondaria: La rete primaria resta la cluster network.



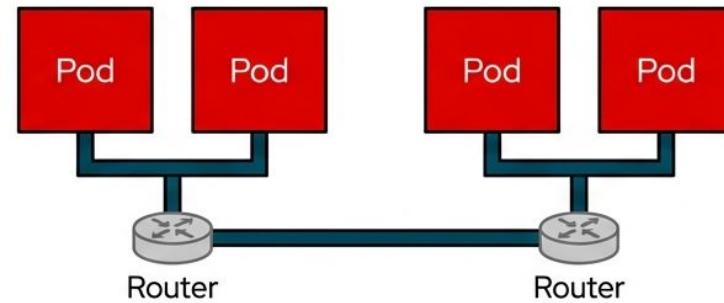
## Architetture UDN: Layer 2 vs Layer 3

### Layer 2 (Flat)



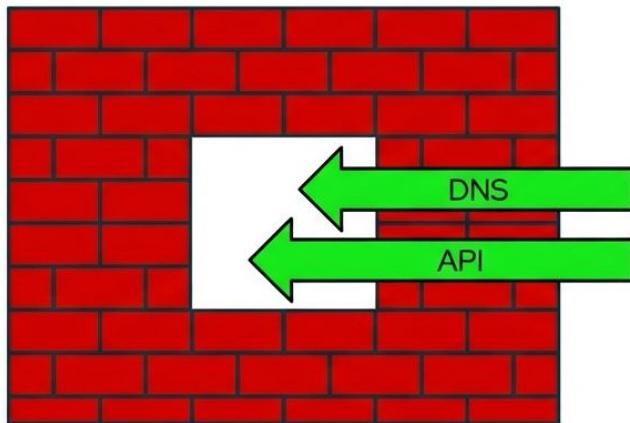
Connettività diretta, broadcast domain unico.

### Layer 3 (Routed)



IP per nodo, simile alla Pod Network standard.

## Gestione del traffico: aprire la Primary UDN

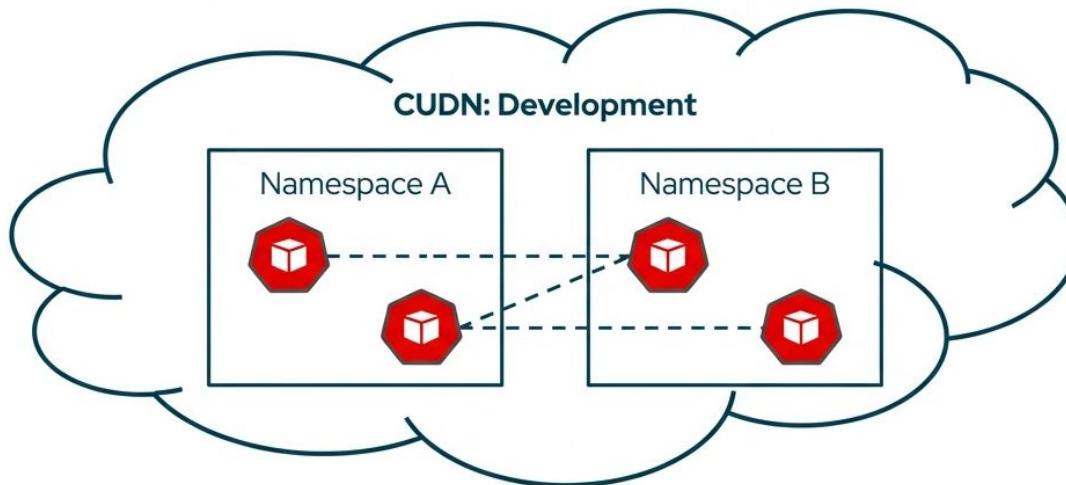


Traffico esterno consentito: DNS, API

L'isolamento implica restrizioni verso la rete K8s di default. Per servizi essenziali usare l'annotazione:

```
metadata:  
  annotations:  
    k8s.ovn.org/open-default-ports: |  
      - protocol: tcp  
        port: 80  
      - protocol: udp  
        port: 53
```

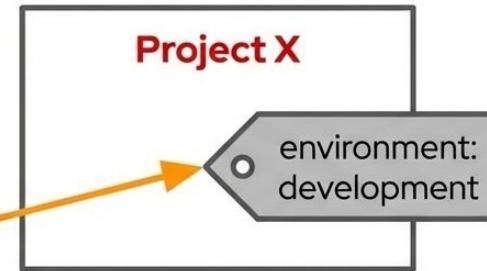
## Cluster User Defined Network (CUDN)



- ▶ Estensione delle UDN per scenari Multi-Namespace.
- ▶ Ideale per teams che gestiscono più progetti correlati.
- ▶ Utilizza *nameSpaceSelector* per agganciare automaticamente i namespace target

