Red Hat OpenShift 與NetApp

NetApp container solutions

NetApp
January 21, 2026

目錄

Red Hat OpenShift 與NetApp

NVA-1160：Red Hat OpenShift 與NetApp

NetApp 的Alan Cowles 和 Nikhil M Kulkarni

本參考文件提供了 Red Hat OpenShift 解決方案的部署驗證，該解決方案透過安裝程式設定基礎架構 (IPI) 在多個不同的資料中心環境中部署，並由NetApp進行驗證。它還詳細介紹如何利用Trident儲存編排器管理持久性存儲，從而實現與NetApp儲存系統的儲存整合。最後，我們探索並記錄了大量解決方案驗證和實際用例。

用例

Red Hat OpenShift 與NetApp解決方案的架構旨在透過以下用例為客戶提供卓越的價值：

- 易於部署和管理使用 IPI（安裝程式設定基礎架構）在裸機、Red Hat OpenStack 平台、Red Hat 虛擬化和 VMware vSphere 上部署的 Red Hat OpenShift。
- 將企業容器和虛擬化工作負載的功能與 Red Hat OpenShift 結合起來，在 OSP、RHV 或 vSphere 上虛擬部署，或在具有 OpenShift Virtualization 的裸機上部署。
- 真實世界的配置和用例突顯了 Red Hat OpenShift 與NetApp儲存和Trident（ Kubernetes 的開源儲存編排器）一起使用時的功能。

商業價值

企業越來越多地採用 DevOps 實踐來創建新產品、縮短發布週期並快速添加新功能。由於其固有的敏捷特性，容器和微服務在支援 DevOps 實踐方面發揮著至關重要的作用。然而，在企業環境中以生產規模實踐 DevOps 也面臨著自身的挑戰，並對底層基礎設施提出了一定的要求，例如：

- 堆疊中所有層的高可用性
- 簡化部署程序
- 無中斷運作和升級
- API 驅動且可編程的基礎設施，以跟上微服務的敏捷性
- 具有效能保證的多租戶
- 能夠同時運行虛擬化和容器化工作負載
- 能夠根據工作負載需求獨立擴展基礎設施

NetApp的 Red Hat OpenShift 承認這些挑戰，並提出了一種解決方案，透過在客戶選擇的資料中心環境中實施 RedHat OpenShift IPI 的全自動部署來幫助解決每個問題。

技術概述

Red Hat OpenShift 與NetApp解決方案由以下主要元件組成：

紅帽 **OpenShift** 容器平台

Red Hat OpenShift Container Platform 是一個完全支援的企業級 Kubernetes 平台。 Red Hat 對開源 Kubernetes 進行了多項改進，以提供一個完全整合所有元件的應用程式平台，用於建置、部署和管理容器化應用程式。

欲了解更多信息，請訪問 OpenShift 網站 "這裡"。

NetApp儲存系統

NetApp擁有多種非常適合企業資料中心和混合雲端部署的儲存系統。 NetApp產品組合包括NetApp ONTAP、NetApp Element和NetApp e-Series 儲存系統，所有這些系統都可以為容器化應用程式提供持久性儲存。

欲了解更多信息，請訪問NetApp網站 "這裡"。

NetApp儲存集成

Trident是一個開源且完全支援的容器和 Kubernetes 發行版（包括 Red Hat OpenShift）儲存編排器。

欲了解更多信息，請訪問Trident網站 "這裡"。

進階配置選項

本節專門介紹實際使用者在將此解決方案部署到生產時可能需要執行的自訂，例如建立專用私有映像登錄檔或部署自訂負載平衡器執行個體。

已驗證版本的目前支援矩陣

科技	目的	軟體版本
NetApp ONTAP	儲存	9.8、9.9.1、9.12.1
NetApp Element	儲存	12.3
NetApp Trident	儲存編排	22.01.0、23.04、23.07、23.10、24.02
紅帽 OpenShift	容器編排	4.6 EUS，4.7，4.8，4.10，4.11，4.12，4.13，4.14
VMware vSphere	資料中心虛擬化	7.0、8.0.2

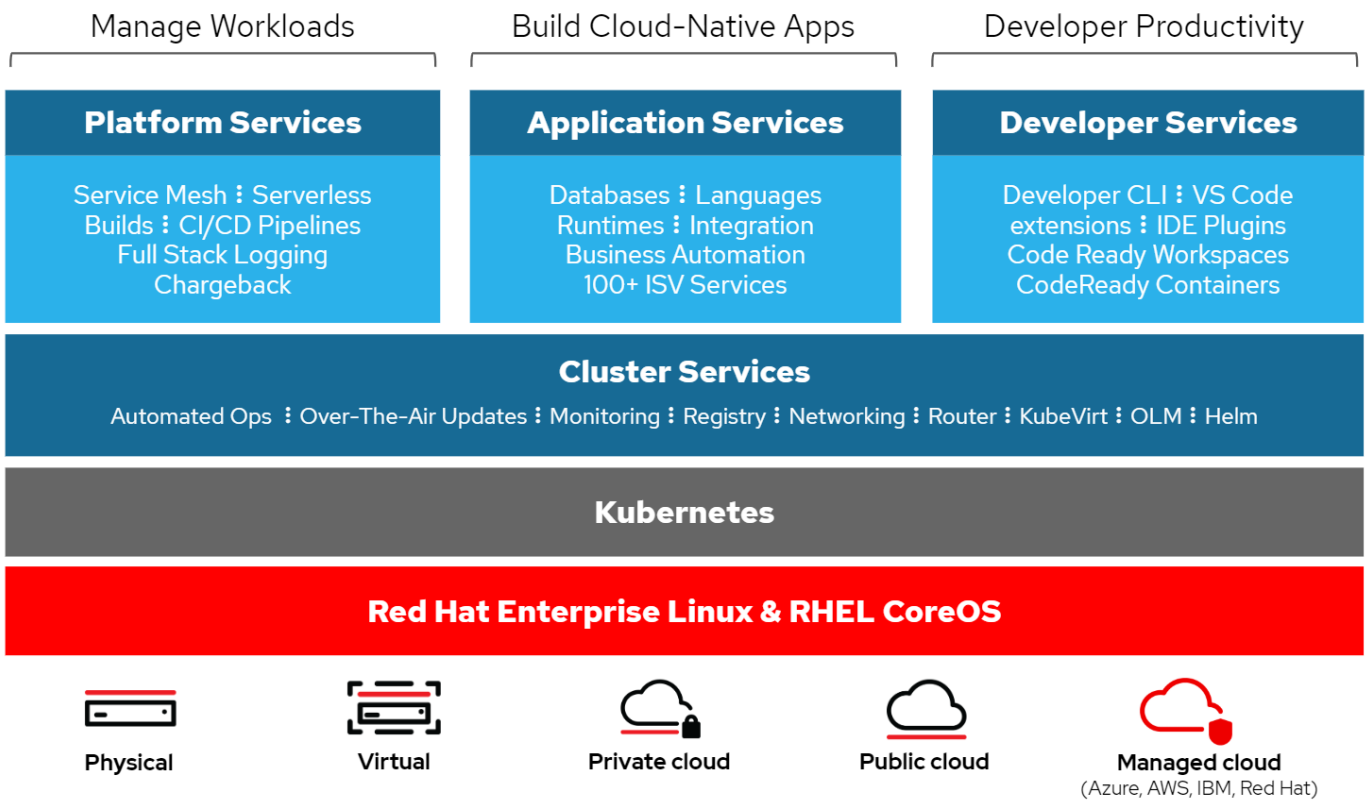
紅帽 **OpenShift**

OpenShift 概述

Red Hat OpenShift 容器平台將開發和 IT 營運整合在一個平台上，以便在本地和混合雲端基礎架構上一致地建置、部署和管理應用程式。 Red Hat OpenShift 基於開源創新和行業標準構建，包括 Kubernetes 和 Red Hat Enterprise Linux CoreOS，後者是專為基於容器的工作負載而設計的全球領先的企業 Linux 發行版。 OpenShift 是雲端原生運算基金會 (CNCF) 認證 Kubernetes 計畫的一部分，提供容器工作負載的可移植性和互通性。

Red Hat OpenShift 提供以下功能：

- 自助服務配置 開發人員可以使用他們最常用的工具快速輕鬆地按需創建應用程序，同時操作保留對整個環境的完全控制。
- 持久性儲存 透過提供對持久性儲存的支持，OpenShift Container Platform 允許您同時執行有狀態應用程式和雲端原生無狀態應用程式。
- 持續整合和持續開發 **(CI/CD)** 此原始碼平台可大規模管理建置和部署映像。
- 開源標準 這些標準除了其他開源技術外，還結合了開放容器計劃 (OCI) 和 Kubernetes 用於容器編排。您不會受到特定供應商的技術或業務路線圖的限制。
- **CI/CD** 管道 OpenShift 為 CI/CD 管道提供開箱即用的支持，以便開發團隊可以自動化應用程式交付過程的每個步驟，並確保在對應用程式的程式碼或配置所做的每個更改上執行它。
- 基於角色的存取控制 **(RBAC)** 此功能提供團隊和使用者跟踪，以幫助組織大型開發人員群體。
- 自動建置和部署 OpenShift 為開發人員提供了建置容器化應用程式的選項，或讓平台從應用程式原始程式碼甚至二進位檔案建置容器。然後，該平台根據為應用程式定義的特性自動在整個基礎架構中部署這些應用程式。例如，為了符合第三方許可證，應該分配多少資源以及在基礎架構的什麼位置部署它們。
- 一致的環境 OpenShift 確保為開發人員提供的環境以及應用程式整個生命週期的環境都是一致的，從作業系統到程式庫、執行時間版本（例如 Java 執行時間），甚至正在使用的應用程式執行階段（例如 tomcat），以消除源自不一致環境的風險。
- 配置管理 平台內建配置和敏感資料管理，以確保無論使用哪種技術建立應用程式或部署在哪種環境中，都為應用程式提供一致且與環境無關的應用程式配置。
- *應用程式日誌和指標。 *快速反饋是應用程式開發的一個重要方面。 OpenShift 整合監控和日誌管理為開發人員提供即時指標，以便他們研究應用程式在變更中的行為，並能夠在應用程式生命週期中儘早解決問題。
- 安全性和容器目錄 OpenShift 提供多租用戶功能，並使用安全增強 Linux (SELinux)、CGroups 和安全計算模式 (seccomp) 建立的安全性來隔離和保護容器，從而保護使用者免受有害程式碼執行的侵害。它還透過 TLS 憑證為各個子系統提供加密，並提供對 Red Hat 認證容器（access.redhat.com/containers）的訪問，這些容器經過掃描和分級，特別強調安全性，以便為最終用戶提供經過認證、值得信賴和安全的應用程式容器。



