Realizzazione e gestione di una nuova infrastruttura informatica al servizio della Pubblica Amministrazione denominata Polo Strategico Nazionale (“PSN”), di cui al comma 1 dell’articolo 33-septies del d.l. n. 179 del 2012

CUP: J51B21005710007

CIG: 9066973ECE

PSN

Manuale Operativo

CaaS Open

Data:10/12/2023

PSN_UserGuide_CaaS_Open_v3

Ed. 3 - ver. 1.0

QUESTA PAGINA È LASCIATA INTENZIONALMENTE BIANCA

STATO DEL DOCUMENTO

TITOLO DEL DOCUMENTO			
PSN User Guide CaaS			
EDIZ.	REV.	DATA	AGGIORNAMENTO
1	1.0	22/07/2023	Prima versione del documento di User Guide del servizio CaaS
1	1.1	27/07/2023	Aggiunta sezione Integrazione API
2	1.0	20/11/2023	Seconda versione del documento di User Guide del servizio CaaS – modificati puntamenti ed aggiunti dettagli
3	1.0	10/12/2023	Terza versione del documento di User Guide del servizio CaaS – aggiornamento e approfondimenti

NUMERO TOTALE PAGINE:	73

AUTORE:	
Team di lavoro PSN	Unità operative Solution Development, Technology Hub e Sicurezza

REVISIONE:	
Referente del Servizio	Paolo Trevisan

APPROVAZIONE:	
Direttore del Servizio	Antonio Garelli

LISTA DI DISTRIBUZIONE

INTERNA A:

- Funzione Sviluppo della soluzione
- Funzione Technology Hub
- Funzione Sicurezza
- Referente Servizio
- Direttore Servizio

ESTERNA A:

Direttore dell'Esecuzione Contrattuale (DEC)	PSN ing. Fabrizio Marchese

INDICE

LISTA DELLE FIGURE

LISTA DELLE TABELLE

1 Definizioni e Acronimi

1. Definizioni

Definizione	Descrizione
NewCo	È la nuova società (New Company) che è stata costituita nell'ambito del progetto del Cloud Nazionale
TBC	Il tema è stato discusso ma è in attesa di conferma dalle parti coinvolte
TBD	Il tema non è ancora stato discusso

Tabella 1. Glossario Definizioni

Definizione	Descrizione
Cloud Portal IaaS VMware	Identifica il Portale di accesso alla Piattaforma offerta dal PSN
Console Tecnica IaaS	Identifica il Portale Tecnico di amministrazione del Servizio IaaS
Cloud IaaS VMware	Identifica il Servizio IaaS offerto dal PSN

Tabella 2. Nomenclatura

2. Acronimi

Acronimo	Descrizione
ALB	Application Load Balancer
CaaS	Container as a Service
CSE	Cluster e Container Service Extension
CSI	Container Storage Interface
DNAT	Traduzione degli indirizzi di rete Destinazione
FQDN	Fully Qualified Domain Name
HA	Alta Affidabilità
HLD	High Level Design
IaaS	Infrastructure as a Service
IAM	Identity and Access Management
GUI	Grafhical User Interface
LCM	Gestione del ciclo di vita
PA	Pubblica Amministrazione
PSN	Polo Strategico Nazionale
RBAC	Controllo Degli Accessi Basato Sui Ruoli
SNAT	Traduzione degli indirizzi di rete Sorgente
TKG	Tanzu Kubernetes Grid
NAT	Traduzione degli indirizzi di rete
NSX	Network Security
SVC	Service Openshift
UI	Interfaccia Utente
vCD	VMware Cloud Director
VDC	Virtual Datacenter
VIP	Virtual IP

Acronimo	Descrizione
VM	Macchina Virtuale

Tabella 3.Glossario Acronimi

2 Panoramica Generale

Il "*Containers as a Service*" è un modello di servizio basato su cloud computing che aiuta a gestire e distribuire applicazioni utilizzando l'astrazione basata su container.

Il PSN offre una piattaforma di orchestrazione gestita basata sul software RedHat Openshift Container Platform, sulla quale i container vengono distribuiti e gestiti.
Tale servizio si rivela utile soprattutto per gli sviluppatori, ai quali consente di realizzare app containerizzate più sicure e scalabili semplificando la gestione infrastrutturale.

1. Scopo del documento

Questo documento rappresenta un manuale con le linee guida di utilizzo della soluzione oltre a contenere una raccolta delle domande più comuni sul servizio e le relative risposte.

Il manuale e le FAQ saranno integrati nel tempo in base ad eventuali altri argomenti che si riveleranno di interesse comune

3 Descrizione del Servizio

Questo capitolo descrive l'architettura del servizio *"Container as a Service Open Source"*. Il servizio è basato sullo stack tecnologico che eroga i servizi IaaS, con l'integrazione di ulteriori due tecnologie:

- **Openshift Container Platform**
- **VMware vSphere CSI Driver Operator**

Il servizio CaaS Open Source è disponibile nel modello di servizio CaaS Shared con Worker Node dedicati ad un singolo cliente

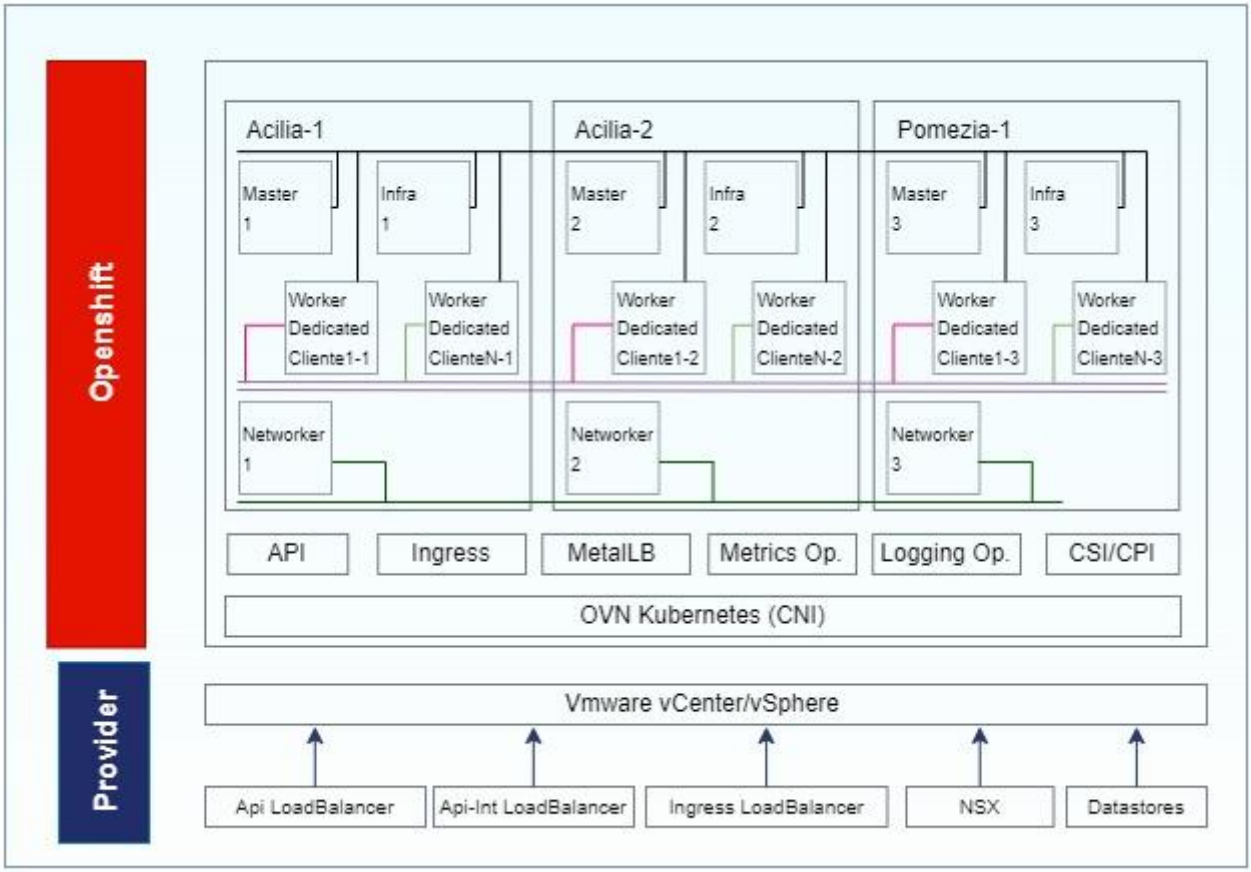


Figura 1. Schema del Servizio CaaS

4 Introduzione al servizio

In questa sezione sono riportate alcune definizioni dei costrutti logici utilizzati all'interno del servizio CaaS Open Source. Tali principi consentiranno una migliore comprensione dei successivi capitoli del manuale che si focalizzeranno su alcune aree 'verticali' di gestione del servizio.

1. Qual è lo scopo del servizio offerto nel CaaS Open?

Red Hat OpenShift è una piattaforma applicativa di cloud ibrido basata su tecnologia Kubernetes, leader del settore.
Unisce servizi testati e affidabili per ridurre gli attriti tra sviluppo, rinnovamento, distribuzione, esecuzione e gestione delle applicazioni.

Inoltre, OpenShift garantisce un'esperienza coerente tra cloud pubblico, on-premise, cloud ibrido o architettura edge.

2. Chi può utilizzare il servizio CaaS

Tutte le PA che decidono di sottoscrivere il Servizio CaaS Open mediante il Piano dei Fabbisogni.

3. Come accedo al Servizio?

Il servizio è usufruibile sia mediante l'accesso alla Dashboard messa a disposizione dal prodotto Openshift che direttamente dall'interno della Console (Unica) PSN, oltre che tramite API Kubernetes Standard, grazie alle quali le PA, che hanno sottoscritto la soluzione CaaS Open Source, facendone esplicita richiesta nel piano dei Fabbisogni, potranno deployare direttamente le proprie applicazioni containerizzate.
Dopo avere effettuato l'accesso alla Dashboard Openshift, ogni cliente avrà la visibilità del proprio Project:

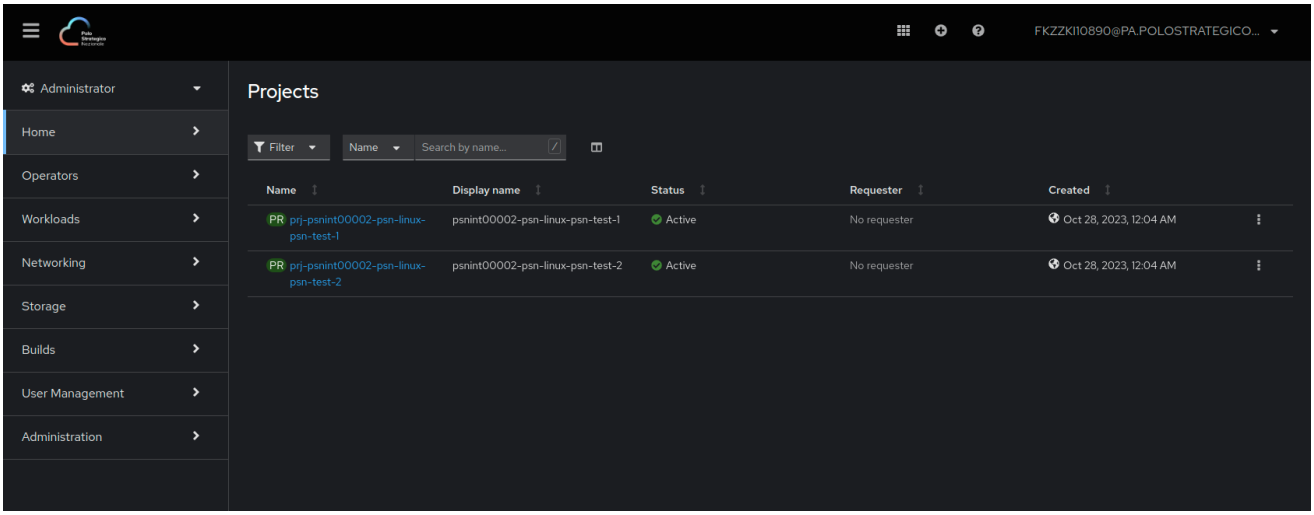


Figura 2.Dashboard Openshift

NOTA:

Ricordiamo che la Dashboard Caas Open è raggiungibile attraverso la seguente URL:

https://console-openshift-console.apps.pvc-os-caas01-rs.polostrategiconazionale.it

O direttamente dalla Console Unica PSN :

https://console.polostrategiconazionale.it

4. Quali Ruoli si possono assegnare alle PA che attivano il Servizio CaaS?

All'attivazione del Servizio CaaS Open, verrà associata l'utenza del Referente Tecnico della PA come utenza amministrativa di tipo "***Admin***" dei Project richiesti; successivamente sarà possibile definire utenze secondarie ed associarle sia ad un ruolo di tipo "***Admin***", che di tipo "***Viewer***".
L' utenza "***Admin***" avrà i permessi di "**edit**" sul proprio Project, e quindi la possibilità di istanziare containers, esporre servizi, allocare Persistent Volume (PV), ecc., nei limiti imposti al fine di garantire la corretta MultiTenancy e garantire la corretta continuità di servizio.

L'utenza "***Viever***" avrà invece permessi molto più limitati, con la possibilità di accedere al Project in modalità di sola lettura.

La piattaforma OCP utilizza un modello di accesso basato su ruoli (***RBAC***) che permette la separazione dei ruoli tra gli amministratori IT del PSN e i referenti tecnici della PA. Il modello di accesso garantisce privilegi commisurati alla funzione svolta e consente pertanto di implementare il modello del minimo privilegio necessario.

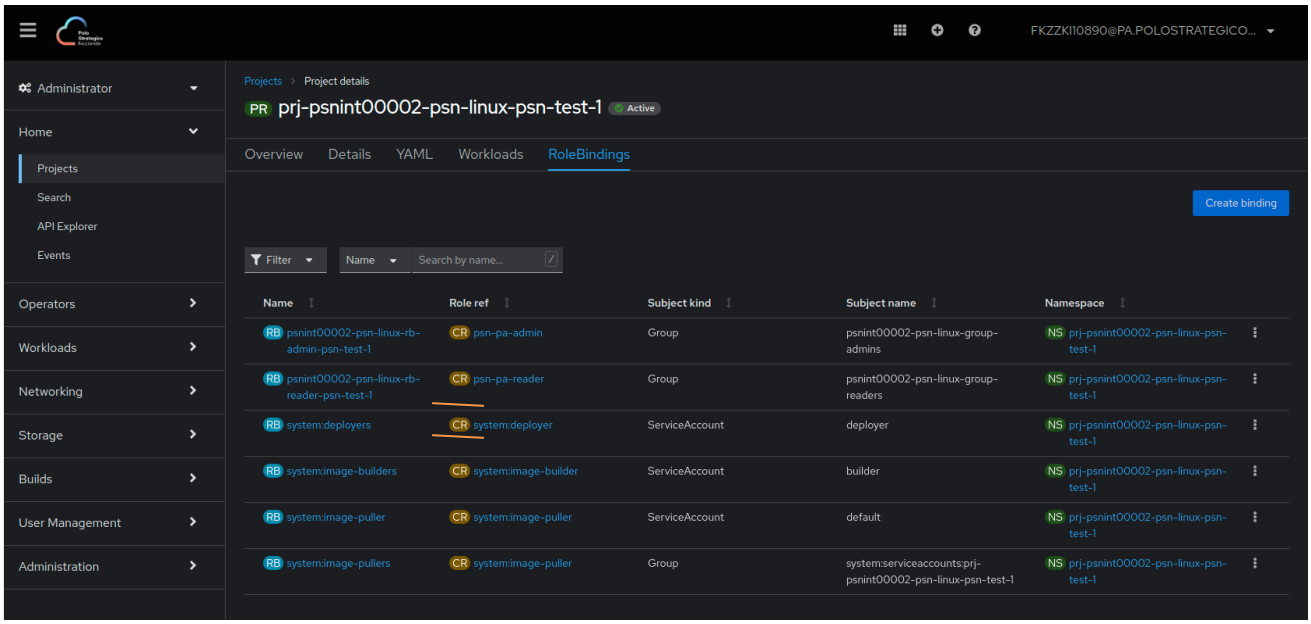


Figura 3.Vista Utenze

5. Quali versioni sono Disponibili?

Il servizio CaaS Opens sarà basato sul software RedHat Openshift Container Platform, nella versione **4.12** LTS.

Tale versione è basata sulle seguenti componenti tecnologiche:

- Kubernetes 1.25
- CRI-o 1.25
- OVN-Kubernetes network plugin

Verranno introdotte nuove versioni, coerentemente con le evoluzioni del prodotto CaaS Open/Openshift.

6. Quali opzioni sono disponibili per le istanze CaaS Open?

Le istanze CaaS Open saranno basate su un'unità minima di 4 vCPU, 16Gb di RAM per ogni worker node, nella quantità minima di 3 worker (1 per AZ) e 50GB di storage persistente cross tenant.
Sarà possibile acquistare più istanze, in relazione alle proporzioni indicate (quindi sempre pool minimo di 3 worker e multipli, mantenendo le proporzioni anche in termini di risorse computazionali) da destinare sullo stesso Project o su Project differenti (farà sempre fede il Piano dei Fabbisogni),

Nota: qualora ci siano più istanze distribuite su Project differenti, nel Piano dei Fabbisogni andranno indicate anche eventuali interconnessioni tra i Project.

7. Dopo aver acquistato un'istanza CaaS, la PA può modificarla?

Si, a seguito del rilascio di un Project con delle quote prestabilite, la PA, tramite flusso di variazione, può richiedere la modifica delle risorse riservate, incrementandole o diminuendole.

8. Quante istanze CaaS Open posso istanziare?

Non ci sono limiti, dipende dalla disponibilità della piattaforma, al netto di possibili espansioni hardware future.

9. Esistono dei prerequisiti per accedere e gestire un'istanza CaaS Open?

Si, per poter accedere e gestire un'istanza CaaS è necessario installare sul proprio client i seguenti software:

- "Openshift CLI" (oc client)
- "Podman" oppure "Docker Client", o in alternativa qualsiasi software in grado di gestire i Container registry che implementano la specifica OCI Registry v1
- "Browser Web"

Nota bene: molti software vedasi Podman e Docker su Windows potrebbero richiedere l'installazione del componente aggiuntivo Windows Subsystem for Linux (WSL)

Come installare "Openshift CLI "?

Di seguito viene descritta la procedura di installazione e inizializzazione di "**Openshift CLI**".

Appena entrati sulla GUI, cliccando sul "**?**" sarà possibile scaricare direttamente il client "**oc**" dalla sezione "**Command Line tools**" , sia per sistemi operativi Windows che per sistemi operativi Linux.