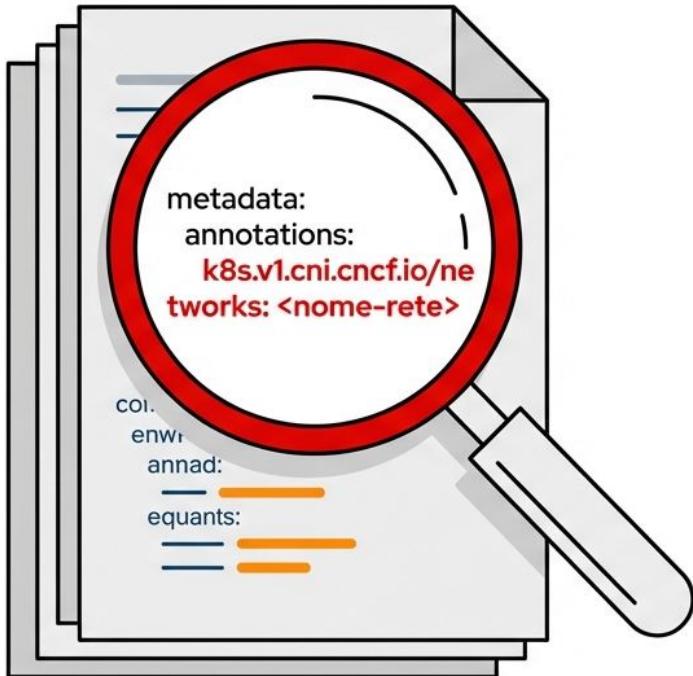
## Configurare CUDN: La logica di selezione

```
kind: ClusterUserDefinedNetwork
spec:
  network:
    topology: Layer2
    namespaceSelector:
      matchLabels:
        environment: development
```



- ▶ Lifecycle IPAM persistente e sottoreti condivide (es. 10.0.0.0/16)

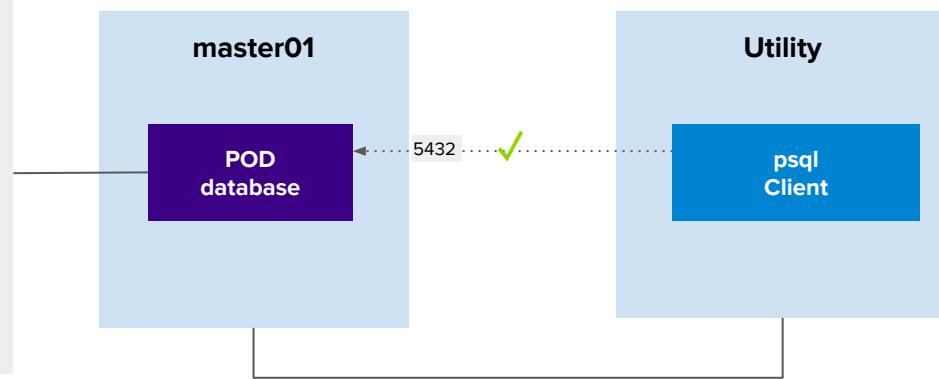
## Collegare i Workloads alla Network



- ▶ Come si attiva la rete secondaria sui Pods?
- ▶ Inserire questa annotation nel Workload
- ▶ Questo meccanismo è universale. funziona per NAD, Secondary UDN e CUDN

# Multus - lab non-http-multus

```
apiVersion: apps/v1
kind: Deployment
metadata:
name: database
namespace: database
spec:
selector:
matchLabels:
app: database
name: database
template:
metadata:
annotations:
k8s.v1.cni.cncf.io/networks: custom
```



192.168.51.10/24

Segregazione di un DB PostgreSQL database in modo che sia accessibile dall'esterno mediante una rete isolata usando una Network interface disponibile su master01



# Resource Quotas and Limits

## Capitolo 6

# Limiting Workloads

## **Requests** → Influenza lo Scheduling

- Definiscono la quantità minima garantita di CPU e memoria per un Pod
- Lo scheduler verifica che il nodo abbia risorse sufficienti per soddisfare le *requests*
- I Pod restano in stato *Pending* se le richieste non possono essere soddisfatte

## **Limits** → Influenza il Runtime e il comportamento di terminazione

- Definiscono la quantità massima di CPU e memoria che un Pod può utilizzare
- Superare il limite di CPU → il Pod viene *throttled* (non terminato)
- Superare il limite di memoria → il Pod viene *OOMKilled* (*Out of Memory Killed*)

# Pod con Requests e Limits

```
apiVersion: v1
kind: Pod
metadata:
  name: example-pod
spec:
  containers:
    - name: app
      image: my-app:latest
      resources:
        requests:
          cpu: "250m"      # Minimum guaranteed CPU (0.25 cores)
          memory: "256Mi" # Minimum guaranteed memory
        limits:
          cpu: "500m"      # Max CPU (can be throttled)
          memory: "512Mi" # Max memory (exceeding kills pod)
```

# Resource Quotas

È una policy applicata a livello di *namespace* per limitare l'uso aggregato delle risorse (CPU, memoria, oggetti) in tutti i resource di quel namespace.