Red Hat OpenShift 的部署方法

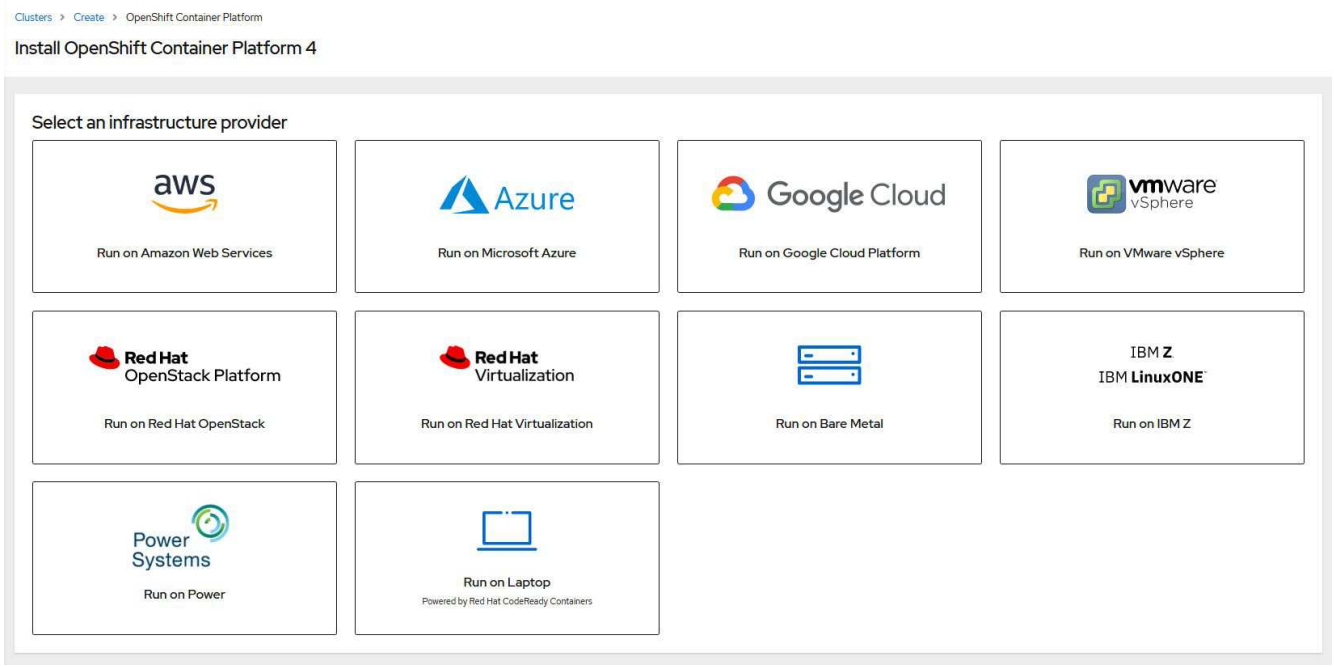
從 Red Hat OpenShift 4 開始，OpenShift 的部署方法包括使用使用者設定基礎架構 (UPI) 進行高度客製化的部署的手動部署或使用安裝程式設定基礎架構 (IPI) 進行全自動部署。

在大多數情況下，IPI 安裝方法是首選方法，因為它允許為開發、測試和生產環境快速部署 OpenShift 叢集。

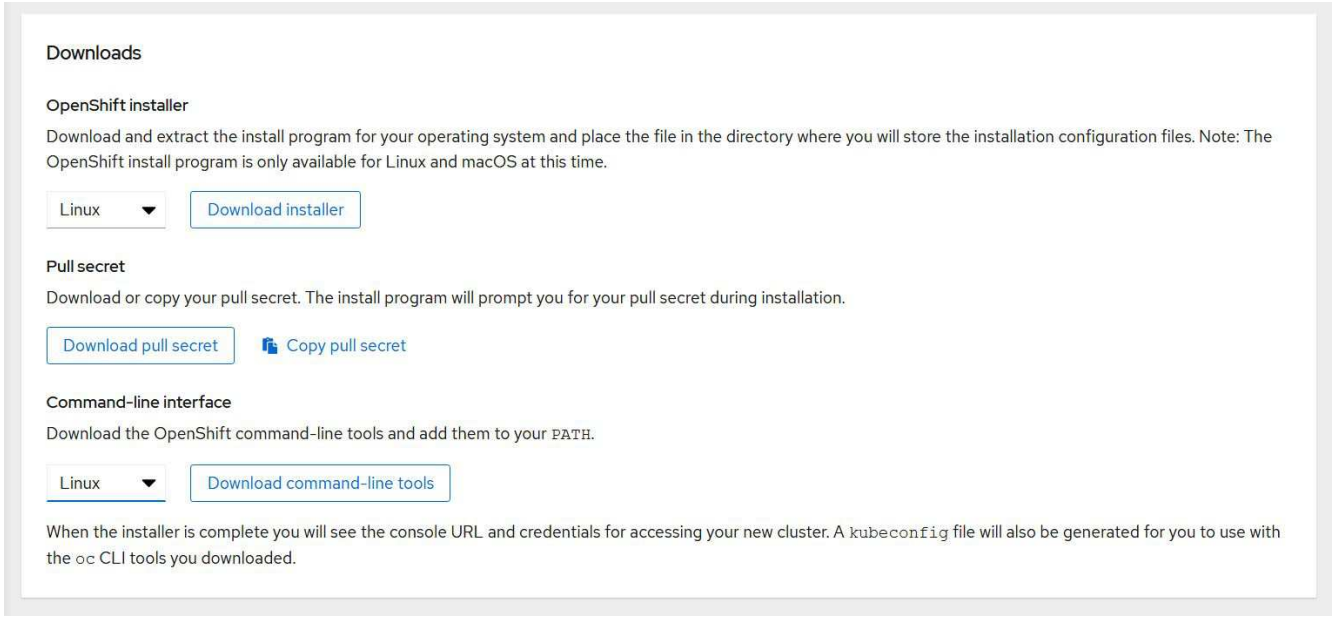
Red Hat OpenShift 的 IPI 安裝

OpenShift 的安裝程式設定基礎架構 (IPI) 部署涉及以下進階步驟：

1. 造訪 Red Hat OpenShift"網站"並使用您的 SSO 憑證登入。
2. 選擇您想要部署 Red Hat OpenShift 的環境。



3. 在下一個畫面下載安裝程式、唯一的拉取金鑰和用於管理的 CLI 工具。



4. 關注"安裝說明"由 Red Hat 提供，可部署到您選擇的環境。

NetApp驗證的 **OpenShift** 部署

NetApp已在其實驗室中使用安裝程式設定基礎架構 (IPI) 部署方法在以下每個資料中心環境中測試並驗證了 Red Hat OpenShift 的部署：

- "裸機上的 OpenShift"
- "Red Hat OpenStack 平台上的 OpenShift"
- "Red Hat 虛擬化上的 OpenShift"
- "VMware vSphere 上的 OpenShift"

裸機上的 **OpenShift**

OpenShift on Bare Metal 可在商用伺服器上自動部署 OpenShift 容器平台。

OpenShift on Bare Metal 類似於 OpenShift 的虛擬部署，它可以輕鬆部署、快速配置和擴展 OpenShift 集群，同時支援尚未準備好容器化的應用程式的虛擬化工作負載。透過在裸機上部署，您不需要除了 OpenShift 環境之外管理主機虛擬機器管理程式環境所需的額外開銷。透過直接在裸機伺服器上部署，您還可以減少主機和 OpenShift 環境之間共用資源的實體開銷限制。

OpenShift on Bare Metal 提供以下功能：

- **IPI** 或輔助安裝程式部署 透過在裸機伺服器上由安裝程式配置基礎架構 (IPI) 部署的 OpenShift 集群，客戶可以直接在商用伺服器上部署高度通用、易於擴展的 OpenShift 環境，而無需管理虛擬機器管理程式層。
- 緊湊的叢集設計 為了最大限度地減少硬體需求，裸機上的 OpenShift 允許使用者部署僅 3 個節點的集群，透過使 OpenShift 控制平面節點也可以充當工作節點和主機容器。
- **OpenShift** 虛擬化 OpenShift 可以使用 OpenShift 虛擬化在容器內執行虛擬機器。此容器原生虛擬化在容器內執行 KVM 虛擬機器管理程序，並為 VM 儲存附加持久性磁碟區。
- **AI/ML** 最佳化基礎架構 透過將基於 GPU 的工作節點合併到您的 OpenShift 環境並利用 OpenShift Advanced Scheduling，為機器學習應用程式部署 Kubeflow 等應用程式。

網路設計

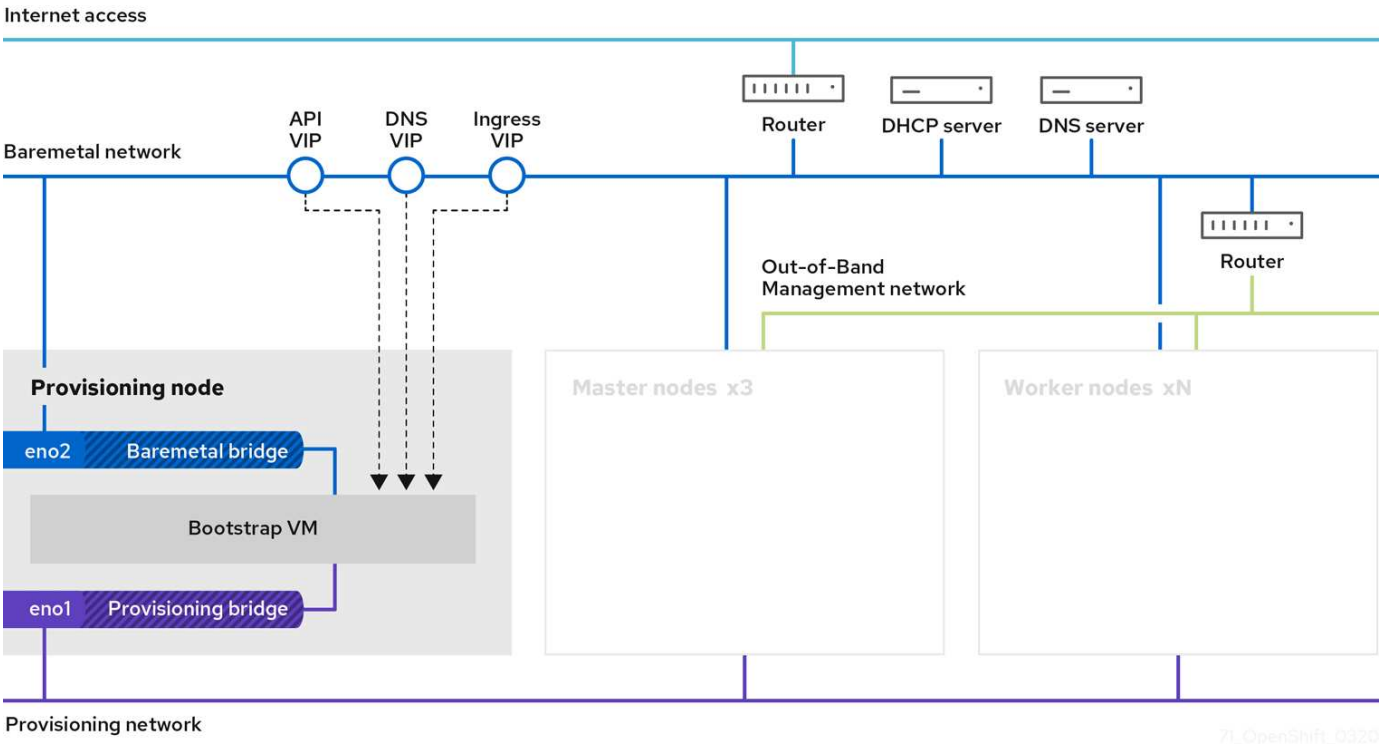
NetApp解決方案上的 Red Hat OpenShift 使用兩台資料交換器以 25Gbps 提供主要資料連線。它還使用兩個管理交換機，以 1Gbps 的速度提供儲存節點的帶內管理連接以及 IPMI 功能的帶外管理連接。