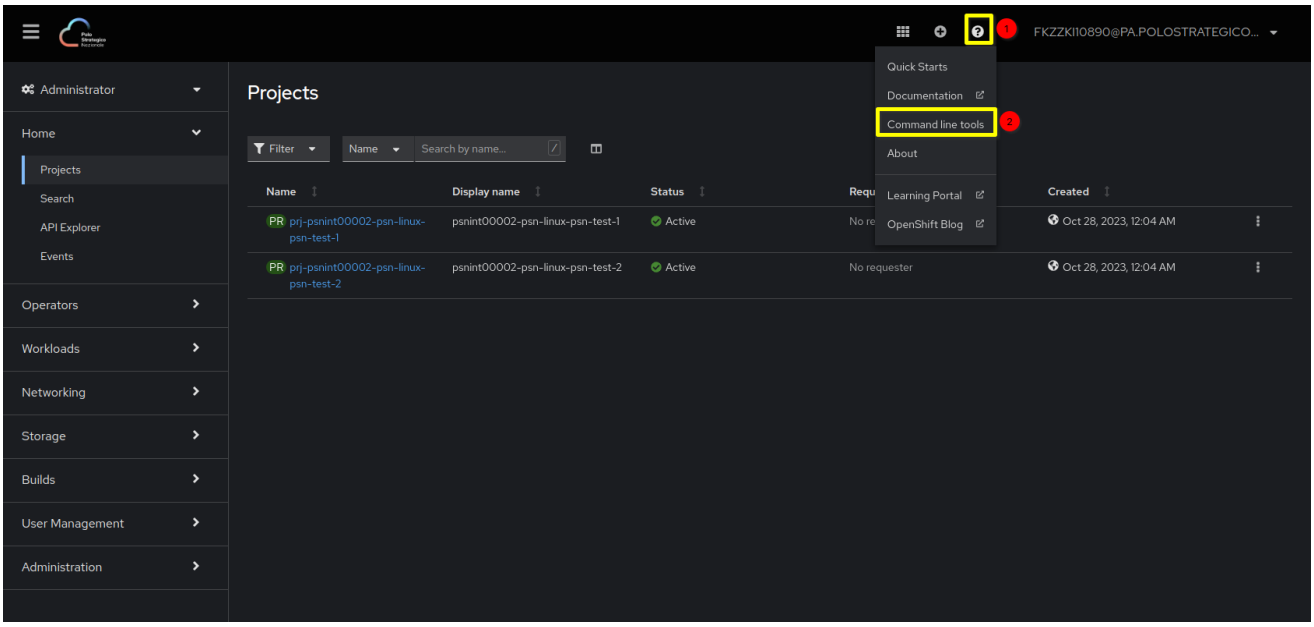


Figura 4.Link OC Client da GUI

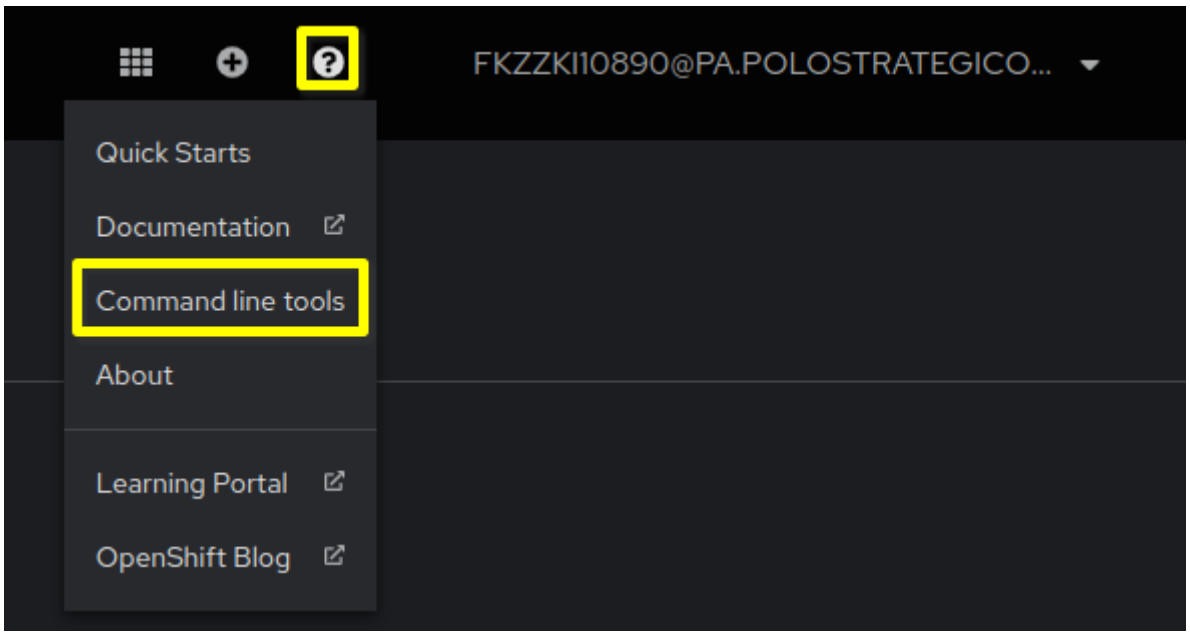


Figura 5. Selezione "Command line tools"

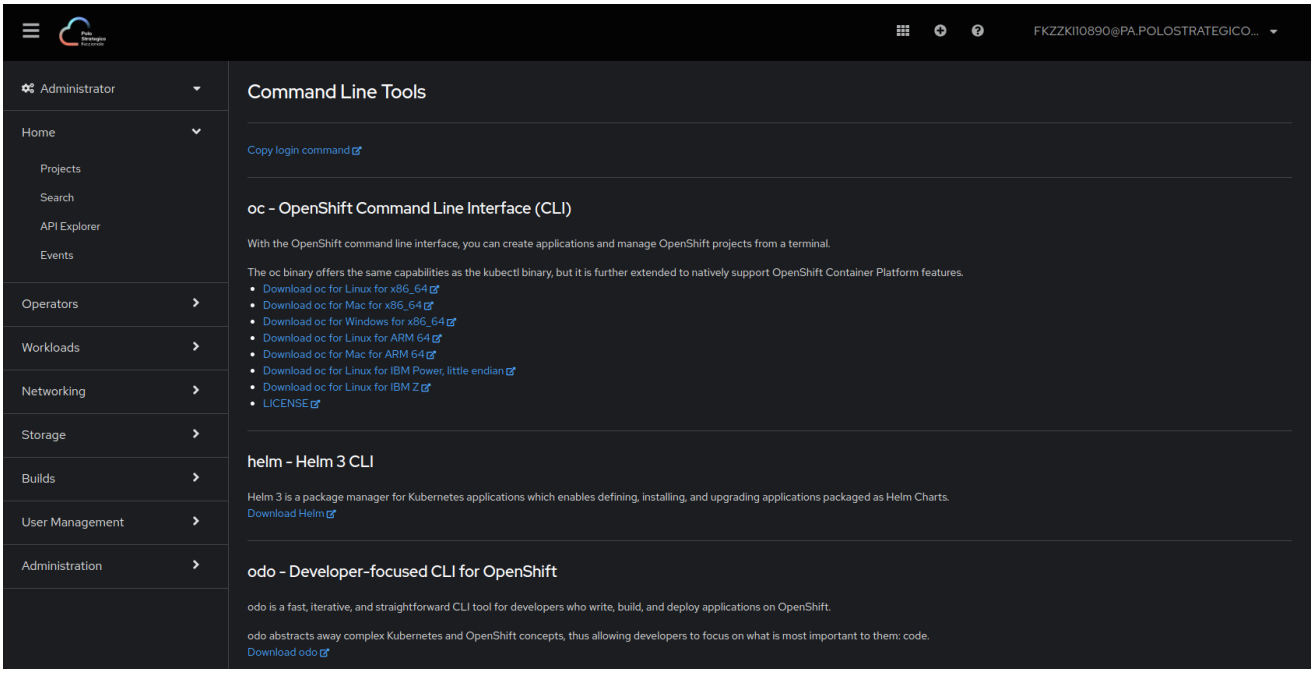


Figura 6.Download OC Client

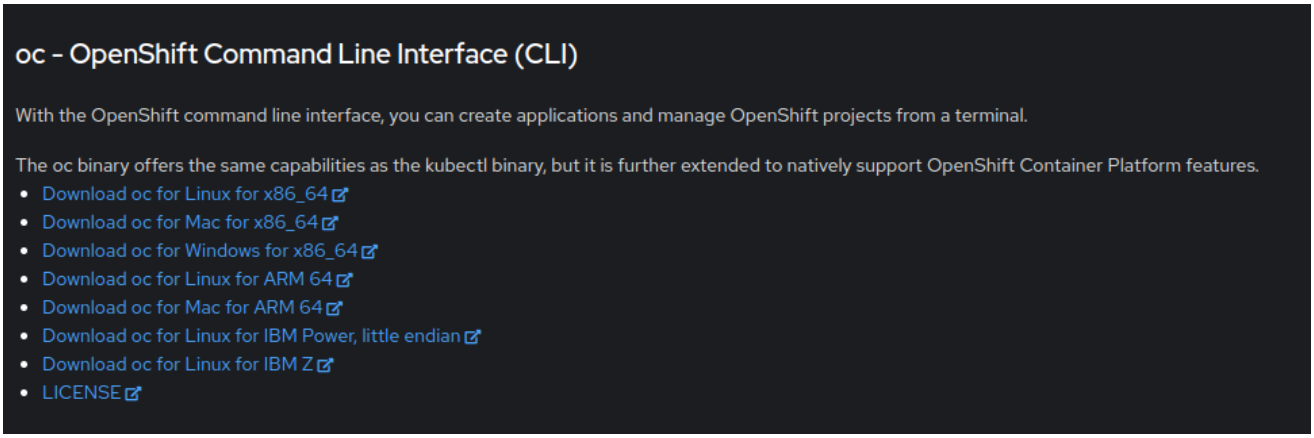


Figura 7. Lista client OC disponibili

I file di installazione saranno inoltre disponibili tramite i seguente link:

- "https://downloads-openshift-console.apps.pvc-os-caas01-rs.polostrategiconazionale.it/amd64/windows/oc.zip"
  o
- "https://downloads-openshift-console.apps.pvc-os-caas01-rs.polostrategiconazionale.it/amd64/linux/oc.tar"

Nel caso di sistemi Windows la procedura di installazione è la seguente:

- Scaricare e scompattare la CLI di Openshift sul proprio client
- Creare una nuova cartella "*Program Files\oc_client*"
- Copiare il file "*oc.exe*" nella Folder appena creata

- Fare clic con il tasto destro del mouse sulla cartella "*oc_client*", selezionare "*Properties> Security*" e assicurarsi che l'account che si sta utilizzando abbia l'autorizzazione di "*Full Control*"

Inoltre, per semplicità si suggerisce di settare il *Path di exe di oc* come variabile di ambiente:

- Utilizzare "*Windows Search*" per cercare "*env*"
- Selezionare "*Edit the System Environment Variables*" e cliccare su " *Environment Variables*"
- Selezionare la riga "*Path*" in Variabili di sistema e fare clic su "Edit"
- Fare clic su "*New*"[Path di exe di oc] per aggiungere una nuova riga e inserire il percorso della CLI di oc
- Aprire un nuovo terminale in "*command line*" e verificare la versione del client appena installato: "*oc version*"

Nel caso di sistemi Linux la procedura di installazione è invece la seguente:

- Scaricare e scompattare la CLI di Openshift sul proprio client
- Eseguire il comando "tar xvf oc.tar.gz"
- Per comodità, è consigliabile copiare l'eseguibile "oc" direttamente all'interno della directory */usr/local/bin* , così da avere il comando sempre disponibile da qualsiasi path.

10. Come accedo al CaaS Open Source?

Una volta richiesta un'istanza CaaS verranno fornite delle credenziali. Con queste credenziali sarà possibile accedere tramite GUI e/o tramite api (client OC).

Accesso via GUI

È sufficiente collegarsi all'indirizzo

https://console-openshift-console.apps.pvc-os-caas01-rs.polostrategiconazionale.it/
Si verrà ridirezionati immediatamente sulla login page

Figura 8.Login via GUI

Selezionare l'opzione "IAM" ed accedere alla GUI.

Accesso via CLI

Di seguito viene descritta la modalità di accesso via CLI:

```
$ oc login --server=https://api.pvc-os-caas01-rs.polostrategiconazionale.it


You must obtain an API token by visiting https://oauth-openshift.apps.pvc-os-caas01-
rs.polostrategiconazionale.it/oauth/token/request
```

Invocato il comando ci chiederà di continuare l'autenticazione attraverso un Browser Web, dove andrà selezionata la modalità IAM (come descritto nel paragrafo precedente).

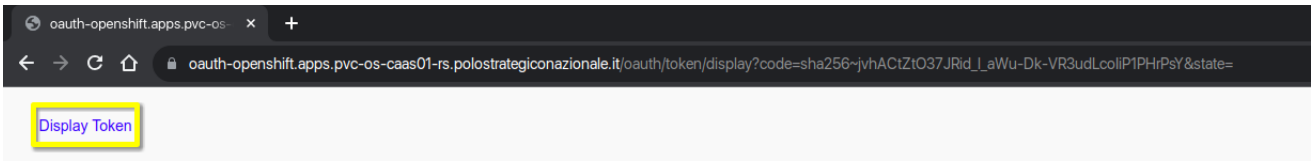


Figura 9. Display Token

Figura 10.Token per l'accesso alla piattaforma

Una volta autenticati, verrà mostrato un comando da lanciare sulla CLI per autorizzare l'accesso (di seguito un esempio).

```
oc login --token=sha256~2iKRWX8yw0Eel1Shc0wND3HhVDdVuySXNQn3O64HAko --server=https://api.pvc-os-caas01-rs.polostrategiconazionale.it
```

Tale procedura provvede anche a generare il file kubeconfig (su Linux e Mac $HOME/.kube/config) compatibile con kubectl e tutti i programmi per l'interazione con i cluster kubernetes (vedi Ansible, Helm, Terraform) fino alla scadenza del token; una volta scaduto sarà necessario ripetere la procedura di login appena descritta.

Accesso via API

Oltre alle modalità descritte nei paragrafi precedenti, è possibile interfacciarsi all'infrastruttura CaaS Open mediante chiamate API.

Dopo aver ottenuto il token di autenticazione utilizzando le indicazioni del paragrafo precedente, è possibile interagire con gli oggetti ospitati all'interno di Openshift attraverso chiamate REST.

```
GET /api/v1/namespaces/myproject/pods?limit=500 HTTP/2
Host: api.pvc-os-caas01-rs.polostrategiconazionale.it:6443
user-agent:
accept: application/json
Authorization: Bearer sha256~UaiIBC8kUKoH1Lk-KAxcpKs4VluG_Hahl8PymaKxgVs
```

Figura 11. Esempio di Richiesta tramite Token


```
{
  "kind":"PodList",
  "apiVersion":"v1",
  "metadata":{
    "resourceVersion":"1449667469"
  },
  "items":[
    {
      "metadata":{
        "name":"my-pod-completed-pods-28173600-mqcm6",
        "generateName":"my-pod-completed-pods-28173600-",
        "namespace":"myproject",
        "uid":"42cb50ee-70d3-453c-93ea-5750fec9884e",
        "resourceVersion":"1449320601",
        "creationTimestamp":"2023-07-27T00:00:00Z",
        "labels":{
          "controller-uid":"13c0d516-b976-4568-ac8b-4bb415b80603",
          "job-name":"my-pod-completed-pods-28173600"
        },
        "spec":{
          "volumes":[
            {
              "name":"kube-api-access-jg4cb",
              "projected":{
                "sources":[
                  {
                    "serviceAccountToken":{
                      "expirationSeconds":3607,
                      "path":"token"
```

```
            }
          },
          {
            "configMap":{
              "name":"kube-root-ca.crt",
              "items":[
                {
                  "key":"ca.crt",
                  "path":"ca.crt"
                }
              ]
            }
          },
          {
            "downwardAPI":{
              "items":[
                {
                  "path":"namespace",
                  "fieldRef":{
                    "apiVersion":"v1",
                    "fieldPath":"metadata.namespace"
                  }
                }
              ]
            }
          },
          {
            "configMap":{
              "name":"openshift-service-ca.crt",
              "items":[
                {
                  "key":"service-ca.crt",
                  "path":"service-ca.crt"
                }
              ]
            }
          }
        ],
        "defaultMode":420
      }
    }
  ],
  "containers":[
    {
```

```
            "name":"my-pod-completed-pods",
            "image":"quay.io/openshift/origin-cli:4.4",
            "command":[
              "/bin/bash",
              "-c",
              "oc get pod -A"
            ],
            "resources":{

            },
            "volumeMounts":[
              {
                "name":"kube-api-access-jg4cb",
                "readOnly":true,
                "mountPath":"/var/run/secrets/kubernetes.io/serviceaccount"
              }
            ],
            "terminationMessagePath":"/dev/termination-log",
            "terminationMessagePolicy":"File",
            "imagePullPolicy":"IfNotPresent"
          }
        ],
        "restartPolicy":"Never",
        "terminationGracePeriodSeconds":30,
        "dnsPolicy":"ClusterFirst",
        "serviceAccountName":"pods",
        "serviceAccount":"pods",
        "nodeName":"node02.caas.polostrategiconazionale.it",
        "securityContext":{

        }
      },
      "priority":0
    }
  }
 ],
}
```


Figura 12.Esempio di Risposta tramite Token

L'accesso alla piattaforma attraverso le API consente inoltre l'utilizzo di strumenti di automazione e l'integrazione con strumenti di CI/CD (in completa gestione del cliente).

```
provider "kubernetes" {
  host = "https://api.pvc-os-caas01-rs.polostrategiconazionale.it"
  token = "sha256~UaiIBC8kUKoH1Lk-KAxcpKs4VluG_Hahl8PymaKxgVs"

}
```

Figura 13. Esempio di configurazione con Provider Terraform

Come carico le immagini all'interno del registry interno

Per poter caricare e gestire le immagini container all'interno dei singoli Project è necessario disporre di un software capace di interagire con la specifica OCI Registry v1, strettamente legata alla specifica Docker Registry API v2.

Si suggerisce l'uso dei seguenti client che sono stati testati sulla piattaforma

- Docker
- Podman
- Skopeo

Una volta caricata l'immagine all'interno del Project verrà creato l'oggetto "Image"

Come installare Podman e Docker

Sia "**Podman**" che "**Docker**" richiedono che la funzionalità WSL sia installata sui client Windows.

Per l'installazione e la configurazione di Docker è possibile fare riferimento alla documentazione ufficiale del prodotto, consultabile all'indirizzo https://docs.docker.com/engine/install/

Per quanto riguarda Podman, il file di installazione per i sistemi Windows è disponibile al seguente link:

- https://github.com/containers/podman/releases/download/v4.8.0/podman-4.8.0-setup.exe

Mentre per tutti gli altri sistemi operativi è possibile fare riferimento alla documentazione ufficiale del prodotto, consultabile all'indirizzo https://podman.io/docs/installation

Con l'installazione di podman, su tutte le piattaforme viene installato anche il client skopeo.

Come utilizzare Podman e Docker

1. Effettuare il login alla piattaforma mediante client OC (vedasi capitolo Accesso via CLI)
2. Mostrare a video il token rilasciato dalla piattaforma per l'utente

```
$ oc whoami
FKZZKI10890@PA.POLOSTRATEGICONAZIONALE.IT
$ oc whoamit -t
sha256~n3kbM5nmVzluhkw7OBzOnDHbrigao4FM22xcGucLGn
```

Questo ci fornirà il token temporaneo d'accesso per effettuare il caricamento delle immagini Container all'interno della piattaforma

L'indirizzo del registry è il seguente https://registry.apps.pvc-os-caas01-rs.polostrategiconazionale.it

I due client possono essere trattati insieme in quanto condividono la medesima sintassi

1. Login sul registry interno

```
$ docker login -u $(oc whoami) -p $(oc whoami -t) registry.apps.pvc-os-caas01-rs.polostrategiconazionale.it
…
Login Succeeded!
```

oppure

```
podman login -u $(oc whoami) -p $(oc whoami -t) registry.apps.pvc-os-caas01-rs.polostrategiconazionale.it
Login Succeeded!
```

2. Pull di un'immagine di demo

```
$ docker pull registry.access.redhat.com/ubi9/ubi-minimal:9.3-1361.1699548032
```

```
9.3-1361.1699548032: Pulling from ubi9/ubi-minimal
a032f50e22ae: Pull complete
Digest: sha256:c77792b8084ce5946c68f39024fa460ef7769c0eef3fce995e70299e21a7e166
Status: Downloaded newer image for registry.access.redhat.com/ubi9/ubi-minimal:9.3-1361.1699548032
registry.access.redhat.com/ubi9/ubi-minimal:9.3-1361.1699548032
```

oppure con podman

```
$ podman pull registry.access.redhat.com/ubi9/ubi-minimal:9.3-1361.1699548032
Trying to pull registry.access.redhat.com/ubi9/ubi-minimal:9.3-1361.1699548032...
Getting image source signatures
Checking if image destination supports signatures
Copying blob a032f50e22ae done   |
Copying config 4d9a019b47 done   |
Writing manifest to image destination
Storing signatures
4d9a019b47a0d1da42b109f80186ba866d3048d47f7ab0a8c1ba40b27f5017cb
```

3. Identificare il Project sul quale caricare l'immagine

```
$ oc project -q
prj-psnint00002-psn-linux-psn-test-1
```

4. Inserire un tag tra l'immagine sorgente e la destinazione, il formato di destinazione deve essere il seguente

   registry.apps.pvc-os-caas01-rs.polostrategiconazionale.it/$Project /$IMAGE:$TAG

   il che si trasformerà nel nostro caso in

   registry.apps.pvc-os-caas01-rs.polostrategiconazionale.it/prj-psnint00002-psn-linux-psn-test-1/ubi-minimal:9.3-1361.1699548032

   con Docker

```
$ docker tag registry.access.redhat.com/ubi9/ubi-minimal:9.3-1361.1699548032 registry.apps.pvc-os-caas01-rs.polostrategiconazionale.it/prj-psnint00002-psn-linux-psn-test-1/ubi-minimal:9.3-1361.1699548032
```

con Podman

```
$ podman tag registry.access.redhat.com/ubi9/ubi-minimal:9.3-1361.1699548032 registry.apps.pvc-os-caas01-rs.polostrategiconazionale.it/prj-psnint00002-psn-linux-psn-test-1/ubi-minimal:9.3-1361.1699548032
```

5. Effettuiamo il push.
   ATTENZIONE: il comando qui tra i due è leggermente differente e per podman prevede un parametro in più

con Docker

```
$ docker push registry.apps.pvc-os-caas01-rs.polostrategiconazionale.it/prj-psnint00002-psn-linux-psn-test-1/ubi-minimal:9.3-1361.1699548032
```

con Podman

```
$ podman push registry.apps.pvc-os-caas01-rs.polostrategiconazionale.it/prj-psnint00002-psn-linux-psn-test-1/ubi-minimal:9.3-1361.1699548032 --remove-signatures
```

Skopeo

Client Skopeo ha il compito di effettuare l'introspezione e la copia di immagini container sia da locale a remoto che remoto su remoto, nel caso in esame vedremo la copia di un immagine da DockerHub verso il Registry interno senza necessità di pull :

1. Login sul registry interno

```
$ skopeo login -u $(oc whoami) -p $(oc whoami -t) registry.apps.pvc-os-caas01-rs.polostrategiconazionale.it
...
Login Succeeded!
```

Info: se in precedenza si è fatto il login con podman condividono la stessa autenticazione

2. Effettuare la copia dell'immagine Jenkins su registry

```
$ skopeo copy docker://docker.io/jenkins/jenkins docker://registry.apps.pvc-os-caas01-rs.polostrategiconazionale.it/prj-psnint00002-psn-linux-psn-test-1/jenkins:lts
Getting image source signatures
Copying blob 56c1327728ce done
Copying blob 28d2d8d6013d done
Copying blob 1b3412a767b4 done
Copying blob e2aae4e0e12e done
Copying blob 45d9289576a1 done
Copying blob 90e5e7d8b87a done
Copying blob d729df913513 done
Copying blob 0e5fdac6b409 done
Copying blob 90fec0e0380d done
Copying blob 2b8cb86329d8 done
Copying blob a4392fb54cbc done
Copying blob 212cb98682aa done
Copying config a8f4c4533d done
Writing manifest to image destination
```

3. Verifichiamo la creazione dell'Image Stream relativo a Jenkins all'interno del Project

```
$ oc get imagestream
NAME      IMAGE REPOSITORY                                                                TAGS    UPDATED
httpd-24  registry.apps.pvc-os-caas01-rs.polostrategiconazionale.it/prj-psnint00002-psn-linux-psn-test-1/httpd-24  1-268  6 days ago
jenkins   registry.apps.pvc-os-caas01-rs.polostrategiconazionale.it/prj-psnint00002-psn-linux-psn-test-1/jenkins   lts    12 minutes ago
```

11. Che aspetti network posso gestire?

È possibile esporre i microservizi all'esterno della rete del cluster tramite "**route**" ed inoltre è possibile esporre un service tramite ExternalIP.