📌 **Scopo:** prevenire l'uso eccessivo delle risorse del cluster imponendo limiti a livello di team

👉 **Azione:** applica l'utilizzo di *Requests* e *Limits*

🔗 **Integrazione:** funziona insieme a *LimitRange* per definire valori di default e max/min per container

```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: example-quota
spec:
  hard:
    pods: "5"
    requests.cpu: "2"
    requests.memory: 4Gi
```

# Per-Project Resource Constraints: Limit Ranges

## Cos'è un **LimitRange**?

- Definisce i valori di default per *Requests* e *Limits* di CPU e memoria nei pod/container di un progetto (namespace).
- Garantisce un uso equo delle risorse all'interno del progetto.
- Previene che i pod consumino eccessivamente CPU e memoria.

## Come funziona:

- Se un pod/container non specifica *requests/limits*, vengono applicati i valori di default definiti dal *LimitRange*.
- Impone vincoli di max/min per ogni container.

# Limit Types

- **default:** Specifica il Limit di default del Workload
- **defaultRequest:** Specifica la Request di default del Workload
- **max:** Valore massimo sia di *requests* che di *limits*
- **min:** Valore minimo sia di *requests* che di *limits*
- **limit-to-request ratio:** rapporto tra *limit* e *request*

```
apiVersion: v1
kind: LimitRange
metadata:
  name: limitrange-demo
  namespace: demo
spec:
  limits:
    - type: Container
      default:
        cpu: "500m"
        memory: "256Mi"
      defaultRequest:
        cpu: "250m"
        memory: "128Mi"
      max:
        cpu: "1"
        memory: "512Mi"
      min:
        cpu: "100m"
        memory: "64Mi"
```

# L'ordine dei Limit è importante!

- Definisci limits e requests in base a questo ordine:



## 🎯 Verifica su Limits/Limits ranges

- Se il Cluster ha configurato il seguente **LimitRange**, sarai in grado di schedulare questo **Pod** ?



```
apiVersion: v1
kind: Pod
metadata:
  name: example-pod
spec:
  containers:
    - name: app
      image: my-app:latest
      resources:
        requests:
          cpu: "250m"
          memory: "256Mi"
        limits:
          cpu: "1"
          memory: "512Mi"
```



```
apiVersion: v1
kind: LimitRange
metadata:
  name: limit-range
spec:
  limits:
    - type: Container
      max:
        cpu: "500m"
        memory: "512Mi"
```

## 🎯 Verifica su Limits/Limits ranges

- No! Il Pod non verrà schedulato !

```
apiVersion: v1
kind: Pod
metadata:
  name: example-pod
spec:
  containers:
    - name: app
      image: my-app:latest
      resources:
        requests:
          cpu: "250m"
          memory: "256Mi"
        limits:
          cpu: "1"  ✘ Maggiore di LimitRange.max
          memory: "512Mi"
```

```
apiVersion: v1
kind: LimitRange
metadata:
  name: limit-range
spec:
  limits:
    - type: Container
      max:
        cpu: "500m"
        memory: "512Mi"
```

## Perche è utile l'attributo **max** del LimitRange - non bastano i limits?

- A. Perchè il LimitRange è un attributo a livello di cluster
- B. Devono essere utilizzati entrambi per funzionare
- C. I Limits li definisce lo sviluppatore quindi potrebbero essere errati
- D. Perché i limits vengono applicati solo quando il nodo è in overcommit al 100%

```
apiVersion: v1
kind: Pod
metadata:
  name: example-pod
spec:
  containers:
    - name: app
      image: my-app:latest
      resources:
        requests:
          cpu: "250m"
          memory: "256Mi"
        limits:
          cpu: "1"
          memory: "512Mi"
```

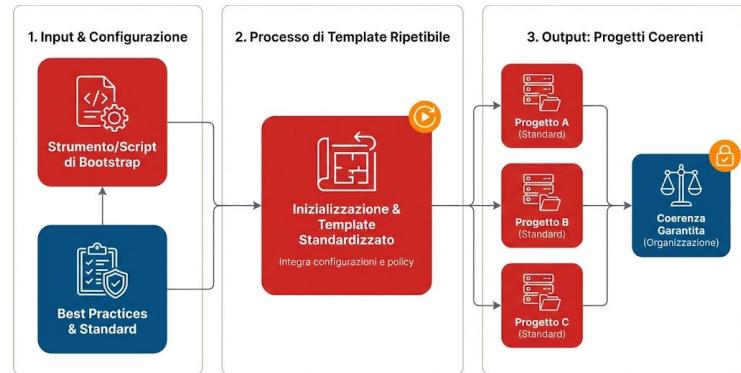
```
apiVersion: v1
kind: LimitRange
metadata:
  name: limit-range
spec:
  limits:
    - type: Container
      max:
        cpu: "500m"
        memory: "512Mi"
```

# OpenShift Bootstrap Template

- Uno strumento/script per inizializzare progetti OpenShift standardizzati
- Fornisce un template ripetibile che integra le *best practices*
- Aiuta a garantire coerenza nelle configurazioni dei progetti all'interno dell'organizzazione

Perché usarlo?