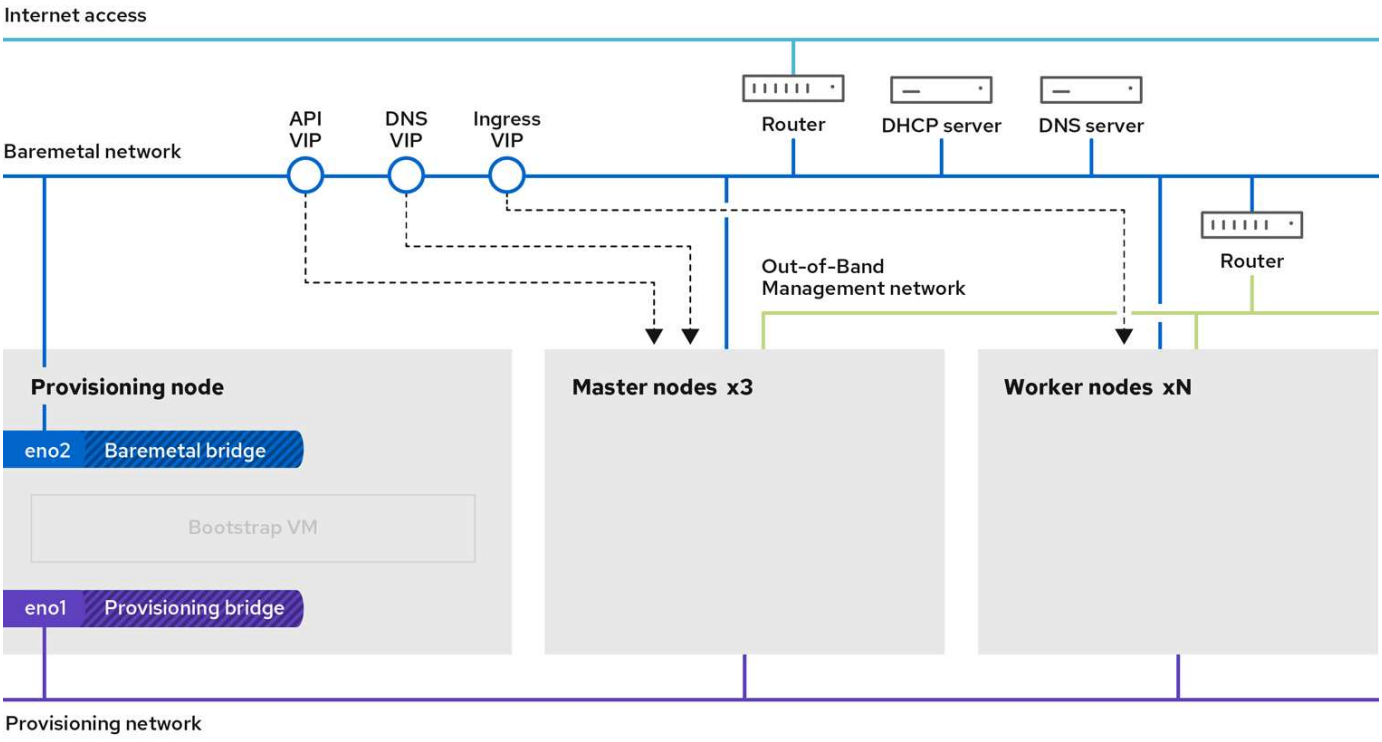
對於 OpenShift 裸機 IPI 部署，您必須建立一個設定節點，也就是一台必須具有連接到單獨網路的網路介面的 Red Hat Enterprise Linux 8 機器。

- 設定網路 此網路用於啟動裸機節點並安裝部署 OpenShift 叢集所需的映像和套件。
- 裸機網路 此網路用於集群部署後面向公眾的通訊。

為了設定配置節點，客戶建立橋接接口，允許流量在節點本身和為部署目的而配置的 Bootstrap VM 上正確路由。叢集部署完成後，API 和 ingress VIP 位址從 bootstrap 節點遷移到新部署的叢集。

下圖描述了 IPI 部署期間和部署完成後的環境。





VLAN 需求

Red Hat OpenShift 與NetApp解決方案旨在透過使用虛擬區域網路 (VLAN) 在邏輯上分離用於不同用途的網路流量。

VLAN	目的	VLAN ID
帶外管理網絡	裸機節點與 IPMI 管理	16
裸機網絡	叢集可用時，OpenShift 服務的網絡	181
設定網路	透過 IPI 進行 PXE 啟動和安裝裸機節點的網絡	3485

儘管這些網路實際上是由 VLAN 分隔的，但每個實體連接埠都必須設定為存取模式並指派主 VLAN，因為在 PXE 啟動序列期間無法傳遞 VLAN 標籤。

網路基礎設施支援資源

在部署 OpenShift 容器平台之前，應具備以下基礎架構：

- 至少有一個 DNS 伺服器提供可從帶內管理網路和 VM 網路存取的完整主機名稱解析。
- 至少一個可從帶內管理網路和 VM 網路存取的 NTP 伺服器。
- （選購）內管理網路和 VM 網路的出站網際網路連線。

Red Hat OpenStack 平台上的 OpenShift

Red Hat OpenStack 平台提供了一個整合的基礎來創建、部署和擴展安全可靠的私有 OpenStack 雲端。

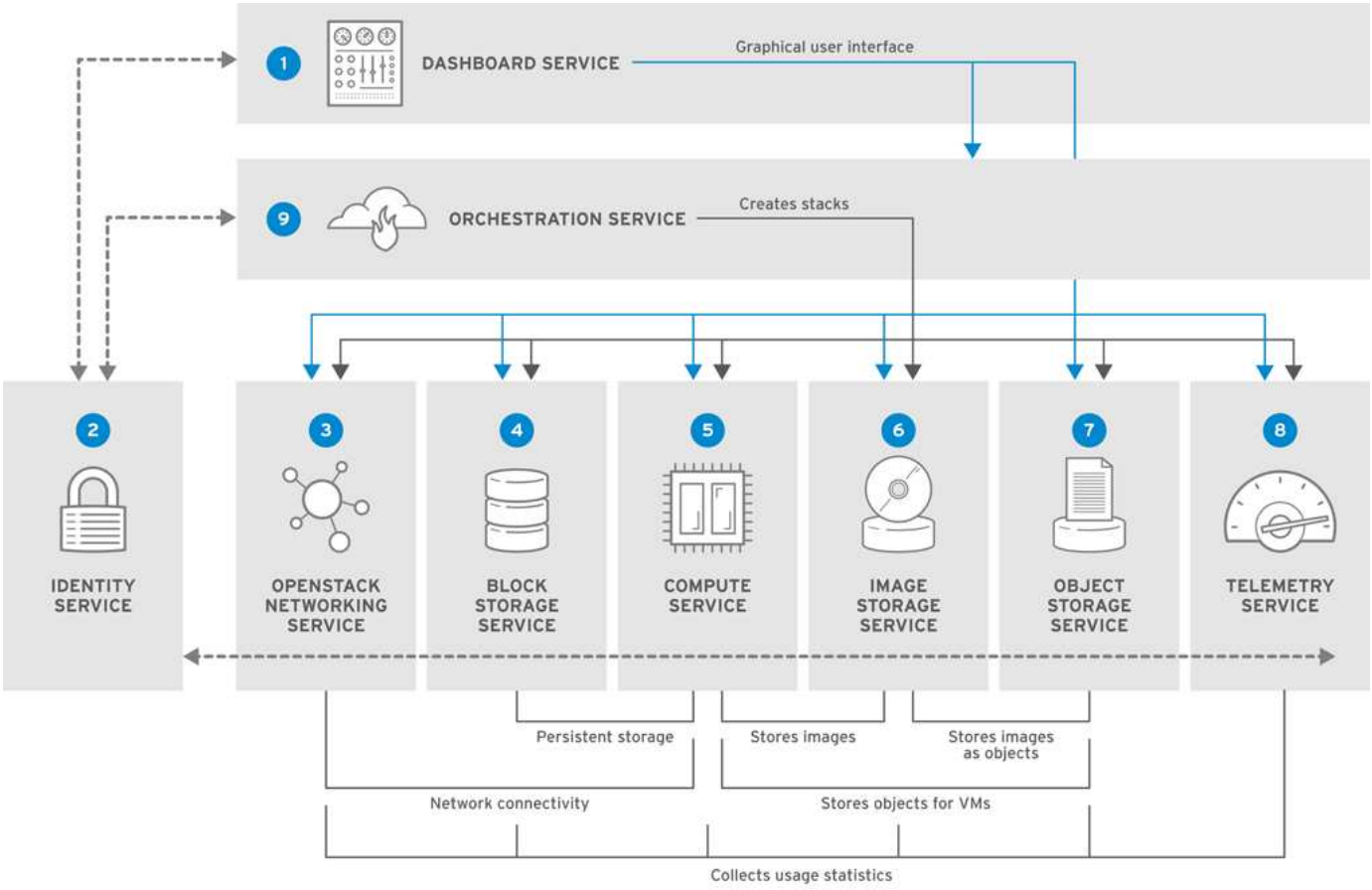
OSP 是一種基礎設施即服務 (IaaS) 雲，由管理運算、儲存和網路資源的一系列控制服務實現。此環境使用基於 Web 的介面進行管理，允許管理員和使用者控制、配置和自動化 OpenStack 資源。此外，OpenStack 基礎架構透過廣泛的命令列介面和 API 實現，為管理員和最終用戶提供完全自動化功能。

OpenStack專案是一個快速發展的社群項目，每六個月提供一次更新版本。最初，Red Hat OpenStack Platform 透過隨每個上游版本發布新版本並為每三個版本提供長期支援來跟上此發布週期。最近，隨著 OSP 16.0 版本（基於 OpenStack Train）的發布，Red Hat 選擇不跟上版本號，而是將新功能反向移植到子版本中。最新版本是 Red Hat OpenStack Platform 16.1，其中包括從 Ussuri 和 Victoria 上游版本移植的高級功能。

有關 OSP 的更多信息，請參閱"紅帽 OpenStack 平台網站"。

OpenStack服務

OpenStack 平台服務以容器形式部署，從而將服務彼此隔離並實現輕鬆升級。 OpenStack 平台使用一組透過 Kolla 建置和管理的容器。服務的部署是透過從 Red Hat Custom Portal 提取容器映像來執行的。這些服務容器使用 Podman 指令進行管理，並使用 Red Hat OpenStack Director 進行部署、設定和維護。