12. Ci sono componenti Open Source già disponibili all'interno del project?

Si, all'interno di ogni istanza CaaS Open sono disponibili dei tool source di default e pronte all'uso, così come riportato nella tabella sottostante:

Pacchetto	Descrizione dei pacchetti
Coredns	Fornisce il servizio DNS (k8s)
vcd-csi	Fornisce la Cloud Storage Interface (CSI) di VMware Cloud Provider.
Prometheus	Time series DB utilizzato per raccogliere le metriche dalle applicazioni cloud native
Grafana	Applicazione che legge le informazioni da Prometheus e permette di creare e visualizzare dashboard.
Registry	Registry immagini integrato

Tabella 4. Componenti aggiuntive installate di default

5 Come effettuare il deploy di un'applicazione

Per il deploy di un'applicazione è possibile utilizzare sia gli strumenti messi a disposizione direttamente dalla GUI che il client OC (soprattutto per i casi di applicazioni meno standard o con opzioni più specifiche).

1. Deploy applicazioni da GUI

Per il deploy di un'applicazione direttamente dalla GUI, posizionarsi nella sezione "Home", selezionare il project di riferimento, spostarsi nella sottosezione "Workload", cliccare quindi su "Deployment" a seconda dell'oggetto da cui si vuole partire.

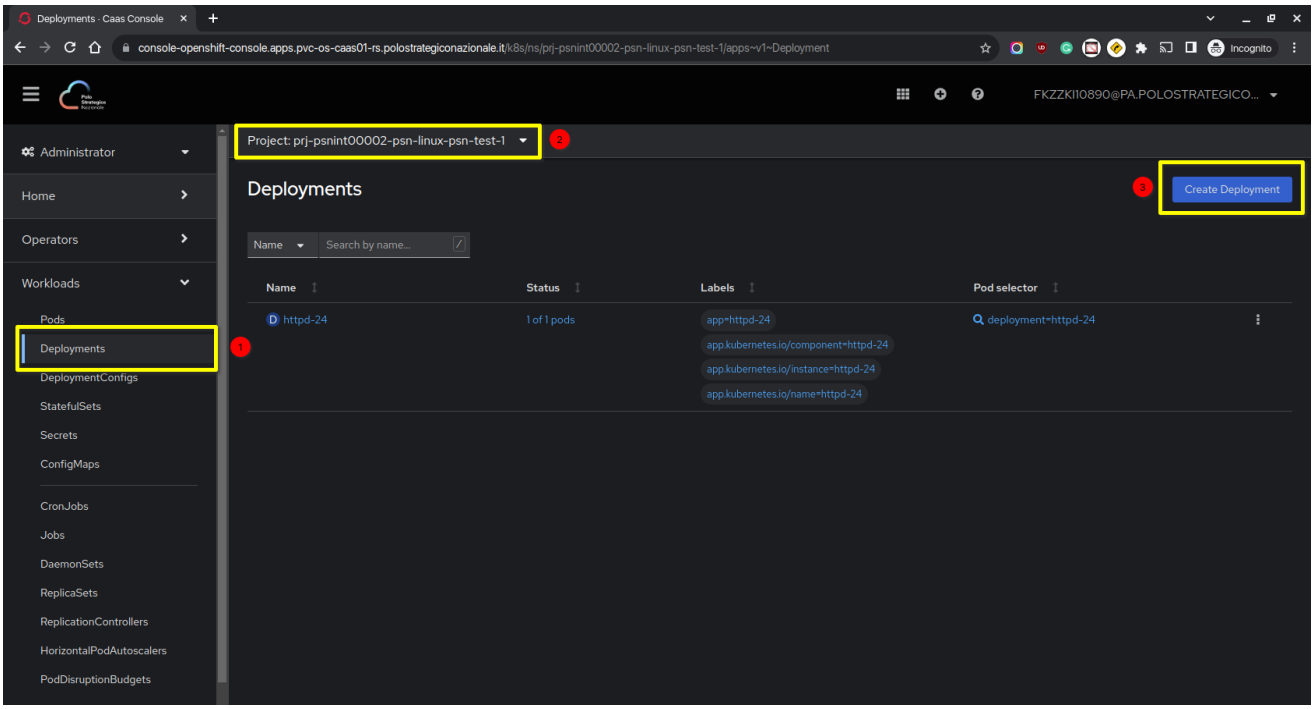


Figura 6. Selezione Project

4. Una volta selezionato "**Workloads**" > "**Deployments"**
5. Selezionare l'opportuno project dal menù modale posto al centro della videata
6. Avviamo la creazione del Deployment effettuando click sul bottone "Create Deployment"

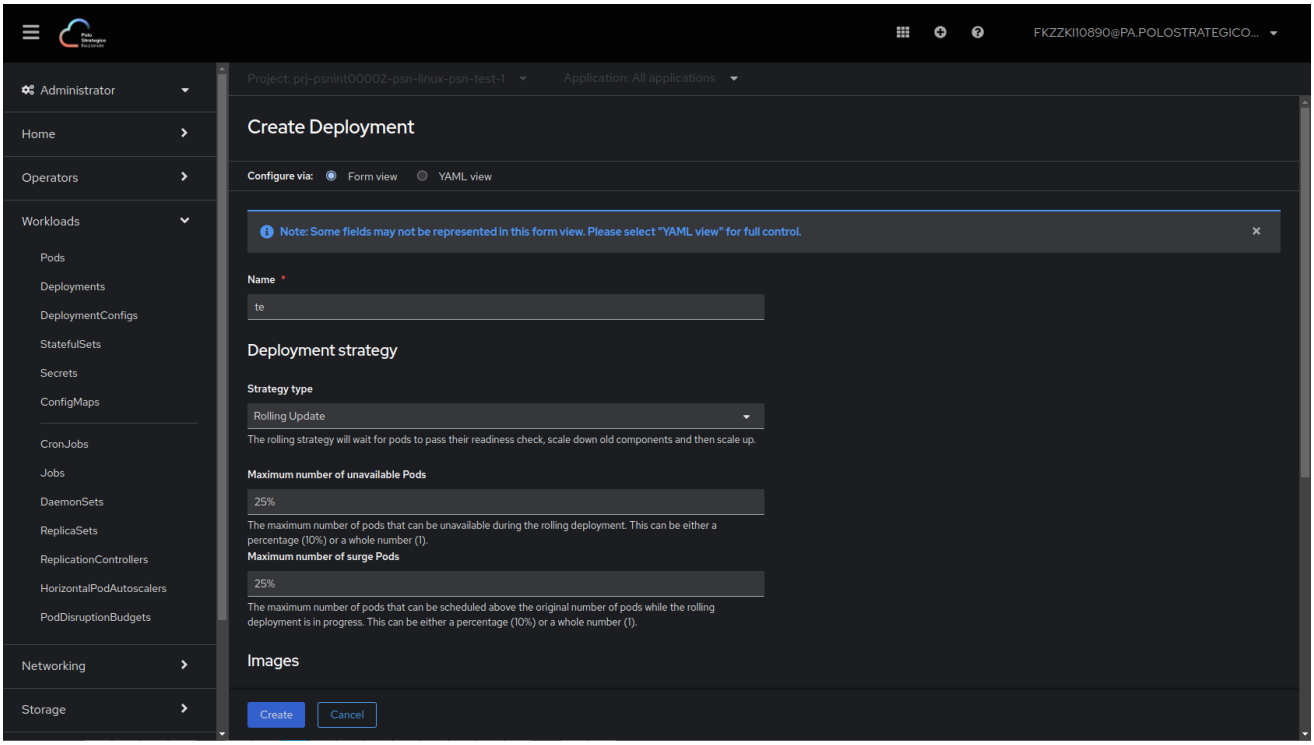


Figura 14. Creazione Deployment

Ci verrà mostrata una lunga videata con diverse opzioni che vanno dal nome, dall'immagine container da utilizzare alla modalità di aggiornamento della distribuzione software.

Nel dettaglio:

Strategie di Deployment

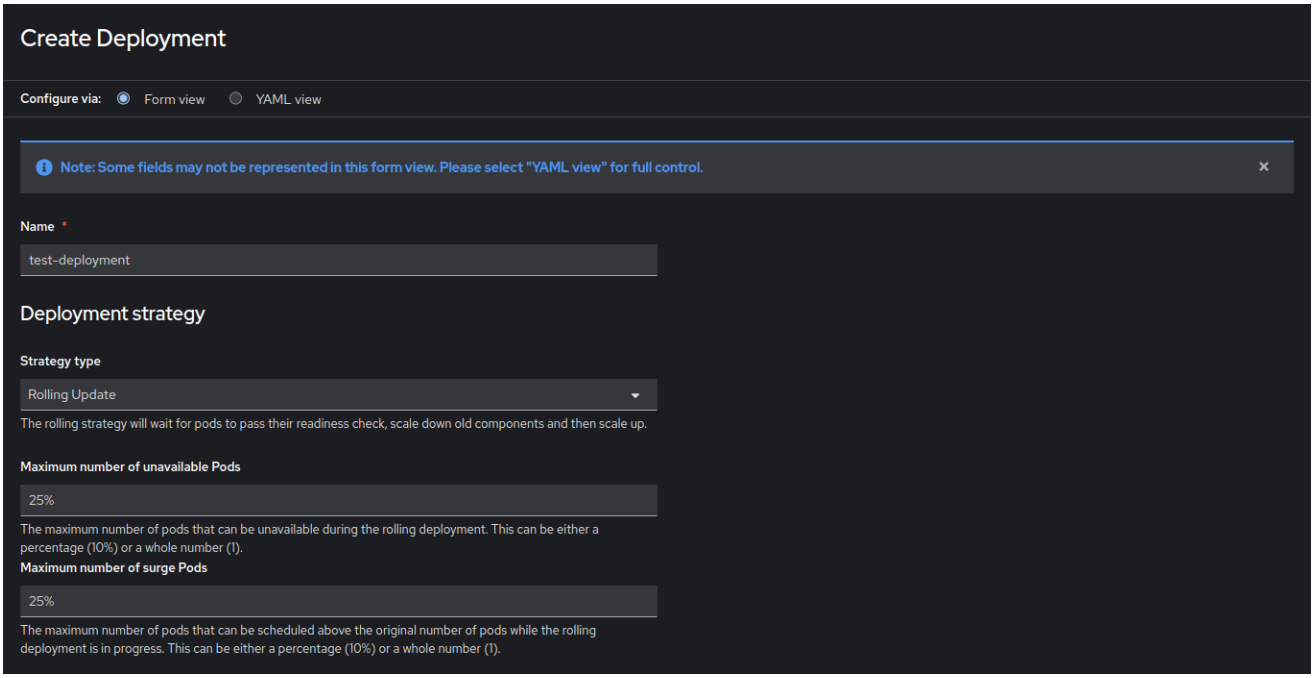


Figura 15.Creazione Deployment - Nome Deploy

Nella prima sezione oltre al nome che assumerà la deployment, ci verrà chiesta in che modalità verrà aggiornata e ci propone due strategie

- **Rolling Update:** Effettua un aggiornamento con sostituzione graduale dei pod man mano diventino operativi i nuovi
- **Recretate**: Termina tutti i pod appartenenti alla deployment e li fa' ripartire con la nuova configurazione

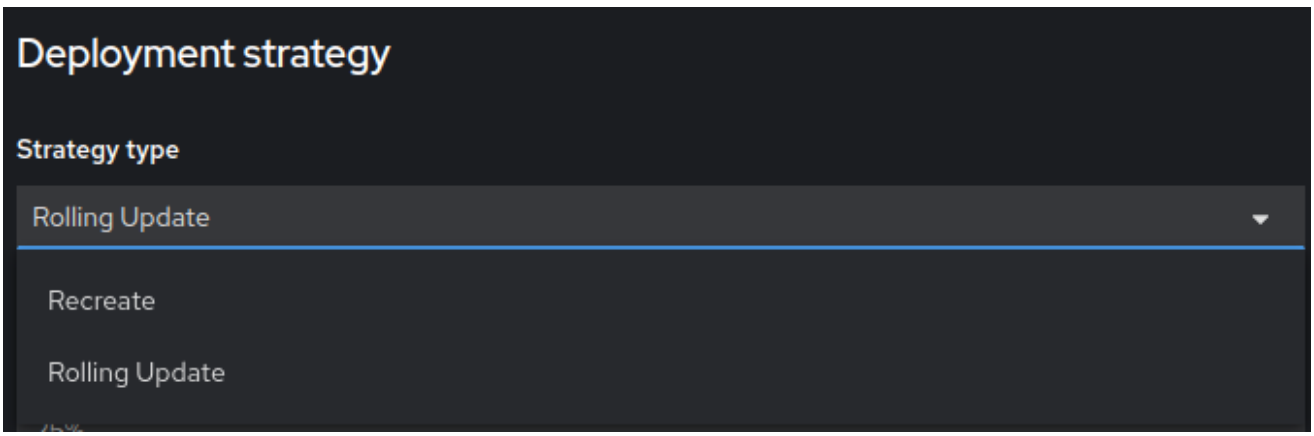


Figura 16.Creazione Deployment - Modalità di Rollout

Rolling Update

Il principale scopo del Rolling Update è limitare al massimo le interruzioni di servizio creando in anticipo un nuovo pod per ogni pod preesistente e nel momento quest'ultimo supera la fase dei Readiness Check il traffico viene immediatamente dirottato sul nuovo ed il vecchio terminato.

Per poter assolvere la sua funzione dovrà aumenterà temporaneamente la numerosità di Pod ed in caso di Deployment con una numerosità elevata di repliche è possibile definire una percentuale di Pod non disponibili al fine di velocizzare gli aggiornamenti senza impattare troppo sulle allocazione di CPU e Ram del Project

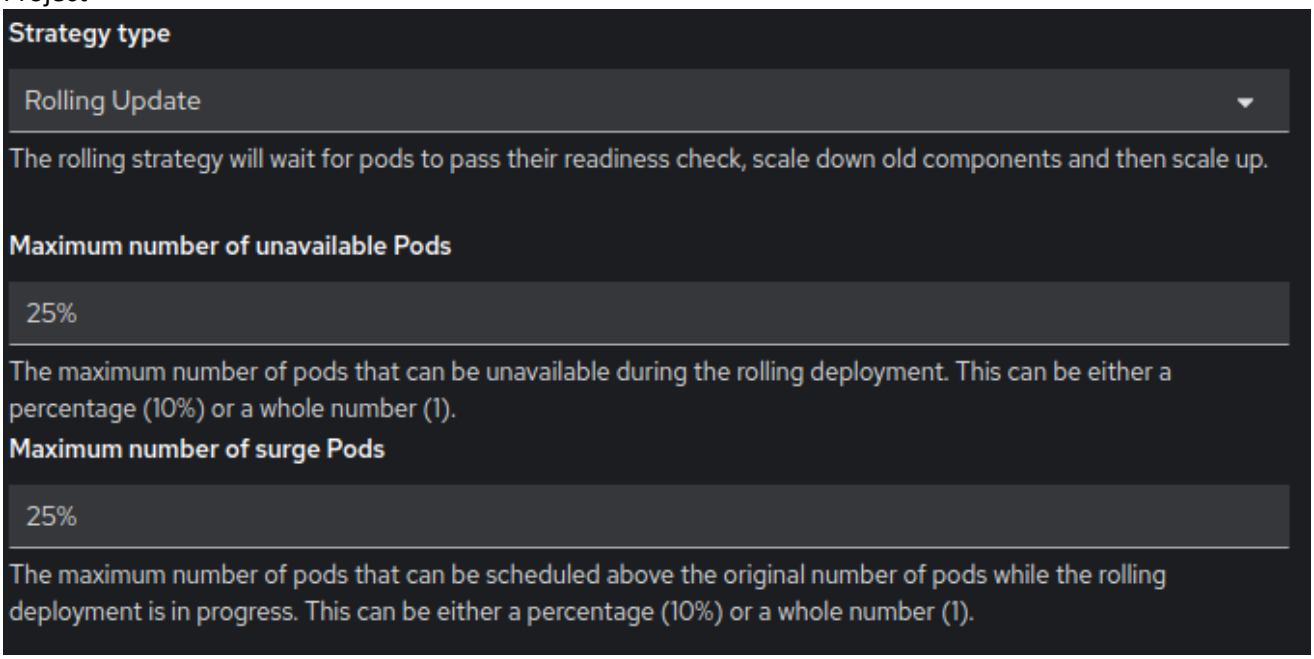


Figura 17.Creazione Deployment - Configurazione Rolling Update

Recreate

Come è facile intuire, tutti i pod vengono terminati e rischedulati all'unisono. Questa strategia a volte si rende necessaria nel caso di applicativi che non supportano la coesistenza con versioni precedenti oppure nell'applicazione delle metodologie Blue/Green (https://en.wikipedia.org/wiki/Blue%E2%80%93green_deployment)

Container Image

Nella sezione Image abbiamo la possibilità configurare il deploy per poter utilizzare un container specifico, esso può provenire da Registry esterni o interni.

Immagine proveniente da Registry Esterni

Non selezionando la spunta “Deploy image from an Image Stream Tag” di fatto dobbiamo specificare la URL dal quale il Container Runtime deve prelevare l’immagine del container.

Tale operazione di download viene effettuata sempre in anonimo a meno che non si faccia riferimento esplicito all’uso di una PullSecret che può essere creata anche contestualmente alla creazione della Deployment, effettuando il click su “Create new Secret”

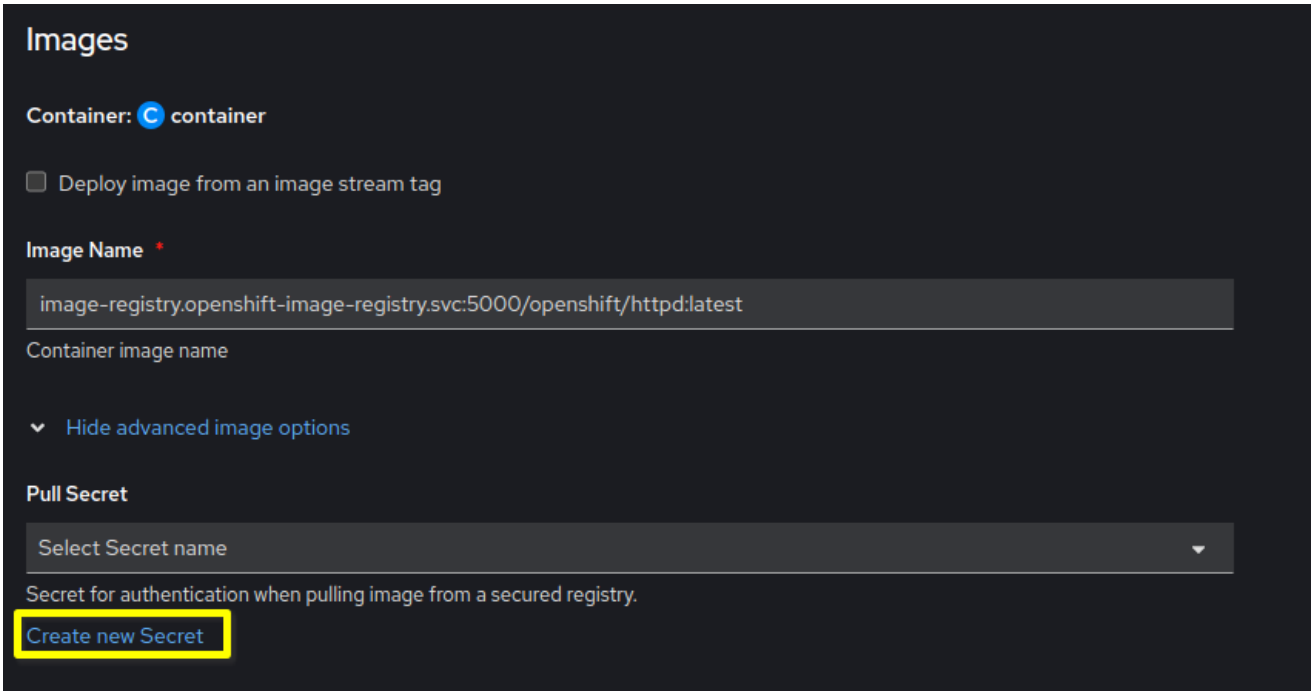


Figura 18. Creazione Deployment - Configurazione Immagine da registry esterni

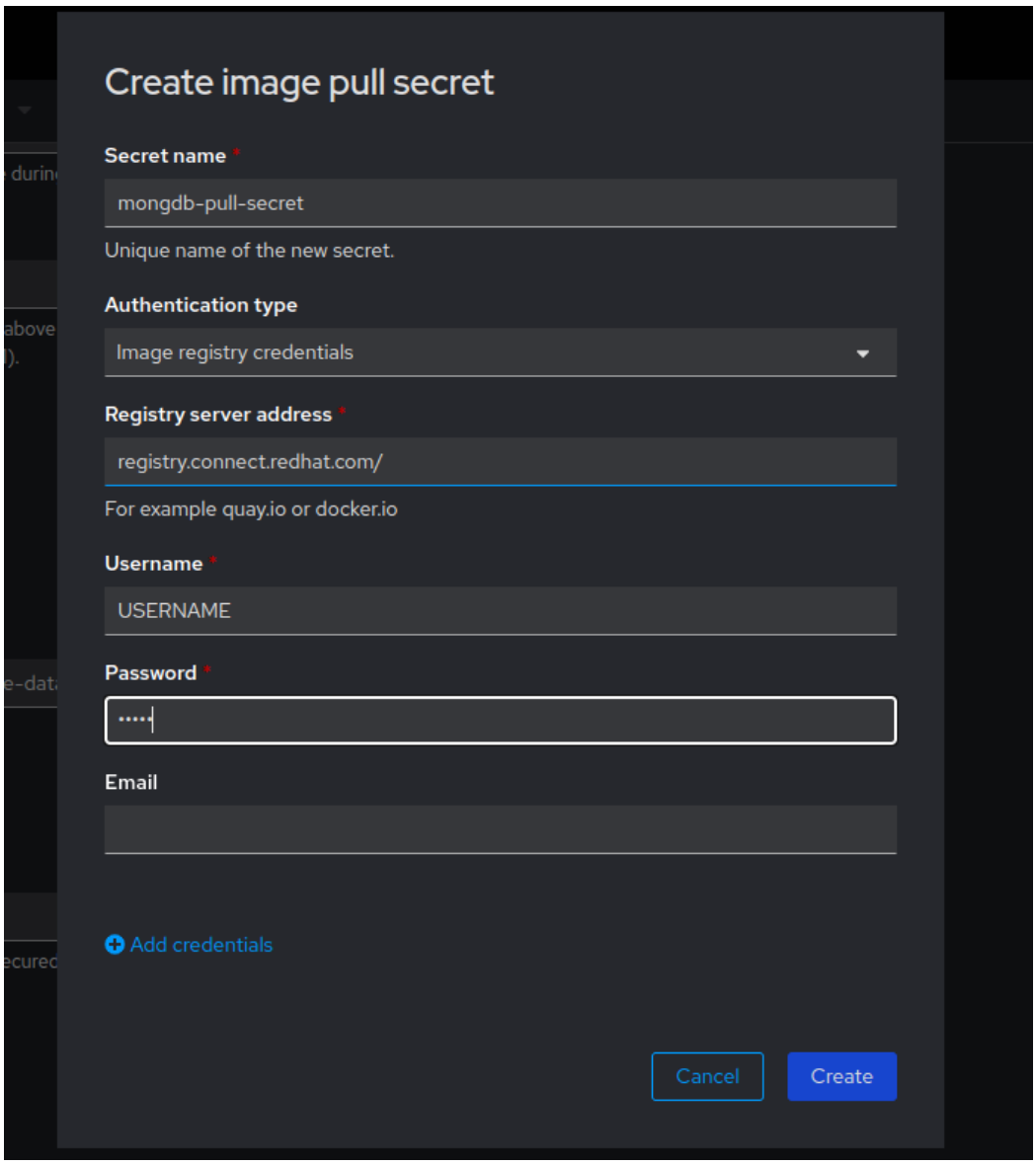


Figura 19. Creazione Deployment - Configurazione Pull Secret

Ci verranno chieste le credenziali per l'accesso al registry.