- Permette la pre-configurazione di risorse come *build configs*, *image streams* e pipeline CI/CD
- Consente un setup rapido dei progetti con intervento manuale minimo



# Come creare un Project Template

- **Export** di un Project Template

```
oc adm create-bootstrap-project-template -o yaml >template.yaml
```

- **Edita** il Template
- **Crea** il Template nel namespace openshift-config
- **Inserisci** il Template nella Risorsa Project

```
apiVersion: config.openshift.io/v1
kind: Project
metadata:
  ...output omitted...
name: cluster
  ...output omitted...
spec:
  projectRequestTemplate:
    name: project-request
```



# OpenShift Operators

## Capitolo 7



## OpenShift Operators

Sono applicazioni *Kubernetes-native* che automatizzano il deployment, la gestione e il ciclo di vita di software complesso su OpenShift. Estendono le funzionalità di Kubernetes incapsulando la conoscenza operativa nel codice.

Punti chiave:

- Automatizzano installazione, aggiornamenti e manutenzione delle applicazioni
- Usano le *Custom Resource Definitions (CRDs)* di Kubernetes per gestire le applicazioni in modo dichiarativo
- Riduzione dell'intervento manuale grazie a scaling, *self-healing* e monitoraggio automatico
- Migliorano affidabilità e coerenza tra i cluster OpenShift
- Disponibili tramite Operator Hub

### Cluster Operators

- Gestiscono l'Infrastruttura di OpenShift
- Esposti tramite Operator APIs
- Si aggiornano in seguito all'update di OpenShift



MachineConfig



ImageRegistry



CloudCredentials



WebConsole



Ingress

### Workload Operators / Add-Ons

- Estendono le risorse applicative del Cluster
- Installati on-demando dall' OperatorHub .
- Orchestrate dall' **OLM**



Red Hat Quay



OpenShift GitOps



OpenShift Pipelines

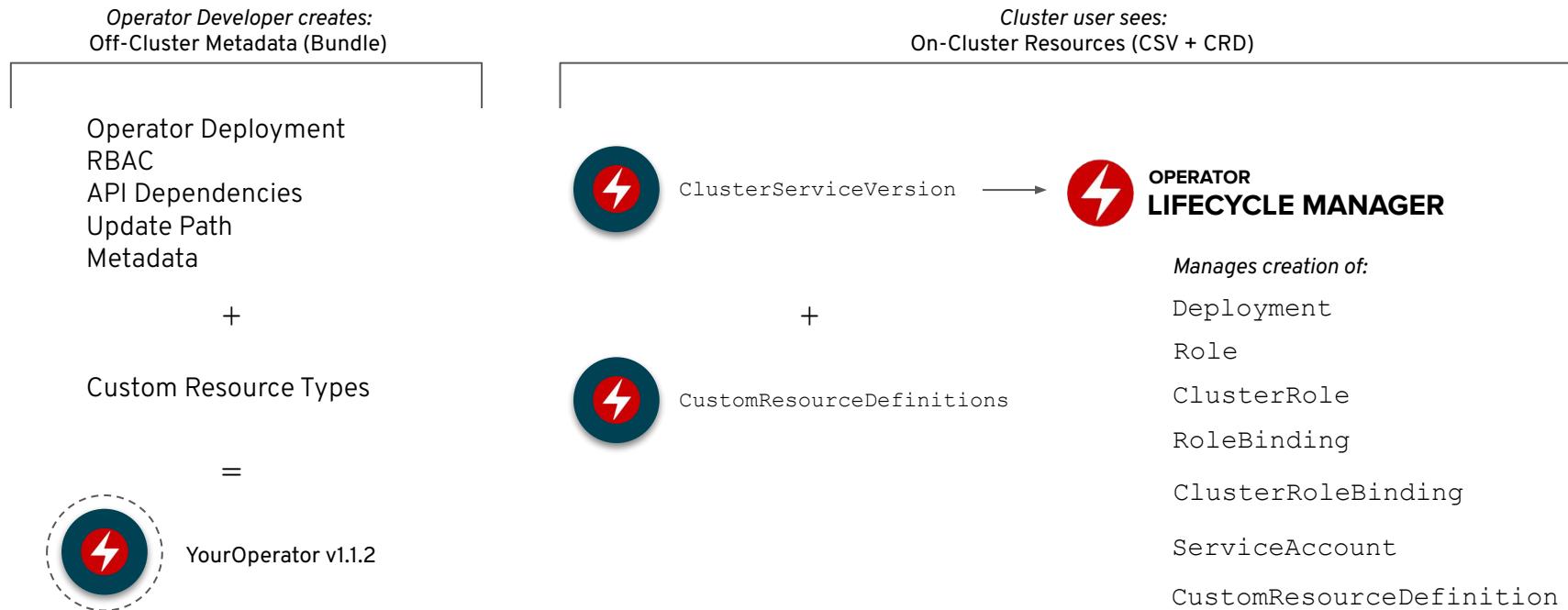


CrunchyDB Postgres



GitLabs Runner

# Orchestrazione Operators



# Installazione Add-On Operators

Red Hat consiglia di utilizzare l'OpenShift Operator Hub per l'installazione degli Operators:

The screenshot shows the Red Hat OpenShift web interface. The left sidebar has sections for Overview, Projects, Search, API Explorer, Events, Operators (with OperatorHub selected), and Workloads (with Pods, Deployments, DeploymentConfigs, and StatefulSets). The main content area is titled "OperatorHub" and describes discovering operators from the Kubernetes community and Red Hat partners. It features a search bar and a list of operators. Two operators are highlighted: "Compliance Operator" and "File Integrity Operator", both provided by Red Hat.