服務	項目名稱	描述
儀表板	地平線	用於管理 OpenStack 服務的基於 Web 瀏覽器的儀表板。
身分	Keystone	用於對 OpenStack 服務進行身份驗證和授權以及管理使用者、專案和角色的集中服務。
OpenStack網路	中子	提供 OpenStack 服務介面之間的連線。
區塊儲存	煤渣	管理虛擬機器 (VM) 的持久性區塊儲存磁碟區。
運算	新星	管理和配置在計算節點上運行的虛擬機器。
影像	一瞥	用於儲存虛擬機器鏡像、磁碟區快照等資源的註冊服務。
物件儲存	迅速	允許使用者儲存和檢索文件和任意資料。
遙測	雲高儀	提供雲端資源使用情況的測量。
編排	熱	基於模板的編排引擎，支援自動建立資源堆疊。

網路設計

Red Hat OpenShift 與NetApp解決方案使用兩台資料交換器以 25Gbps 提供主要資料連線。它還使用兩個額外的管理交換機，以 1Gbps 的速度提供儲存節點的帶內管理連接以及 IPMI 功能的帶外管理連接。

Red Hat OpenStack Director 需要 IPMI 功能來使用 Ironic 裸機設定服務部署 Red Hat OpenStack Platform。

VLAN 需求

Red Hat OpenShift 與NetApp旨在透過使用虛擬區域網路 (VLAN) 在邏輯上分離用於不同用途的網路流量。此配置可以擴展以滿足客戶需求或為特定網路服務提供進一步的隔離。下表列出了在NetApp驗證解決方案時實施該解決方案所需的 VLAN。

VLAN	目的	VLAN ID
帶外管理網絡	用於管理 Ironic 的實體節點和 IPMI 服務的網路。	16
儲存基礎設施	用於控制器節點直接對應磁碟區以支援 Swift 等基礎架構服務的網路。	201
儲存煤渣	用於將區塊磁碟區直接對應並附加到環境中部署的虛擬執行個體的網路。	202
內部 API	用於使用 API 通訊、RPC 訊息和資料庫通訊的 OpenStack 服務之間的通訊的網路。	301
租戶	Neutron 透過 VXLAN 隧道為每個租戶提供自己的網路。網路流量在每個租戶網路內是隔離的。每個租用戶網路都有一個與之關聯的 IP 子網，網路命名空間意味著多個租用戶網路可以使用相同的位址範圍而不會造成衝突。	302
儲存管理	OpenStack 物件儲存（Swift）使用此網路在參與的副本節點之間同步資料物件。代理服務充當使用者請求和底層儲存層之間的中介介面。代理接收傳入的請求並找到必要的副本以檢索請求的資料。	303
PXE	OpenStack Director 提供 PXE 啟動作為 Ironic 裸機設定服務的一部分，以協調 OSP Overcloud 的安裝。	3484
外部的	公共可用網絡，託管用於圖形管理的 OpenStack 儀表板（Horizon），並允許公共 API 呼叫來管理 OpenStack 服務。	3485
帶內管理網絡	提供對系統管理功能（例如 SSH 存取、DNS 流量和網路時間協定 (NTP) 流量）的存取。該網路還充當非控制器節點的網關。	3486

網路基礎設施支援資源

在部署 OpenShift 容器平台之前，應具備以下基礎架構：

- 至少一個提供完整主機名稱解析的 DNS 伺服器。
- 至少有三個NTP伺服器可以為解決方案中的伺服器保持時間同步。
- （可選）OpenShift 環境的出站網路連線。

生產部署的最佳實踐

本節列出了組織在將此解決方案部署到生產中之前應考慮的幾種最佳實踐。

將 OpenShift 部署到至少三個運算節點的 OSP 私有雲

本文檔中所述的經過驗證的架構透過部署三個 OSP 控制節點和兩個 OSP 運算節點，提供了適合 HA 操作的最小硬體部署。此架構確保了容錯配置，其中兩個運算節點都可以啟動虛擬實例，並且部署的虛擬機器可以在兩個虛擬機器管理程式之間遷移。

由於 Red Hat OpenShift 最初部署了三個主節點，因此雙節點配置可能會導致至少兩個主節點佔用同一個節點，如果該特定節點不可用，則可能導致 OpenShift 中斷。因此，Red Hat 的最佳實踐是部署至少三個 OSP 運算節點，以便 OpenShift 主機可以均勻分佈，並且解決方案可以獲得額外的容錯程度。

配置虛擬機器/主機親和性

透過啟用 VM/主機親和性，可以將 OpenShift 主機分散在多個虛擬機器管理程式節點上。

親和性是一種為虛擬機器和/或主機集合定義規則的方法，用於確定虛擬機器是否在同一主機或群組中的主機上一起執行，還是在不同的主機上運行。它透過建立由具有一組相同參數和條件的虛擬機器和/或主機組成的親和性群組應用於虛擬機器。根據親和性群組中的虛擬機器是運行在群組內的同一主機上還是運行在群組內的多個主機上，還是分別運行在不同的主機上，親和性群組的參數可以定義正親和性或負親和性。在 Red Hat OpenStack Platform 中，可以透過建立伺服器群組和設定篩選器來建立和強制執行主機親和性和反親和性規則，以便 Nova 在伺服器群組中部署的執行個體部署在不同的運算節點上。

一個伺服器群組最多可以管理 10 個虛擬實例。可以透過更新 Nova 的預設配額來修改這一點。

> OSP 伺服器群組存在特定的硬親和性/反親和性限制；如果沒有足夠的資源部署在單獨的節點上或沒有足夠的資源允許共享節點，則虛擬機器將無法啟動。

若要設定關聯群組，請參閱"如何為 OpenStack 實例配置親和性和反親和性？"。

使用自訂安裝檔進行 **OpenShift** 部署

IPI 透過本文檔前面討論的互動式精靈使 OpenShift 叢集的部署變得簡單。但是，您可能需要在叢集部署過程中變更一些預設值。

在這種情況下，您可以執行精靈並執行任務，而無需立即部署叢集；它會建立一個設定文件，稍後可以從中部署叢集。如果您需要更改任何 IPI 預設值，或者如果您想在環境中部署多個相同的叢集以用於多租戶等其他用途，這將非常有用。有關建立 OpenShift 自訂安裝配置的更多信息，請參閱 "Red Hat OpenShift 在 OpenStack 上安裝自訂集群"。

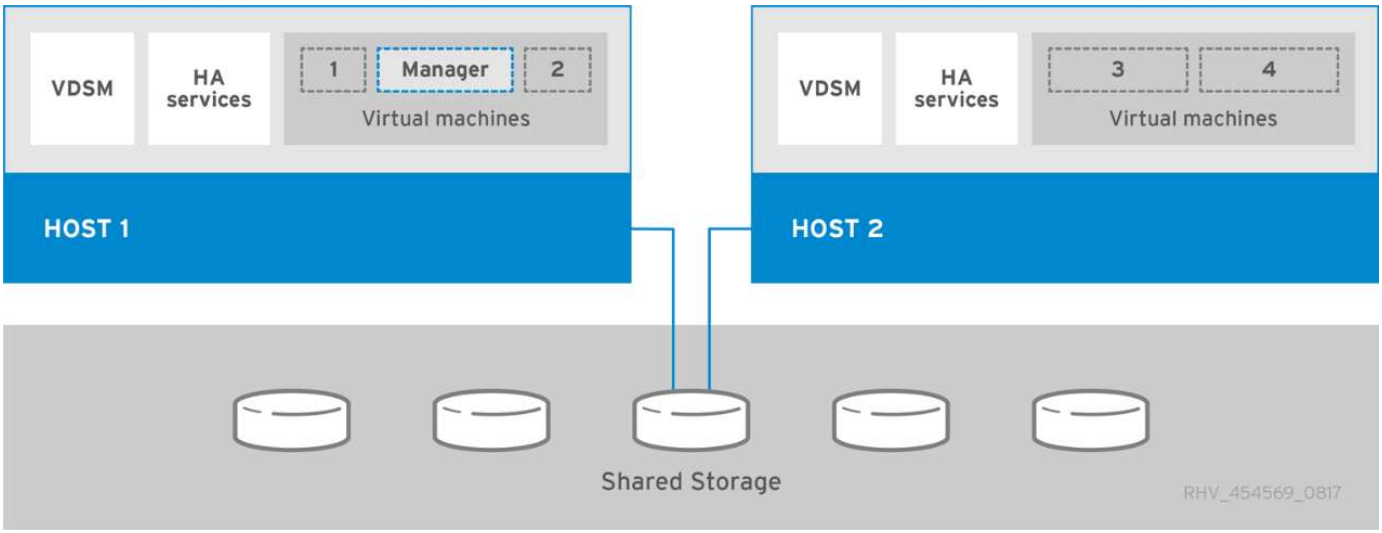
Red Hat 虛擬化上的 **OpenShift**

Red Hat Virtualization (RHV) 是一個企業虛擬資料中心平台，運行在 Red Hat Enterprise Linux (RHEL) 上並使用 KVM 虛擬機器管理程式。

有關 RHV 的更多信息，請參閱"Red Hat 虛擬化網站"。

RHV 提供以下功能：

- 虛擬機器和主機的集中管理 RHV 管理器在部署中作為實體機或虛擬機器 (VM) 運行，並提供基於 Web 的 GUI，以便從中央介面管理解決方案。
- 自託管引擎 為了最大限度地減少硬體需求，RHV 允許將 RHV 管理器 (RHV-M) 作為 VM 部署在運行來賓 VM 的同一主機上。
- 高可用性 為避免主機故障時造成中斷，RHV 允許對虛擬機器進行高可用性配置。高可用性虛擬機器在叢集層級使用彈性策略進行控制。
- 高可擴展性 單一 RHV 叢集最多可擁有 200 個虛擬機器管理程式主機，使其能夠支援大量虛擬機器託管資源密集型企業級工作負載的需求。
- 增強的安全性 從 RHV 繼承而來，RHV 採用安全虛擬化 (sVirt) 和安全增強 Linux (SELinux) 技術來提高主機和虛擬機器的安全性和強化程度。這些功能的主要優勢是虛擬機器及其相關資源的邏輯隔離。



網路設計

NetApp解決方案上的 Red Hat OpenShift 使用兩台資料交換器以 25Gbps 提供主要資料連線。它還使用兩個額外的管理交換機，以 1Gbps 的速度提供儲存節點的帶內管理連接以及 IPMI 功能的帶外管理。 OCP使用RHV上的虛擬機器邏輯網路進行叢集管理。本節介紹解決方案中使用的每個虛擬網段的安排和用途，並概述部署解決方案的先決條件。

VLAN 需求

RHV 上的 Red Hat OpenShift 旨在透過使用虛擬區域網路 (VLAN) 在邏輯上分離用於不同目的的網路流量。此配置可以擴展以滿足客戶需求或為特定網路服務提供進一步的隔離。下表列出了在NetApp驗證解決方案時實施該解決方案所需的 VLAN。

VLAN	目的	VLAN ID
帶外管理網絡	實體節點和IPMI的管理	16
虛擬機器網絡	虛擬訪客網路訪問	1172
帶內管理網絡	RHV-H 節點、RHV-Manager 和 ovirtmgmt 網路的管理	3343
儲存網路	NetApp Element iSCSI 的儲存網絡	3344
移民網絡	虛擬客戶遷移網絡	3345

