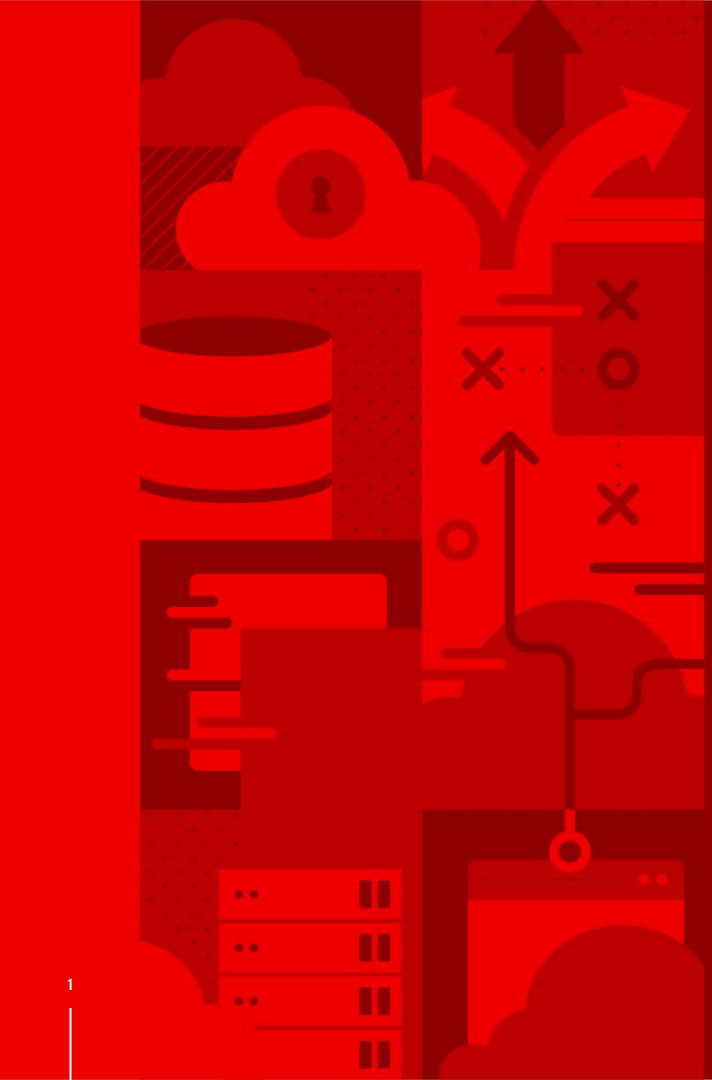
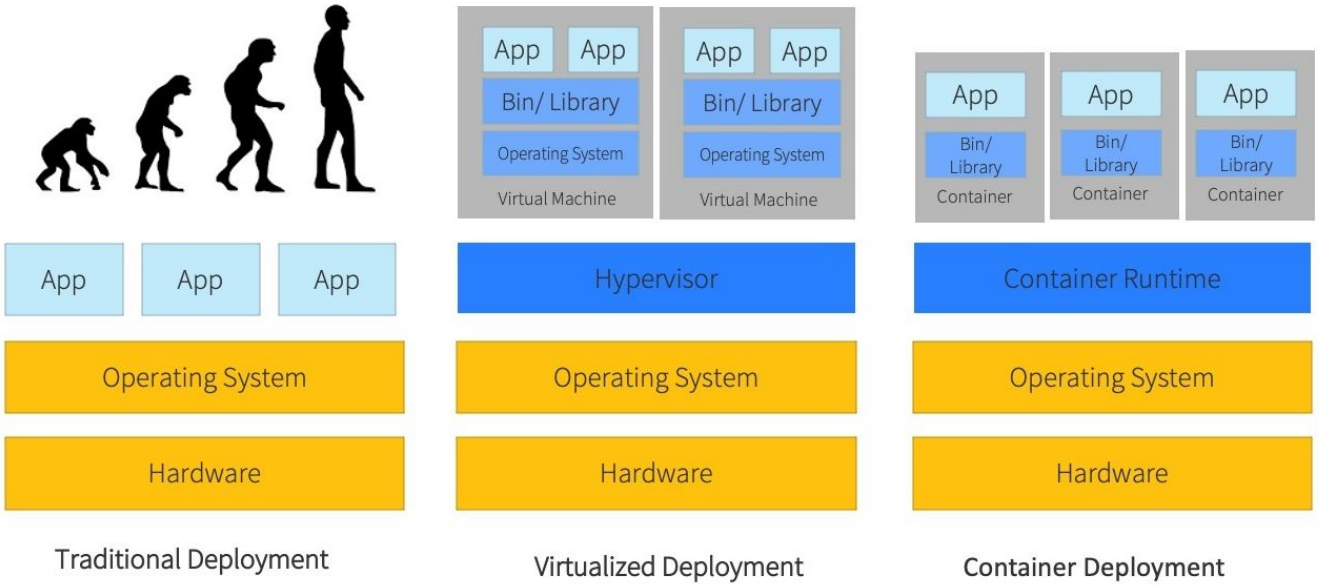


Containers Overview

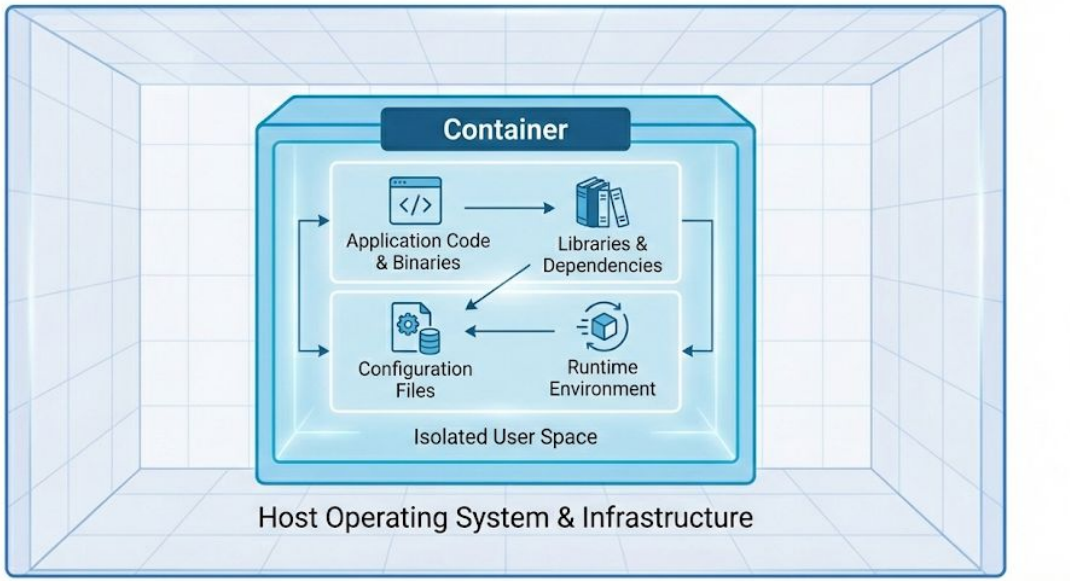
Capitolo 1

Evolution of Application Deployments

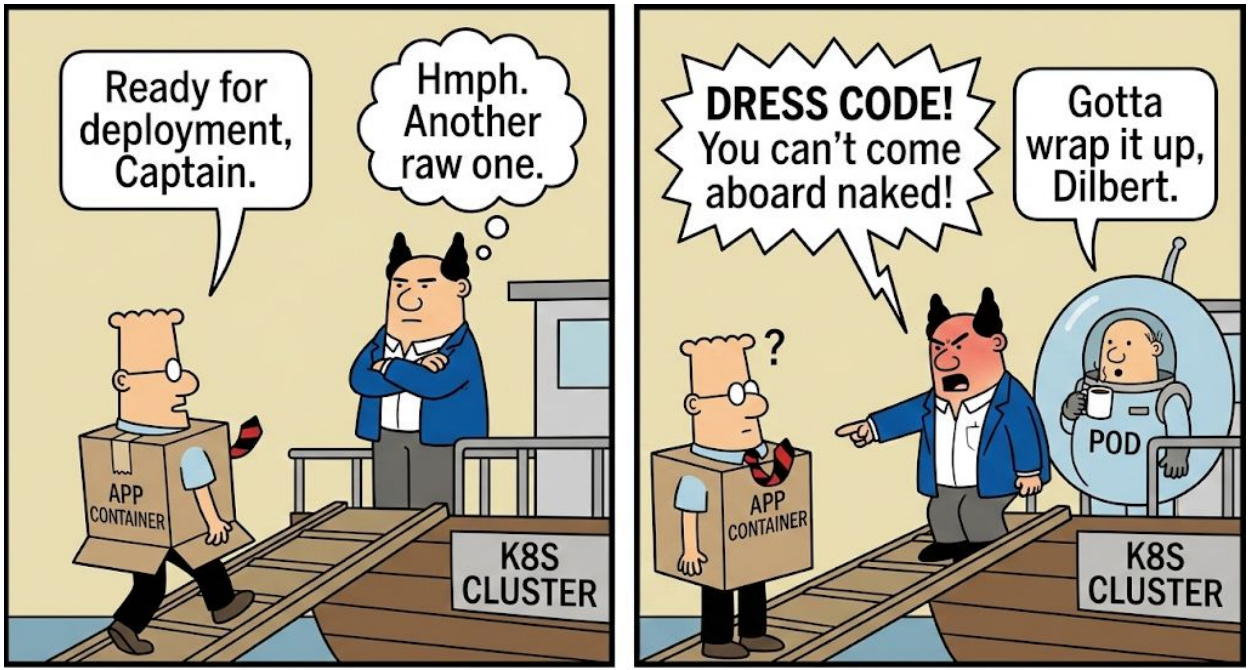


Cosa è un Container ?

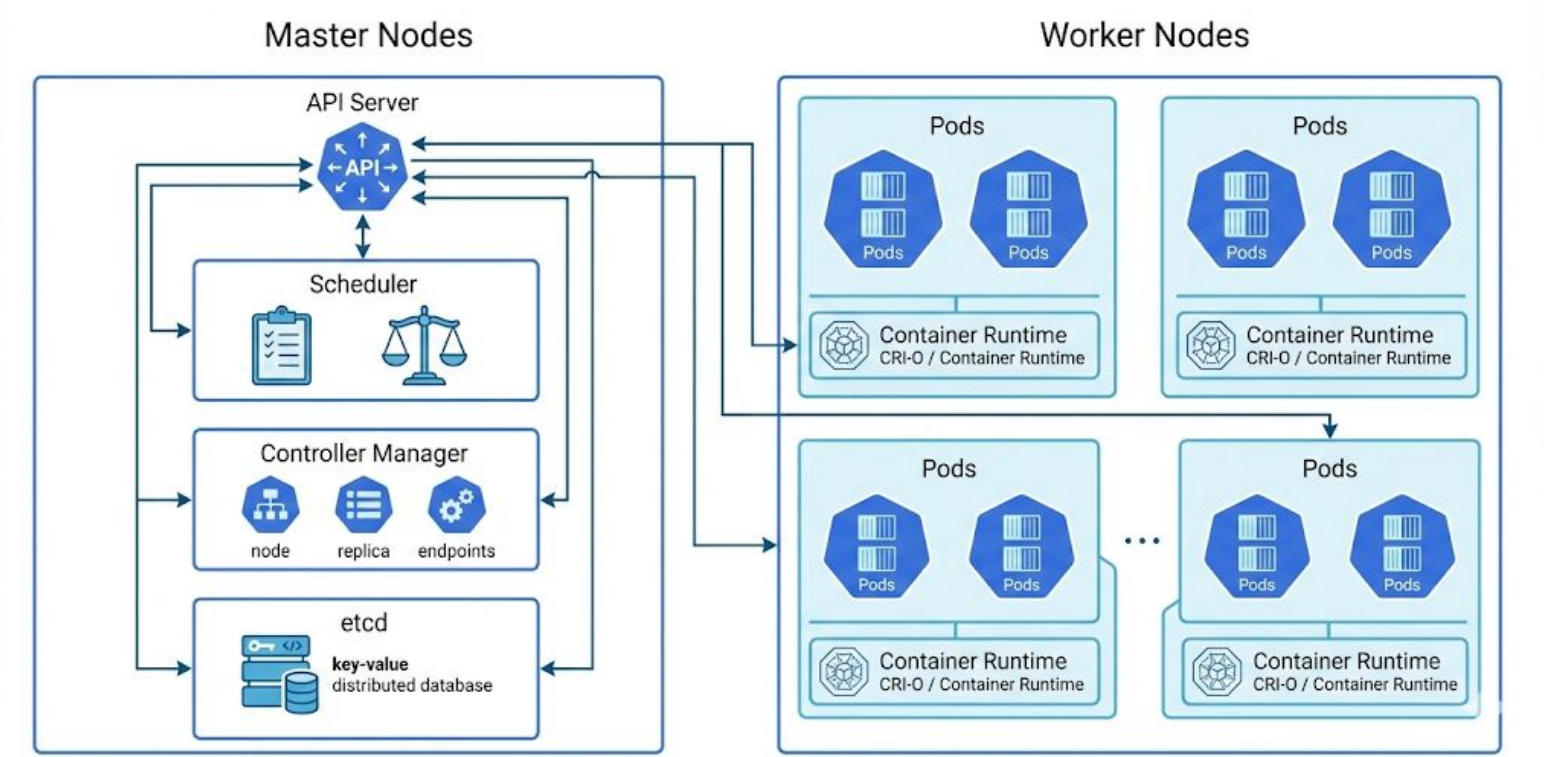
Cosa è un Container ?