In alternativa possiamo scegliere di effetuare il deploy da un'immagine caricata sul registry interno

Immagine proveniente dal Registry Interno

Selezionando la spunta su "Deploy image from an image stream tag" possiamo referenziare un'immagine già presente all'interno del registry Openshift. È possibile specificare se avviare il rollout una volta effettuato il

push di una nuova versione del container con un determinato tag, selezionando l'opzione "Auto deploy when new Image is available".

Questa metodologia è particolarmente utile nel caso in cui il processo di deploy delle applicazioni è regolamentato da pipeline e vogliamo avere un tag di export (es. deploy);sarà quindi sufficiente taggare l'immagine che vogliamo distribuire con il tag scelto.

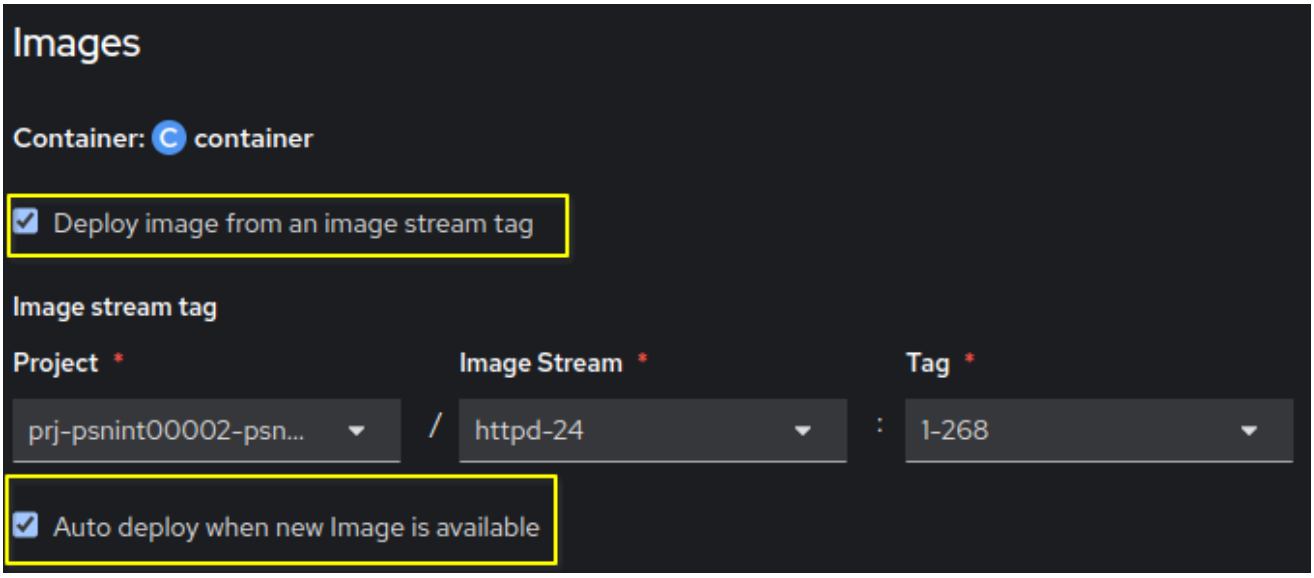


Figura 20. Creazione Deployment - Configurazione Immagine da registry interno

Opzioni Avanzate

In aggiunta a quanto già descritto possiamo impostare altri aspetti legati all'oggetto Deployment, come ad esempio la numerosità delle repliche iniziali e la messa in pausa delle modifiche

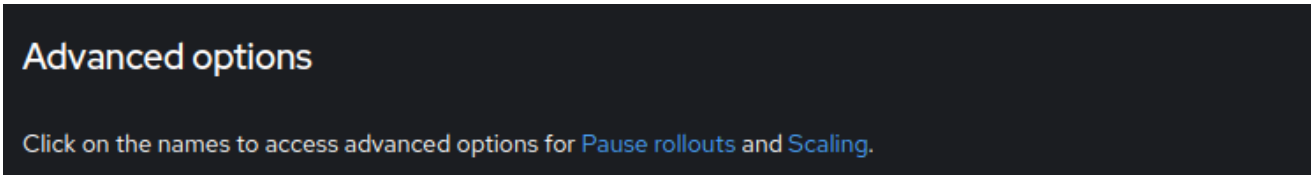


Figura 21. Creazione Deployment - Opzioni Avanzate

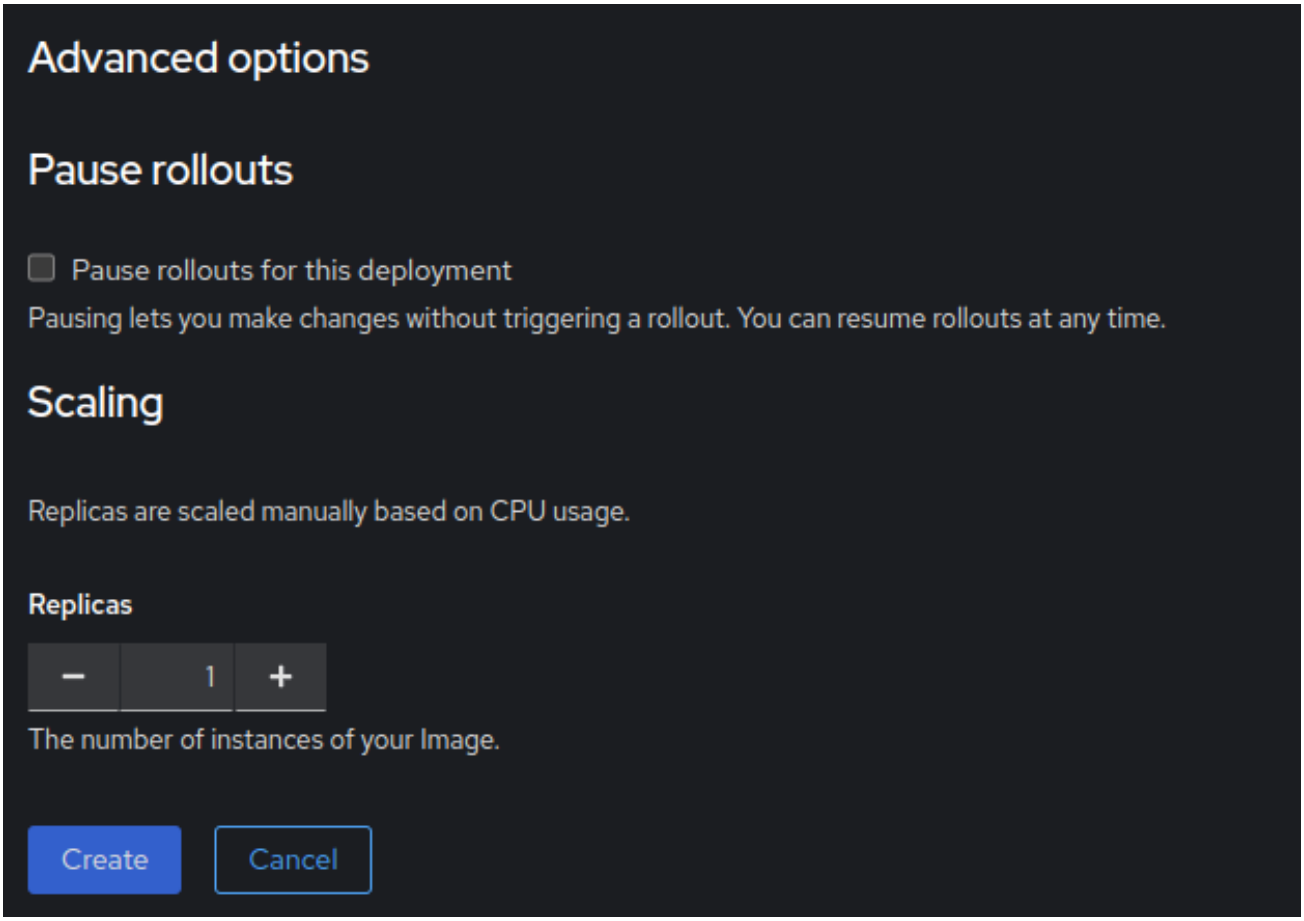


Figura 22. Creazione Deployment - Opzioni avanzate Scaling e Pausa

Pause Rollout

Qualsiasi modifica apportata all'oggetto Deployment (al netto delle annotation) comporta il rollout secondo lo schema prescelto per i riavvi.

Al fine di evitare un impatto sull'operatività applicativa in determinate condizioni, è possibile mettere in pausa i rollout sin da subito selezionando l'opzione "Pause rollout for this deployment".

Per rendere operative le modifiche, successivamente è possibile farlo sia tramite Web Interface sia tramite client oc .

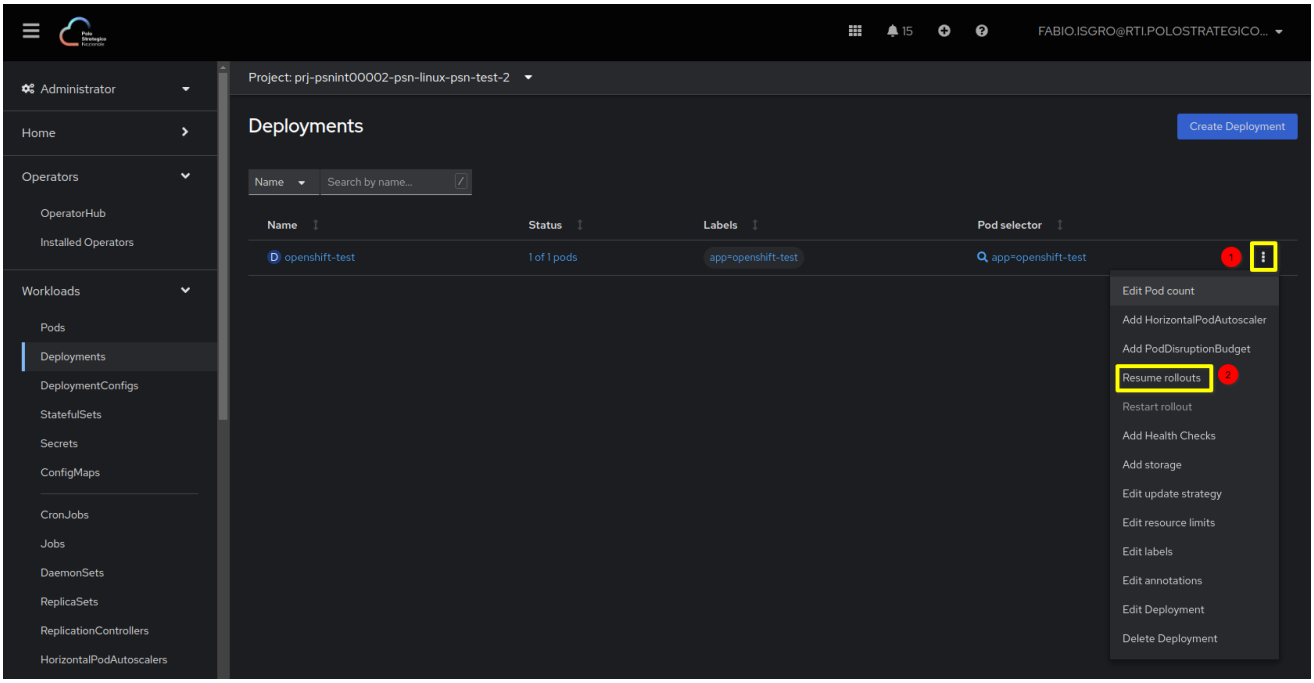


Figura 23. Creazione Deployment - Workflow Pausa rollout

1. Selezionare la Deployment da riavviare facendo click sui tre pallini sulla sinistra
2. Effettuare un click su “Resume rollouts”

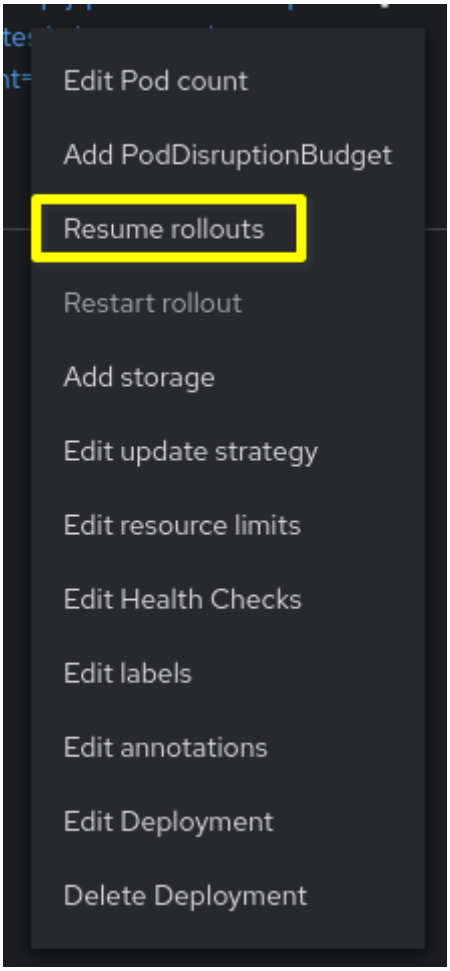


Figura 24.Creazione Deployment - Opzioni avanzate Scaling e Pausa Dettaglio

E possibile cambiarne lo stato anche via OC client od API (attraverso la modifica dell' attributo "***.spec.paused*** " con i valori "*true*" o "*false*"

```
$ oc get deploy -o custom-columns="DEPLOY:.metadata.name,PAUSED:.spec.paused" -n prj-psnint00002-psn-linux-psn-test-2
DEPLOY          PAUSED
openshift-test   true
```

Figura 25. Interrogazione dello stato dei deploy presenti su un determinato Project

```
$ oc patch deploy openshift-test -n prj-psnint00002-psn-linux-psn-test-2  --type="merge" -p
'{"spec":{"paused":true}}'
deployment.apps/openshift-test patched (no change)
```

Figura 26. Messa in Pausa dei rollout per il deployment openshift-test

Il messaggio “no change” è dovuto al fatto che la Deployment era già in stato di pausa

```
$ oc patch deploy openshift-test -n prj-psnint00002-psn-linux-psn-test-2  --type="merge" -p
'{"spec":{"paused": false}}'
$ oc get deploy -o custom-columns="DEPLOY:.metadata.name,PAUSED:.spec.paused" -n prj-
psnint00002-psn-linux-psn-test-2
DEPLOY         PAUSED
openshift-test   <none>
```

Figura 27. Messa in esercizio dei rollout

Il fatto che restituisca “<none>” è da considerarsi normale.

Per maggiori informazioni sul comando “*get*” con l’opzione custom-colum e “merge” si rimanda alla documentazione del comando kubectl di kubernetes su cui il client OC è basato

https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands

Scaling - Numero di Repliche

Ultimo ma non meno importate è l’impostazione relativa al numero di repliche richieste, che per default sono 3; tale opzione non è sempre l’ideale, sopratutto quando la Deployment in questione fa o farà uso di PersistentVolume, in quanto al momento della scrittura di questo documento non è disponibile una storage class di tipo ReadWriteMany ovvero accedibile in lettura e scrittura contemporanea da parte di più repliche.

Pertanto, se si opta per quest’ultima topologia, è opportuno impostare tale valore ad 1.

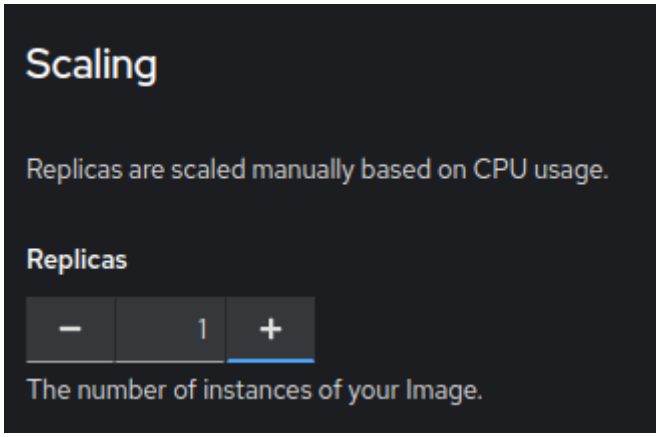


Figura 28. Creazione Deployment - Opzioni avanzate Scaling, numero di repliche

È possibile cambiare il valore del numero di repliche successivamente, sia tramite interfaccia grafica che via OC client o API:

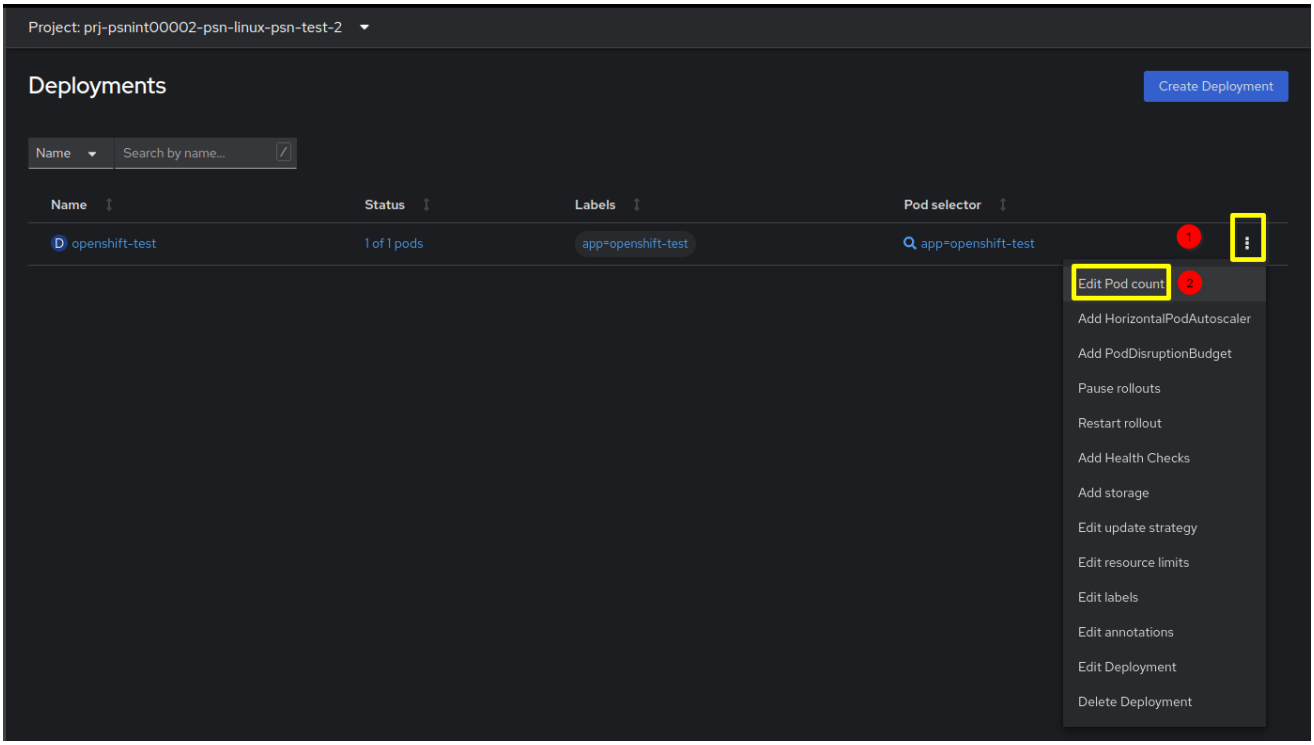


Figura 29. Creazione Deployment - Workflow numero di repliche

1. Selezionare la Deployment da modificare facendo click sui tre pallini sulla sinistra
2. Effettuare un click su “Edit Pod Count”