網路基礎設施支援資源

在部署 OpenShift 容器平台之前，應具備以下基礎架構：

- 至少有一個 DNS 伺服器提供完整的主機名稱解析，可從帶內管理網路和 VM 網路存取。
- 至少一個可從帶內管理網路和 VM 網路存取的 NTP 伺服器。
- （選購）內管理網路和 VM 網路的出站網際網路連線。

生產部署的最佳實踐

本節列出了組織在將此解決方案部署到生產中之前應考慮的幾種最佳實踐。

將 **OpenShift** 部署到至少三個節點的 **RHV** 集群

本文檔中所述的經過驗證的架構透過部署兩個 RHV-H 虛擬機器管理程式節點並確保容錯配置（其中兩個主機都可以管理託管引擎並且部署的虛擬機器可以在兩個虛擬機器管理程式之間遷移）提供了適合 HA 操作的最小硬體部署。

由於 Red Hat OpenShift 最初部署有三個主節點，因此在雙節點配置中可以確保至少兩個主節點佔用同一個節點，如果該特定節點不可用，則可能導致 OpenShift 中斷。因此，Red Hat 的最佳實踐是將至少三個 RHV-H 虛擬機器管理程式節點作為解決方案的一部分進行部署，以便 OpenShift 主機可以均勻分佈，並且解決方案可以獲得額外的容錯程度。

配置虛擬機器/主機親和性

您可以透過啟用 VM/主機親和性將 OpenShift 主機分散在多個虛擬機器管理程式節點上。

親和性是一種為虛擬機器和/或主機集合定義規則的方法，用於確定虛擬機器是否在同一主機或群組中的主機上一起執行，還是在不同的主機上運行。它透過建立由具有一組相同參數和條件的虛擬機器和/或主機組成的親和性群組應用於虛擬機器。根據親和性群組中的虛擬機器是運行在群組內的同一主機上還是運行在群組內的多個主機上，還是分別運行在不同的主機上，親和性群組的參數可以定義正親和性或負親和性。

參數定義的條件可以是硬執行，也可以是軟執行。硬執行確保親和性群組中的虛擬機器始終嚴格遵循正親和性或負親和性，而不考慮任何外部條件。軟執行確保為親和性群組中的虛擬機器設定更高的優先級，以便在可行的情況下遵循正或負親和性。在本文檔所描述的兩個或三個虛擬機器管理程式配置中，軟親和性是建議的設定。在較大的叢集中，硬親和性可以正確分配 OpenShift 節點。

若要設定關聯群組，請參閱"紅帽 6.11。親和性群組文檔"。

使用自訂安裝檔進行 **OpenShift** 部署

IPI 透過本文檔前面討論的互動式精靈使 OpenShift 叢集的部署變得簡單。但是，作為叢集部署的一部分，可能需要更改一些預設值。

在這些情況下，您可以執行並執行精靈，而無需立即部署叢集。而是創建一個配置文件，以便以後可以部署叢集。如果您想要變更任何 IPI 預設值，或想要在您的環境中部署多個相同的叢集用於其他用途（例如多租用戶），這將非常有用。有關為 OpenShift 建立自訂安裝配置的更多信息，請參閱"Red Hat OpenShift 在 RHV 上安裝具有自訂的集群"。

VMware vSphere 上的 OpenShift

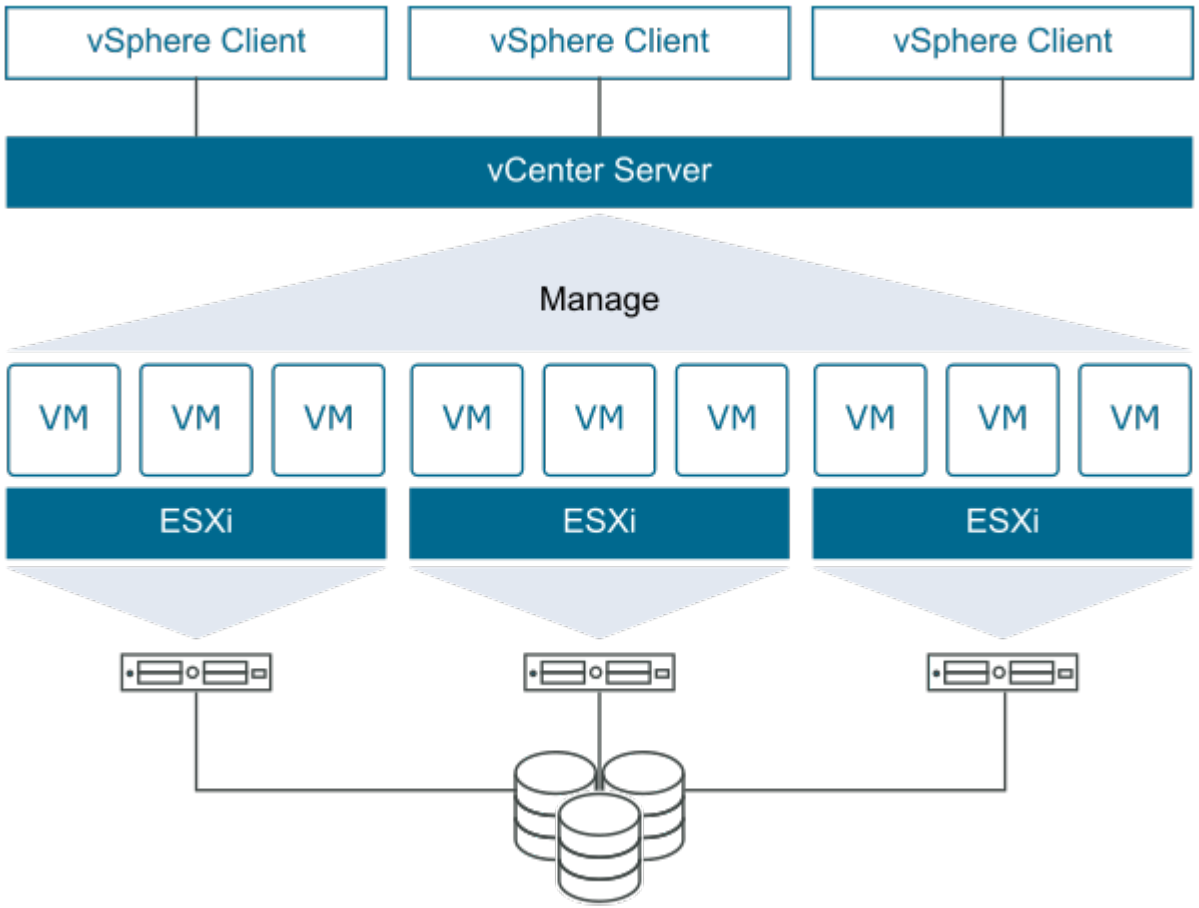
VMware vSphere 是一個虛擬化平台，用於集中管理在 ESXi 虛擬機器管理程式上運行的大量虛擬化伺服器和網路。

有關 VMware vSphere 的更多信息，請參見"VMware vSphere 網站"。

VMware vSphere 提供以下功能：

- **VMware vCenter Server** VMware vCenter Server 從單一控制台統一管理所有主機和虛擬機，並彙總叢集、主機和虛擬機的效能監控。
- **VMware vSphere vMotion** VMware vCenter 可讓您以無中斷的方式根據請求在叢集中的節點之間熱遷移虛擬機器。
- **vSphere** 高可用性 為避免主機發生故障時造成中斷，VMware vSphere 允許將主機叢集化並配置為高可用性。因主機故障而中斷的虛擬機器將在叢集中的其他主機上短暫重新啟動，從而恢復服務。

- 分散式資源調度程式 **(DRS)** 可以設定 VMware vSphere 叢集以平衡其託管的虛擬機器的資源需求。存在資源爭用的虛擬機器可以熱遷移到叢集中的其他節點，以確保有足夠的資源可用。



網路設計

NetApp解決方案上的 Red Hat OpenShift 使用兩台資料交換器以 25Gbps 提供主要資料連線。它還使用兩個額外的管理交換機，以 1Gbps 的速度提供儲存節點的帶內管理連接以及 IPMI 功能的帶外管理連接。 OCP 使用 VMware vSphere 上的 VM 邏輯網路進行叢集管理。本節描述了解決方案中使用的每個虛擬網路段的安排和用途，並概述了部署解決方案的先決條件。

VLAN 需求

VMware vSphere 上的 Red Hat OpenShift 旨在透過使用虛擬區域網路 (VLAN) 在邏輯上分離用於不同目的的網路流量。此配置可以擴展以滿足客戶需求或為特定網路服務提供進一步的隔離。下表列出了在NetApp驗證解決方案時實施該解決方案所需的 VLAN。

VLAN	目的	VLAN ID
帶外管理網絡	實體節點和IPMI的管理	16
虛擬機器網絡	虛擬訪客網路訪問	181
儲存網路	ONTAP NFS 的儲存網絡	184
儲存網路	ONTAP iSCSI 的儲存網絡	185
帶內管理網絡	ESXi 節點、VCenter Server、 ONTAP Select的管理	3480

VLAN	目的	VLAN ID
儲存網路	NetApp Element iSCSI 的儲存網絡	3481
移民網絡	虛擬客戶遷移網絡	3482

網路基礎設施支援資源

在部署 OpenShift 容器平台之前，應具備以下基礎架構：

- 至少有一個 DNS 伺服器提供完整的主機名稱解析，可從帶內管理網路和 VM 網路存取。
- 至少一個可從帶內管理網路和 VM 網路存取的 NTP 伺服器。
- （選購）內管理網路和 VM 網路的出站網際網路連線。

生產部署的最佳實踐

本節列出了組織在將此解決方案部署到生產中之前應考慮的幾種最佳實踐。

將 **OpenShift** 部署到至少三個節點的 **ESXi** 集群

本文檔中所述的經過驗證的架構透過部署兩個 ESXi 虛擬機器管理程式節點並透過啟用 VMware vSphere HA 和 VMware vMotion 確保容錯配置，提供了適合 HA 操作的最低硬體部署。此配置允許已部署的虛擬機器在兩個虛擬機器管理程式之間遷移，並在一個主機不可用時重新啟動。

由於 Red Hat OpenShift 最初部署有三個主節點，因此在某些情況下，雙節點配置中至少有兩個主節點可以佔用同一個節點，如果該特定節點不可用，則可能導致 OpenShift 中斷。因此，Red Hat 的最佳實踐是必須部署至少三個 ESXi 虛擬機器管理程式節點，以便 OpenShift 主機可以均勻分佈，從而提供額外的容錯程度。

配置虛擬機器和主機關聯性

透過啟用 VM 和主機親和性，可以確保 OpenShift 主機分佈在多個虛擬機器管理程式節點上。

親和性或反親和性是一種為虛擬機器和/或主機集定義規則的方法，用於確定虛擬機器是否在同一主機或群組中的主機上一起執行，還是在不同的主機上運行。它透過建立由具有一組相同參數和條件的虛擬機器和/或主機組成的親和性群組應用於虛擬機器。根據親和性群組中的虛擬機器是運行在群組內的同一主機上還是運行在群組內的多個主機上，還是分別運行在不同的主機上，親和性群組的參數可以定義正親和性或負親和性。