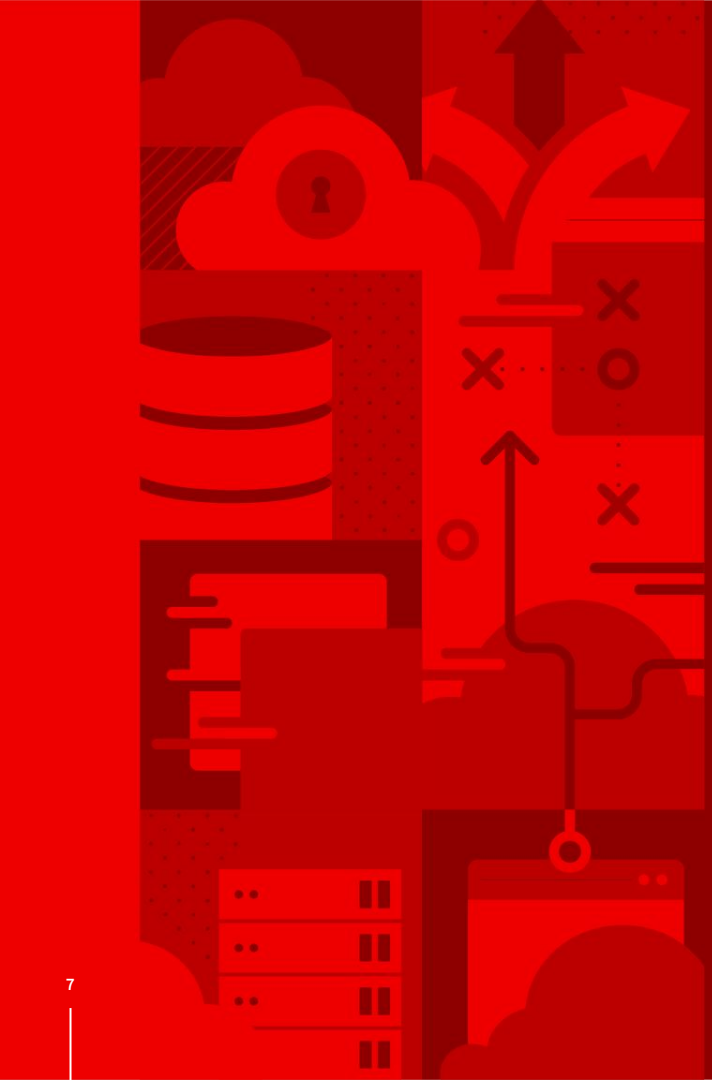
Containers e Kubernetes



Kubernetes: Architettura



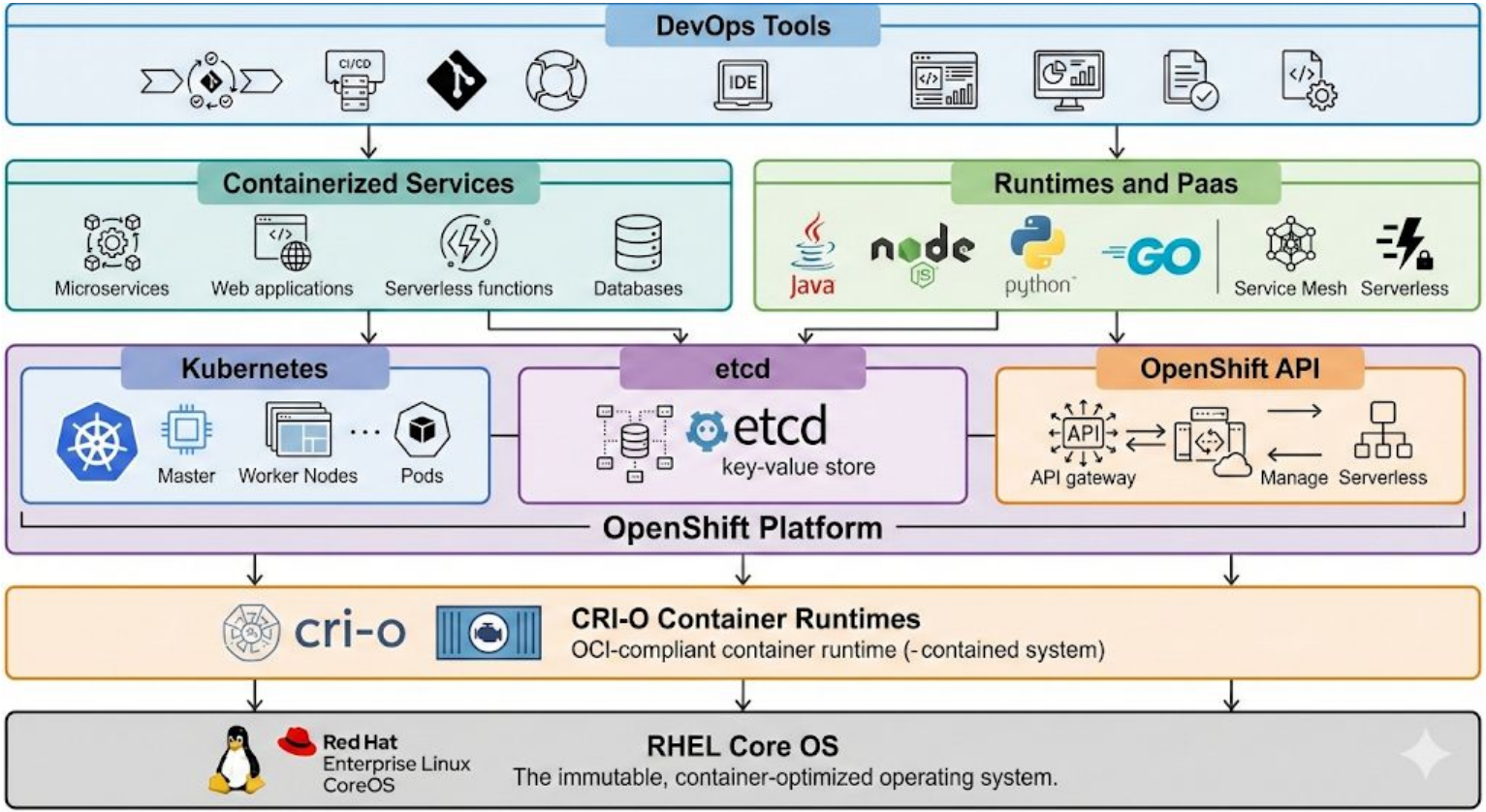
Red Hat



Red Hat OpenShift Components and Editions

Capitolo 1

OpenShift: Architettura



Openshift Offering

Self-Managed Editions

- Red Hat OpenShift Container Platform
- Red Hat OpenShift Kubernetes Engine
- Red Hat OpenShift Virtualization Engine

Managed Editions

Red Hat | aws

Red Hat | Microsoft Azure

Red Hat OpenShift Dedicated

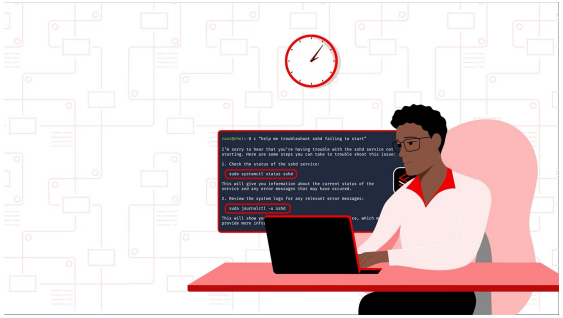
Available on Google Cloud

Red Hat | IBM

Gestione risorse su OpenShift

Gestione Imperativa

- Comandi dalla CLI per creare, modificare e cancellare le risorse
- Esecuzione con il tool oc (openShift client): oc expose service httpd



Gestione Dichiarativa

- Utilizzo dei Manifest YAML o JSON per definire lo stato desiderato
- Uso tipico: oc apply -f <file>.yaml

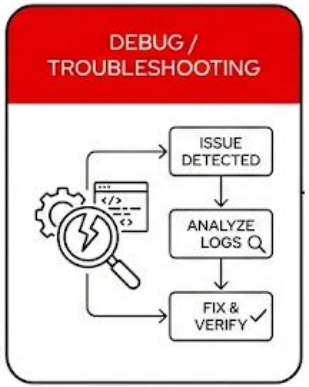
Come è fatta una risorsa Kubernetes ?

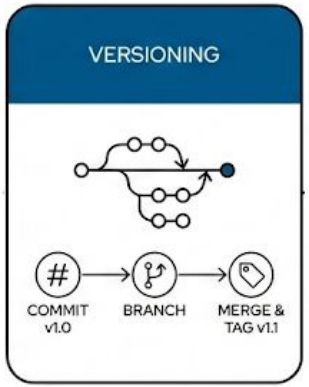
```yaml
apiVersion: v1
kind: Pod
metadata:
  name: my-sample-pod
  labels:
      app: my-app
      env: dev
spec:
  containers:
      - name: my-container
      image: nginx:latest
      ports:
      - containerPort: 80
      resources:
        requests:
                memory: "64Mi"
                cpu: "250m"
```

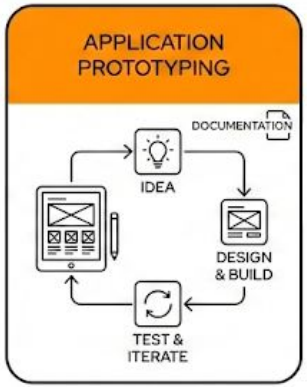
- **apiVersion**: identifica la versione dello schema dell'oggetto.
- **Kind**: indica il tipo di risorsa.
- **labels**: coppie chiave-valore per categorizzare e selezionare le risorse.
- **annotations**: metadati non identificativi (es. informazioni di versione, link).
- **spec**: definisce la configurazione richiesta per la risorsa.

Casi d’uso

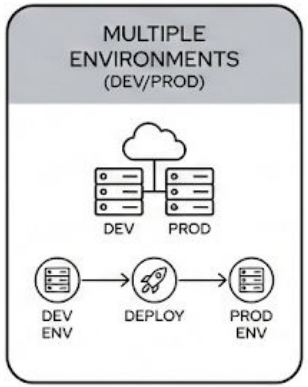
Gestione Imperativa

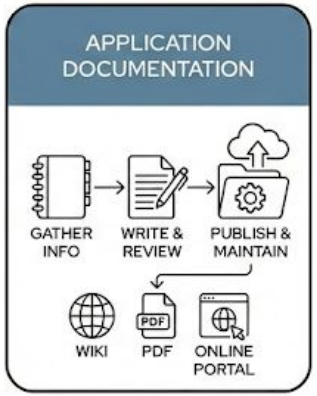




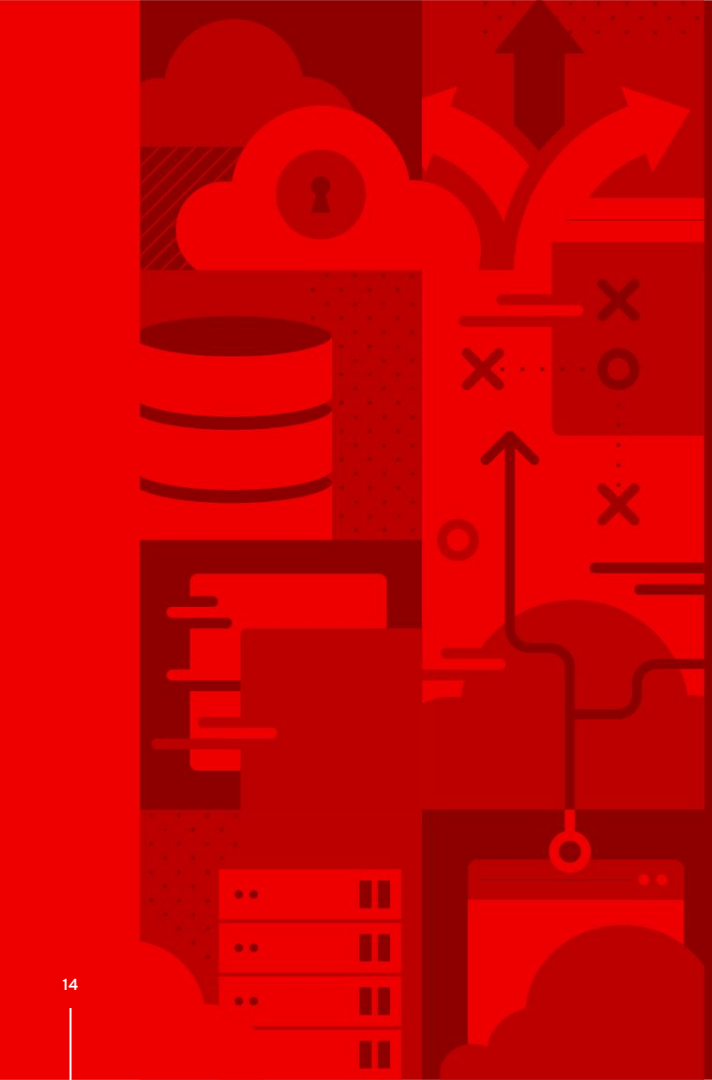


Gestione Dichiarativa





Andiamo sul Bookshelf!



OpenShift Command Line Interfaces

Capitolo 2

kubectl vs oc

kubectl:

- Kubernetes usa kubectl per gestire le risorse.

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

oc

- OpenShift lo estende con oc, aggiungendo funzioni extra per sviluppatori e amministratori.
- oc include tutti i comandi di kubectl e introduce strumenti specifici di OpenShift.
- Quando lavori su OpenShift, preferisci oc per un'integrazione migliore e maggiore produttività.

```
oc new-app --image=httpd:1.0
```

Comandi esclusivi di *oc* - 1

🔑 Autenticazione & Gestione Utenti

- oc login – Autenticazione su un Cluster OpenShift.
- oc whoami – Visualizza l'utente corrente e il suo Token.

📂 Gestione Project & Namespace

- oc new-project <name> – Crea un Project OpenShift.
- oc project <name> – Cambia il Project corrente.

🚀 Deploy Applicazioni & Builds

- oc new-app <source> – Crea una applicazione da sorgente, Image o Template.
- oc start-build <buildconfig> – Trigger manuale di una Build.

Comandi esclusivi di *oc* - 2

🌐 Networking & Routing

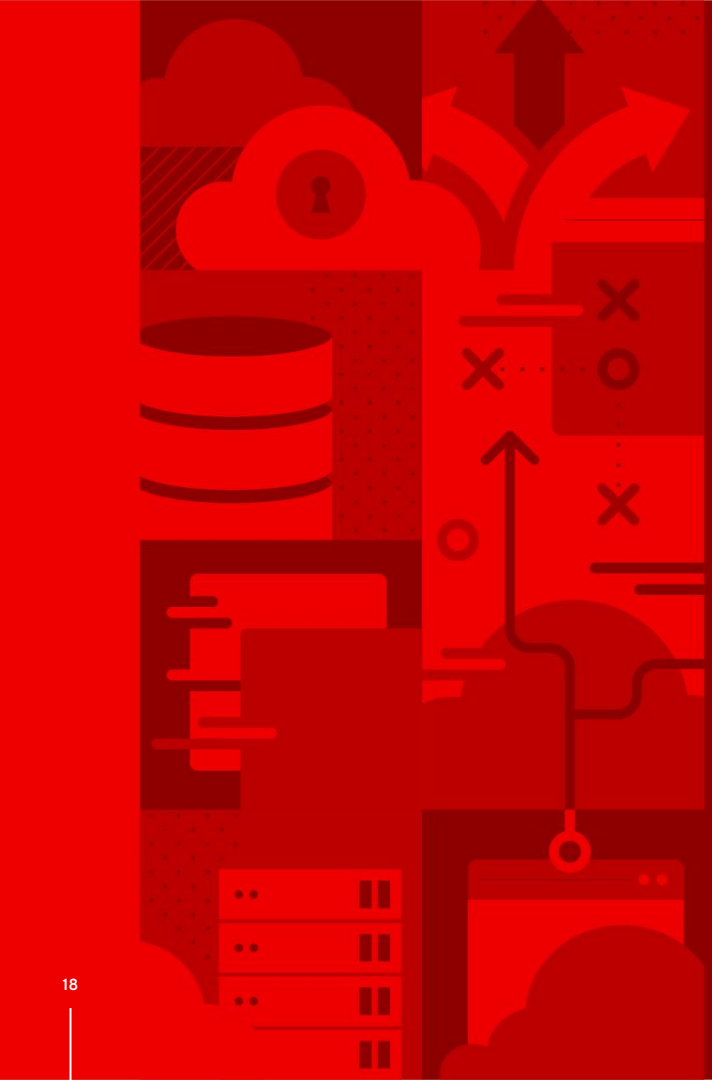
- `oc get routes` – Elenca la Routes (per accesso HTTP ad un Service).

📦 Gestione Immagini

- `oc import-image <image-stream>` – Importa Immagini esterne in un ImageStream.
- `oc tag <source> <destination>` – Gestisce i tags degli ImageStream tags.

🔧 Amministrazione

- `oc adm <subcommand>` – Comandi di amministrazione del Cluster (e.g., `oc adm policy`, `oc adm prune`).



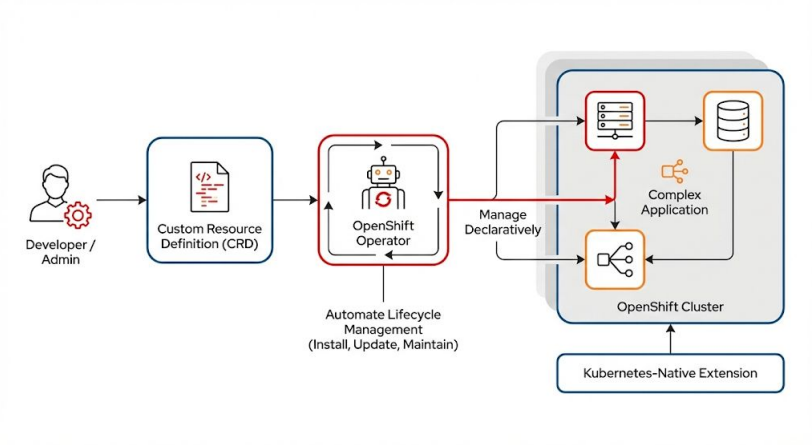
OpenShift Operators

Capitolo 2

OpenShift Operators

- Gli **Operators** sono applicazioni native Kubernetes che estendono Kubernetes incorporando nel codice la logica operativa.

❖ Automatizzano installazione, aggiornamenti e manutenzione.
❖ Usano CRD per gestire le applicazioni in modo dichiarativo.
❖ Riducono l'intervento manuale gestendo scaling, self-healing e monitoraggio.
❖ Disponibili su OperatorHub, con operatori certificati Red Hat e della community.