Project: All Projects

## OperatorHub

Discover Operators from the Kubernetes community and Red Hat partners, curated by Red Hat. You can purchase commercial software through [Red Hat Marketplace](#). You can install Operators on your clusters to provide optional add-ons and shared services to your developers. After installation, the Operator capabilities will appear in the [Developer Catalog](#) providing a self-service experience.

All Items	All Items	6 items
Monitoring	<input type="text"/> Filter by keyword...	
Networking		
OpenShift Optional		
Security		
Storage		
Other		
<b>Source</b>		
<input type="checkbox"/> do280 Operator Catalog Cs (6)	<b>do280 Operator Catalog Cs</b>	
	Compliance Operator	
	provided by Red Hat Inc.	
	An operator which runs OpenSCAP and allows you to keep your cluster compliant with...	
<b>Provider</b>		
<input type="checkbox"/> Red Hat (5)	<b>do280 Operator Catalog Cs</b>	
	File Integrity Operator	
	provided by Red Hat	
	An operator that manages file integrity checks on nodes.	

# Subscription Modes

Gli Add-On Operators possono essere installati in due modalità:

- ◆ **Modalità Automatica**

- L'Operator si aggiorna automaticamente ogni volta che è disponibile una nuova versione
- Garantisce l'applicazione immediata di nuove funzionalità, bug fix e patch di sicurezza
- Ideale per ambienti di sviluppo/test che richiedono aggiornamenti continui

- ◆ **Modalità Manuale**

- Gli aggiornamenti richiedono approvazione manuale prima di essere applicati
- Offre maggiore controllo, consentendo di testare prima del rilascio
- Raccomandata per ambienti di produzione dove la stabilità è prioritaria



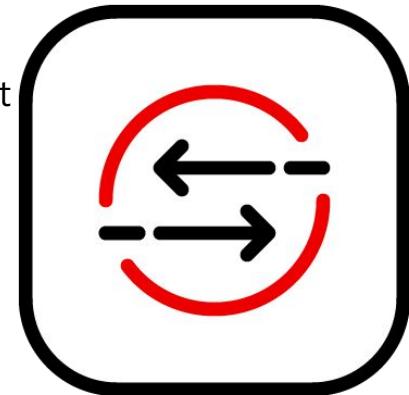
# File Integrity Operator

## Utilizzo File Integrity Operator:

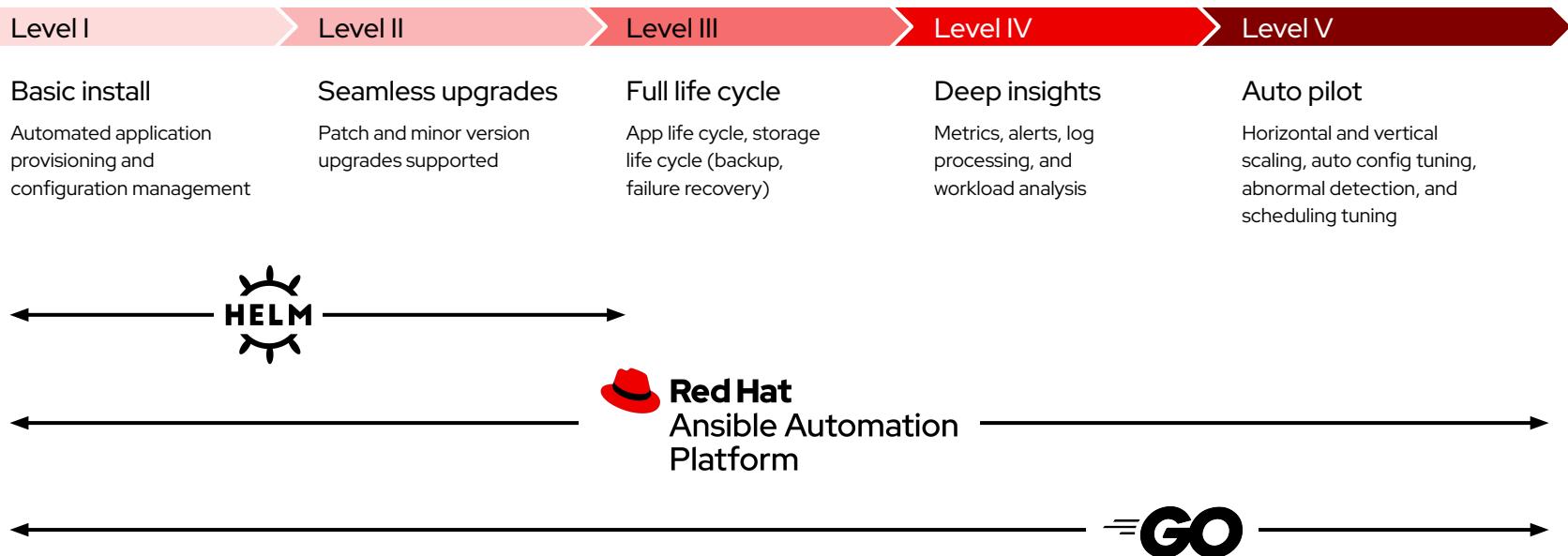
- Monitora continuamente l'integrità dei file sui nodi del cluster OpenShift utilizzando AIDE (Advanced Intrusion Detection Environment)
- Distribuisce un DaemonSet che esegue container AIDE privilegiati su ciascun nodo, registrando le modifiche ai file rispetto alla scansione iniziale