若要配置親緣組，請參閱 "vSphere 9.0 文件：使用 DRS 關聯性規則"。

使用自訂安裝檔進行 **OpenShift** 部署

IPI 透過本文檔前面討論的互動式精靈使 OpenShift 叢集的部署變得簡單。但是，您可能需要在叢集部署過程中變更一些預設值。

在這些情況下，您可以執行精靈並執行任務而不立即部署集群，而是精靈會建立一個設定文件，稍後可以從中部署集群。如果您需要更改任何 IPI 預設值，或者想要在您的環境中部署多個相同的叢集用於其他用途（例如多租戶），這將非常有用。有關為 OpenShift 建立自訂安裝配置的更多信息，請參閱"Red Hat OpenShift 在 vSphere 上安裝具有自訂設定的集群"。

AWS 上的 Red Hat OpenShift 服務

AWS 上的 Red Hat OpenShift 服務 (ROSA) 是一項託管服務，您可以使用它在 AWS 上透

過 Red Hat OpenShift 企業 Kubernetes 平台建置、擴充和部署容器化應用程式。 ROSA 簡化了將本機 Red Hat OpenShift 工作負載遷移到 AWS 的過程，並提供了與其他 AWS 服務的緊密整合。

有關 ROSA 的更多信息，請參閱此處的文檔："AWS 上的 Red Hat OpenShift 服務（AWS 文件）"。"AWS 上的 Red Hat OpenShift 服務（Red Hat 文件）"。

NetApp儲存系統

NetApp ONTAP

NetApp ONTAP是一款功能強大的儲存軟體工具，具有直覺的 GUI、具有自動化整合的 REST API、基於 AI 的預測分析和糾正措施、無中斷硬體升級以及跨儲存導入等功能。

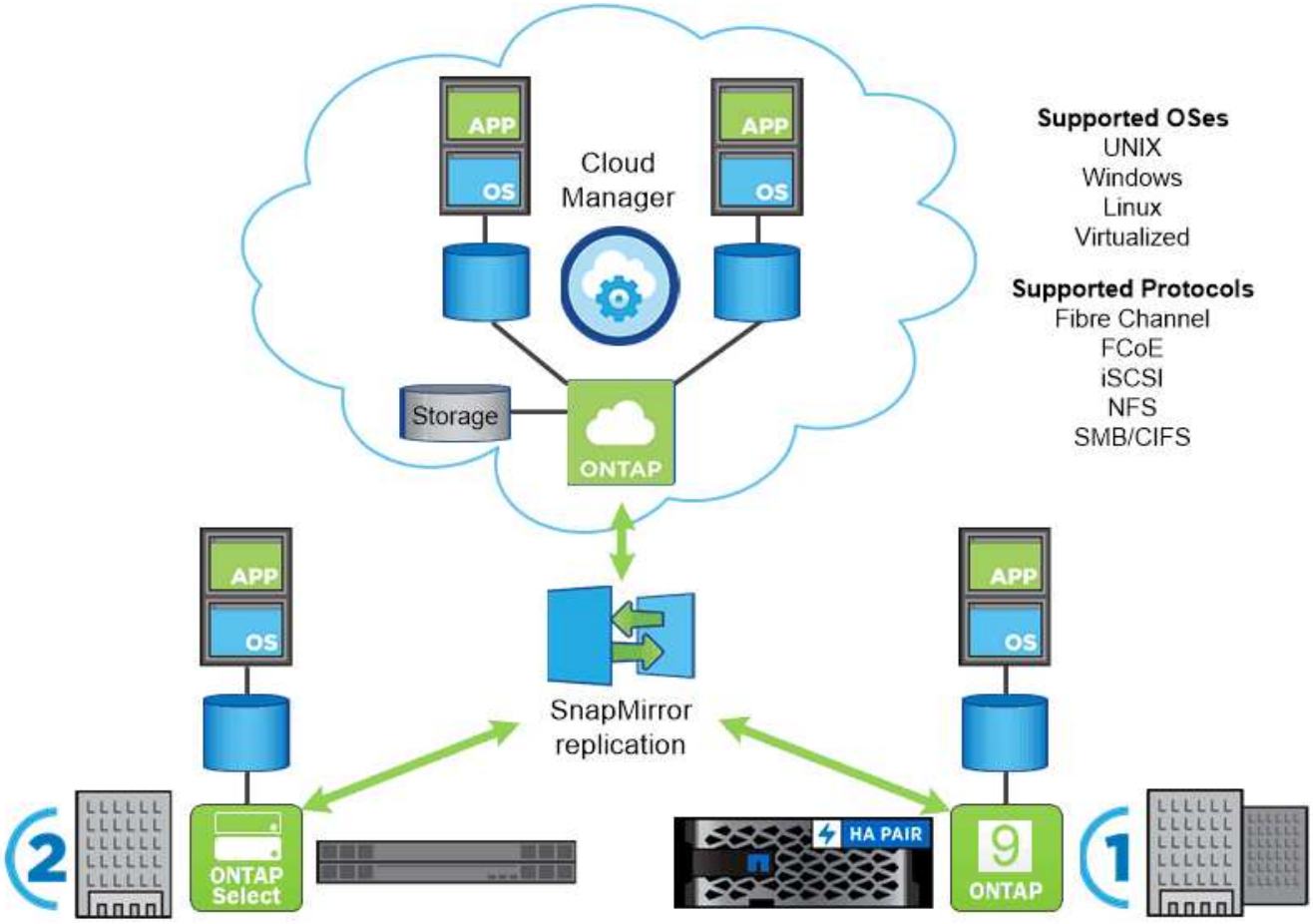
有關NetApp ONTAP儲存系統的更多信息，請訪問 "NetApp ONTAP網站"。

ONTAP提供以下功能：

- 一個統一的儲存系統，可同時存取和管理 NFS、CIFS、iSCSI、FC、FCoE 和 FC-NVMe 協定的資料。
- 不同的部署模型包括全快閃、混合和全 HDD 硬體配置上的本地部署；受支援的虛擬機器管理程式（如ONTAP Select）上的基於 VM 的儲存平台；以及在雲端中的Cloud Volumes ONTAP。
- 透過支援自動資料分層、內聯資料壓縮、重複資料刪除和壓縮，提高了ONTAP系統上的資料儲存效率。
- 基於工作負載、QoS 控制的儲存。
- 與公有雲無縫集成，實現資料分層與保護。 ONTAP還提供強大的資料保護功能，使其在任何環境中都脫穎而出：
  - * NetApp Snapshot 副本。 *使用最少的磁碟空間快速進行時間點資料備份，且不產生額外的效能開銷。
  - * NetApp SnapMirror。 *將資料的 Snapshot 副本從一個儲存系統鏡像到另一個儲存系統。 ONTAP也支援將資料鏡像到其他實體平台和雲端原生服務。
  - * NetApp SnapLock。 *透過將不可重寫資料寫入在指定時間內無法覆寫或擦除的特殊捲，可以有效地管理這些資料。
  - * NetApp SnapVault。 *將來自多個儲存系統的資料備份到中央 Snapshot 副本，作為所有指定係統的備份。
  - * NetAppSyncMirror。 *為實體連接到同一控制器的兩個不同磁碟叢提供即時 RAID 等級資料鏡像。
  - * NetApp SnapRestore。 *提供從 Snapshot 副本按需快速恢復備份資料的功能。
  - * NetApp FlexClone。 *根據 Snapshot 副本提供NetApp磁碟區的完全可讀、可寫入副本的即時配置。

有關ONTAP的更多信息，請參閱 "ONTAP 9 文檔中心"。

NetApp ONTAP可在本機、虛擬化或雲端使用。



NetApp平台

NetApp AFF/ FAS

NetApp提供強大的全快閃 (AFF) 和橫向擴展混合 (FAS) 儲存平台，這些平台具有低延遲效能、整合資料保護和多協定支援等特點。

這兩種系統均由NetApp ONTAP資料管理軟體提供支持，這是業界最先進的資料管理軟體，具有高可用性、雲端整合、簡化的儲存管理功能，可提供您的資料結構所需的企業級速度、效率和安全性。

有關 NETAPP AFF/ FAS平台的更多信息，請按一下 "這裡"。

ONTAP Select

ONTAP Select是NetApp ONTAP的軟體定義部署，可部署到您環境中的虛擬機器管理程式。它可以安裝在 VMware vSphere 或 KVM 上，並提供基於硬體的ONTAP系統的全部功能和體驗。

有關ONTAP Select的更多信息，請單擊 "這裡"。

Cloud Volumes ONTAP

NetApp Cloud Volumes ONTAP是NetApp ONTAP的雲端部署版本，可部署在許多公有雲中，包括：Amazon AWS、Microsoft Azure 和 Google Cloud。

有關Cloud Volumes ONTAP 的更多信息，請單擊 "這裡"。

Amazon FSx ONTAP

Amazon FSx ONTAP透過ONTAP流行的資料存取和管理功能在 AWS 雲端中提供完全託管的共用儲存。有關Amazon FSx ONTAP 的更多信息，請單擊 "這裡"。

Azure NetApp Files

Azure NetApp Files是 Azure 原生的、第一方的、企業級的高效能檔案儲存服務。它提供卷宗服務，您可以為其建立NetApp帳戶、容量池和磁碟區。您還可以選擇服務和效能等級並管理資料保護。您可以使用熟悉且依賴的相同協定和工具來建立和管理高效能、高可用性和可擴展的檔案共用。有關Azure NetApp Files 的更多信息，請單擊 "這裡"。

Google Cloud NetApp Volumes

Google Cloud NetApp Volumes是一種完全託管的基於雲端的資料儲存服務，可提供進階資料管理功能和高度可擴展的效能。它允許您將基於檔案的應用程式移至 Google Cloud。它內建支援網路檔案系統（NFSv3 和 NFSv4.1）和伺服器訊息區塊（SMB）協議，因此您無需重新建置應用程序，並可以繼續為您的應用程式取得持久儲存。有關 Google Cloud NetApp VolumesP 的更多信息，請點擊 "這裡"。

NetApp Element：Red Hat OpenShift 與NetApp

NetApp Element軟體提供模組化、可擴展的效能，每個儲存節點為環境提供保證的容量和吞吐量。 NetApp Element系統可在單一叢集中從 4 個節點擴展到 100 個節點，並提供許多進階儲存管理功能。

有關NetApp Element存儲系統的更多信息，請訪問 "NetApp Solidfire 網站"。

iSCSI 登入重新導向和自我修復功能

NetApp Element軟體利用 iSCSI 儲存協議，這是在傳統 TCP/IP 網路上封裝 SCSI 命令的標準方法。當 SCSI 標準變更或乙太網路效能提升時，iSCSI 儲存協定將受益，而無需進行任何變更。