Cluster Operators

- Gestiscono l'Infrastruttura del Control Plane
- 40+ Operators, governati dal Cluster Version Operator
- Installati al giorno 0

MachineConfig

ImageRegistry

CloudCredentials

WebConsole

Ingress

```
oc get cluster operators
```

Workload Operators / Add-Ons

- Estendono le funzionalità di OpenShift con Applicazioni o Servizi
- Installati dall' OperatorHub on demand.
- Gestiti da **OLM**

Red Hat Quay

OpenShift Pipelines

CrunchyDB Postgres

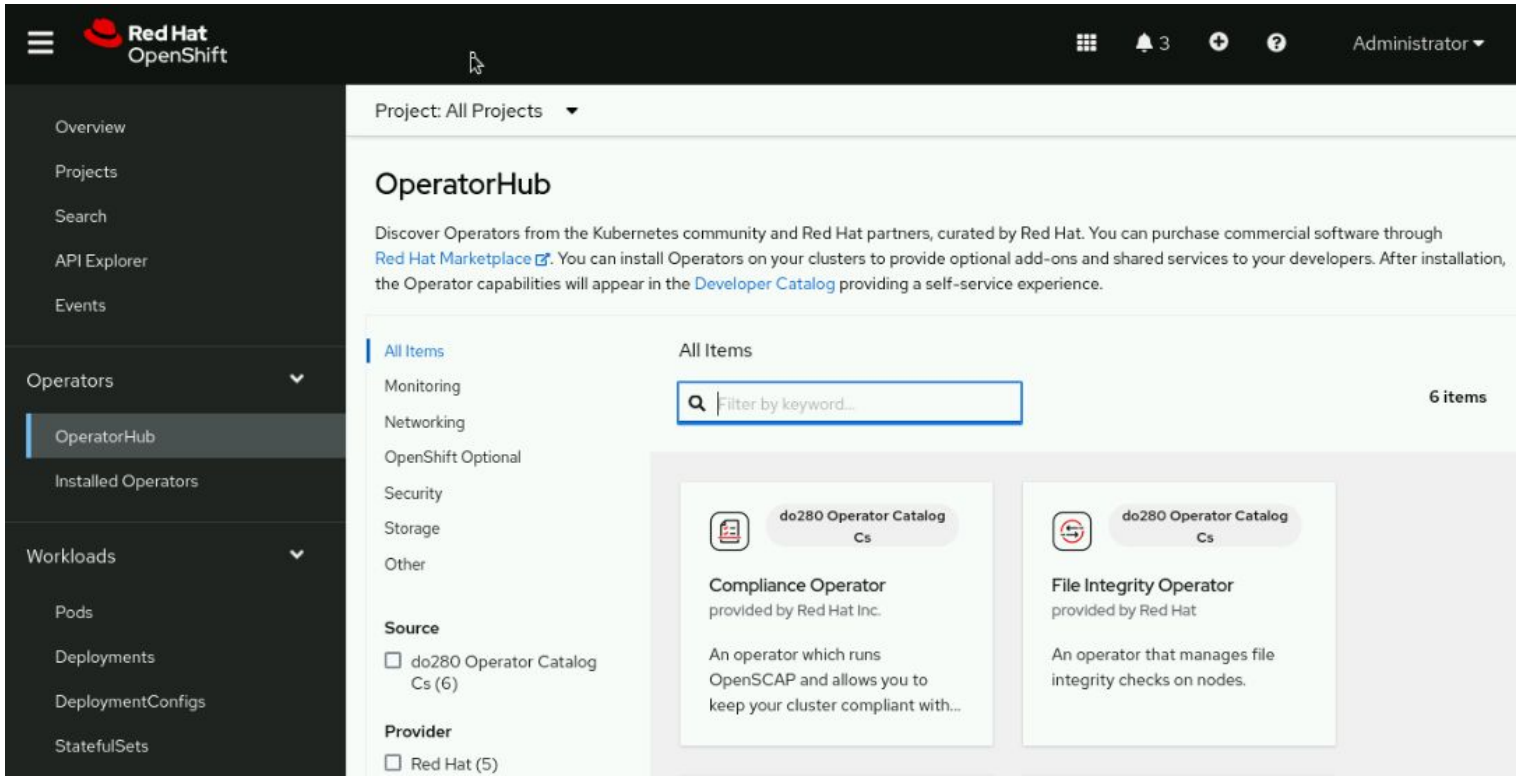
OpenShift GitOps

GitLabs Runner

```
oc get operators
```

Installazione degli Operators dell'Hub

Gli add-on Operator si possono installare da CLI o dall' Operator Hub:



Aggiornamento Add-on Operators

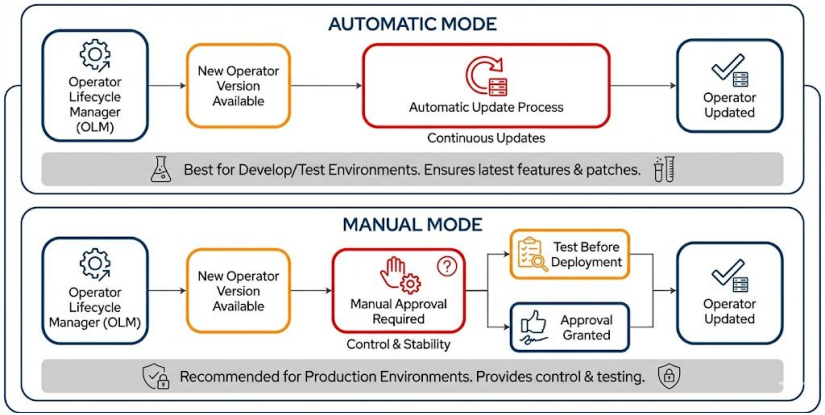
Gli **Operators** si possono installare in due modalità diverse:

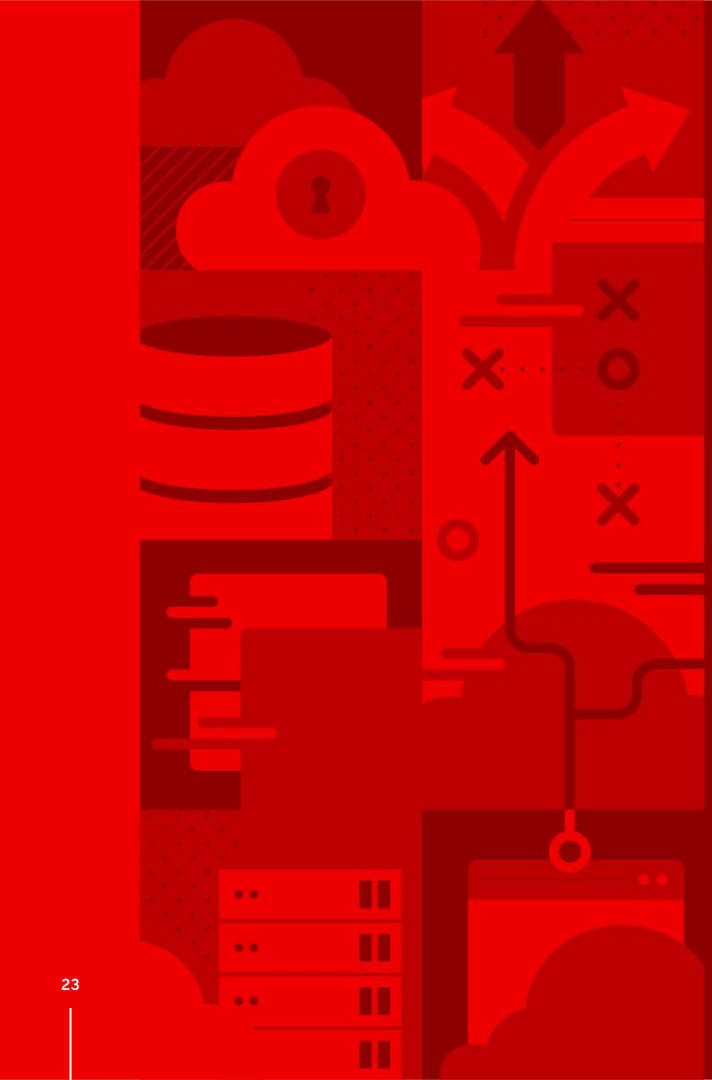
Modalità Automatica

- L'Operator si aggiorna da solo quando esce una nuova versione.
- Garantisce funzionalità, bugfix e patch di sicurezza sempre aggiornate.
- Ideale per ambienti di sviluppo e test

Modalità Manuale

- Gli aggiornamenti richiedono un'approvazione manuale.
- Offre maggiore controllo e consente di testare prima di applicare.
- Consigliata per ambienti di produzione

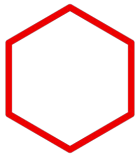




Run Applications as Containers and Pods

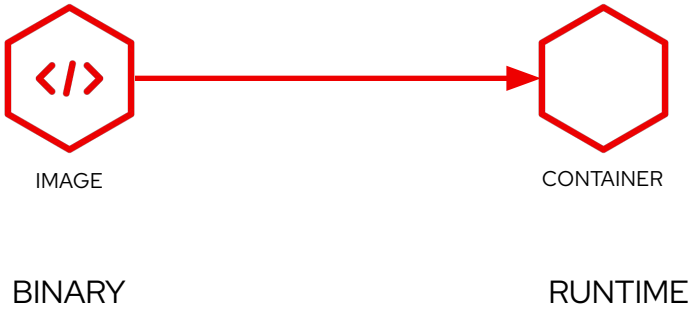
Chapter 3

Il Container è l'elemento più piccolo nel mondo Kubernetes

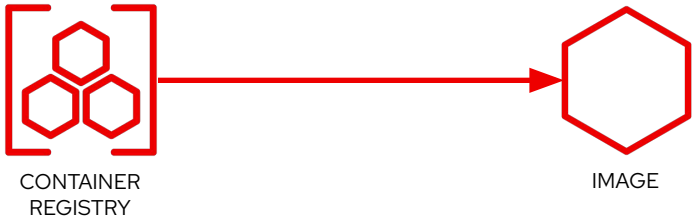


CONTAINER

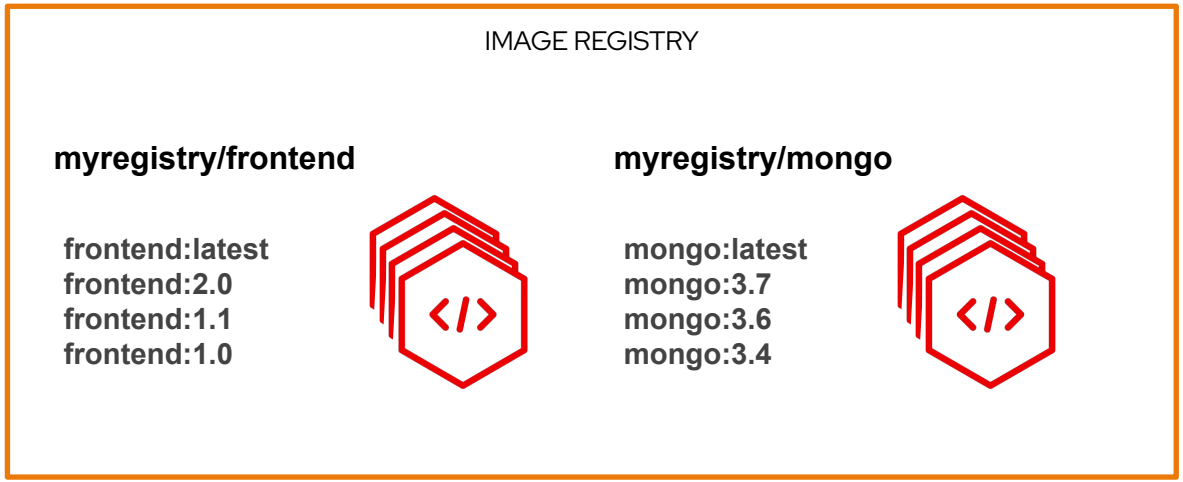
I Containers sono creati dalle Immagini



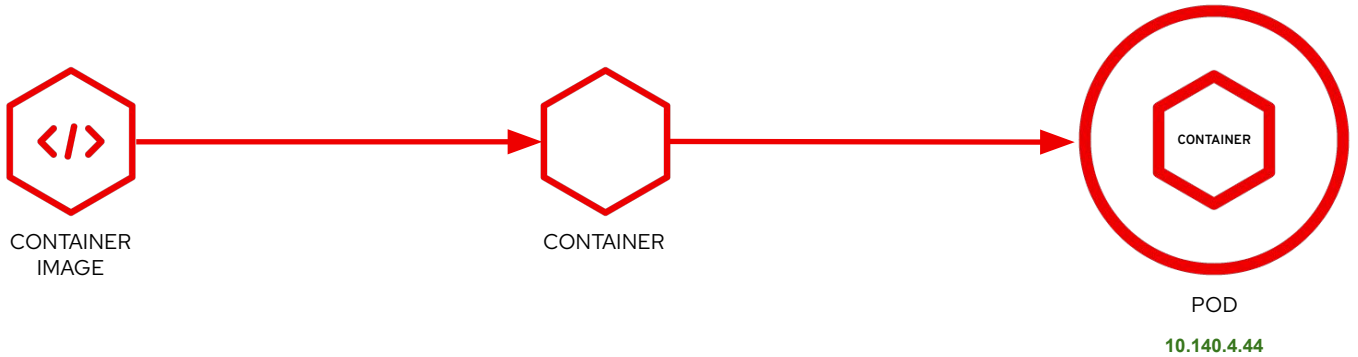
Le Immagini sono conservata in un Registry



Il Registry contiene tutte le versioni delle Immagini

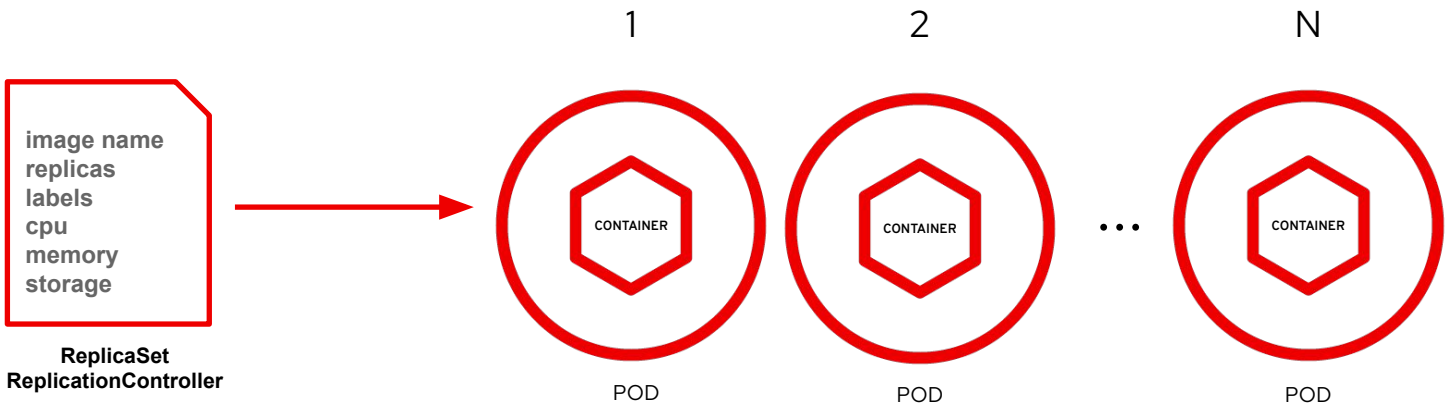


In OpenShift i workloads girano nei Pod

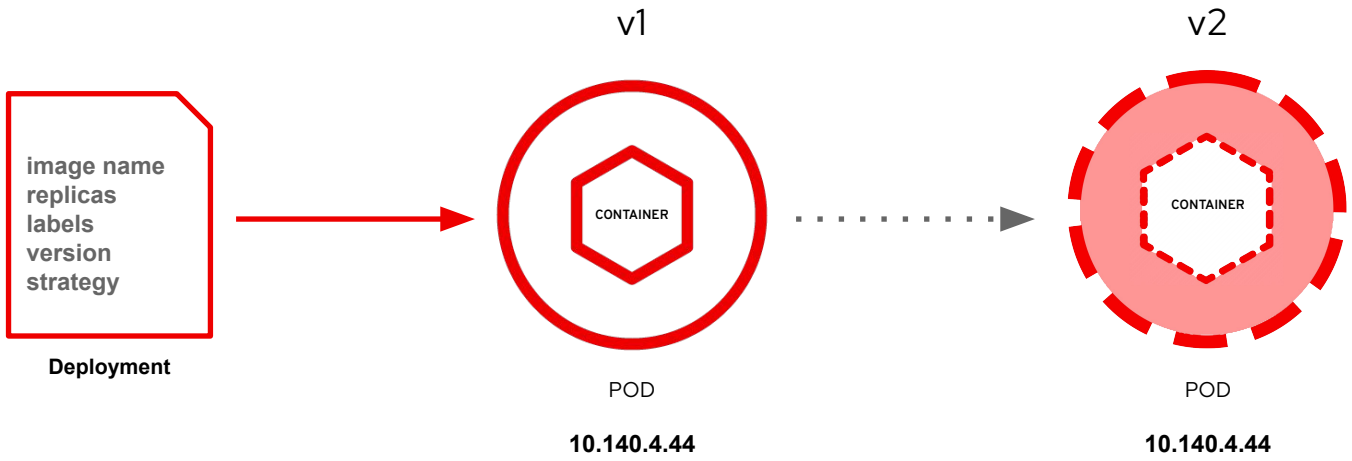


Ogni Pod contiene 1 o più Container

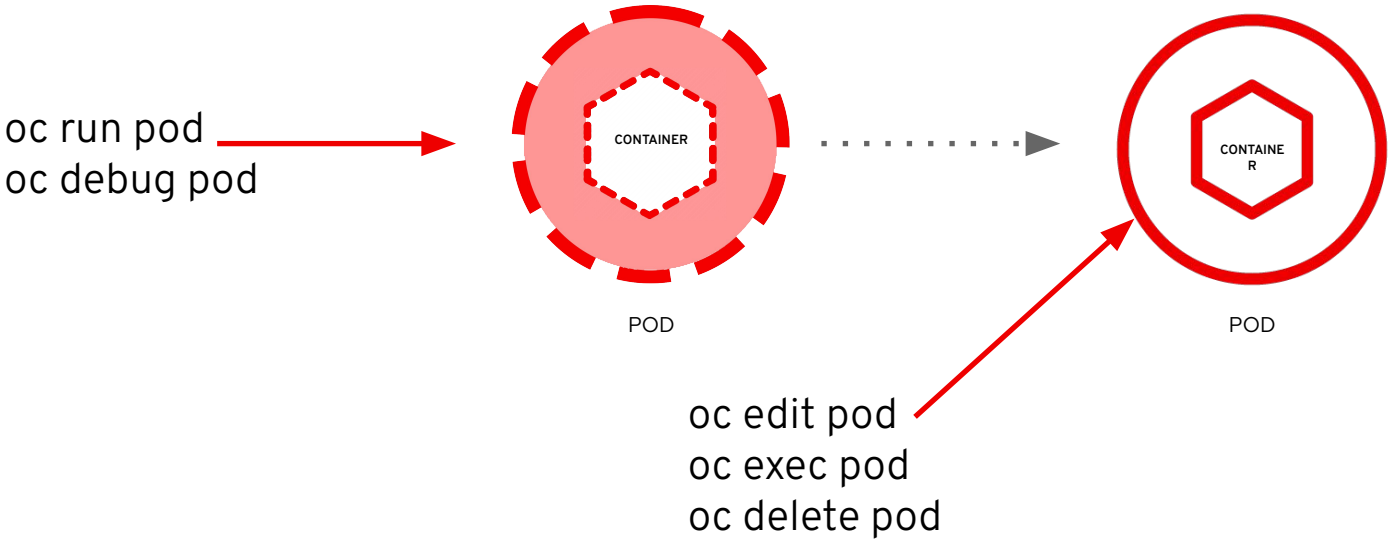
ReplicationControllers & ReplicaSets ensure a specified number of pods are running at any given time