3. Selezionare il numero repliche desiderate
4. Salvare con “Save”

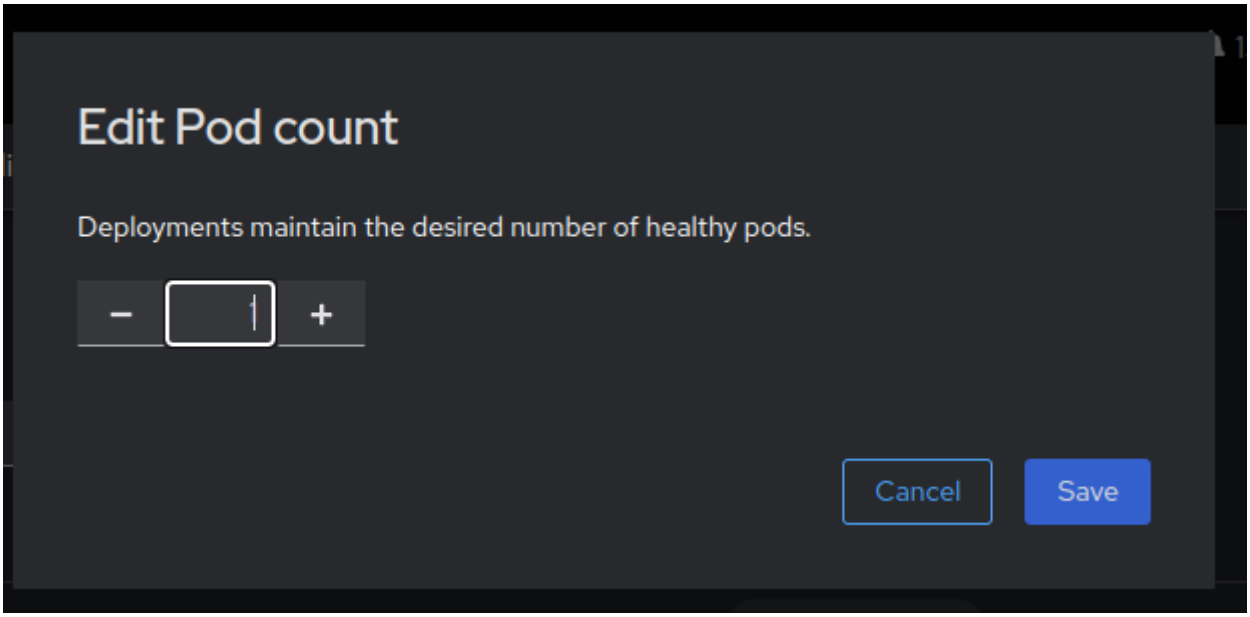


Figura 30. Creazione Deployment - Modifica numero di repliche, dettaglio

E possibile cambiarne il numero di replicas e lo stato anche via OC client o API (attraverso la modifica del seguente attributo “**.spec.replicas** “ .

```
$ oc get deploy -n prj-psnint00002-psn-linux-psn-test-2
NAME           READY   UP-TO-DATE   AVAILABLE   AGE
openshift-test   1/1     1            1           2d9h
```

Figura 31. Interrogazione dello stato dei deploy presenti su un determinato Project

```
$ oc scale --replicas=2 deploy/openshift-test
deployment.apps/openshift-test scaled
$ oc get deploy
NAME           READY   UP-TO-DATE   AVAILABLE   AGE
openshift-test   2/2     2            2           2d9h
```

Figura 32. Incremento delle repliche della deployment “openshift-test”, specificando oltre il nome anche il tipo di oggetto

```
$ oc scale --replicas=0 deploy/openshift-test
deployment.apps/openshift-test scaled
$ oc get deploy
NAME             READY   UP-TO-DATE   AVAILABLE   AGE
openshift-test   0/0     0            0           2d9h
$ oc get pods
No resources found in prj-psnint00002-psn-linux-psn-test-2 namespace.
```

Figura 33. Azzeramento delle repliche della deployment "openshift-test", specificando oltre il nome anche il tipo di oggetto

Revisione Manifest

Tutte le informazioni inserite all'interno della web interface vengono convertite implicitamente in un **manifest** in formato YAML, che è possibile revisionare effettuando un click sul OptionBox "Yaml View" o scaricare tramite l'apposito bottone "Download" (utile come template per deploy successivi oppure applicarlo tramite il client OC).

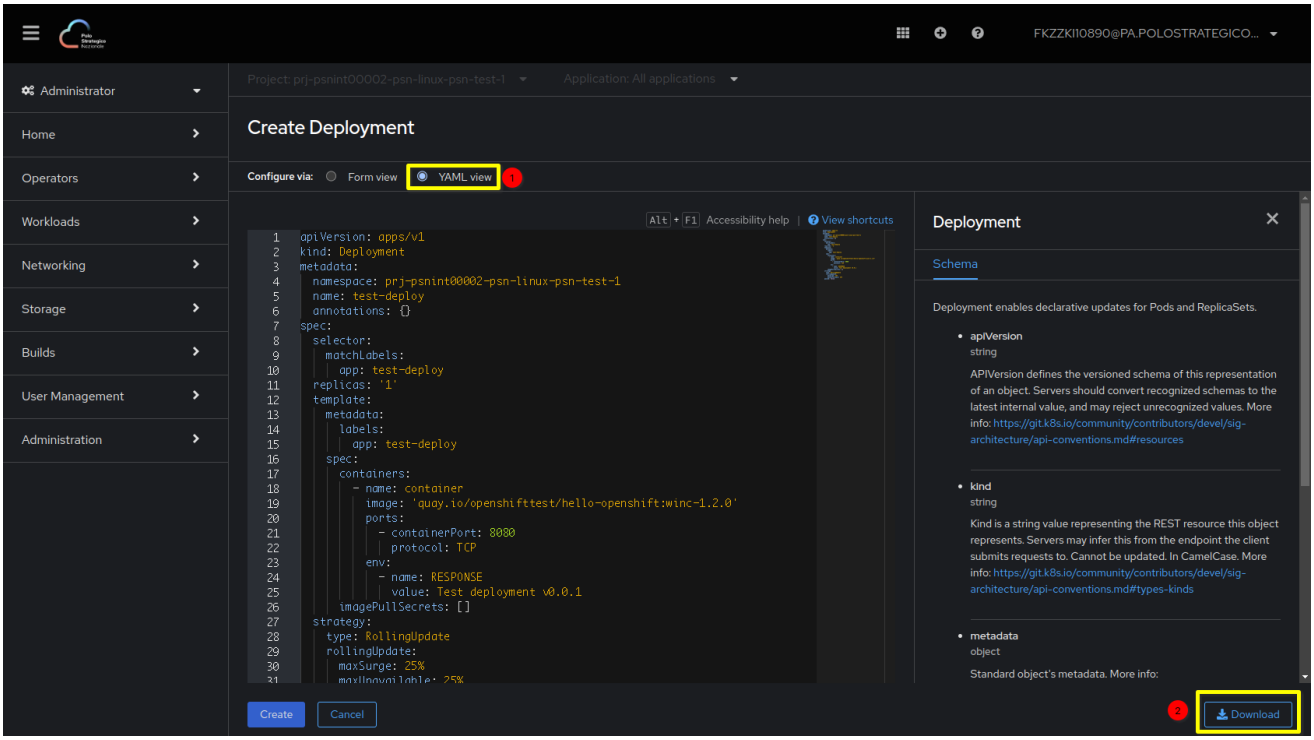


Figura 34. Creazione Deployment - Revisione Manifest YAML

È inoltre possibile applicare modifiche direttamente allo YAML e confermare con “Create” per avallare la creazione del deployment.

L’editor darà suggerimenti sui vari parametri e sui valori ammessi al passaggio del mouse, si consiglia la visione della documentazione ufficiale a riguardo.

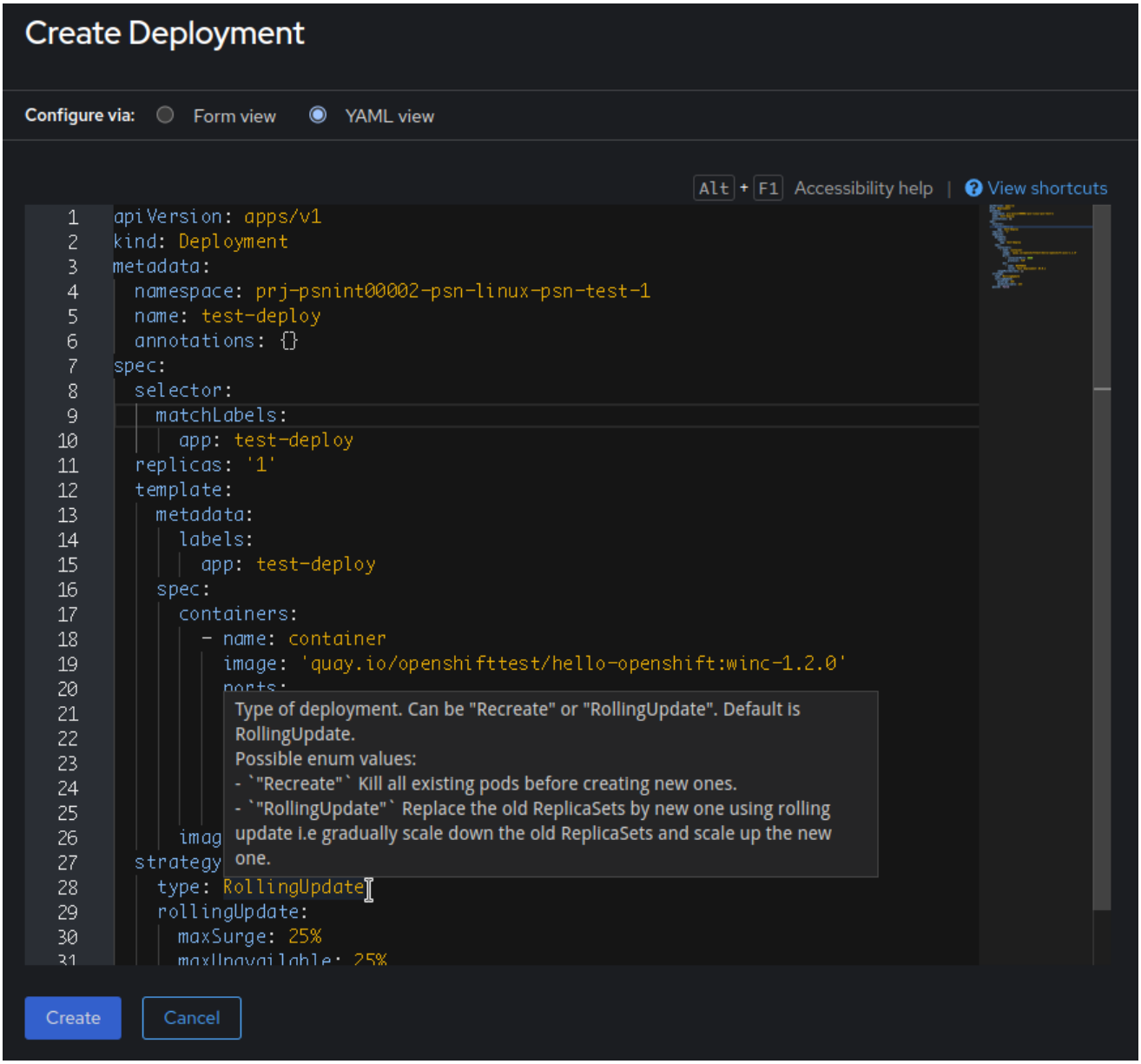


Figura 35. Creazione Deployment - Revisione Manifest YAML, suggerimento sul tipo di update

NOTA: Si rammenda che nelle prossime versioni (4.14) il costrutto DeploymentConfig sarà deprecato e pertanto ne è sconsigliato l'uso.

https://access.redhat.com/documentation/en-us/openshift_container_platform/4.12/html/building_applications/deployments

2. Deploy applicazione da CLI

Per il deploy di un'applicazione da CLI, è possibile utilizzare il client OC direttamente da terminale.

Per una modalità di deploy più "standard" basterà utilizzare il comando "*oc new-app applicazione opzioni*".

Es.

```
$ oc new-app centos/ruby-25-centos7~https://github.com/sclorg/ruby-ex.git

$ oc new-app --image=registry.redhat.io/ubi8/httpd-24:1-268 --name test2
```

Figura 36. Esempio deploy new-app

Il client OC permette inoltre la possibilità di utilizzare metodi di deploy più "avanzati" sfruttando dei file locali al client, attraverso il comando "*oc create -f file.yaml*".

```
$ oc apply -f deploy1.yaml

$ oc apply -f deploy2.yaml -n project
```

Figura 37. Esempio deploy da file YAML

NOTA: Si consiglia tale opzione prettamente agli utenti più esperti e sempre dopo aver verificato di essere all'interno del project corretto (nel caso di più istanze caas e project differenti) o di specificare l'opzione "*-n project*" per evitare errori.

6 Come pubblico i servizi applicativi?

La pubblicazione dei servizi applicativi può avvenire in due modalità:

- **Route:** Oggetto peculiare del mondo Openshift che esiste da ben prima che il progetto Kubernetes implemetasse l'oggetto **Ingress**
- **Ingress:** Oggetto peculiare del mondo Kubernetes di cui Openshift comunque fa parte ed è perfettamente compatibile

Per entrambe le soluzioni cambia il manifest con cui vengono applicate le configurazioni, ma nella sostanza all'interno di Openshift l'oggetto principale, per ragioni storiche, è le Route, da cui gli Ingres si sono fortemente ispirati (vedasi anche le Deployment Config che esistevano ben prima delle Deployment di Kubernetes).

Entrambi implementano l'esposizione di applicazioni su protocollo HTTP (su porta 80) oppure HTTPS (su porta 443) al quale ogni PA è libera di associare un nome DNS e volendo i propri certificati TLS (storicizzati sulla piattaforma tramite opportuna Secret (nel caso di ingress) oppure all'interno dei manifest per le Route

Nota bene: La creazione di un Ingress all'interno di Openshift genera la sua Route Gemella gestita dalla piattaforma e qualsiasi modifica sull'oggetto verrà riflessa anche su quest'ultima, per gli utenti più esperti non è necessaria l'imposizione del valore di IngressClass all'interno, in quanto direttamente gestite dalla piattaforma.

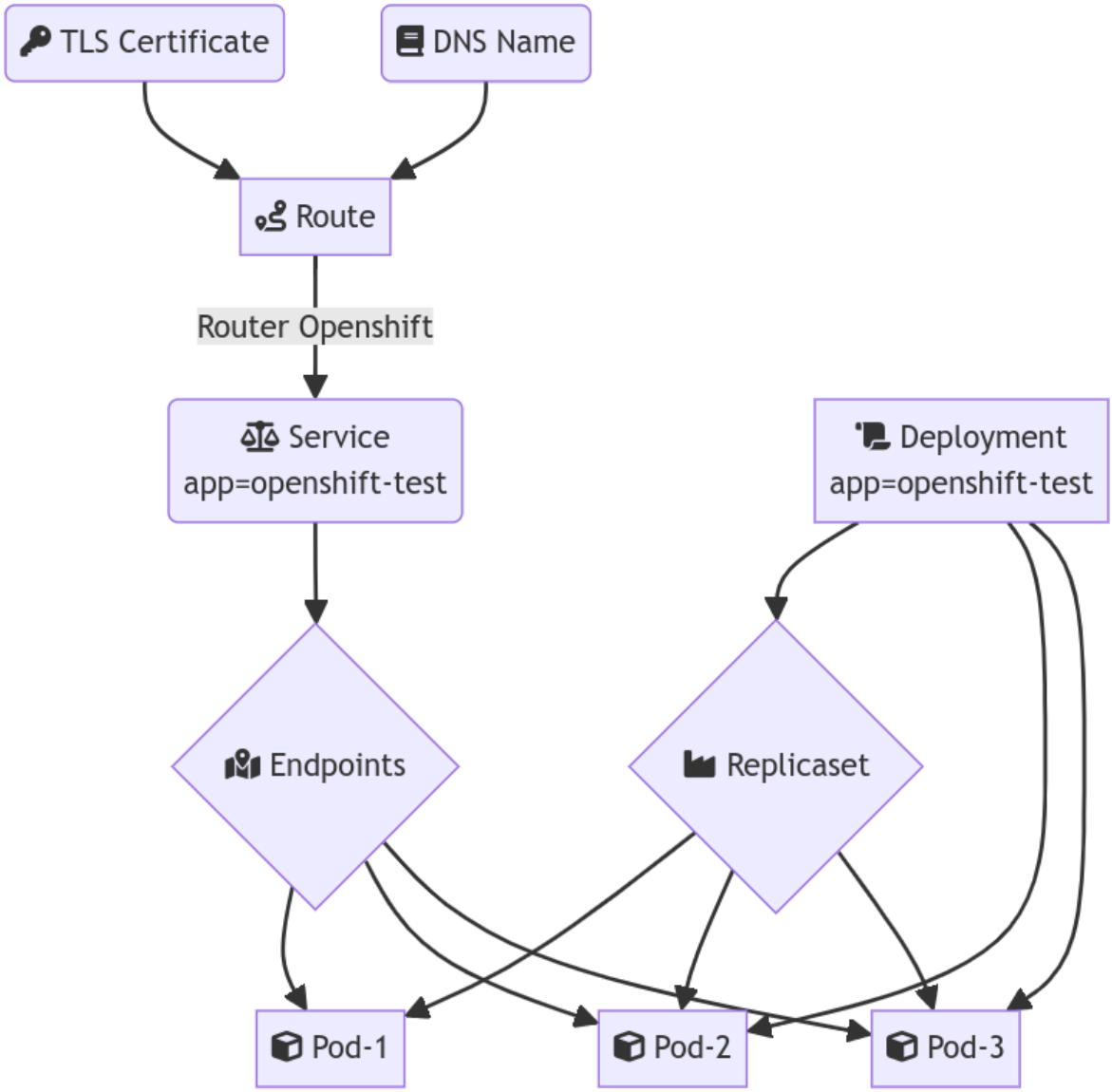


Figura 38. Esposizione Servizi - Schema generale componenti

Per pubblicare un servizio è necessario:

- **Deployment**
- **Service** che punta al medesimo selector della deployment (es. app=openshift-test), creato contestualmente al deployment se l'operazione è fatta da GUI

- **Secret** contenente il nostro certificato TLS (Opzionale)
- **Route** oppure **Ingress**

Assumendo che il deployment effettuato nel capitolo Deploy applicazioni da GUI sia concluso, è possibile procedere all'esposizione del servizio.

Come sempre, è possibile farlo sia tramite GUI che tramite client OC o API

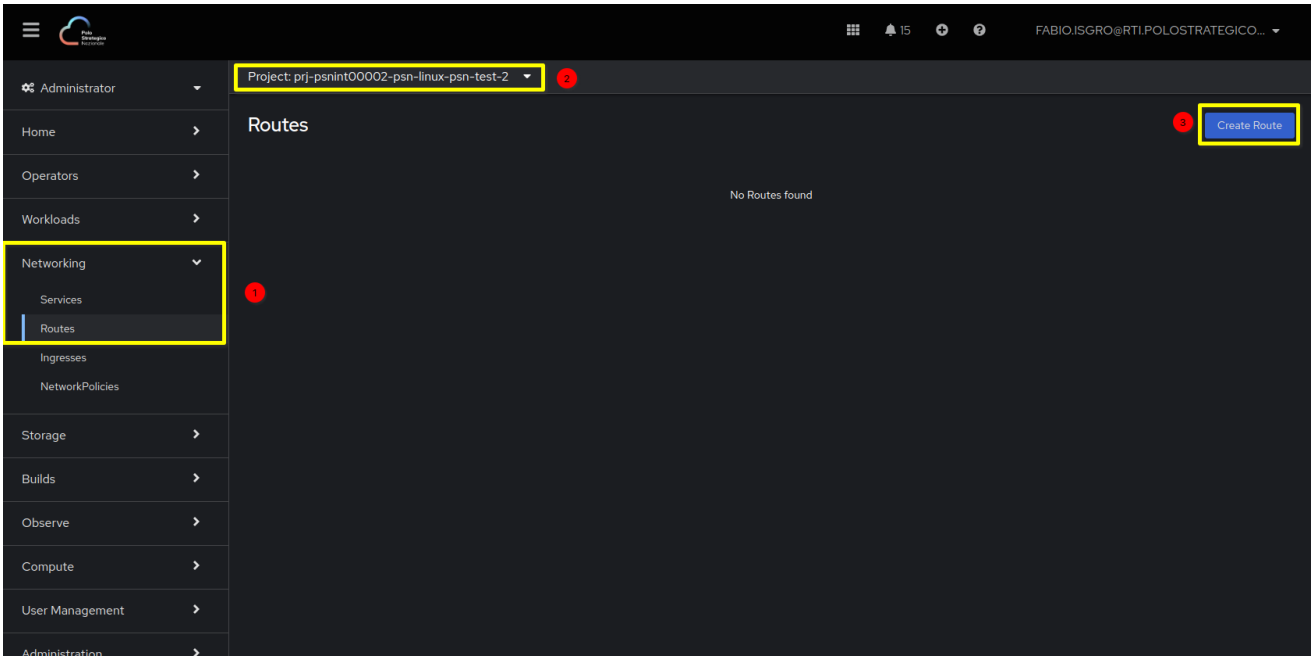


Figura 39. Esposizione Servizi - Creazione Route Openshift

1. Una volta selezionato "**Networking**" > "**Routes"**
2. Selezionare l'opportuno project dal menù modale posto al centro della videata
3. Avviare la creazione della Deployment effettuando click sul bottone "Create Route"

Ci verranno mostrate a video le opzioni peculiari delle Route dove dovremmo fornire