儘管所有儲存節點都有一個管理 IP 和一個儲存 IP，但NetApp Element軟體會為叢集中的所有儲存流量公佈一個儲存虛擬 IP 位址（SVIP 位址）。作為 iSCSI 登入程序的一部分，儲存可以回應目標磁碟區已移至不同的位址，因此無法繼續協商流程。然後，主機會向新位址重新發出登入請求，整個過程不需要主機端重新配置。此過程稱為 iSCSI 登入重定向。

iSCSI 登入重新導向是NetApp Element軟體叢集的關鍵部分。當收到主機登入要求時，節點根據 IOPS 和磁碟區的容量需求決定叢集中的哪個成員應該處理流量。卷分佈在NetApp Element軟體叢集中，如果單一節點處理的捲流量過多或新增了新節點，則會重新分配。在整個陣列中分配給定卷的多個副本。

透過這種方式，如果節點故障後進行磁碟區重新分配，則除了登出並登入並重新導向到新位置之外，對主機連線沒有影響。透過 iSCSI 登入重新導向， NetApp Element軟體叢集是一種可自我修復、橫向擴展的架構，能夠實現無中斷升級和運作。

NetApp Element軟體叢集 QoS

NetApp Element軟體叢集允許根據每個磁碟區動態配置 QoS。您可以使用每個磁碟區的 QoS 設定根據您定義的 SLA 來控制儲存效能。以下三個可設定參數定義 QoS：

- *最低 IOPS。 * NetApp Element軟體叢集為磁碟區提供的最小持續 IOPS 數。為磁碟區配置的最小 IOPS 是磁碟區的保證效能等級。每卷的性能不會低於這個水平。
- *最大 IOPS。 * NetApp Element軟體叢集為特定磁碟區提供的最大持續 IOPS 數。
- *突發 IOPS。 *短時間突發場景下允許的最大 IOPS 數。突發持續時間設定是可設定的，預設為 1 分鐘。如果卷的運行速度低於最大 IOPS 水平，則會累積突發積分。當效能等級變得非常高並被推動時，磁碟區上允許出現超出最大 IOPS 的短時間突發 IOPS。

多租戶

安全多租戶透過以下功能實現：

- *安全認證。 *質詢握手身份驗證協定 (CHAP) 用於安全磁碟區存取。輕量級目錄存取協定 (LDAP) 用於安全存取叢集以進行管理和報告。
- 卷訪問組 (VAG)。或者，可以使用 VAG 代替身份驗證，將任意數量的 iSCSI 啟動器特定的 iSCSI 限定名稱 (IQN) 對應到一個或多個磁碟區。若要存取 VAG 中的捲，啟動器的 IQN 必須位於該卷組允許的 IQN 清單中。
- 租戶虛擬區域網路 (VLAN)。在網路級別，透過使用 VLAN 實現 iSCSI 啟動器和NetApp Element軟體叢集之間的端對端網路安全性。對於為隔離工作負載或租用戶而建立的任何 VLAN， NetApp Element Software 都會建立一個單獨的 iSCSI 目標 SVIP 位址，該位址只能透過特定的 VLAN 存取。
- *啟用 VRF 的 VLAN。 *為了進一步支援資料中心的安全性和可擴充性， NetApp Element軟體可讓您啟用任何租戶 VLAN 以實現類似 VRF 的功能。此功能增加了以下兩個關鍵功能：
  - *L3 路由到租戶 SVIP 位址。 *此功能可讓您將 iSCSI 啟動器放置在與NetApp Element軟體叢集不同的網路或 VLAN 上。
  - *重疊或重複的 IP 子網路。 *此功能可讓您為租用戶環境新增模板，從而允許每個對應的租用戶 VLAN 指派來自相同 IP 子網路的 IP 位址。此功能對於服務提供者環境中非常有用，因為 IP 空間的規模和保存非常重要。

企業儲存效率

NetApp Element軟體叢集提高了整體儲存效率和效能。以下功能以內聯方式執行，始終處於開啟狀態，且不需要使用者手動設定：

- *重複資料刪除*系統僅儲存唯一的 4K 區塊。任何重複的 4K 區塊都會自動與已儲存的資料版本相關聯。資料位於區塊驅動器上，並使用NetApp Element軟體 Helix 資料保護進行鏡像。該系統大大減少了容量消耗和系統內的寫入操作。
- *壓縮。 *在資料寫入NVRAM之前，壓縮是內聯執行的。資料被壓縮，儲存在 4K 區塊中，並在系統中保持壓縮狀態。這種壓縮顯著減少了整個叢集的容量消耗、寫入操作和頻寬消耗。
- *精簡配置。 *此功能可在您需要時提供適量的存儲，從而消除因磁碟區過度配置或捲利用不足而導致的容量消耗。

- *螺旋。 *單一磁碟區的元資料儲存在元資料磁碟機上，並複製到輔助元資料磁碟機以實現冗餘。

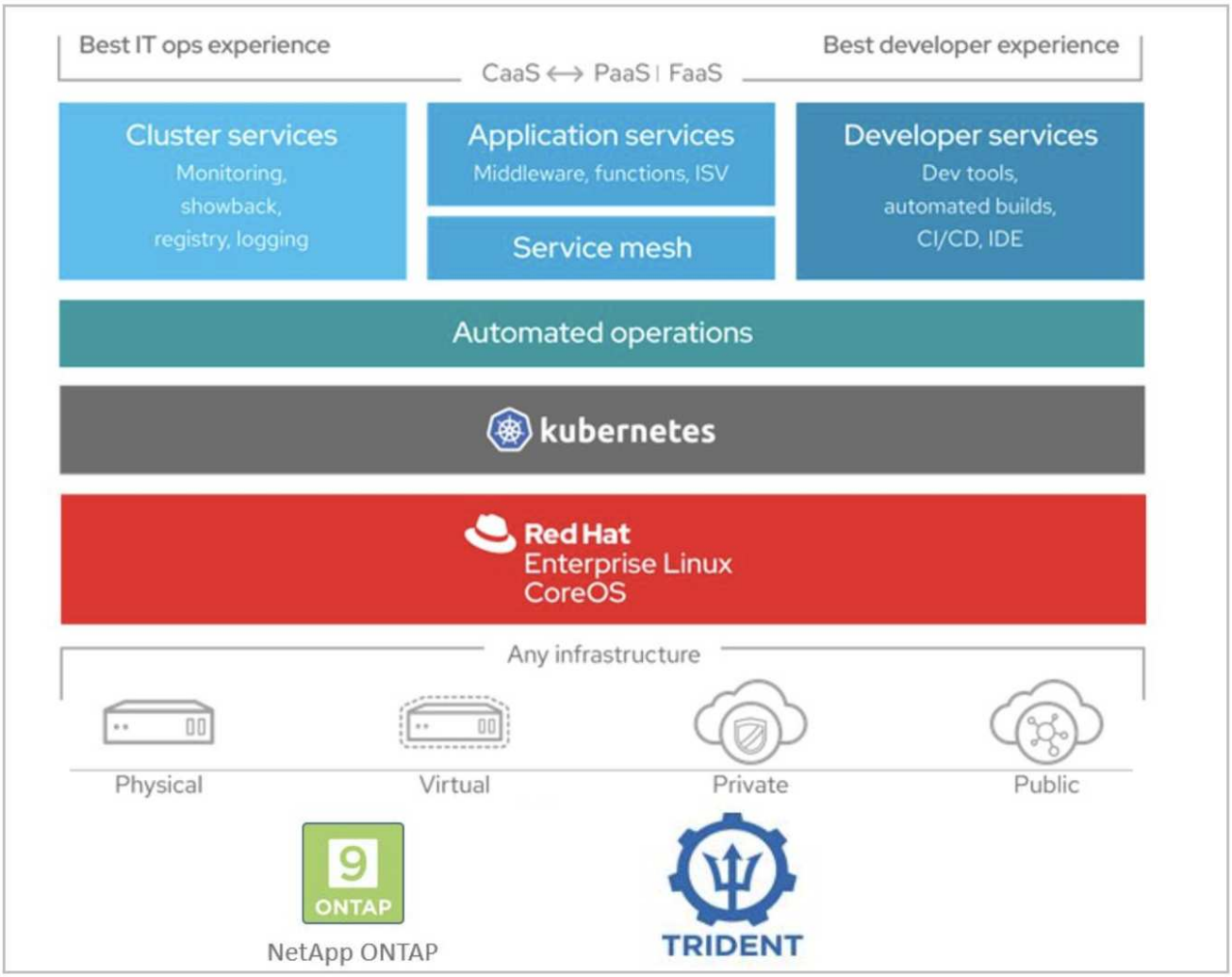
Element 是為自動化而設計的。所有儲存功能均可透過 API 取得。這些 API 是 UI 用來控制系統的唯一方法。

NetApp儲存集成

了解NetApp Trident與 Red Hat OpenShift 的集成

了解NetApp Trident保護，該保護已通過 OpenShift 虛擬化解決方案的應用程式和持久性儲存管理驗證。

Trident是由NetApp維護的開源儲存配置器和編排器， NetApp Trident保護可協助您在基於容器的環境（例如 Red Hat OpenShift）中編排和管理持久性資料。



以下頁麵包含有關NetApp產品的更多信息，這些產品已在 Red Hat OpenShift with NetApp解決方案中經過應用程式和持久性儲存管理驗證：

- "Trident文檔"

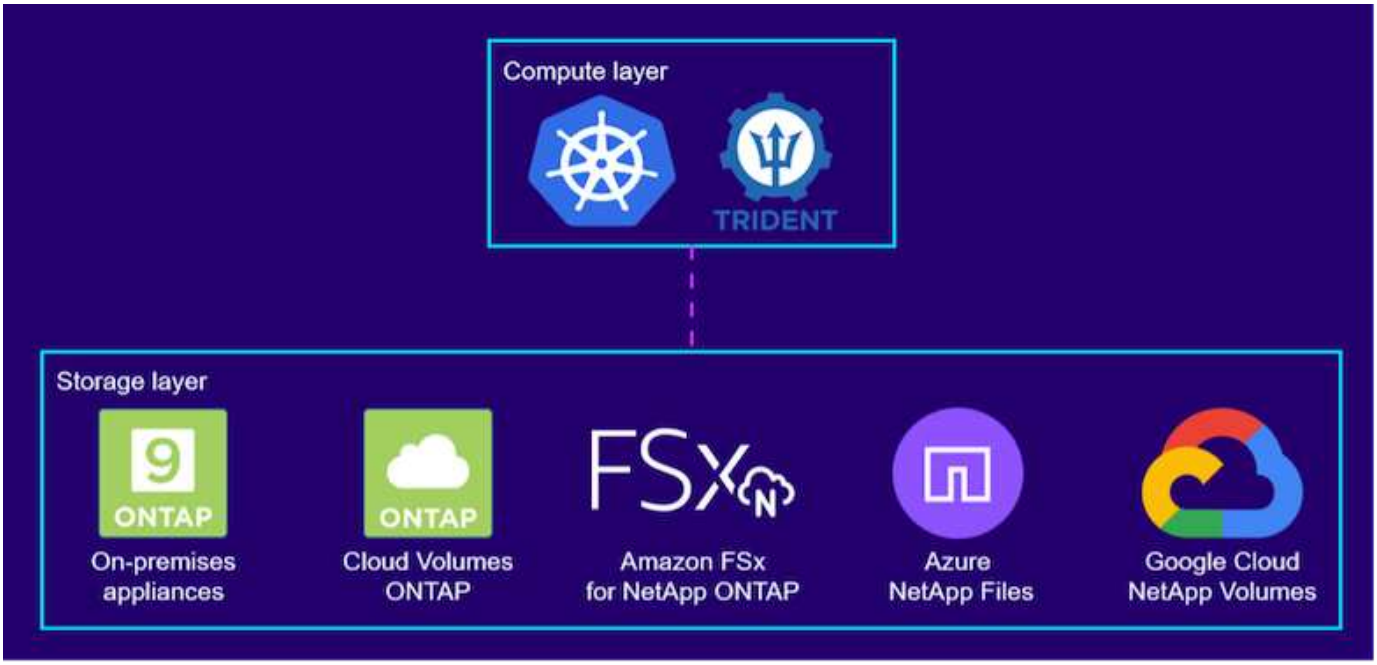
- "Trident保護文檔"