## Caratteristiche principali

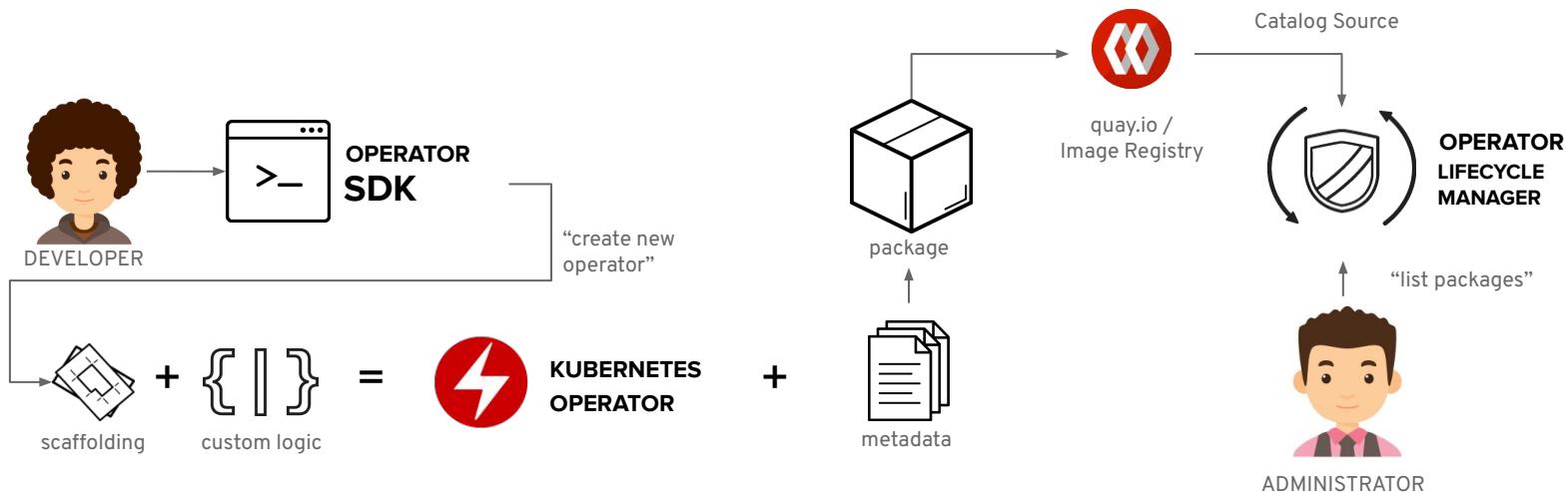
- Supporta esclusivamente nodi RHEL CoreOS
- Funziona anche in ambienti disconnessi (*air-gapped*)
- Rileva modifiche inattese ai file, indicando potenziali intrusioni



# Operator Maturity



# Operator Framework in Action



Package Discovery:

```
$ oc get packagemanifests
```

## Come sviluppare un Operator ?

Sono disponibili diverse Operators SDK per sviluppare Operators: Tra queste: Java, Go Programming Language, Bash ed altre ancora.





# Application Security

## Capitolo 8

# Application Security

Un Service Account è un'identità tecnica che può ricevere permessi da due domini:

## SCC (Security Context Constraints)

- Specifico di OpenShift (non presente nel Kubernetes "vanilla")
- Controlla come vengono eseguiti pod/container: privilegi, UID/GID, hostPath, NET\_ADMIN, ecc.
- Collegato al SA tramite *RoleBinding* verso un oggetto SCC

## Role / ClusterRole (RBAC)

- Oggetti standard di Kubernetes/OpenShift
- Definiscono cosa può fare il SA tramite le API di Kubernetes: creare pod, leggere secrets, scalare



# OpenShift Enforced Security

Per impostazione predefinita, OpenShift applica policy di sicurezza rigorose per impedire ai container di essere eseguiti come utente root. Questo consente di:

- Migliorare la sicurezza del cluster
- Ridurre il rischio di attacchi di *container breakout*
- Applicare l'accesso *least-privilege* di default

Come abilitare operazioni privilegiate?