- Nome
- Hostname
- Eventuale Contesto (es. /)
- Eventuale Supporto TLS

- Relativi certificati e modalità di cifratura

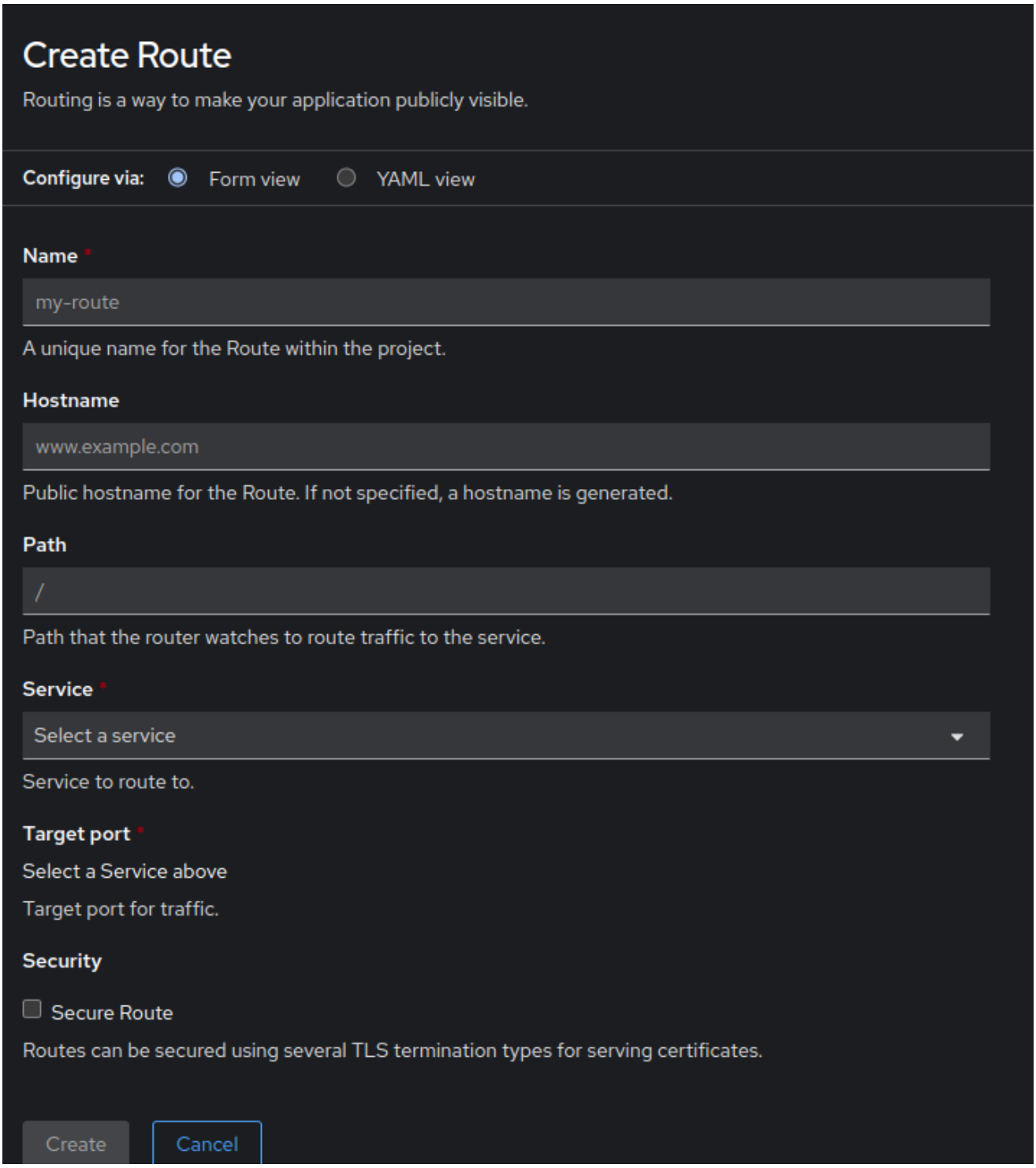


Figura 40. Esposizione Servizi - Creazione Route Openshift, Impostazioni Generali

Lo scopo principale delle Route Openshift è quello di rendere disponibile all'esterno la nostra applicazione, alla stregua di un Reverse Proxy; pertanto oltre al nome della risorsa dobbiamo definire l'hostname ed il contesto per i quali i Router di Openshift (componente infrastrutturale gestita dalla piattaforma basato a sua volta su HAProxy) esporranno il servizio

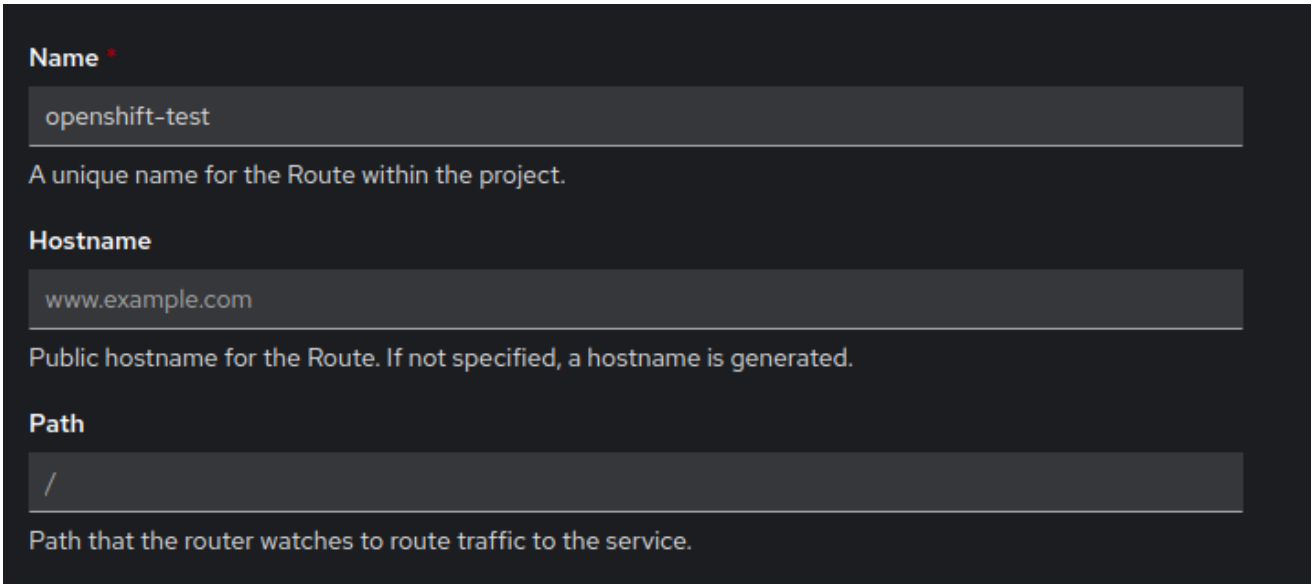


Figura 41. Esposizione Servizi - Creazione Route Openshift, Dettaglio Hostname di pubblicazione

Il nome è mandatorio, mentre hostname in caso non venga specificato segue la seguente logica

http[s]://${NOME_ROUTE}-${NOME_PROJECT}.apps.apps.pvc-os-caas01-rs.polostrategiconazionale.it

Dove, nel caso preso in esame diventerà

http[s]://openshift-test-prj-psnint00002-psn-linux-psn-test-2.apps.pvc-os-caas01-rs.polostrategiconazionale.it

In alternativa la PA potrà inserire il suo hostname arbitrario e dovrà pubblicare sulla propria zona DNS il seguente CNAME

Esempio:

La rotta sarà configurata nel seguente modo

Hostname Route Openshift: miaapp.miapa.it

Record DNS sulla zona della PA:

```
miaapp IN CNAME cname.apps.pvc-os-caas01-rs.polostrategiconazionale.it.
```

Nota Bene: Ricordarsi il . (punto) alla fine del CNAME

La risoluzione risultante sarà la seguente

```
$ host miaapp.miapa.it
miaapp.miapa.it is an alias for cname.apps.apps.pvc-os-caas01-rs.polostrategiconazionale.it.
miaapp.miapa.it has address 81.126.72.22
$ host cname.apps.apps.pvc-os-caas01-rs.polostrategiconazionale.it
cname.apps.apps.pvc-os-caas01-rs.polostrategiconazionale.it has address 81.126.72.22
```

1. Configurazione Service di Destinazione

Una volta configurato l' hostname dell'applicazione sarà necessario indirizzarlo verso il Service che bilancia le repliche della nostra Deployment.

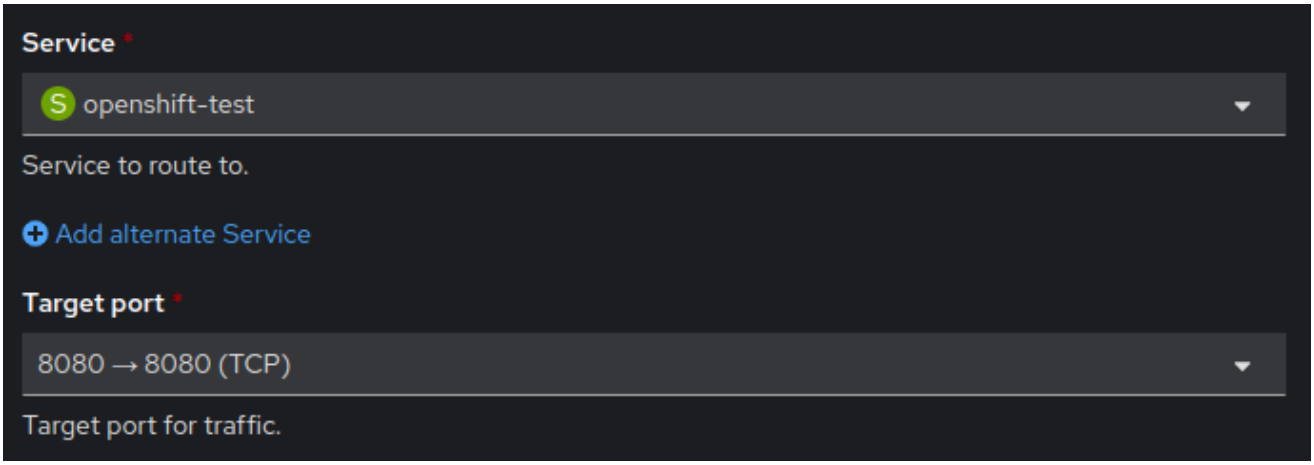


Figura 42. Esposizione Servizi - Creazione Route Openshift, Configurazione service di destinazione

Possiamo sceglierlo dal menù a tendina relativo e selezionare contestualmente anche la porta d'erogazione nel momento in cui quest'ultimo presentasse più porte.

In alternativa possiamo effettuare il balance su più Service permettendo l'erogazione di un servizio in modalità A/B testing, dove una quota parte del traffico viene ripartita su due versioni di un applicativo.

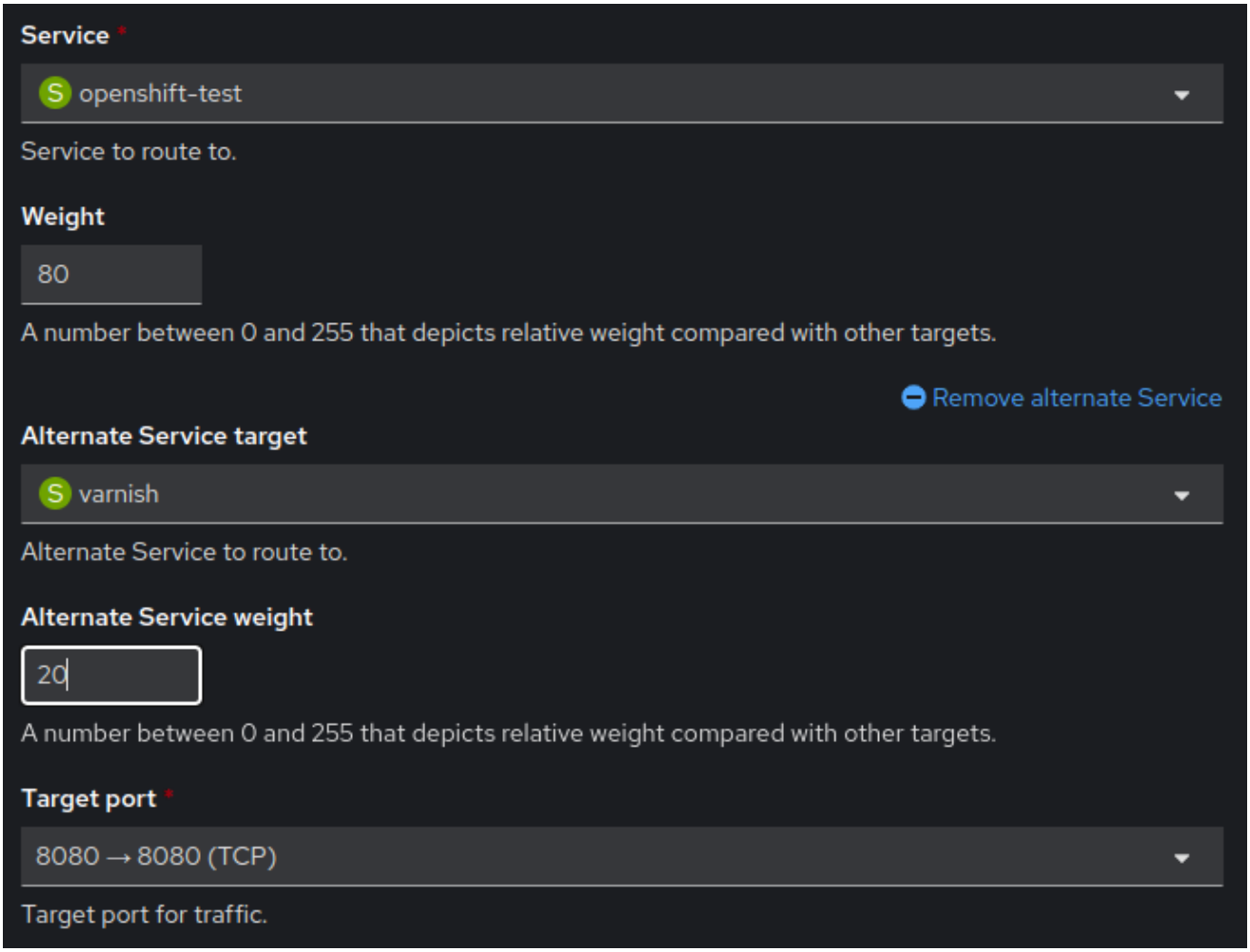


Figura 43. Esposizione Servizi - Creazione Route Openshift, Configurazione avanzata Service multiplo in modalità frazionaria

2. Configurazione TLS

Altro aspetto non meno importante dell'esporre i servizi riguarda la cifratura TLS.

I browser moderni in genere tendono a con collegarsi a servizi privi cifratura TLS pertanto è sempre consigliabile abilitare sulle Route il flag "**Secure Route**"

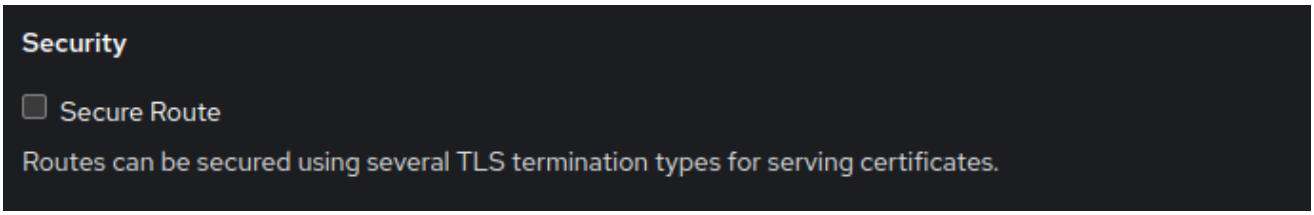


Figura 44. Esposizione Servizi - Creazione Route Openshift, Abilitazione TLS

La piattaforma chiederà come terminare la connessione sui Router Openshift e cosa fare nel momento in cui si tenta di contattare l'applicazione in chiaro (porta 80)

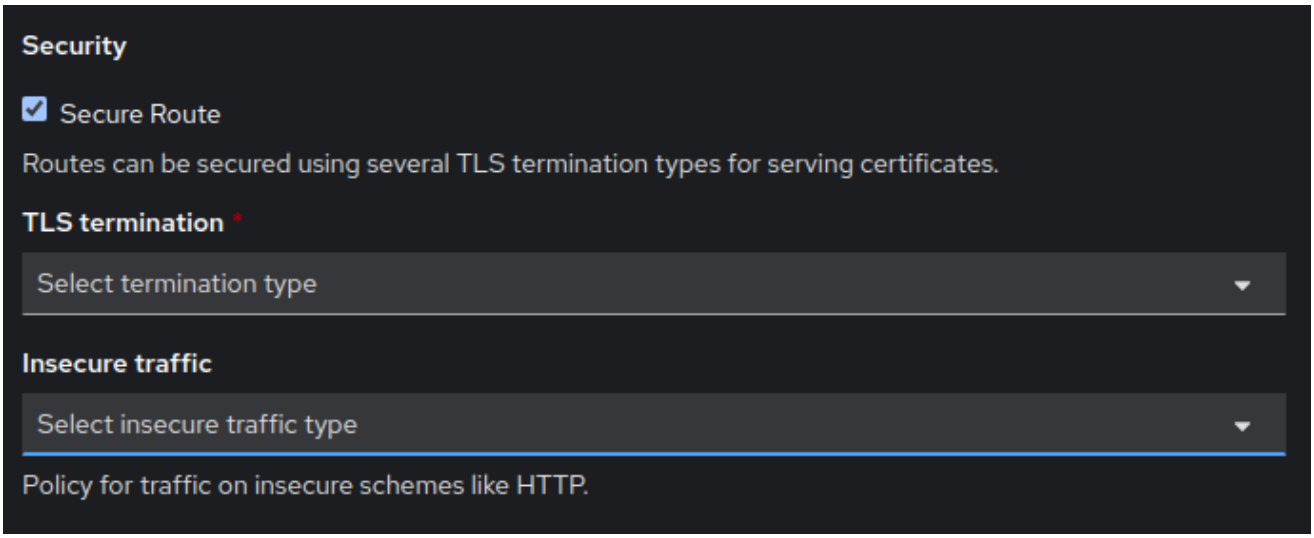


Figura 45. Esposizione Servizi - Creazione Route Openshift, Configurazione TLS

Openshift ci presenta tre modalità di terminazione:

- **Edge**: Il traffico viene terminato sui Router Openshift ed in chiaro all'interno del Project Openshfit, in quanto erogato in chiaro dai Pod finali.
- **Passthrough**: Il traffico non viene terminato sui Router ma fatto arrivare ai server ed esposto in TLS direttamente dai Pod che dovranno erogare il servizio in modalità TLS e non richiede configurazioni aggiuntive.
- **Re-Ecrypt**: È la combinazione delle due modalità precedenti, il traffico viene terminato sui Router Openshfit ma il service espone il proprio servizio in formato TLS

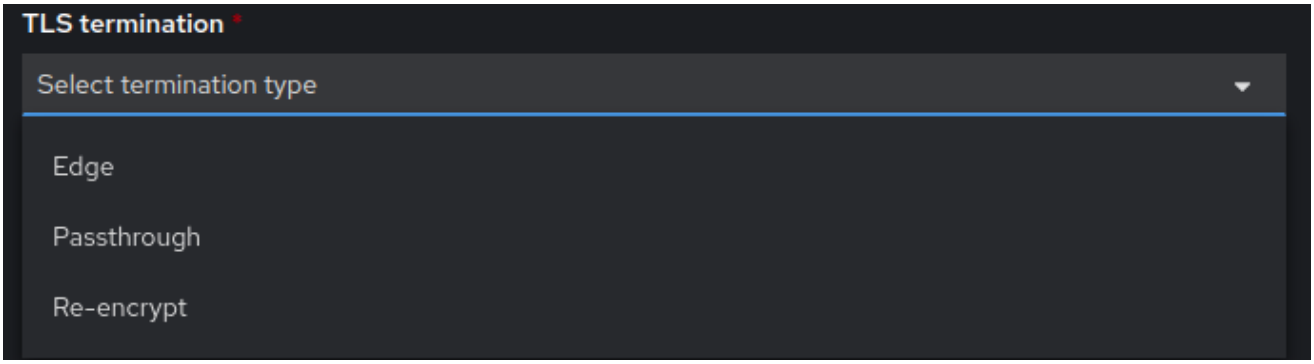


Figura 46. Esposizione Servizi - Creazione Route Openshift, Configurazione Terminazione TLS

Per le modalità Edge e Re–Encrypt appariranno ulteriori impostazioni che permettono "**eventualmente**" di caricare la chiave ed il certificato.

Se la rotta viene pubblicato sotto apps.pvc-os-caas01-rs.polostrategiconazionale.it esso avrà by default un certificato TLS verificato da una CA attendibile a costo zero.