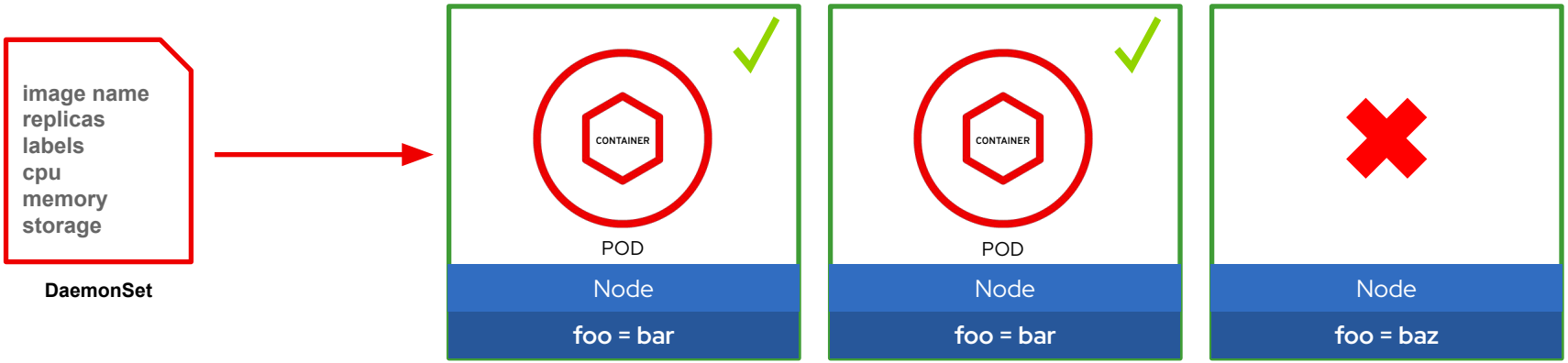
I Pods possono essere governati dai Deployments



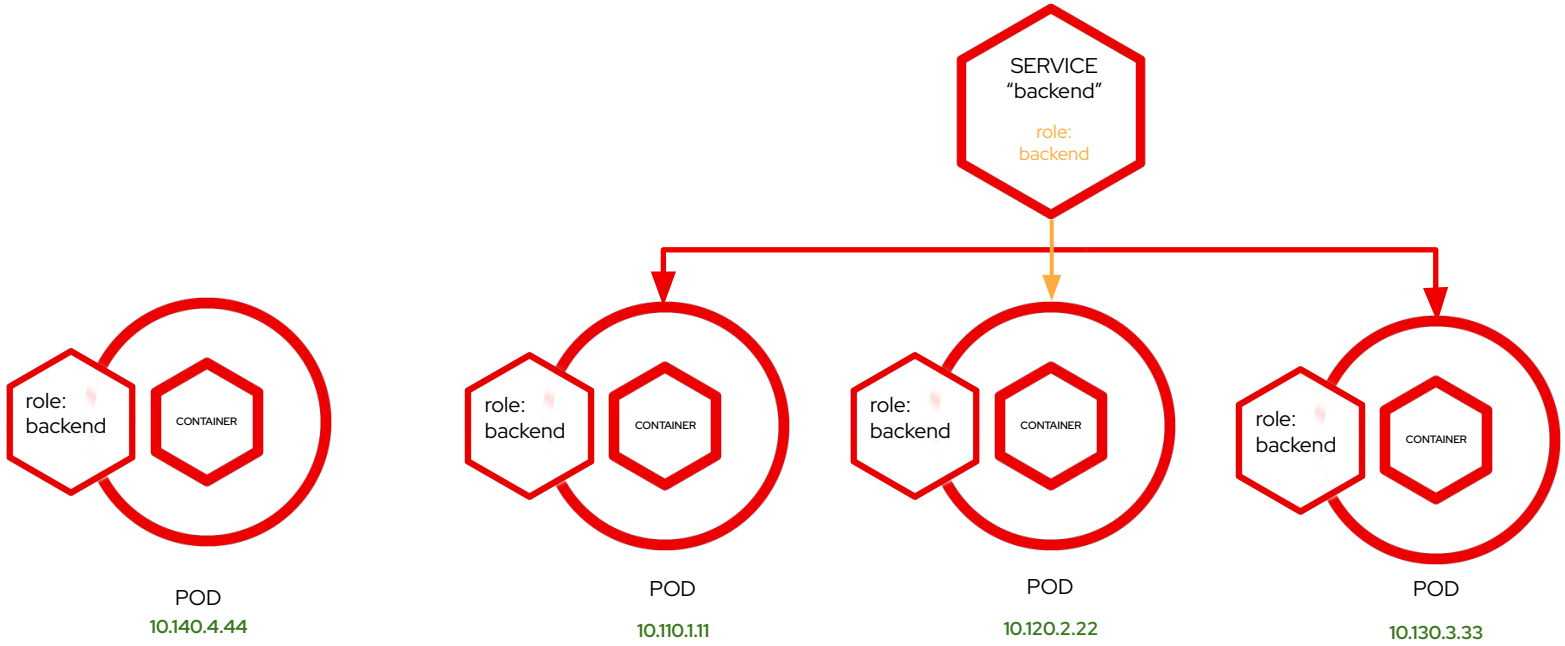
Comandi di gestione dei Pods



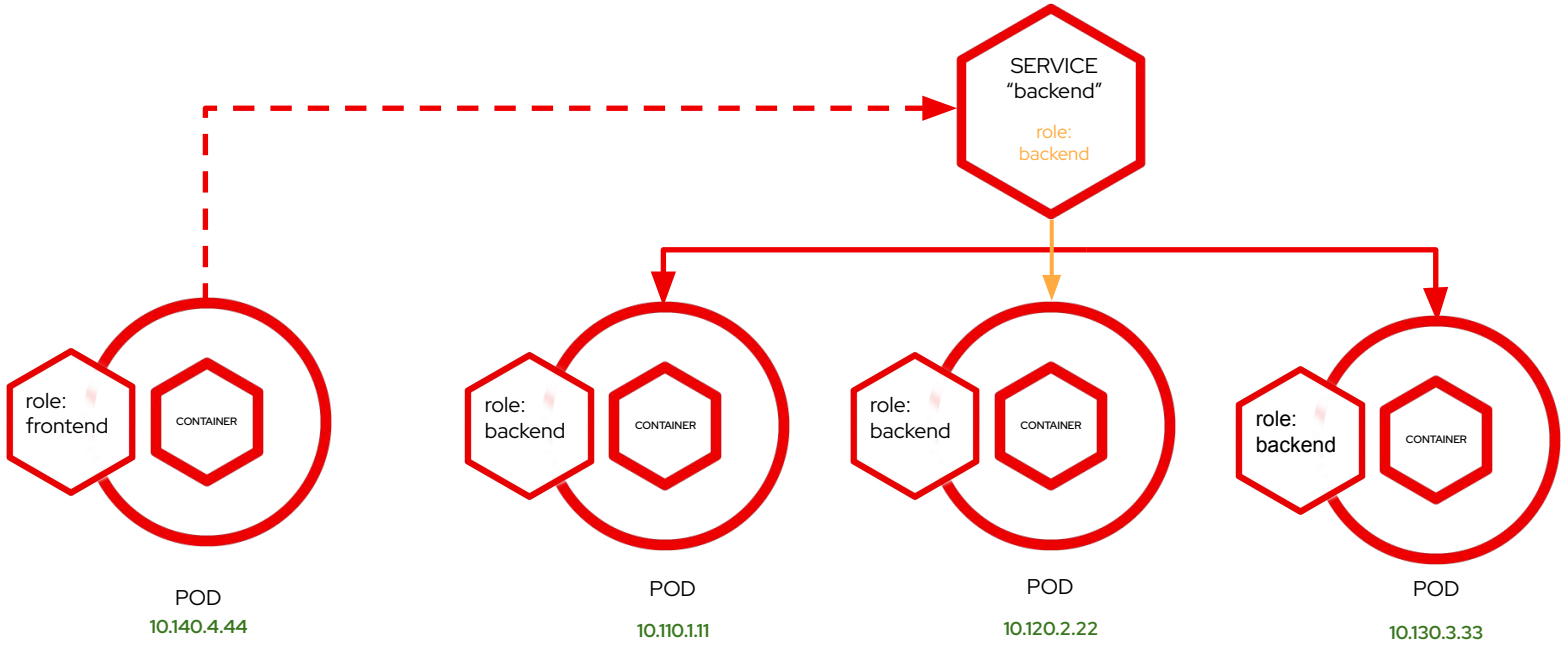
A daemonset ensures that all (or some) nodes run a copy of a pod



I Services distribuiscono le chiamate tra Pods



Le applicazioni comunicano mediante Services



Quiz: Se OpenShift fosse un cantiere ?

SERVICE
"backend"

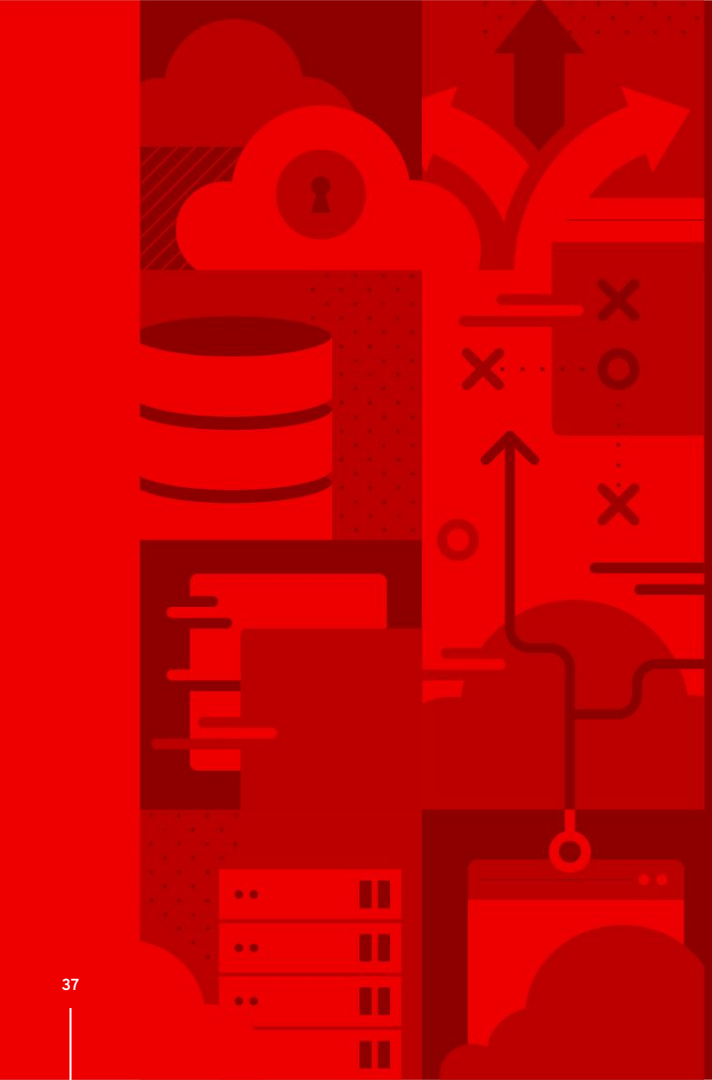
image
name
replicas
labels
version
strategy

Deployment

C

POD

Route
app-prod.acme.com



Deploy Applications on Kubernetes

Chapter 4

Kubernetes Workloads

Deployment 🔄

- Gestisce applicazioni *stateless* con rolling update.
- Ideale per web app e API.

StatefulSet 📦

- Gestisce applicazioni *stateful*, garantendo identità stabili e storage persistente.
- Usato spesso per database come PostgreSQL o MongoDB.

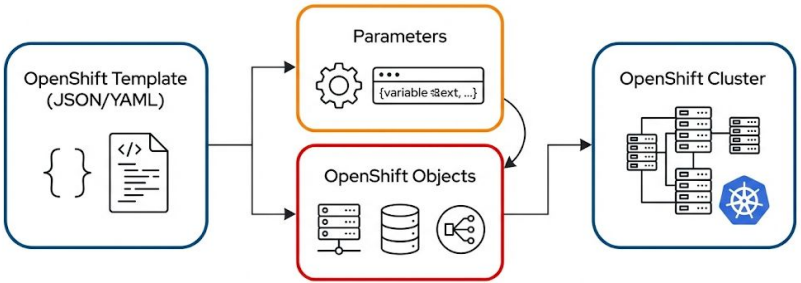
Job / CronJob ⏰

- Esegue un task una sola volta o a intervalli programmati.
- Utile per batch, migrazioni dati o job.

OpenShift Templates

I Templates sono modelli JSON/YAML riutilizzabili che definiscono oggetti OpenShift con parametri.

- ✅ Integrati nativamente in OpenShift
- ✅ Semplici da usare e condividere
- ✅ Possono definire più risorse correlate
- ✅ Usano parametri per personalizzare i workload



OpenShift Templates: Comandi

Controllo dei Templates disponibili:

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

Creazione di un Template:

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

Processing di un Template:

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

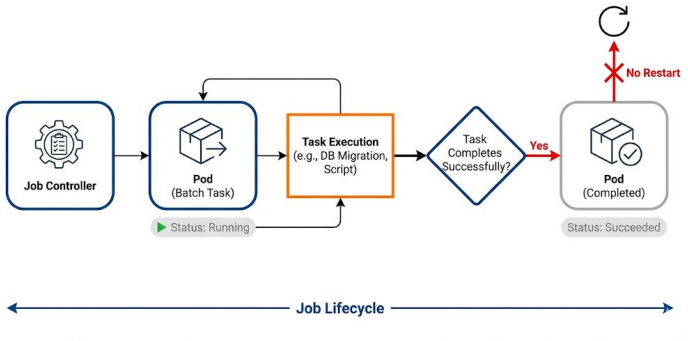
Creazione di una Applicazione da Template:

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

Jobs

- I Job eseguono un Pod fino al completamento (task batch).
- Utili per operazioni una tantum come migrazioni di database, batch processing o script.
- Quando il Job termina con successo, i Pod non vengono riavviati.