- Per consentire l'accesso come utente root o altre azioni privilegiate, è necessario assegnare una **Security Context Constraint (SCC)** che permetta permessi elevati
- Senza modifiche alle SCC, OpenShift applica policy rigorose che eseguono i container come utenti *non-root* di default



# Security Context Constraints

Cosa sono i SCC? I **Security Context Constraints (SCC)** in OpenShift definiscono le policy di sicurezza per controllare come i pod interagiscono con le risorse di sistema. Aiutano ad applicare le *best practices* di sicurezza limitando permessi e capacità.

Perché le SCC sono importanti?

- Prevengono l'escalation di privilegi
- Controllano l'accesso a risorse sensibili
- Definiscono policy di sicurezza dei pod a livello granulare



```
oc adm policy add-scc-to-user restricted -z myserviceaccount -n mynamespace
```

## Esempio di Custom SecurityContextConstraint:

```
apiVersion: security.openshift.io/v1
kind: SecurityContextConstraints
metadata:
  name: custom-scc
allowPrivilegedContainer: false ←
runAsUser:
  type: MustRunAs
  uid: 1000 ←
fsGroup:
  type: MustRunAs
  ranges:
    - min: 1000 ←
      max: 2000 ←
supplementalGroups:
  type: RunAsAny ←
users:
  - system:serviceaccount:myproject:default ←
```

### Cosa fa questo SecurityContextConstraint ?

🚫 Disabilita i container privilegiati  
(`allowPrivilegedContainer: false`)

👤 Definisce l'User ID obbligatorio per i pod  
(`runAsUser: MustRunAs uid=1000`)

📁 Imposta il gruppo FS (fsGroup) con vincolo di range (`MustRunAs` tra 1000 e 2000)

👥 Permette gruppi supplementari liberi  
(`supplementalGroups: RunAsAny`)

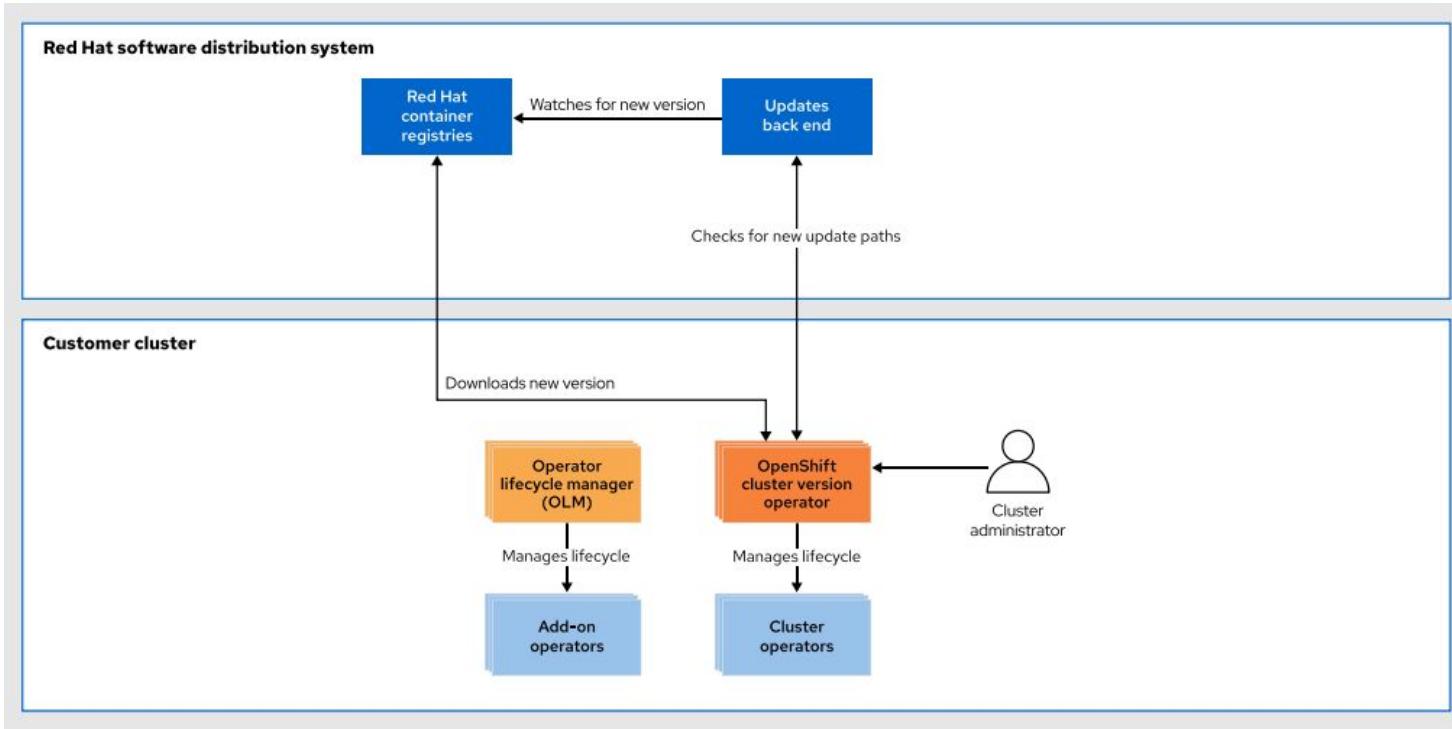
🔗 Associa l'SCC a un ServiceAccount specifico