Nel caso venga esposto con l'hostname personalizzato e non verranno impostati certificati, tali Route verranno comunque erogate ma con il certificato TLS della piattaforma valido per "*.apps.pvc-os-caas01-rs.polostrategiconazionale.it"

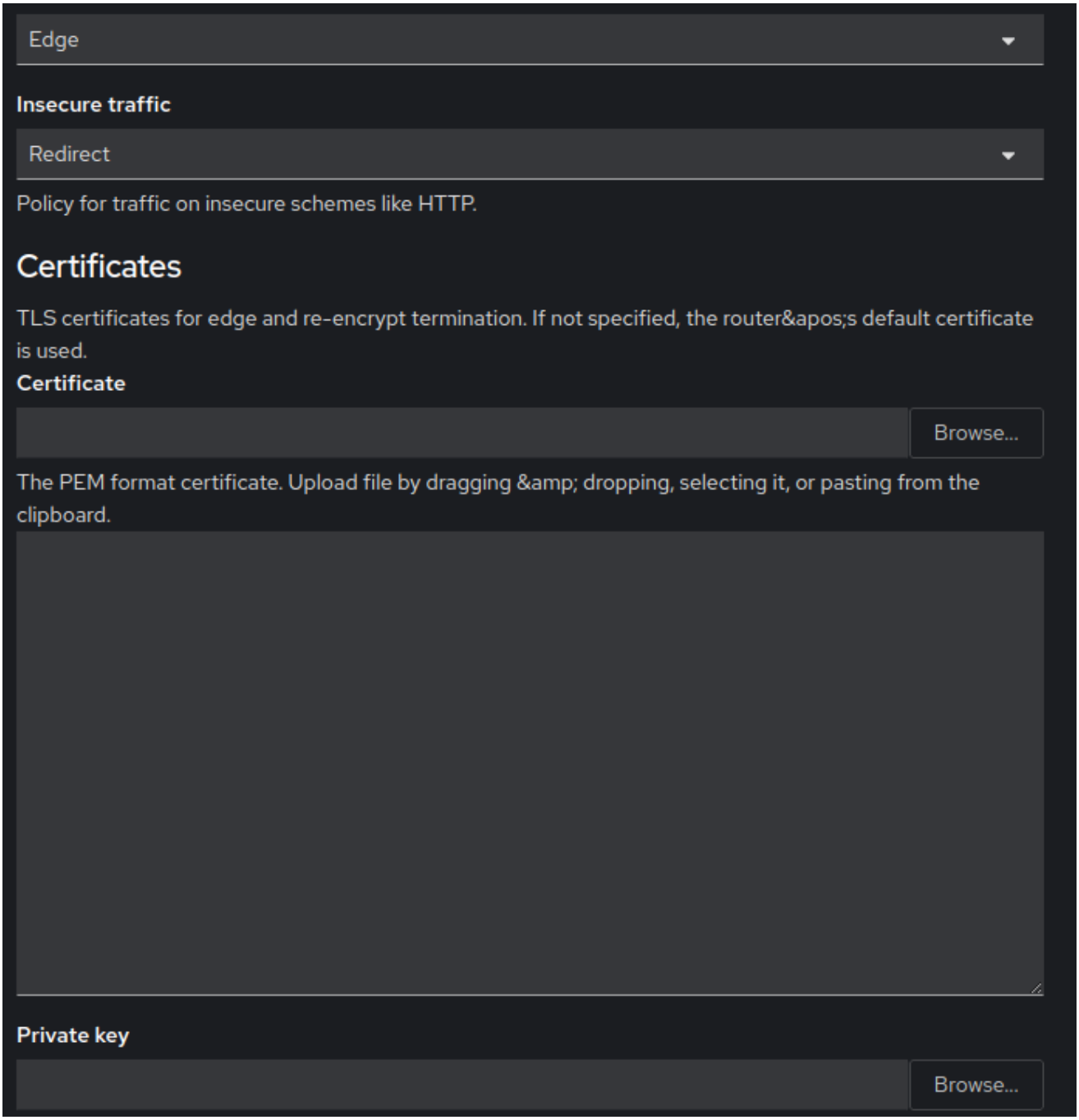


Figura 47. Esposizione Servizi - Creazione Route Openshift, Configurazione Certificati TLS su Router Openshift

Infine, resterà da configurare il traffico in chiaro nel momento in cui esso arriva ai Router Openshift ed anche qui abbiamo tre scelte:

- **None:** Ovvero non pubblicare il servizio in chiaro
- **Allow**: Abilitare il traffico in chiaro
- **Redirect**: Viene inviato il codice HTTP 302 e fatto redirect in modalità TLS

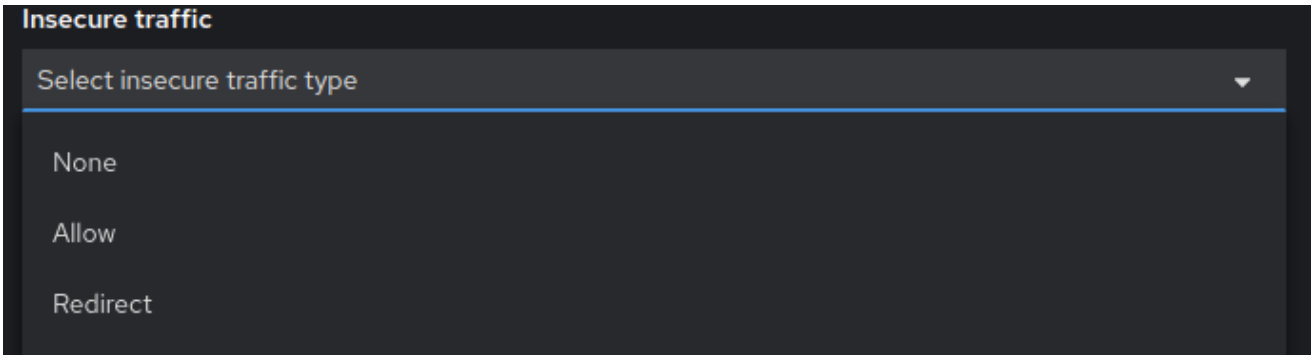


Figura 48. Esposizione Servizi - Creazione Route Openshift, Configurazione Redirect traffico non TLS

3. Revisione dei Manifest in formato Yaml

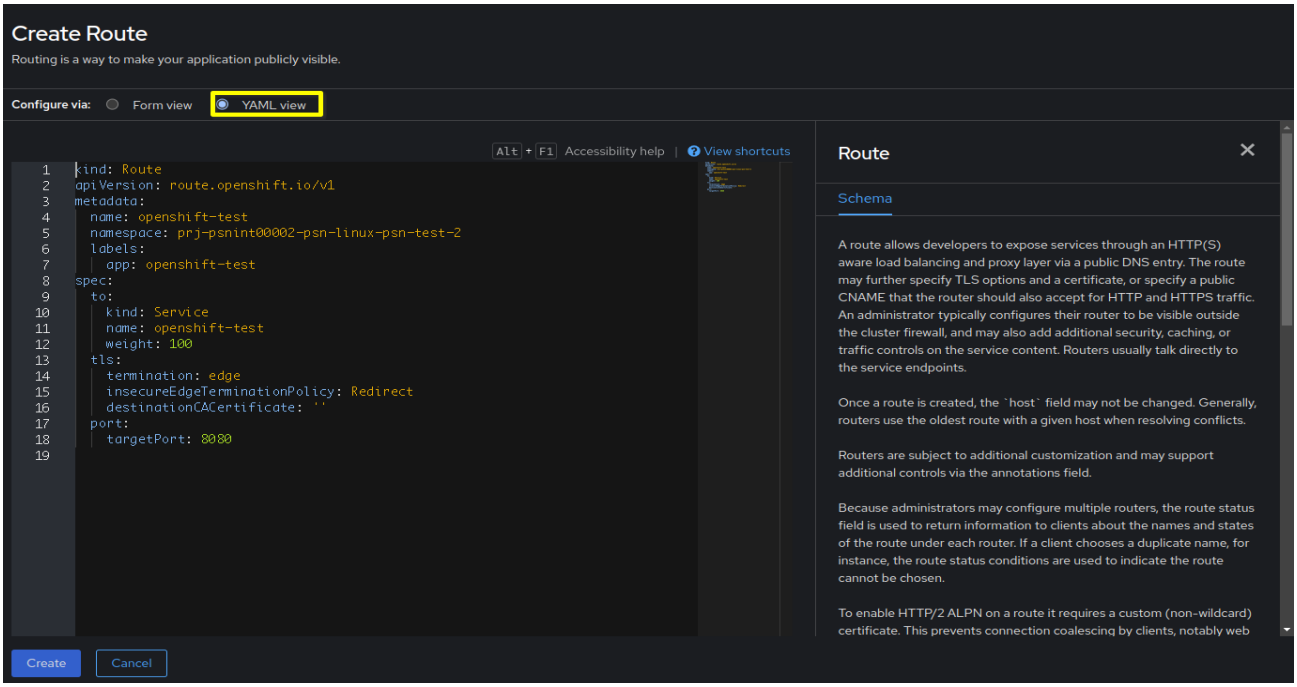


Figura 49. Esposizione Servizi - Creazione Route Openshift, Revisione Manifest in modalità YAML

Tutte le configurazioni effettuate con la GUI hanno la loro contropartita in formato YAML, che permette il salvataggio ed eventualmente la possibilità di riapplicare lo stesso modello in seguito, tramite OC client.

4. Esposizione servizi da CLI

È possibile effettuare tutte le operazioni sulla piattaforma sia tramite Gui che tramite CLI, in quanto entrambe le modalità non fanno altro che consultare le API messe a disposizione da Openshift.

Quando si opera da GUI bisogna fare due distinzioni:

- Creazione Deployment via "*oc new-app*"
    - Il Service relativo all'applicazione è creato contestualmente
- Creazione Deployment via manifest YAML "*oc apply -f* "
    - Il Service relativo deve essere creato manualmente

L' applicazione sarà raggiungibile dall'esterno del cluster solo dopo aver esposto l'oggetto "**service**" con il comando "***oc expose***", specificando la Deployment e la porta.
Nel dettaglio, dopo aver completato il deploy di un'applicazione, utilizzare i seguenti comandi:

```
$ oc expose deploy openshift-test --name svc-openshift-test --port 8080
$ oc get svc
NAME              TYPE       CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
svc-openshift-test   ClusterIP   100.68.102.9    <none>       8080/TCP   33s
$ oc get ep
NAME              ENDPOINTS         AGE
svc-openshift-test   100.64.42.33:8080   2m15s
```

Nota Bene: Il Service, tramite opportune configurazioni, potrebbe rispondere su una porta e bilanciare il l'applicativo su una porta diversa dei Pod (https://kubernetes.io/docs/concepts/services-networking/service/).

Una volta creato il service procediamo a creare la relativa Rotta come segue

```
$ oc expose service svc-openshift-test
route.route.openshift.io/svc-openshift-test exposed
$ oc get route.route.openshift.io/svc-openshift-test -o jsonpath='{.spec.host }{"\n"}'
svc-openshift-test-prj-psnint00002-psn-linux-psn-test-2.apps.pvc-os-caas01-rs.polostrategiconazionale.it
```

Una volta lanciato il comando di "**expose**" esso sarà privo di TLS ed esposto all'indirizzo http://svc-openshift-test-prj-psnint00002-psn-linux-psn-test-2.apps.pvc-os-caas01-rs.polostrategiconazionale.it

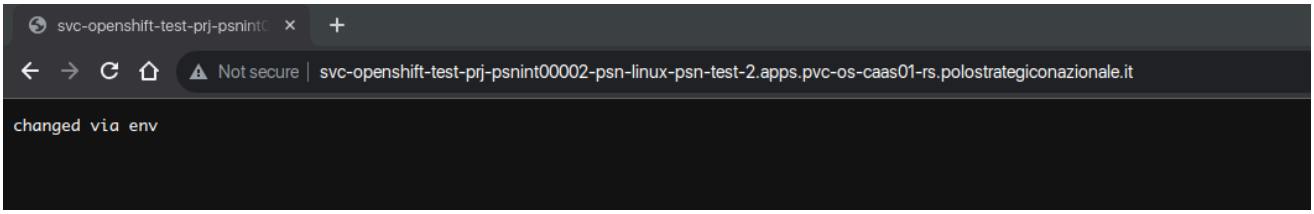


Figura 50. . Esposizione Servizi - Creazione Route Openshift via cli senza TLS

Nel caso si volesse esporre direttamente in TLS è sufficiente cambiare il comando "oc expose" con il seguente

```
$ oc create route  edge --service svc-openshift-test
```

7 Persistenza delle Applicazioni

La natura effimera dei container e dei loro contesti applicativi porta alla perdita dei dati d'esecuzione al momento della normale chiusura dell'applicazione oppure in caso di Fault Applicativo.

Pertanto, se l'applicazione prevede la persistenza del dato, l'ambiente Openshift mette a disposizione i seguenti oggetti:

- **PersistentVolumeClaim:** Rappresenta la richiesta allo schedulatore di mettere a disposizione una determinata quantità di spazio e con quale tipologia d'accesso (Multiplo , Singolo , Sola lettura)
- **StorageClass:** Rappresenta la tipologia di storage da consumare e può essere riferita ad un particolare tipologia di Storage
- **PersistentVolume:** Esso rappresenta lo storage utilizzato dallo schedulatore e montato all'interno del contesto d'esecuzione
- **CSI Provisioner:** Esso è un automatismo che genera i PersistentVolume a seguito di un PersistentVolumeClaim

Il cliente ha la facoltà creare quanti PersistentVolumeClaim desidera a patto che essi abbiano come dimensione minima 1Gigabyte (nei limiti delle quote richieste nel piano dei fabbisogni o successive espansioni).

All'interno del Caas Open Source vi sono diverse StorageClass, tuttavia è stato predisposto un meccanismo tale per cui viene generata una storage class per singolo cliente, ma ad un cliente verrà impostata sempre e solo la propria, senza facoltà di scelta, al fine di tutelare la tenancy di ogni cliente.

Per limiti imposti dal provisioner attualmente in uso sulla piattaforma è possibile consumare i volumi nella sola modalità (RWO) ovvero ReadWriteOnce e ne va tenuto conto nel caso si voglia scalare una Deployment con accesso concorrente al disco.

1. Creazione Volumi da GUI

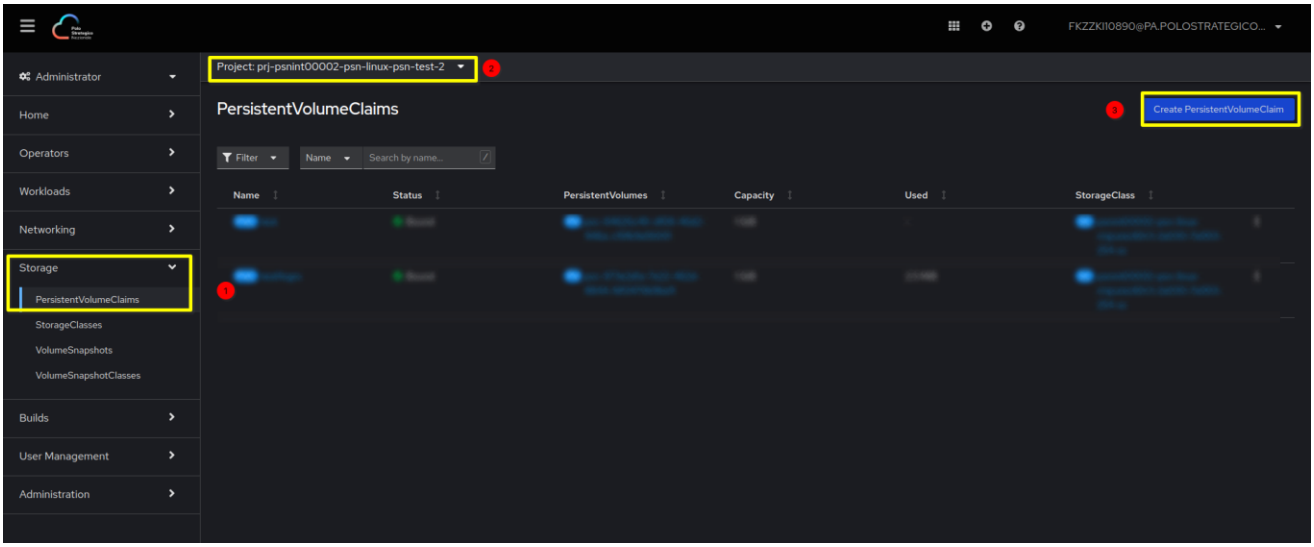


Figura 51. Creazione Volumi - Workflow creazione Persistent Volume Claim

4. Una volta selezionato “**Storage**” > “**PersistentVolumeClaim”**
5. Selezionare l’opportuno project dal menù modale posto al centro della videata
6. Avviare la creazione del Deployment effettuando click sul bottone “**PersistentVolumeClaim**”

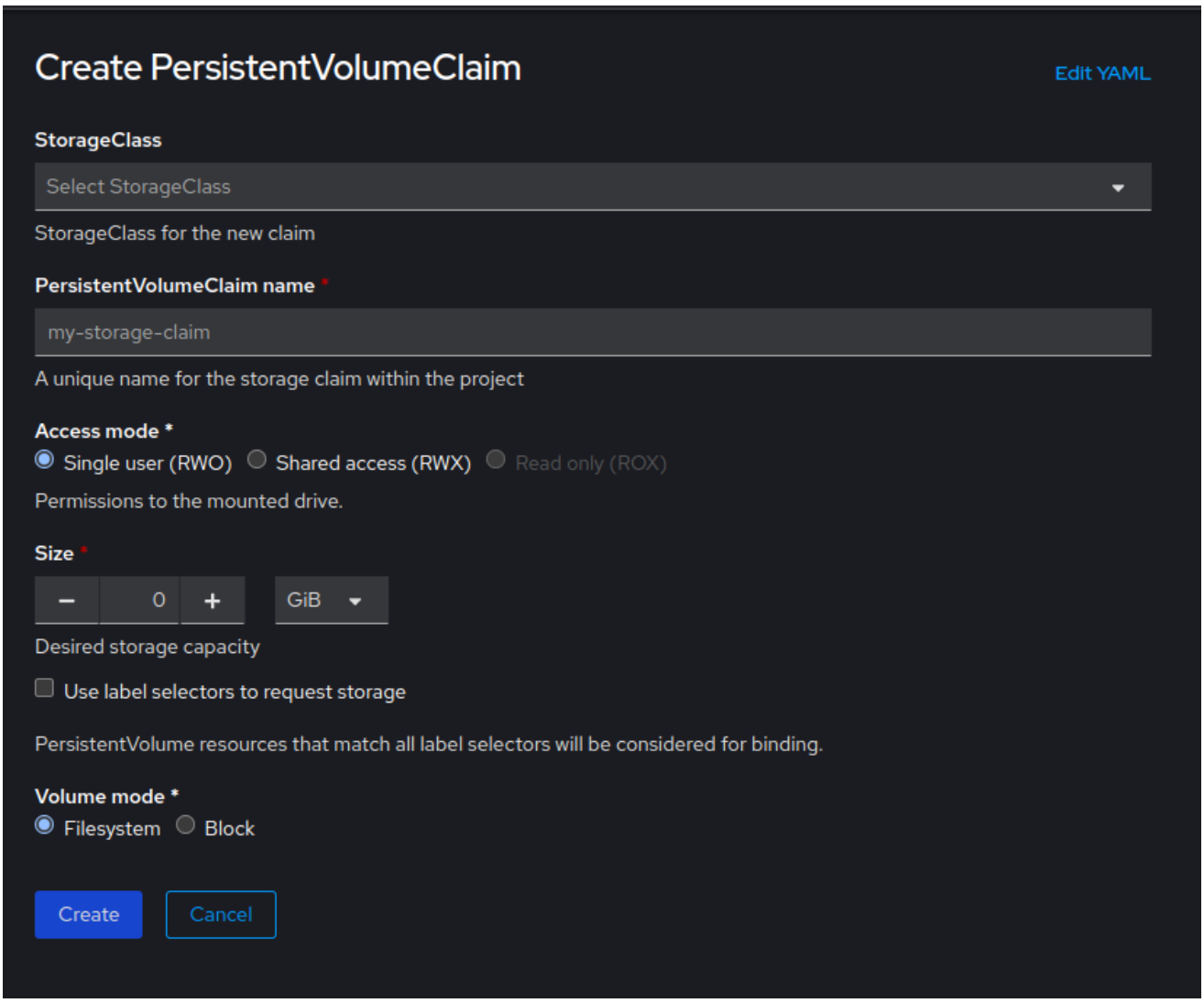


Figura 52. Creazione Volumi - Dettaglio creazione PersistentVolumeClaim

Le informazioni richeste sono abbastanza semplici:

- **StorageClass:** Indica quale storage utilizzare , selezionare quella che ha lo stesso prefisso del project dove create il PVC (in questo caso "*psnint00002-psn-linux-rmpurec60r3-2a030-7a003-254-sc*") , anche in caso d'errore la piattaforma indirizzerà correttamente la richiesta
- **Nome del Volume:** Indica il nome del Claim che sarà referenziato dalle Deployment (ad esempio pvc-openshift-test)
- **Modalità accesso**: Indicare sempre RWO (è l'unica opzione attualmente supportata)

- **Dimensione**: Indica la dimensione richiesta, attenzione non può essere minore di 1Gi
- **Volume Mode**: Può essere Filesystem o Blocchi; se non si hanno esigenze particolari utilizzare "Filesystem"

Una volta terminato l'inserimento dei dati è possibile confermare con **create**, o in alternativa è possibile vedere la versione YAML risultante facendo click sull'apposta opzione in alto destra.

2. Creazione Volumi da CLI

L'allocazione di nuovi PersistentVolumeClaim tramite OC client è legata all'applicazione di un manifest YAML in quanto non c'è un verbo specifico per la loro creazione.