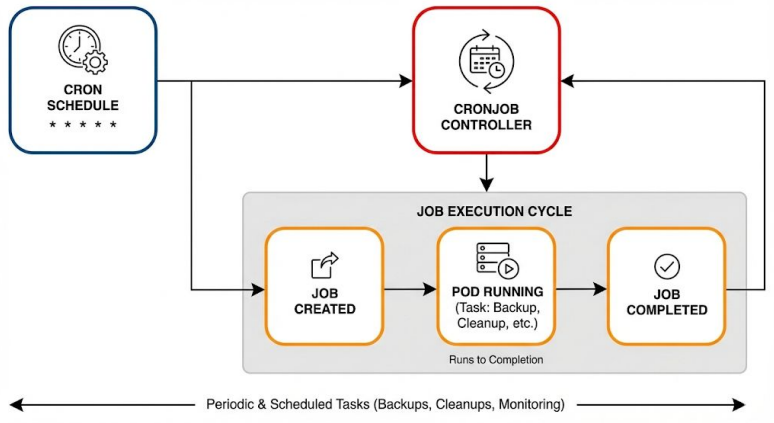
```
apiVersion: batch/v1
kind: Job
metadata:
  name: simple-job
spec:
  template:
    spec:
      containers:
      - name: simple-task
        image: busybox
        command: ["sh", "-c", "echo Hello
OpenShift; sleep 5"]
      restartPolicy: OnFailure
```



CronJobs

- I CronJob creano Jobs in base a una pianificazione (come *cron* in Linux).
- La schedule usa la sintassi cron (* * * * *).
- Utili per task periodici, backup, pulizie o script di monitoraggio.

```
apiVersion: batch/v1
kind: CronJob
metadata:
  name: example-cronjob
spec:
  schedule: "0 */6 * * *"    # every 6 hours
  jobTemplate:
    spec:
      template:
        spec:
          containers:
          - name: my-cron-job
            image: busybox
            command: ["sh", "-c", "echo Running
CronJob; sleep 10"]
          restartPolicy: OnFailure
```



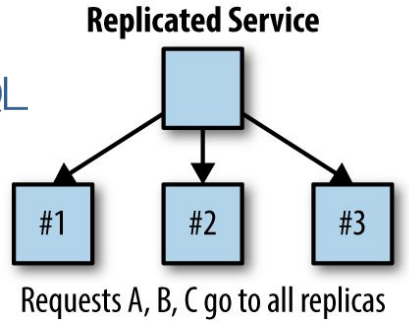
Stateful Sets and Databases

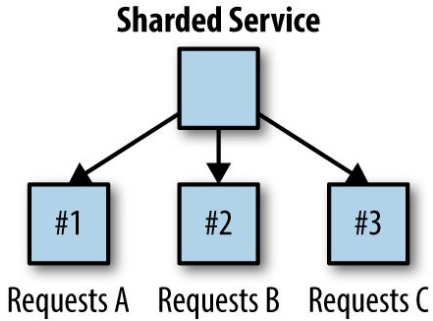
Primary-Secondary Replication:

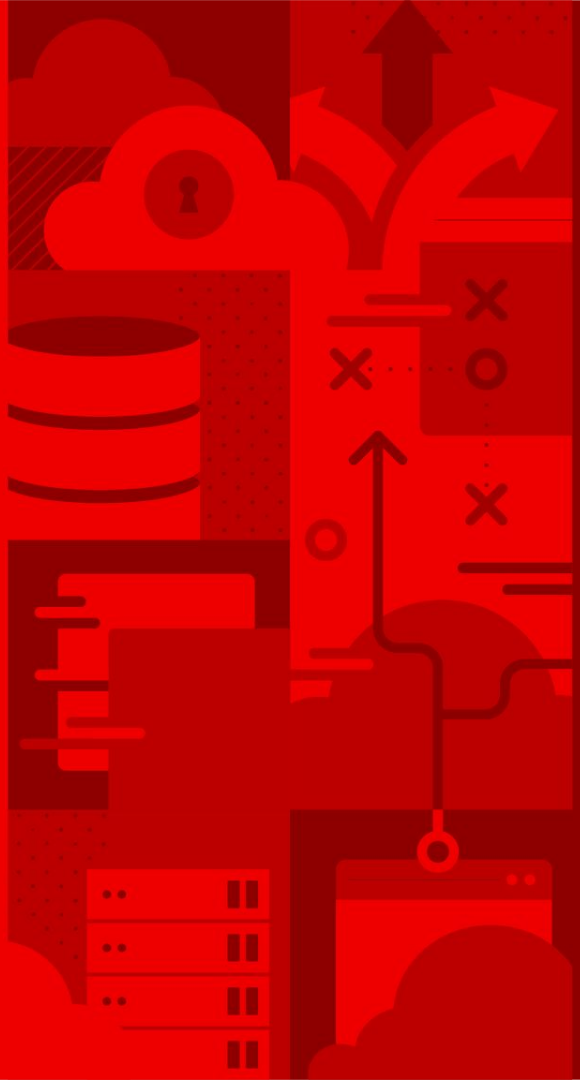
- Primary pod handles writes.
- Secondary pods replicate data and serve read queries.

Sharded Databases:

- Each pod manages a subset of data.
- No shared storage between shards.



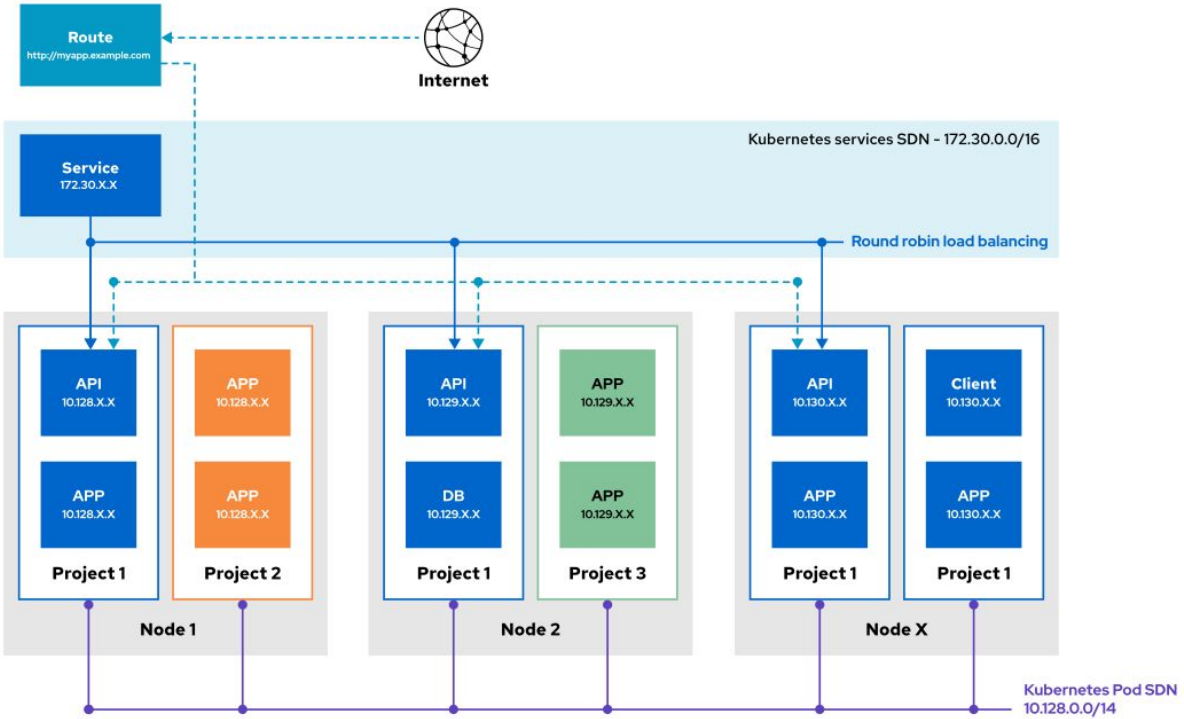




OpenShift Network

Chapter 4

OpenShift Network: vista alto livello



OpenShift Network: vista basso livello

```
$ cat install-config.yaml
[...]
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  networkType:
OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
[...]
```

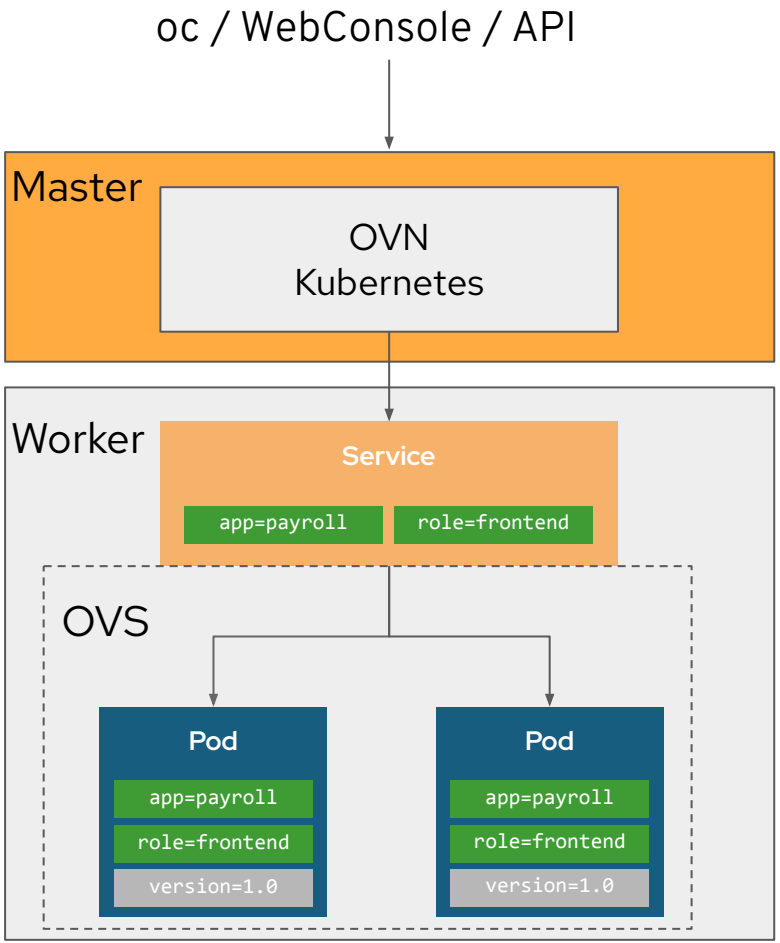
clusterNetwork.cidr: La subnet dove i **Pod** vengono allocati

- 10.128.0.0/14
  - Host min: 10.128.0.0
  - Host max: 10.131.255.255
  - Addresses in network: 262144 (max pods per cluster)

serviceNetwork cidr: La subnet dei **Service**:

- 172.30.0.0/16
  - Host min: 172.30.0.0
  - Host max: 172.30.255.255
  - Addresses in network: 65536 (max number of services)

OpenShift Network Plugins



OVN Kubernetes

- È il plug-in CNI (Container Network Interface) utilizzato da OpenShift quando si sceglie OVN come soluzione di rete.
- **Traduce** le richieste di rete di Kubernetes (creazione dei Pod, servizi, network policy) in **regole** applicate sui Nodi da OVS.

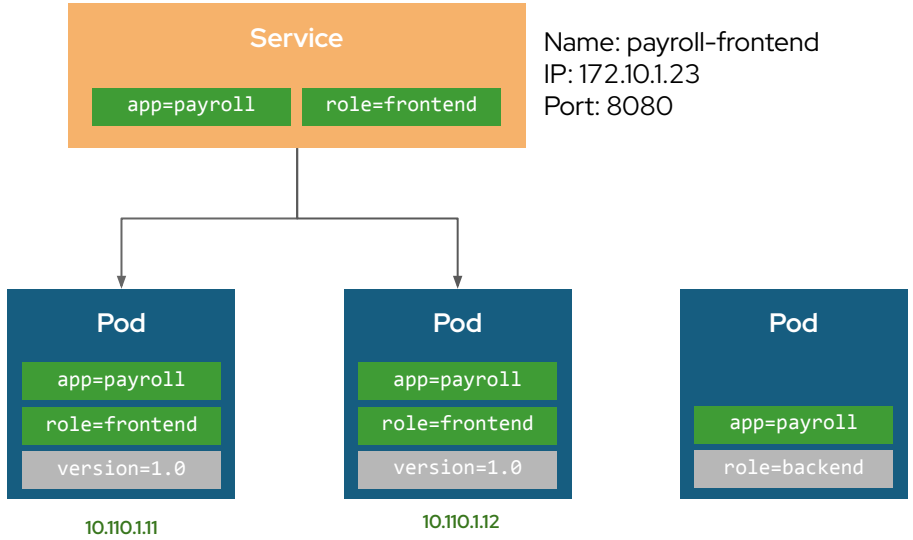
OVS (Open VSwitch)

- È uno switch virtuale che gira su ogni nodo del cluster.
- Esegue la traduzione degli indirizzi (DNAT), trasformando l'IP virtuale del Service nell'IP reale del Pod prima dell'inoltro.

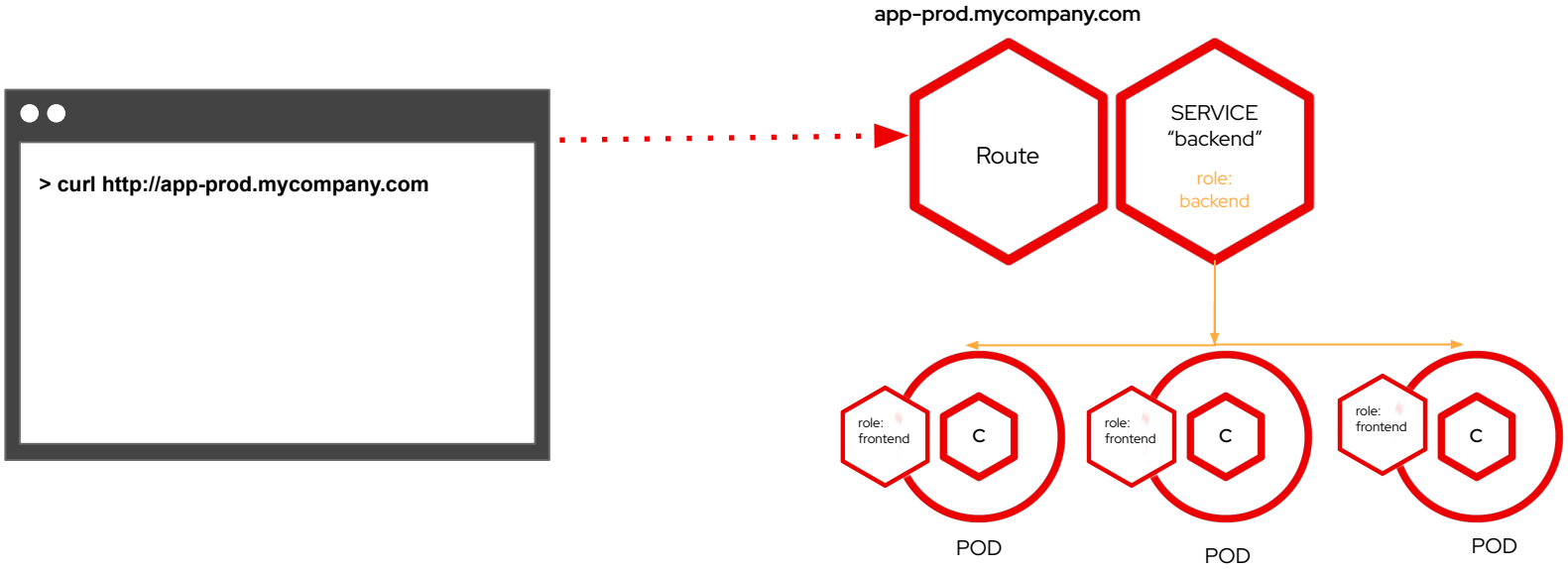
Kubernetes Services

Un **Service** è un'astrazione che definisce un insieme logico di Pod e la policy per accedervi. Fornisce un indirizzo IP dell'overlay network che effettua il load balancing verso i Pod che corrispondono al selector definito.

```yaml
apiVersion: v1
kind: Service
metadata:
  name: payroll-frontend
spec:
  selector:
    app: payroll
    role: frontend
  clusterIP: 172.10.1.23
  ports:
  - port: 8080
    protocol: TCP
    targetPort: 8080
```



Le Route rendono i Service accessibili ai client esterni tramite HTTP/S



Esposizione del traffico con NodePort

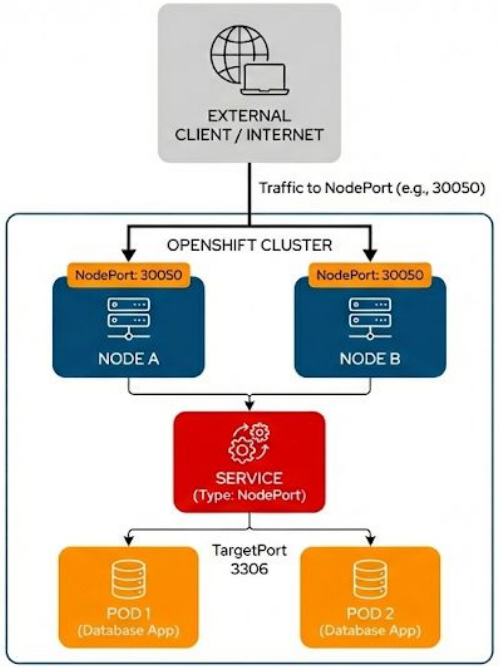
- Espone un Service su una porta dedicata di tutti i nodi del cluster.
- Usa porte nel range 30000–60000, di solito diverse dalla porta del Service.

✅ Vantaggi

- Funziona anche senza load balancer esterni.

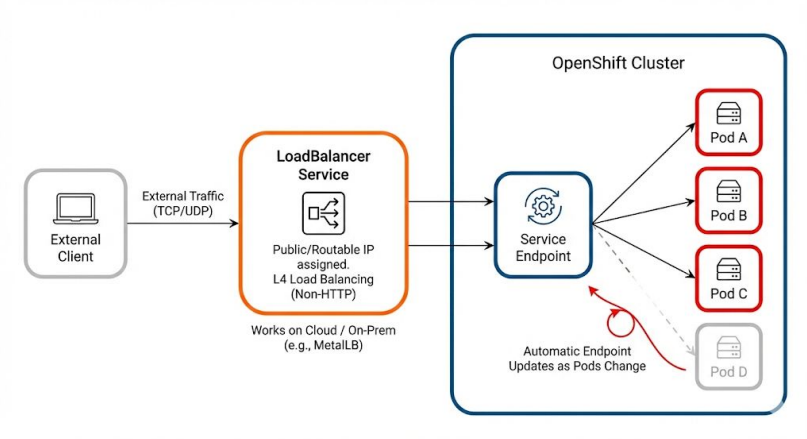
❌ Svantaggi

- Richiede conoscenza IP Nodo e settaggio regole di Firewall



Esposizione del traffico con Load Balancers

- Espone un Service verso l'esterno assegnandogli un IP pubblico
- Funziona sia su cloud provider sia on-prem tramite MetalLB.
- Bilanciamento L4 (TCP/UDP), ideale per traffico non-HTTP.