# OpenShift Update

## Chapter 9

# OpenShift Update



## Why OpenShift Updates Matters

Mantieni i cluster allineati al ciclo di supporto Red Hat (EUS vs non-EUS)

-  Fondamentale per la conformità (CVEs di sicurezza, preparazione agli audit)
-  Gli aggiornamenti migliorano l'osservabilità SRE e la fedeltà della telemetria del cluster
-  Sbloccano nuove API e funzionalità Kubernetes nei workload gestiti
-  Consentono la compatibilità con nuove versioni di Operator e ISV

OpenShift support:

<https://access.redhat.com/articles/4763741>

## OpenShift Update Checklist

- ❖  Fate un backup recente dei workload (Velero/OADP)
- ❖  Verificare che gli Add-on Operators siano compatibili con la nuova versione di RHOCP
- ❖  Controllare le API rimosse/deprecated
- ❖  Mettere in pausa i Machine Health Checks
- ❖  Assicurarsi che tutti i Machine Config Pools (MCPs)

OpenShift update graph:

[https://access.redhat.com/labs/ocpupgradegraph/update\\_path/](https://access.redhat.com/labs/ocpupgradegraph/update_path/)

# Check prima dell' Update !

General Availability

► GA in OpenShift 4.20

► Il comando **oc adm upgrade recommend**  
mostra:

- La prossima versione consigliata per l'aggiornamento
- Precheck feature: indicazioni su possibili alert ed issues relative all'aggiornamento

Da lanciare per verifica prima di un aggiornamento.

```
$ oc adm upgrade recommend

Failing=True:

Reason: ClusterOperatorNotAvailable
Message: Cluster operator monitoring is not available

The following conditions found no cause for concern in updating this cluster to later releases:
recommended/NodeAlerts (AsExpected), recommended/PodImagePullAlerts (AsExpected)

The following conditions found cause for concern in updating this cluster to later releases:
recommended/PodDisruptionBudgetAlerts/PodDisruptionBudgetAtLimit/1

recommended/PodDisruptionBudgetAlerts/PodDisruptionBudgetAtLimit/1=False:

Reason: Alert:firing
Message: warning alert PodDisruptionBudgetAtLimit firing, which might slow node drains.
Namespace=openshift-monitoring, PodDisruptionBudget=prometheus-k8s. The pod disruption budget is preventing further disruption to pods. The alert description is: The pod disruption budget is at the minimum disruptions allowed level. The number of current healthy pods is equal to the desired healthy pods.
https://github.com/openshift/runbooks/blob/master/alerts/cluster-kube-controller-manager-operator/PodDisruptionBudgetAtLimit.md

Upstream update service: https://api.integration.openshift.com/api/upgrades_info/graph
Channel: candidate-4.18 (available channels: candidate-4.18, candidate-4.19, candidate-4.18, eus-4.18, fast-4.18, fast-4.19, stable-4.18, stable-4.19)

Updates to 4.18:
VERSION      ISSUES
4.18.32      no known issues relevant to this cluster
4.18.30      no known issues relevant to this cluster
And 2 older 4.18 updates you can see with '--show-outdated-releases' or '--version VERSION'.
```

# OpenShift Update Status

## General Availability

### ► GA in OpenShift 4.20

### ► oc adm upgrade

**status** mostra lo status del processo di aggiornamento

- Non modifica lo stato dell'aggiornamento

```
$ oc adm upgrade status

= Control Plane =
Assessment: Progressing
Completion: 12%
Duration: 12m5s
Operator Status: 33 Healthy

Control Plane Nodes
NAME                           ASSESSMENT   PHASE      VERSION   EST   MESSAGE
ip-10-0-30-217.us-east-2.compute.internal  Outdated Pending  4.14.0   ?
ip-10-0-53-40.us-east-2.compute.internal  Outdated Pending  4.14.0   ?
ip-10-0-92-180.us-east-2.compute.internal  Outdated Pending  4.14.0   ?

= Worker Upgrade =
= Worker Pool =
Worker Pool: worker
Assessment: Excluded
Completion: 0%
Worker Status: 3 Total, 3 Available, 0 Progressing, 3 Outdated, 0 Draining, 3 Excluded, 0 Degraded

Worker Pool Nodes
NAME                           ASSESSMENT   PHASE      VERSION   EST   MESSAGE
ip-10-0-20-162.us-east-2.compute.internal  Excluded    Paused    4.14.0   -
ip-10-0-4-159.us-east-2.compute.internal  Excluded    Paused    4.14.0   -
ip-10-0-99-40.us-east-2.compute.internal  Excluded    Paused    4.14.0   -

= Update Health =
SINCE LEVEL      IMPACT MESSAGE
- Warning Update Stalled Outdated nodes in a paused pool 'worker' will not be updated

Run with --details=health for additional description and links to related online documentation
```