A titolo d'esempio si riporta il manifest YAML per replicare il PVC creato in precedenza

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-openshift-test
  namespace: prj-psnint00002-psn-linux-psn-test-2
spec:
 accessModes:
  - ReadWriteOnce
 resources:
  requests:
   storage: 1Gi
 storageClassName: psnint00002-psn-linux-rmpurec60r3-2a030-7a003-254-sc
 volumeMode: Filesystem
```

Figura 53. Esempio Manifest YAML per creazione PVC

È suffciente applicarlo e questo sarà il risultato finale

```
$ oc get pvc
NAME              STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS
AGE
pvc-openshift-test   Bound    pvc-b5117e49-0672-4377-9fe2-839a7230aa8c   1Gi        RWO
psnint00002-psn-linux-rmpurec60r3-2a030-7a003-254-sc   21s
```

3. Aggiunta Volumi persistenti da GUI

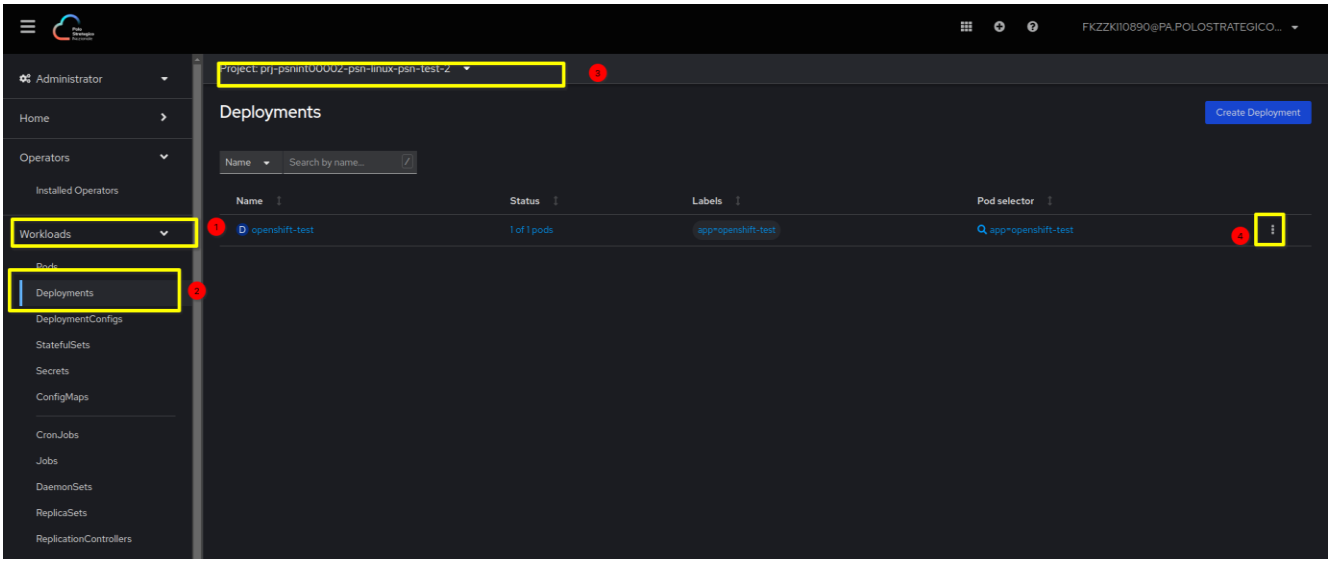


Figura 54. Aggiunta Volumi - Workflow aggiunta Persistent Volume ad una Deployment già esistente

1. Una volta selezionato "**Workload**" > "**Deployments"**
2. Selezionare l'opportuno project dal menù modale posto al centro della videata
3. Effettuare click sull'icona con i tre pallini per accedere al menu delle opzioni

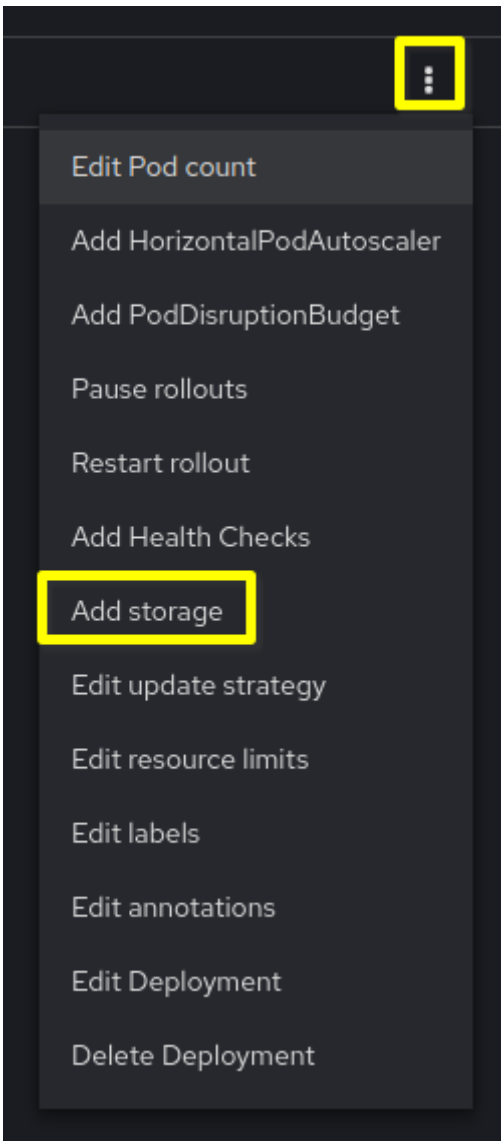


Figura 55. Aggiunta Volumi - Dettaglio aggiunta Persistent Volume ad una Deployment già esistente

4. Facciamo click sull'elemento "Add Storage"

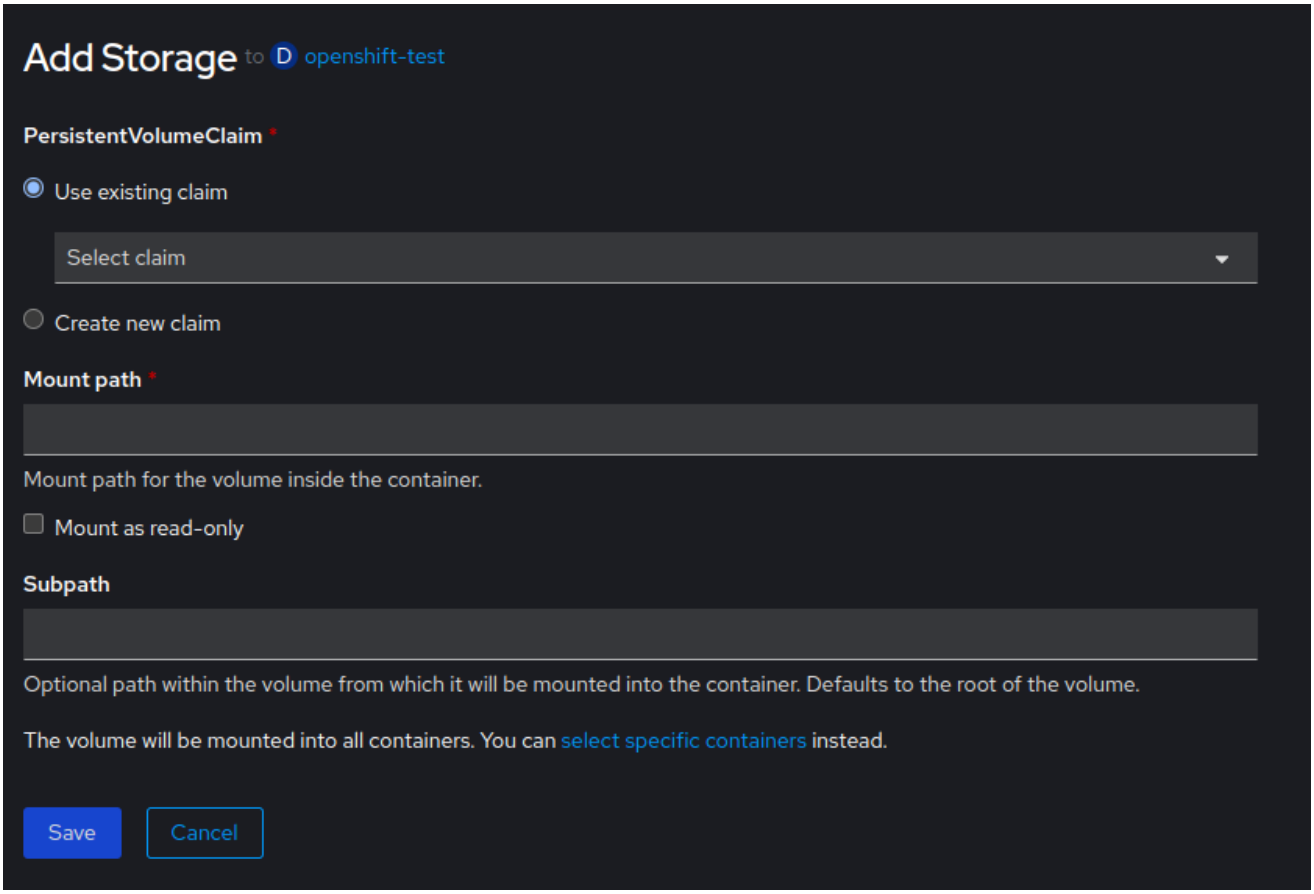


Figura 56. Aggiunta Volumi - Dettaglio aggiunta Persistent Volume ad una Deployment già esistente

Da questa schermata è possibile agganciare un PVC creato in precedenza oppure crearne uno contestualmente e montarlo all'interno del pod in un Path specifico.

4. Aggiunta Volumi persistenti da CLI

È possibile replicare la stessa operazione effettuata nel paragrafo precedente anche in modalità "da command line" con l'ausilio del client OC; nell'esempio che segue si aggiungerà il PesistentVolumeClaim denominato "pvc-openshift-test" alla Deployment "openshift-test

```
$ oc get pvc
NAME              STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS
AGE
pvc-openshift-test   Bound    pvc-b5117e49-0672-4377-9fe2-839a7230aa8c   1Gi        RWO
psnint00002-psn-linux-rmpurec60r3-2a030-7a003-254-sc   3h49m
$ oc get deploy
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
openshift-test   1/1     1            1           3d2h
$ oc set volume deploy/openshift-test --add --name=pvc-openshift-test -t pvc --claim-name=test-
openshift-test     -m /pvcvol/
deployment.apps/openshift-test volume updated
```

Per maggiori informazioni si rimanda alla documentazione ufficiale di Openshift

https://docs.openshift.com/container-platform/4.12/storage/persistent_storage/persistent_storage_local/persistent-storage-local.html

8 Limiti e Quote

1. Visualizzazione dei limiti applicate alle risorse

Per massimizzare l'allocazione delle risorse e garantire stabilità alle applicazioni l'ambiente Openshift prevede per la componente computazionali due Limiti:

- **Request**: Rappresenta la quota di CPU (requests.cpu) e Memoria (requests.memory) richieste per allocazione di un singolo Pod su un determinato nodo Worker
- **Limits**: Rappresenta il limite massimo di CPU (limits.cpu) e Memora (limits.memory) che un Pod può consumare una volta in esecuzione.
    - **Eccessivo consumo di CPU**: Al Pod non viene allocato ulteriore CPU
    - **Eccessivo consumo di Memoria**: Il Pod che lo contiene viene terminato dalla piattaforma

Al fine di applicare delle buone pratiche nell'uso corretto del servizio sono state imposti dei limiti che permettono definire dei valori di minimo, massimo e predefinito, qualora il valore non venga definito. Oltre ad imporre i limiti dimensionali su Pod e relativi Container vengono impostati altri limiti come si evince dalla tabella che segue:

Tipo	Risorsa	Min	Max	Default Request	Default Limit	Rapporto Limit/Request	Note
Pod	cpu	200m	4				
Pod	memory	128Mi	16Gi				
Container	cpu	100m	2	200m	300m	10	Il rapporto masismo tra Limit e request 10:1
Container	memory	64Mi	16Gi	128Mi	200Mi		
openshift.io/Image			2Gi				Limite massimo sulla dimensione della singola immagine all'interno registry interno

openshift.io/ImageStream	openshift.io/image-tags		20				Limite di TAG per singola immagine Container
openshift.io/ImageStream	openshift.io/images		30				Limite massimi di immagini sul registry interno per Project
PersistentVolumeClaim	storage	1Gi					Dimensione minima dei Volumi Persistenti

Tabella 5. Limiti applicativi

Tali ***limit*** vengono impostati in fase di creazione dei singoli Project Openshift; qualora non dovessero risultare compatibili con il dimensionamento applicativo è possibile chiederne l'ampliamento tramite service request, o segnalarlo già in fase d'invio del piano dei fabbisogni.

Inoltre, tali limiti rappresentano il massimale permesso, l'effettiva disponibilità di risorse è regolata dalle quote di servizio, ed in caso la quota disponibile sia minore del massimale prevarrà il valore di quota.

2. Visualizzazione del consumo delle Quote di servizio

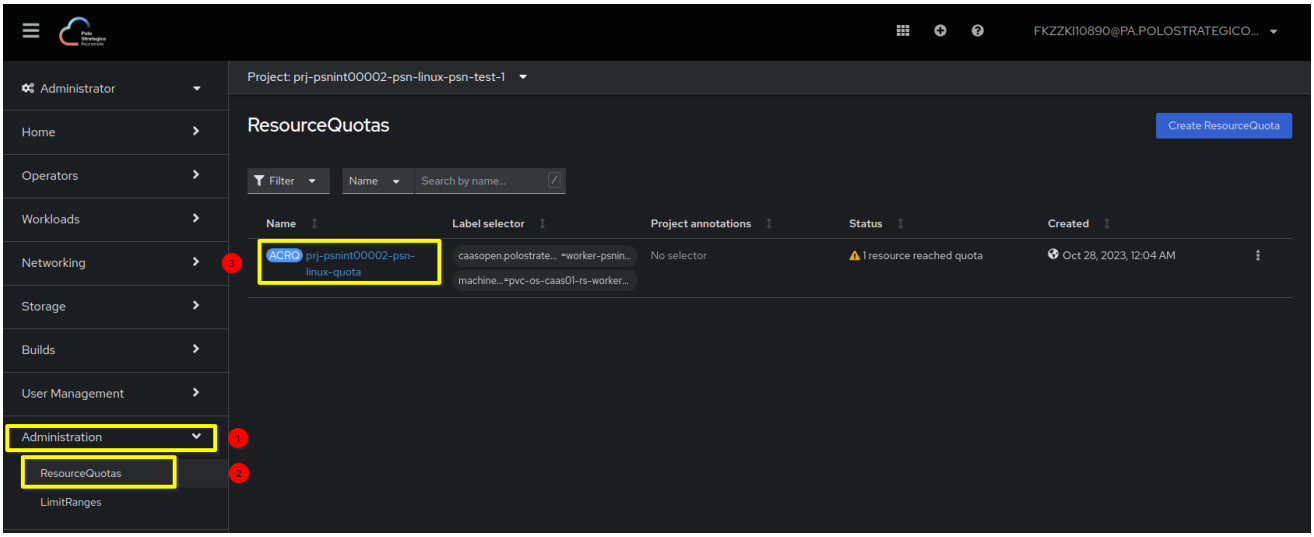


Figura 57. ResourceQuotas

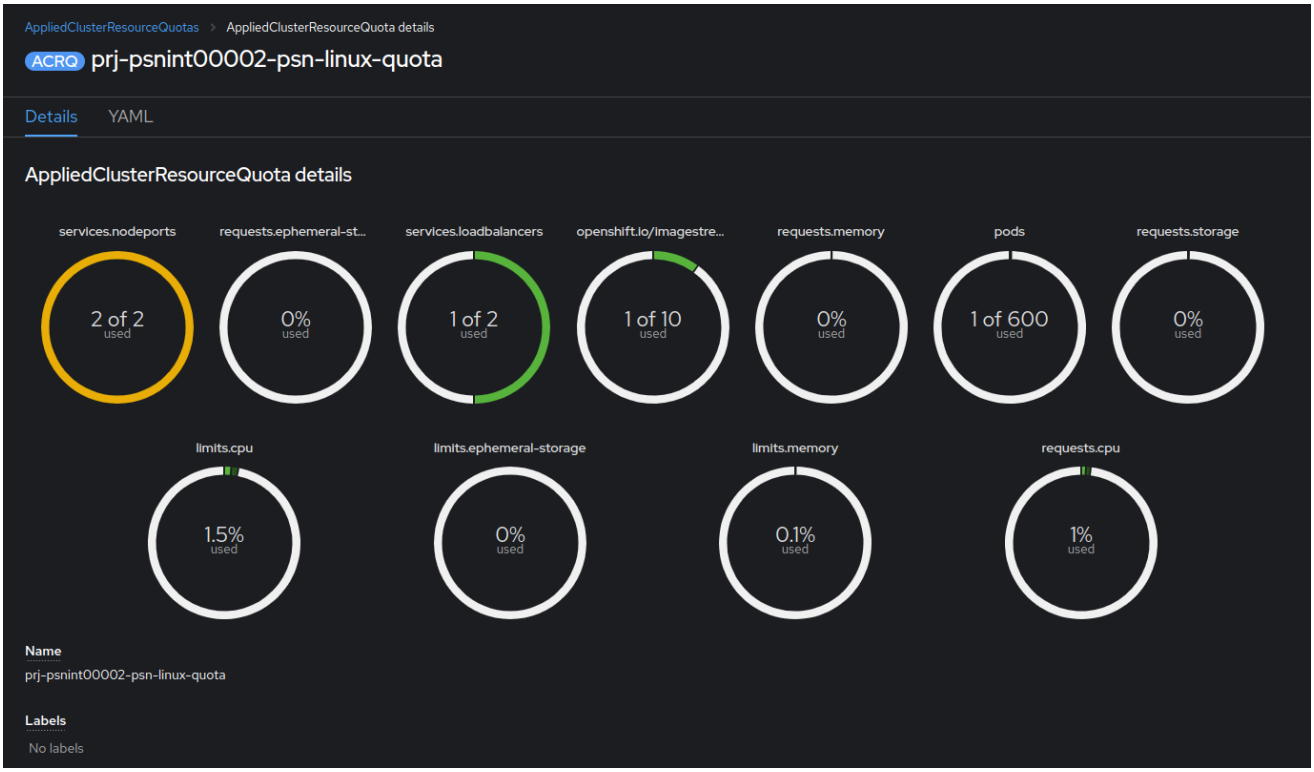
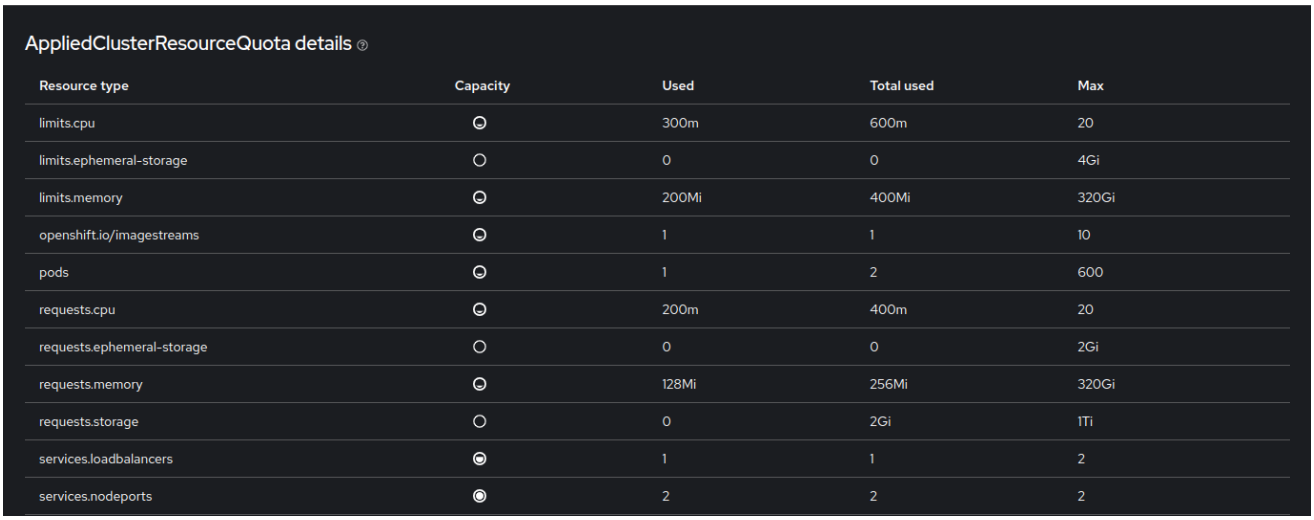


Figura 58. ClusterResourceQuota



AppliedClusterResourceQuota details

Resource type	Capacity	Used	Total used	Max
limits.cpu		300m	600m	20
limits.ephemeral-storage		0	0	4Gi
limits.memory		200Mi	400Mi	320Gi
openshift.io/imagestreams		1	1	10
pods		1	2	600
requests.cpu		200m	400m	20
requests.ephemeral-storage		0	0	2Gi
requests.memory		128Mi	256Mi	320Gi
requests.storage		0	2Gi	1Ti
services.loadbalancers		1	1	2
services.nodeports		2	2	2

Figura 59. Dettaglio Quotas

1. Una volta selezionato “**Administration**” > “**ResourceQuotas”**
2. Selezionare l’opportuno project dal menù modale posto al centro della videata
3. Effettuare lo scroll sulla pagina per vedere il dettaglio numerico per ogni tipologia di quota

Come per gli altri oggetti, è possibile ottenere le stesse informazioni sulla quota tramite il client oc/kubectl con i seguenti comandi:

```
$ oc get appliedclusterresourcequotas
NAME                    LABELS SELECTOR
ANNOTATIONS SELECTOR
prj-psnint00002-psn-linux-quota   caasopen.polostrategiconazionale.it/machineset=worker-psnint00002-psn-linux,machineset_name=pvc-os-caas01-rs-worker-psnint00002-psn-linux   <none>
```

Ed il dettaglio delle stesse con l’ opzione “**describe**” dell’AppliedResourceQuotas sul project:

```
$ oc describe appliedclusterresourcequotas prj-psnint00002-psn-linux-quota
Name:         prj-psnint00002-psn-linux-quota
Created:      6 weeks ago
Labels:       <none>
```

Annotations: <none>
Namespace Selector: ["prj-psnint00002-psn-linux-psn-test-1" "prj-psnint00002-psn-linux-psn-test-2"]
Label Selector: caasopen.polostrategiconazionale.it/machineset=worker-psnint00002-psn-linux,machineset_name=pvc-os-caas01-rs-worker-psnint00002-psn-linux
AnnotationSelector: map[]

Resource	Used	Hard
limits.cpu	600m	20
limits.ephemeral-storage	0	4Gi
limits.memory	400Mi	320Gi
openshift.io/imagestreams	1	10
pods	2	600
requests.cpu	400m	20
requests.ephemeral-storage	0	2Gi
requests.memory	256Mi	320Gi
requests.storage	2Gi	1Ti
services.loadbalancers	1	2
services.nodeports	2	2