✅ Vantaggi

- Accesso esterno semplice e diretto tramite per ogni protocollo

❌ Svantaggi

- Richiede un componente esterno che gestisca gli IP (cloud LB o MetalLB).

External Traffic to a Service using External IP

Assigns a fixed external IP address to a service.

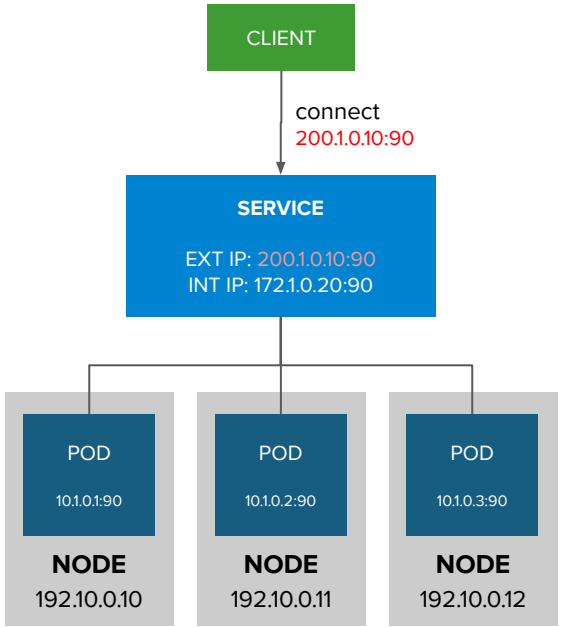
Requires manual network configuration to route traffic to the correct node.

✅ Advantages

- Useful for integrating with on-premise or legacy networks using external LBs.

❌ Disadvantages

- OpenShift does not manage external IPs automatically.

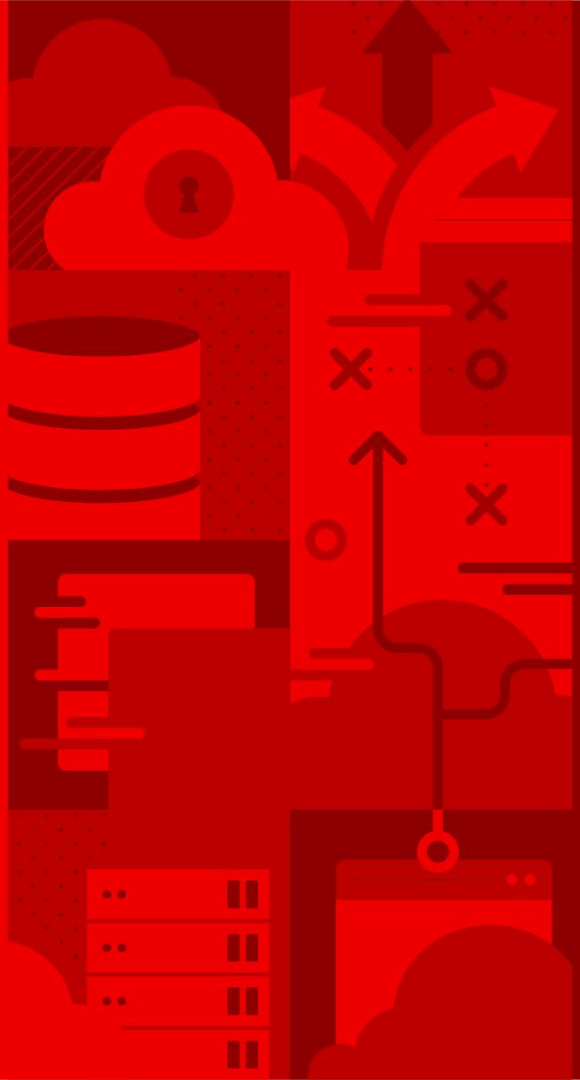


Load Balancer Services

- Load balancer services require the use of network features that are not available in all environments.
- Cloud providers typically provide their own load balancer services.

```
$ oc get svc -n openshift-ingress
NAME   TYPE CLUSTER-IP  EXTERNAL-IP                  PORT(S)
service/router-default    LoadBalancer        172.30.22.68          xx-yy.ap-northeast-1.elb.amazonaws.com
80:31155/TCP,443:32009/TCP
```

- MetalLB is a load balancer component that provides a load balancing service for clusters that do not run on a cloud provider

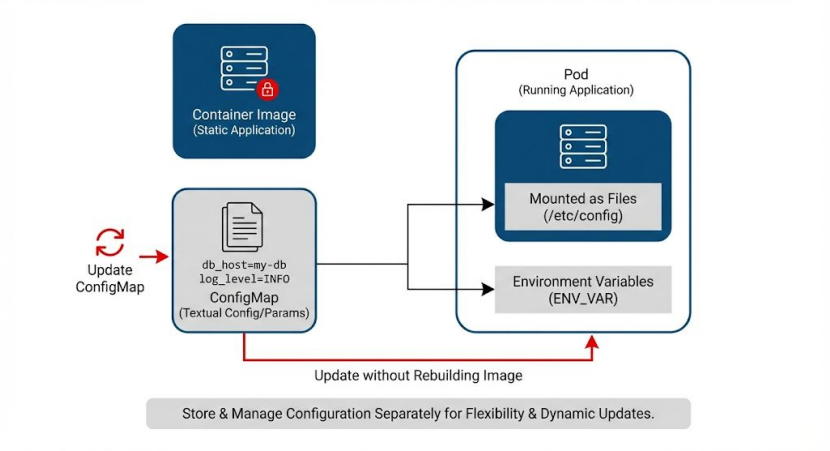


Manage Storage

Chapter 5

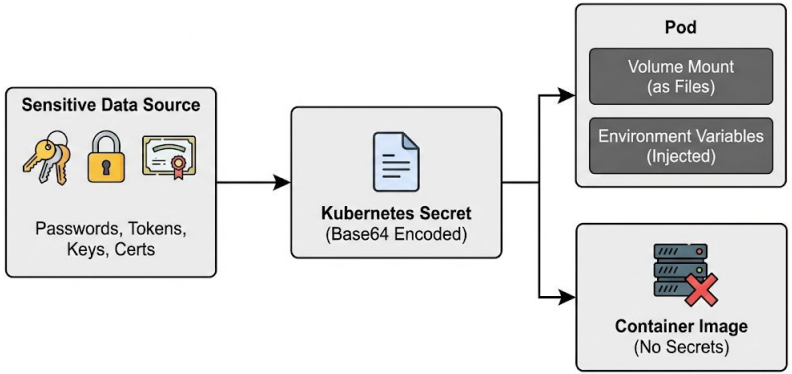
ConfigMaps

- Configurazioni testuali o parametri applicativi separati dall'immagine del container.
- Possono essere montate come file o esposte come variabili d'ambiente nei Pod.
- Permettono di aggiornare la configurazione senza ricostruire l'immagine dell'applicazione.



Secrets

- Conservano informazioni sensibili come password, token, certificati e chiavi, **separandole** dall'immagine del container.
- Possono essere montati come file o esposti come variabili d'ambiente nei Pod, con accesso controllato.
- Sono codificati in base64 e conservati in etcd.

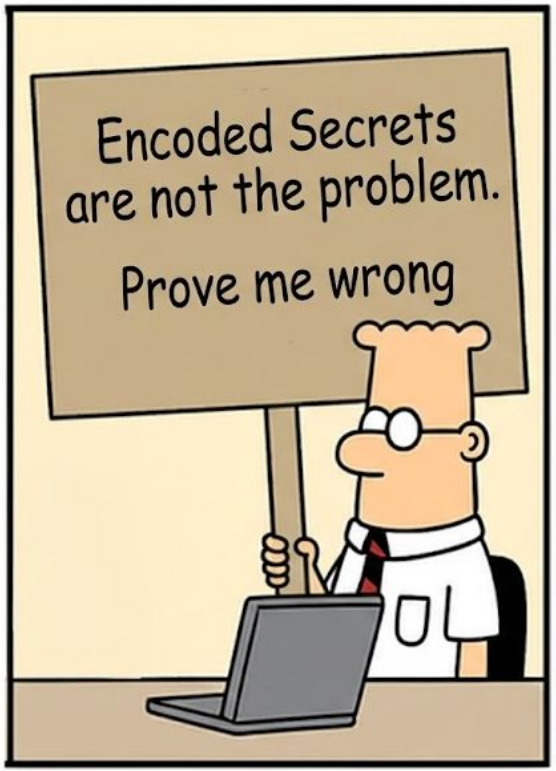


Secrets & encoding

- I secrets sono registrati in formato encoded.
- L'encoding **non è** un meccanismo di sicurezza
- E' un problema per la sicurezza ?

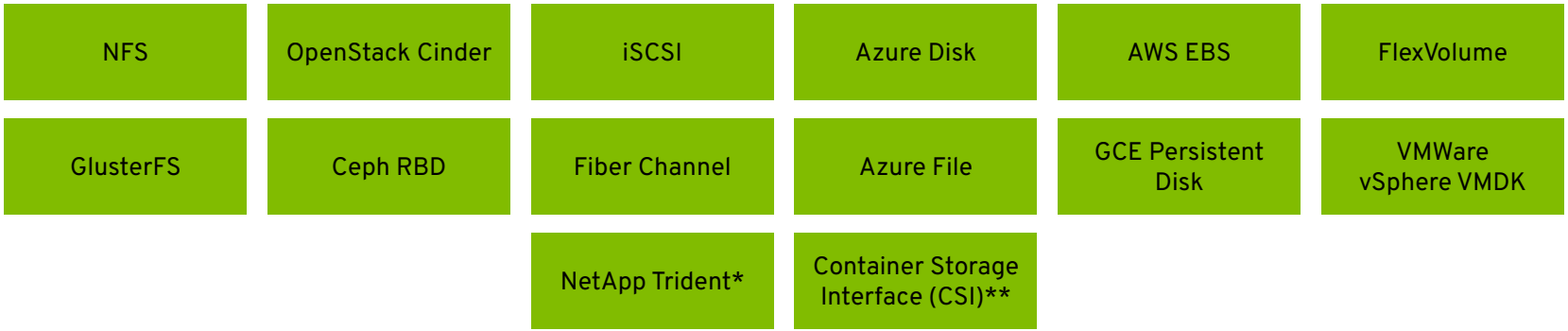
```
oc get secret db-credentials \
  -o jsonpath='{.data.password}' | base64
--decode

S3cr3tP4ss
```



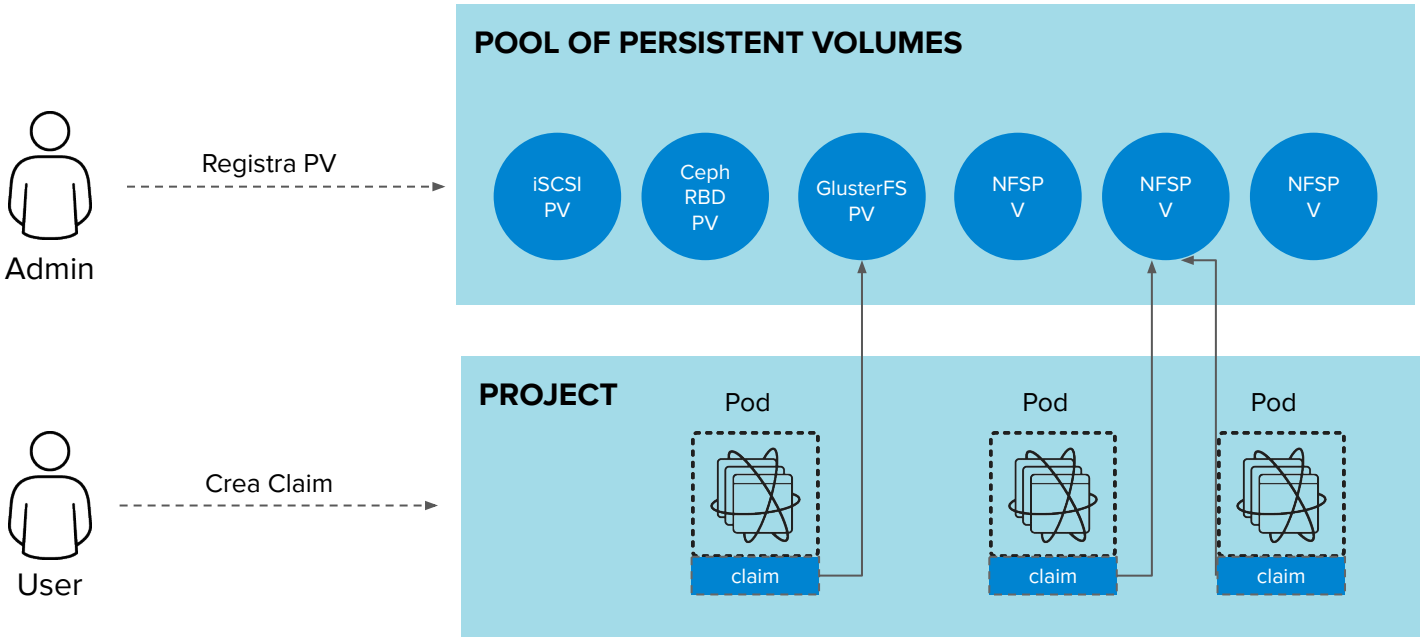
Persistent Storage

- Un Persistent Volume (PV) è associato a una porzione di storage di rete.
- Viene fornito da un amministratore (in modo statico o dinamico).
- Permette agli amministratori di descrivere lo storage e agli utenti di richiederlo.



* Shipped and supported by NetApp via TSANet
** Tech Preview

Provisioning Statico dei Volumi



Persistent Volume (PV)

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv0001
  annotations:
    volume.beta.kubernetes.io/mount-options: rw,nfsvers=4,noexec
spec:
  capacity:
    storage: 1Gi
  accessModes:
  - ReadWriteOnce
  nfs:
    path: /tmp
    server: 172.17.0.2
  persistentVolumeReclaimPolicy: Retain
  claimRef:
    name: claim1
    namespace: default
```

Specify Mount options (Explained in the next slides)

Set the specific the capacity of the PersistentVolume

Set the access Mode (Explained in the next slides)

Indicates how the resource should be handled once it is released

Persistent Volume Claim (PVC)

```yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: myclaim
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 8Gi
  storageClassName: gold
status:
  ...
```

Set the access Mode

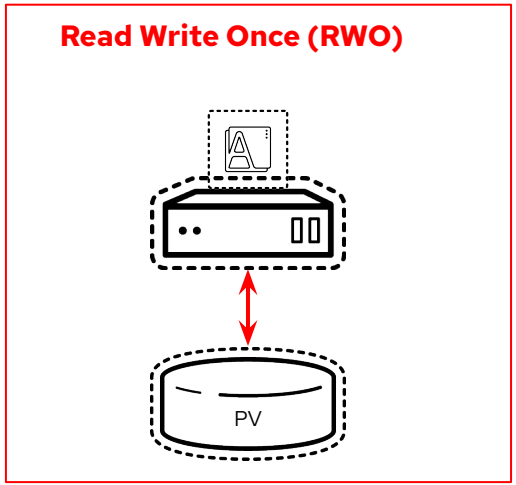
Specify the amount of storage

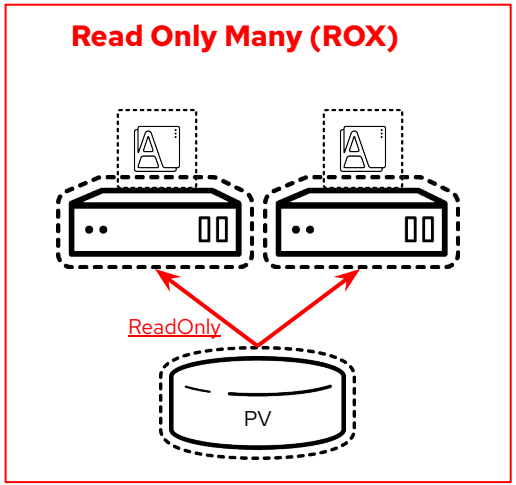
Specify the Storage Class (Explained in the next slides)

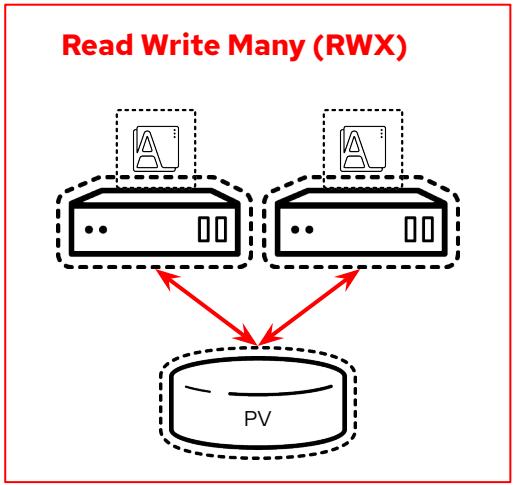
Bind del PVC al PV

I PVCs solo collegati ai Volumi in base alla compatibilità tra:

- accessMode
- Capacity
- storageClass (optional)

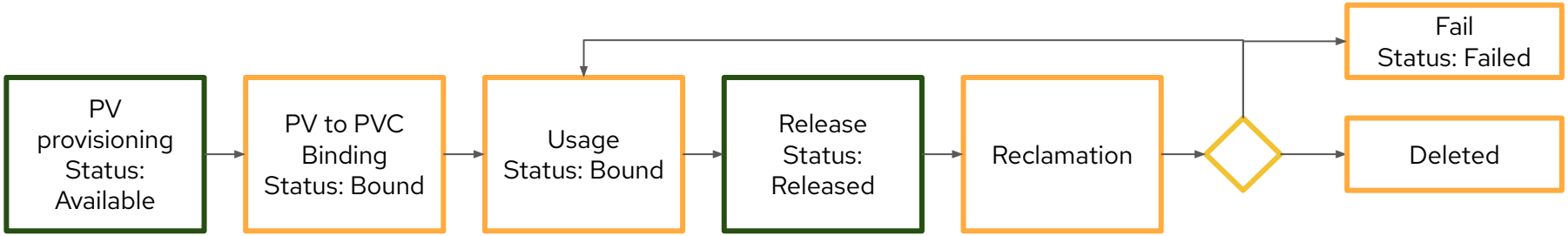






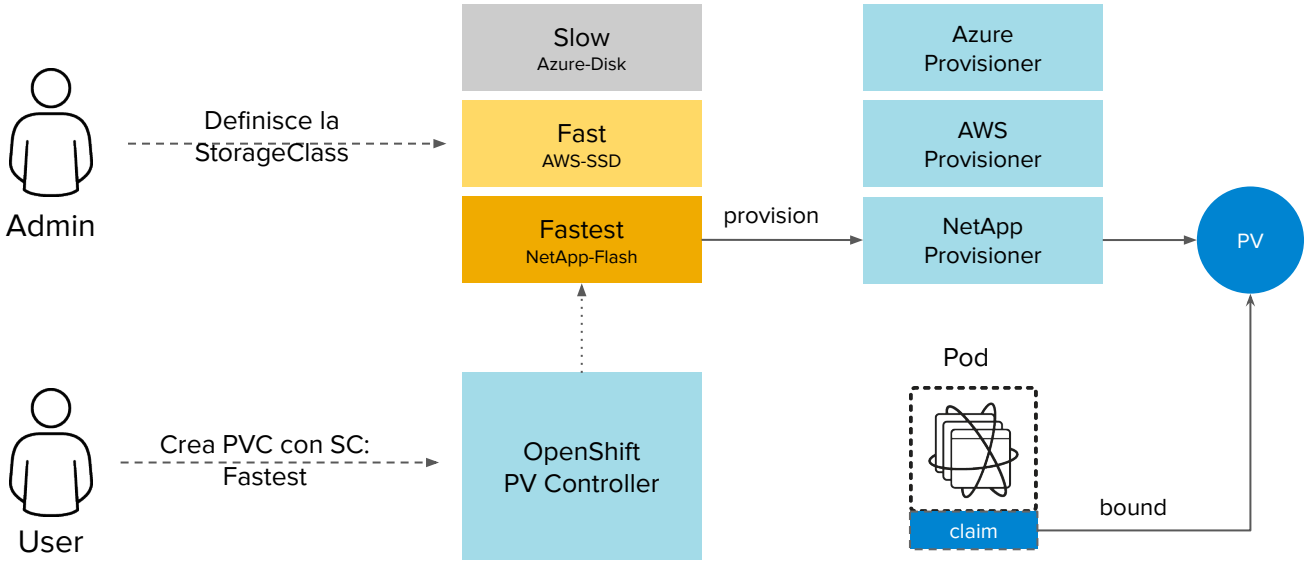
62

Ciclo di vita di un Persistent Volume



Phase	Description
Available	A free resource not yet bound to a claim.
Bound	The volume is bound to a claim.
Released	The claim was deleted, but the resource is not yet reclaimed by the cluster.
Failed	The volume has failed its automatic reclamation.

Provisioning Dinamico dei Volumi



Storage Class

La StorageClass è una risorsa che descrive un tipo di storage che è possibile richiedere

Hanno lo scope a livello di Cluster

Sono definiti dagli utenti cluster-admin di OpenShift

La storageClass classifica lo Storage in base a criteri tecnici:
- Technology (AWS,Vsphere..)
- Service Quality (Example: Disktype for vSphere: zeroedthick or thin)

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: gp2
  annotations:
    storageclass.kubernetes.io/is-default-class: 'true'
    ...
provisioner: kubernetes.io/aws-ebs
parameters:
  type: gp2
```

Annotations for the storage class. Include kubernetes.io/description to add a description.

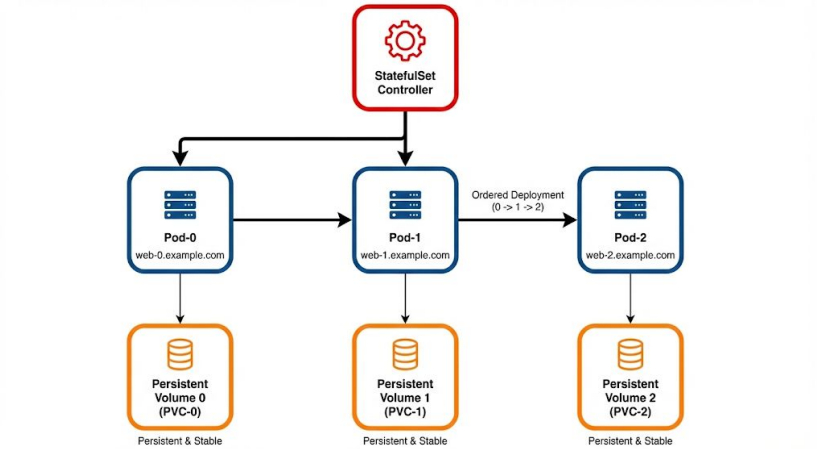
Storage Provisioner (only dynamic)

Parameters specific to the storage provisioner

Stateful Sets

Gli StatefulSet sono una risorsa Kubernetes pensata per gestire applicazioni *stateful* con identità di rete stabili, storage persistente e deployment ordinati.

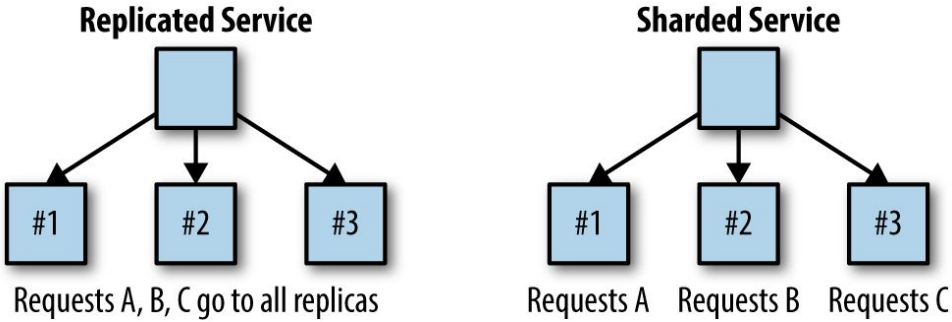
- **Nomi dei Pod stabili**: i Pod vengono creati in ordine prevedibile (es. db-0, db-1).
- **Storage persistente**: ogni Pod mantiene il proprio volume anche dopo un riavvio.
- **Scaling e rolling update ordinati**: creazione e rimozione dei Pod avvengono in modo controllato.

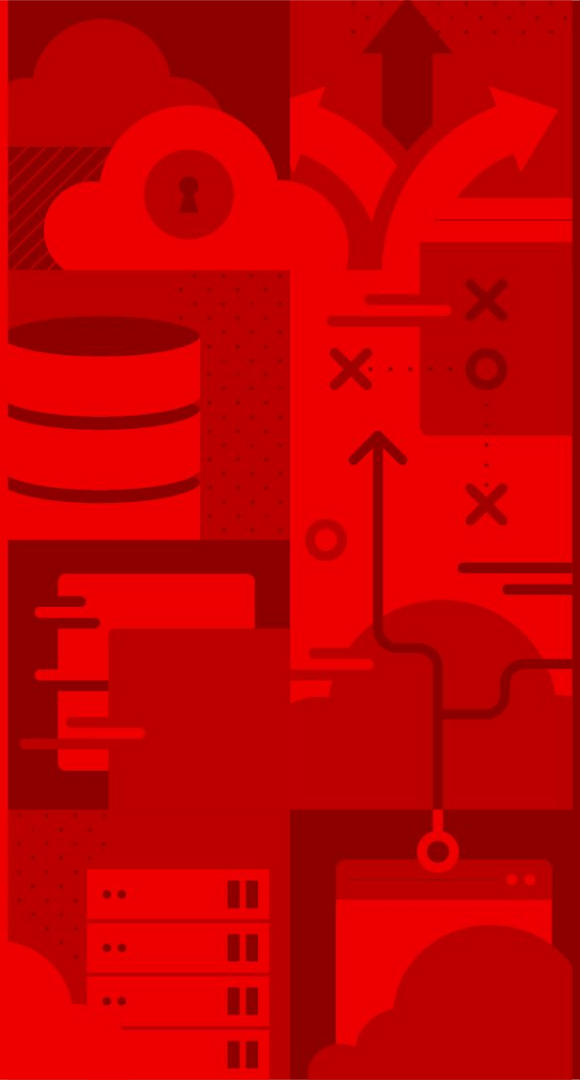


Stateful Sets use cases

Uso tipico: Gestire un database distribuito (es. PostgreSQL, MongoDB, Cassandra) in cui ogni Pod deve mantenere identità stabile e il proprio volume dati persistente.

- Repliche primarie/secondarie (PostgreSQL, MySQL, MongoDB replica set).
- Sharded databases (MongoDB Sharded Cluster, Cassandra, Elasticsearch, CockroachDB).
- Vantaggi: Assicura Consistenza, Failover del Nodo





High Availability

Chapter 6

Kubernetes High Availability

Kubernetes utilizza le seguenti tecniche per ottenere HA dei workloads:

- **Restart dei Pod**: configurando una restart policy, il cluster riavvia automaticamente le istanze dell'applicazione che si comportano in modo anomalo.
- **Probing**: grazie alle health probe, il cluster rileva quando un'applicazione non risponde e può intervenire automaticamente per mitigare il problema.
- **Horizontal AutoScaler**: quando il carico varia, il cluster può aumentare o ridurre il numero di repliche per adeguarsi alla domanda.

Restart Policy

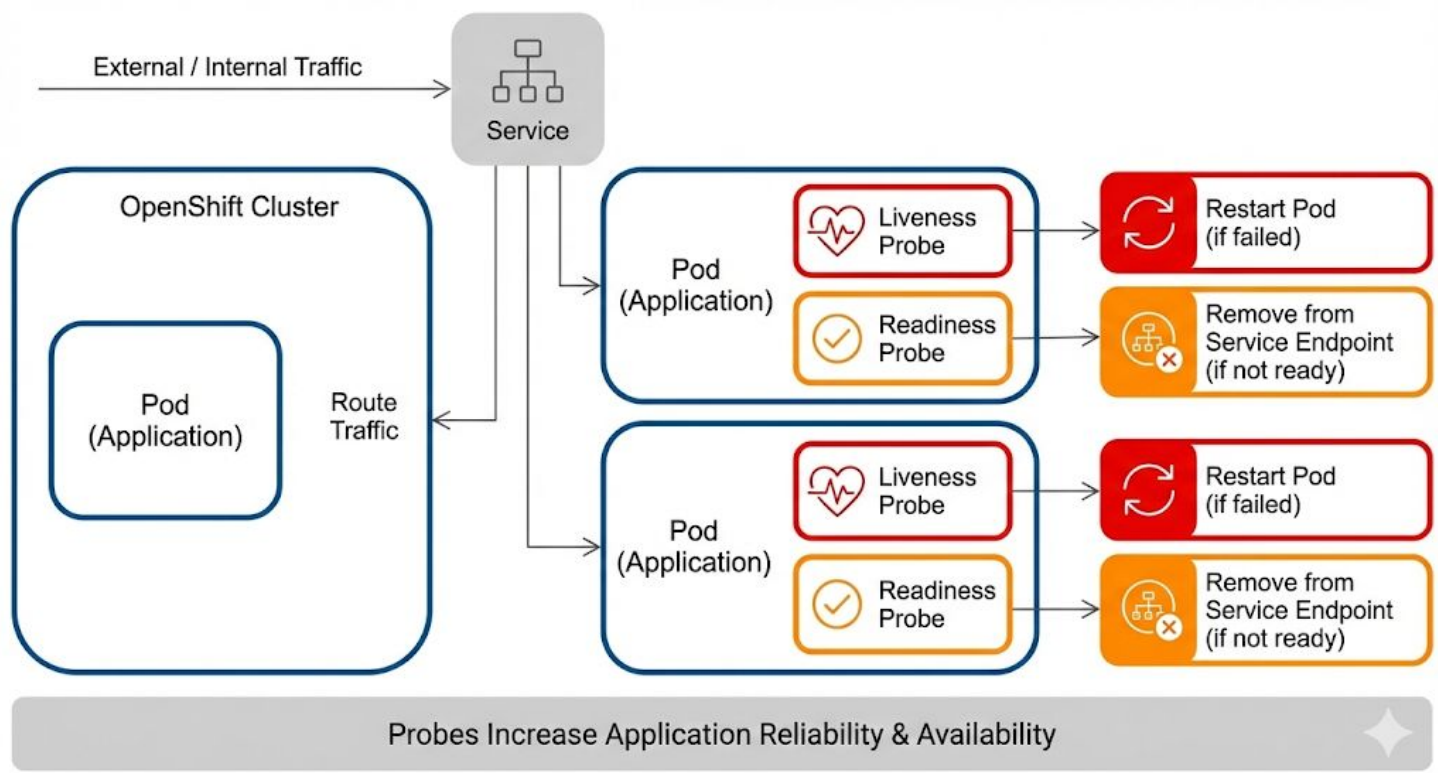
```
apiVersion: v1
kind: Pod
metadata:
  name: example-pod
  namespace: my-namespace
spec:
  restartPolicy: OnFailure
  containers:
      - name: my-container
      image: httpd:latest
      ports:
      - containerPort: 8080
```

The Pod restarts only if the container exits with a failure (non-zero exit code).

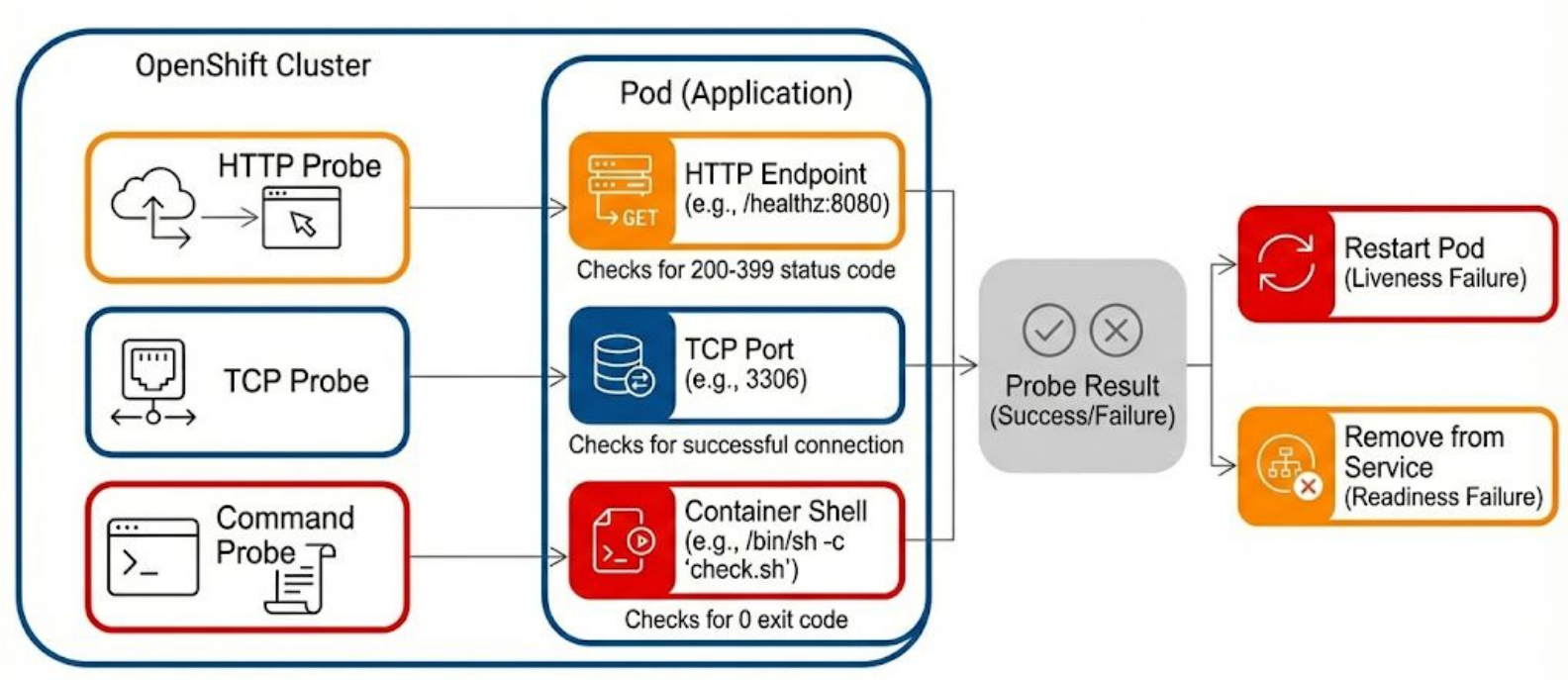
Explanation of restartPolicy Values

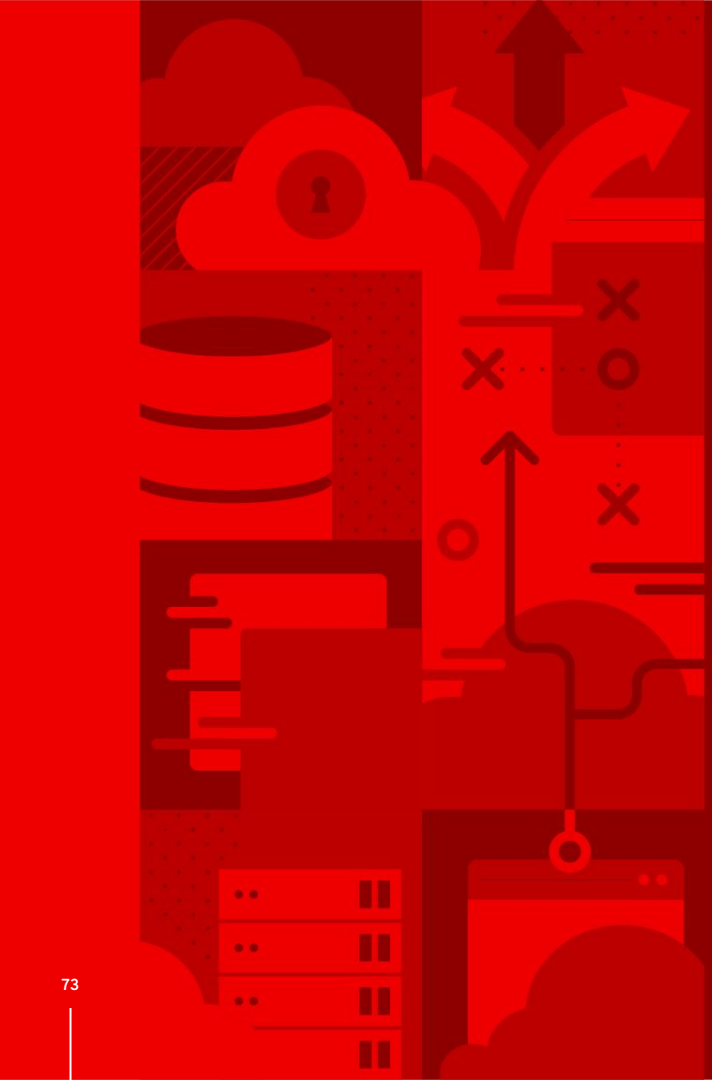
- Always (default) → The Pod restarts automatically if the container stops (even if it exits successfully).
- OnFailure → The Pod restarts only if the container exits with a failure (non-zero exit code).
- Never → The Pod never restarts, even if it fails.

Liveness and Readiness Probes



Probes e Trasporto





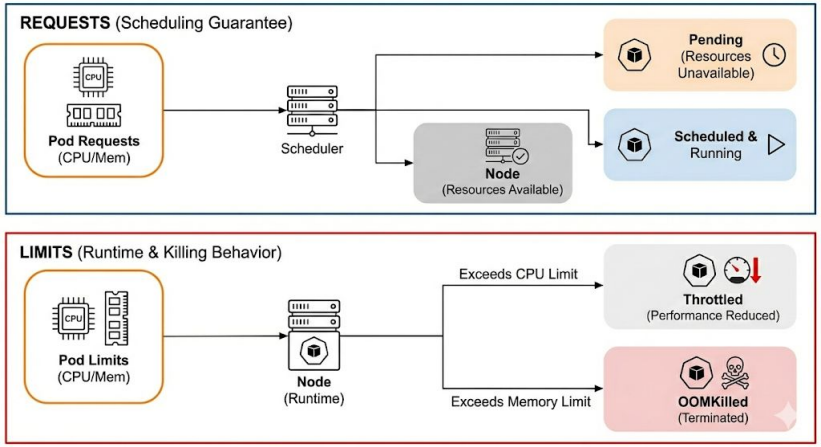
Limit Compute Capacity for Applications

Chapter 6

Gestione risorse dei Workloads

Le *Requests* → influenzano lo Scheduling

- Quantità pianificata di CPU e memoria per un Pod.
- Lo scheduler verifica che il nodo abbia requests sufficienti.
- I Pod restano in Pending se le richieste non possono essere soddisfatte.



I *Limits* → influenzano il Runtime

- Se supera il limite di CPU → il Pod viene throttled (non terminato).
- Se supera il limite di memoria → il Pod viene OOMKilled

Configurazione Requests, 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)
```

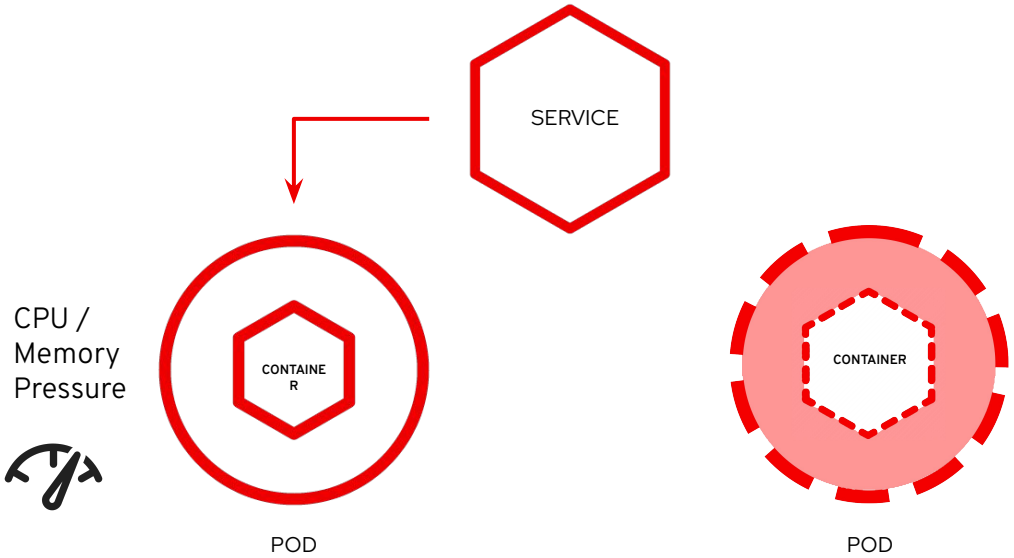
Resource Quotas

Le Resource Quota sono definite dai cluster admin per stabilire la Quota di risorse per un singolo namespace

```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: compute-resources
spec:
  hard:
    requests.cpu: "2"
    requests.memory: "4Gi"
    limits.cpu: "4"
    limits.memory: "8Gi"
```

Horizontal Autoscaler

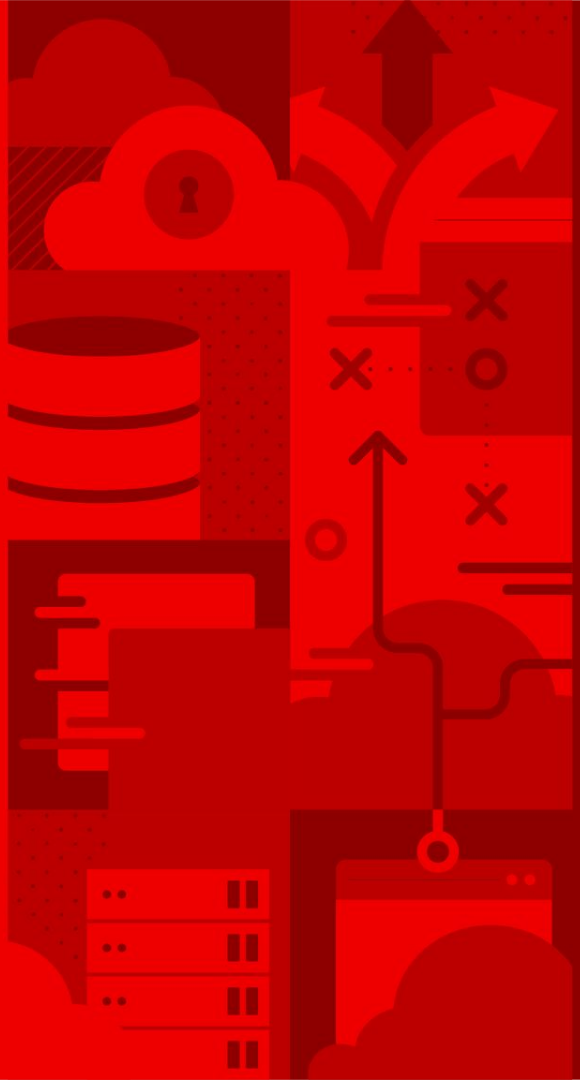
L'Horizontal Pod Autoscaler (HPA) scala automaticamente il numero di pod in un deployment, replica set o stateful set in base a metriche di CPU, memoria o metriche personalizzate.



Come funziona l'HPA

- **Monitora l'utilizzo delle risorse**: l'HPA controlla le metriche (ad esempio CPU e memoria) tramite il Kubernetes Metrics Server.
- **Regola il numero di pod**: se l'utilizzo delle risorse supera la soglia definita, l'HPA aumenta il numero di pod; se l'utilizzo diminuisce, li riduce.
- **Garantisce uno scaling efficiente**: contribuisce a mantenere le prestazioni dell'applicazione ottimizzando al tempo stesso l'uso delle risorse.

```
oc autoscale deployment/hello --min 1 --max 10 --cpu-percent 80
```



ImageStreams

Chapter 7

Cosa sono gli ImageStream ?

Gli **ImageStream** sono componenti che fanno riferimento a immagini presenti nei registry (interni o esterni).

```
oc create deployment version --image hello:v1.0
```

Memorizzano metadati sulle immagini, non le immagini stesse.

Gli aggiornamenti delle immagini attivano CI/CD.

Un ImageStream contiene questi riferimenti:

- ImageStream (riferimento logico)
- ImageStreamTag (versione specifica dell'immagine)
- ImageStreamImage

Esempio di ImageStream

- Definisce un **ImageStream** "hello".
- Punta ad immagini da un registry (es Quay.io o Docker Hub).
- Può essere usato nelle BuildConfigs e/o Deploy in sostituzione di Image

```
oc create is hello

oc tag registry.ocp4.example.com:8443/versioned-hello:v1.0  hello:v1.0

oc create deployment version --image hello:v1.0
```

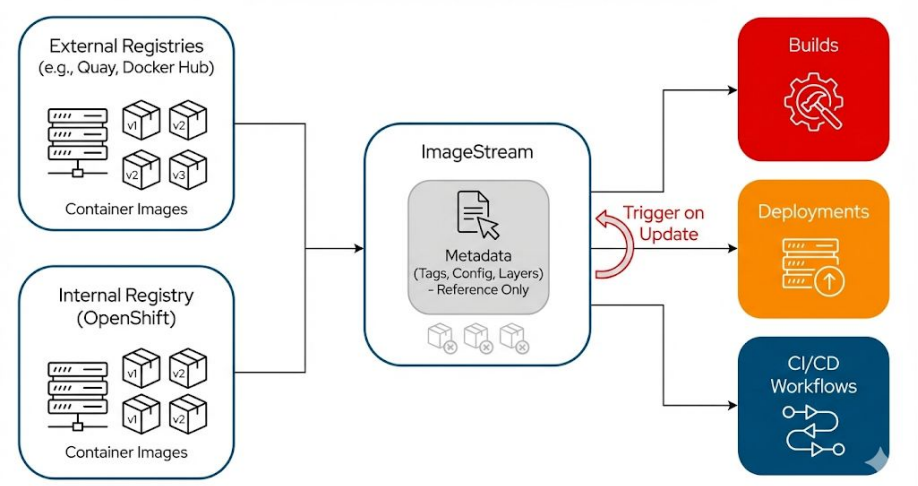
ImageStream Use Cases

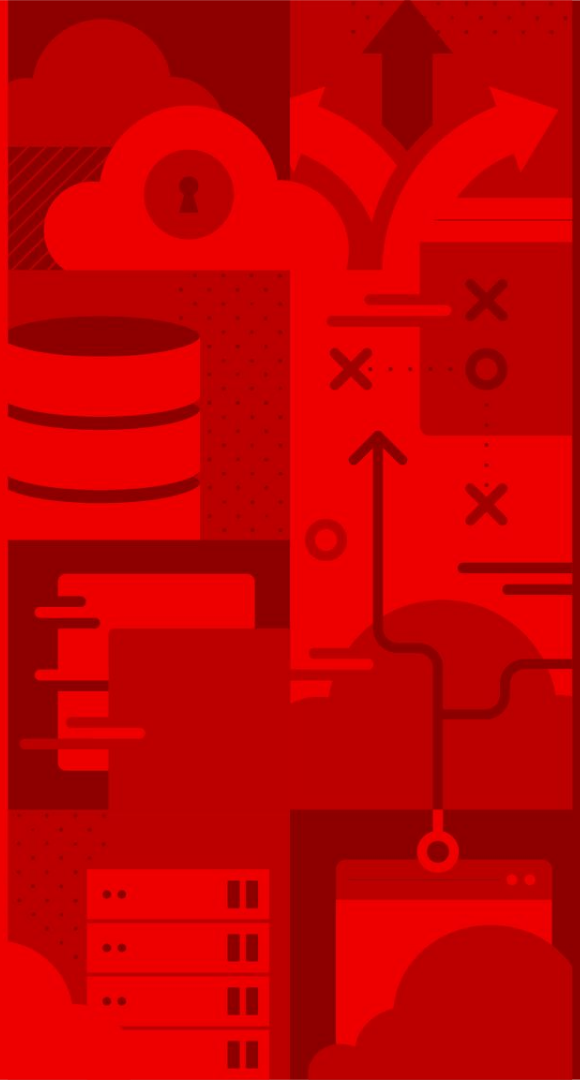
Deployment automatizzati: aggiorna le applicazioni quando una nuova immagine è disponibile.

Integrazione CI/CD: attiva le build quando viene pubblicata una nuova immagine di base.

Supporto al rollback: mantiene più versioni dell'immagine per un ripristino rapido.

Sicurezza: garantisce che vengano distribuite solo immagini approvate.

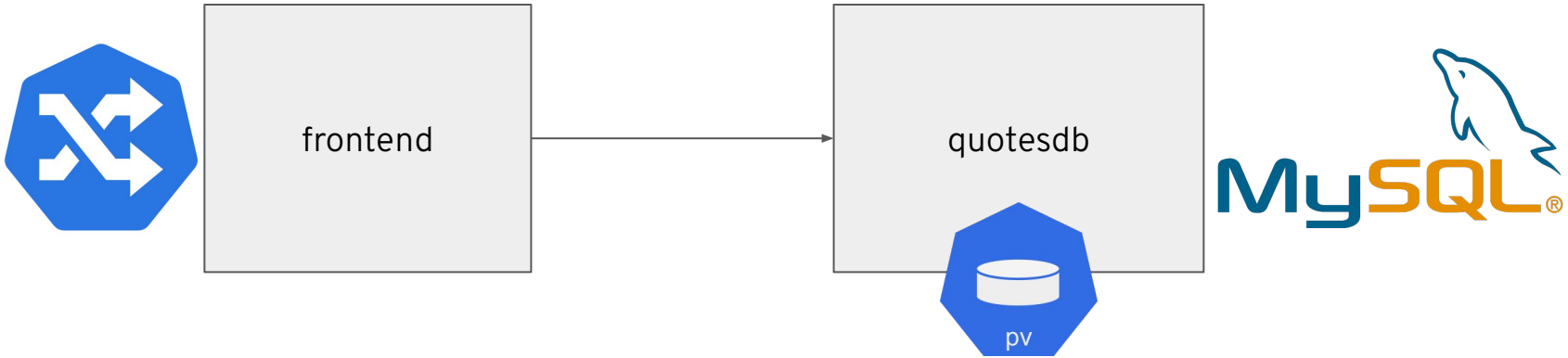




Final Review

Lab compreview-deploy

Estimated time:
45 minutes



Lab compreview-scale

Estimated time:
45 minutes