NetApp Trident

Trident概述

Trident是一個開源且完全支援的容器和 Kubernetes 發行版（包括 Red Hat OpenShift）儲存編排器。 Trident可與整個NetApp儲存產品組合搭配使用，包括NetApp ONTAP和 Element 儲存系統，並且還支援 NFS 和 iSCSI 連線。 Trident允許最終用戶從其NetApp儲存系統配置和管理存儲，而無需儲存管理員的干預，從而加速 DevOps 工作流程。

管理員可以根據專案需求和儲存系統模型配置多個儲存後端，以實現進階儲存功能，包括壓縮、特定磁碟類型或保證一定效能等級的 QoS 等級。定義完成後，開發人員可以在他們的專案中使用這些後端來建立持久性卷聲明 (PVC) 並根據需要將持久性儲存附加到他們的容器。



Trident的開發週期很快，和 Kubernetes 一樣，每年發布四次。

已測試過哪個版本的Trident以及可以找到哪個 Kubernetes 發行版的支援矩陣 "這裡"。

請參閱"Trident產品文檔"有關安裝和配置的詳細資訊。

下載Trident

若要在已部署的使用者叢集上安裝Trident並配置持久性卷，請完成下列步驟：

1. 將安裝檔案下載到管理工作站並提取內容。目前版本的Trident可以下載 "這裡"。
2. 從下載的軟體包中提取Trident安裝。

```
[netapp-user@rhel7 ~]$ tar -xzf trident-installer-22.01.0.tar.gz
[netapp-user@rhel7 ~]$ cd trident-installer/
[netapp-user@rhel7 trident-installer]$
```

使用 **Helm** 安裝**Trident Operator**

1. 首先設定用戶集群的 `kubeconfig`文件作為環境變量，這樣您就不必引用它，因為Trident沒有選項來傳遞此文件。

```
[netapp-user@rhel7 trident-installer]$ export KUBECONFIG=~/ocp-
install/auth/kubeconfig
```

2. 執行 Helm 指令從 helm 目錄中的 tarball 安裝Trident運算符，同時在使用者叢集中建立 trident 命名空間。

```
[netapp-user@rhel7 trident-installer]$ helm install trident
helm/trident-operator-22.01.0.tgz --create-namespace --namespace trident
NAME: trident
LAST DEPLOYED: Fri May  7 12:54:25 2021
NAMESPACE: trident
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
Thank you for installing trident-operator, which will deploy and manage
NetApp's Trident CSI
storage provisioner for Kubernetes.

Your release is named 'trident' and is installed into the 'trident'
namespace.
Please note that there must be only one instance of Trident (and
trident-operator) in a Kubernetes cluster.

To configure Trident to manage storage resources, you will need a copy
of tridentctl, which is
available in pre-packaged Trident releases.  You may find all Trident
releases and source code
online at https://github.com/NetApp/trident.

To learn more about the release, try:

  $ helm status trident
  $ helm get all trident
```

3. 您可以透過檢查命名空間中執行的 pod 或使用 tridentctl 二進位檔案檢查已安裝的版本來驗證Trident是否已成功安裝。

```
[netapp-user@rhel7 trident-installer]$ oc get pods -n trident
NAME                                READY   STATUS    RESTARTS   AGE
trident-csi-5z45l                   1/2     Running   2          30s
trident-csi-696b685cf8-htdb2        6/6     Running   0          30s
trident-csi-b74p2                   2/2     Running   0          30s
trident-csi-lrw4n                   2/2     Running   0          30s
trident-operator-7c748d957-gr2gw    1/1     Running   0          36s

[netapp-user@rhel7 trident-installer]$ ./tridentctl -n trident version
+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 22.01.0        | 22.01.0        |
+----------------+----------------+
```

在某些情況下，客戶環境可能需要客製化Trident部署。在這些情況下，也可以手動安裝Trident操作符並更新包含的清單以自訂部署。

手動安裝**Trident Operator**

1. 首先，設定用戶集群的 `kubeconfig`文件作為環境變量，這樣您就不必引用它，因為Trident沒有選項來傳遞此文件。

```
[netapp-user@rhel7 trident-installer]$ export KUBECONFIG=~/ocp-
install/auth/kubeconfig
```

2. 這 `trident-installer`目錄包含定義所有必要資源的清單。使用適當的清單，創建 `TridentOrchestrator`自訂資源定義。

```
[netapp-user@rhel7 trident-installer]$ oc create -f
deploy/crds/trident.netapp.io_tridentorchestrators_crd_post1.16.yaml
customresourcedefinition.apiextensions.k8s.io/tridentorchestrators.tride
nt.netapp.io created
```

3. 如果不存在，請使用提供的清單在叢集中建立Trident命名空間。

```
[netapp-user@rhel7 trident-installer]$ oc apply -f deploy/namespace.yaml
namespace/trident created
```

4. 建立Trident操作員部署所需的資源，例如 `ServiceAccount`對於操作員來說， `ClusterRole`和 `ClusterRoleBinding`到 `ServiceAccount`，一個專門的 `PodSecurityPolicy`或操作員本身。

```
[netapp-user@rhel7 trident-installer]$ oc create -f deploy/bundle.yaml
serviceaccount/trident-operator created
clusterrole.rbac.authorization.k8s.io/trident-operator created
clusterrolebinding.rbac.authorization.k8s.io/trident-operator created
deployment.apps/trident-operator created
podsecuritypolicy.policy/tridentoperatorpods created
```

5. 您可以使用以下命令在部署操作員後檢查其狀態：

```
[netapp-user@rhel7 trident-installer]$ oc get deployment -n trident
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
trident-operator   1/1     1            1           23s
[netapp-user@rhel7 trident-installer]$ oc get pods -n trident
NAME                                READY   STATUS    RESTARTS   AGE
trident-operator-66f48895cc-lzczk   1/1     Running   0          41s
```

6. 部署操作員後，我們現在可以使用它來安裝Trident。這需要創建一個 `TridentOrchestrator`。

```
[netapp-user@rhel7 trident-installer]$ oc create -f
deploy/crds/tridentorchestrator_cr.yaml
tridentorchestrator.trident.netapp.io/trident created
[netapp-user@rhel7 trident-installer]$ oc describe torc trident
Name:         trident
Namespace:
Labels:       <none>
Annotations:  <none>
API Version:  trident.netapp.io/v1
Kind:         TridentOrchestrator
Metadata:
  Creation Timestamp:  2021-05-07T17:00:28Z
  Generation:          1
  Managed Fields:
    API Version:  trident.netapp.io/v1
    Fields Type:  FieldsV1
    fieldsV1:
      f:spec:
        .:
        f:debug:
        f:namespace:
    Manager:      kubectl-create
    Operation:    Update
```

```
    Time:         2021-05-07T17:00:28Z
    API Version:  trident.netapp.io/v1
    Fields Type:  FieldsV1
    fieldsV1:
      f:status:
        .:
        f:currentInstallationParams:
          .:
          f:IPv6:
          f:autosupportHostname:
          f:autosupportimage:
          f:autosupportProxy:
          f:autosupportSerialNumber:
          f:debug:
          f:enableNodePrep:
          f:imagePullSecrets:
          f:imageRegistry:
          f:k8sTimeout:
          f:kubeletDir:
          f:logFormat:
          f:silenceAutosupport:
          f:tridentimage:
        f:message:
        f:namespace:
        f:status:
        f:version:
    Manager:         trident-operator
    Operation:       Update
    Time:            2021-05-07T17:00:28Z
  Resource Version:  931421
  Self Link:
/apis/trident.netapp.io/v1/tridentorchestrators/trident
  UID:               8a26a7a6-dde8-4d55-9b66-a7126754d81f
Spec:
  Debug:      true
  Namespace:  trident
Status:
  Current Installation Params:
    IPv6:                      false
    Autosupport Hostname:
    Autosupport image:         netapp/trident-autosupport:21.01
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:                     true
    Enable Node Prep:          false
    Image Pull Secrets:
```

```
    Image Registry:
    k8sTimeout:           30
    Kubelet Dir:          /var/lib/kubelet
    Log Format:           text
    Silence Autosupport:  false
    Trident image:        netapp/trident:22.01.0
  Message:                Trident installed
  Namespace:              trident
  Status:                 Installed
  Version:                v22.01.0
Events:
  Type    Reason      Age   From                           Message
  ----    ------      ----  ----                           -------
  Normal  Installing  80s   trident-operator.netapp.io  Installing
Trident
  Normal  Installed   68s   trident-operator.netapp.io  Trident
installed
```

7. 您可以透過檢查命名空間中執行的 pod 或使用 tridentctl 二進位檔案檢查已安裝的版本來驗證Trident是否已成功安裝。

```
[netapp-user@rhel7 trident-installer]$ oc get pods -n trident
NAME                                READY   STATUS    RESTARTS   AGE
trident-csi-bb64c6cb4-lmd6h         6/6     Running   0          82s
trident-csi-gn59q                   2/2     Running   0          82s
trident-csi-m4szj                   2/2     Running   0          82s
trident-csi-sb9k9                   2/2     Running   0          82s
trident-operator-66f48895cc-lzczk   1/1     Running   0          2m39s

[netapp-user@rhel7 trident-installer]$ ./tridentctl -n trident version
+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 22.01.0        | 22.01.0        |
+----------------+----------------+
```

準備工作節點以進行存儲

NFS

大多數 Kubernetes 發行版都附帶預設安裝的用於掛載 NFS 後端的軟體包和實用程序，包括 Red Hat OpenShift。

但是，對於 NFSv3，沒有在客戶端和伺服器之間協商並發的機制。因此，必須手動將客戶端 sunrpc 插槽表條目的最大數量與伺服器上支援的值同步，以確保 NFS 連線的最佳效能，而無需伺服器減小連線的視窗大小。

對於ONTAP，支援的 sunrpc 插槽表條目的最大數量為 128，即ONTAP一次可以處理 128 個並發 NFS 請求。但是，預設情況下，Red Hat CoreOS/Red Hat Enterprise Linux 每個連線最多有 65,536 個 sunrpc 插槽表條目。我們需要將此值設為 128，這可以使用 OpenShift 中的 Machine Config Operator (MCO) 來完成。

若要修改 OpenShift 工作節點中的最大 sunrpc 插槽表條目數，請完成下列步驟：

1. 登入 OCP 網路控制台並導覽至 Compute > Machine Configs。點選建立機器配置。複製並貼上 YAML 文件，然後按一下「建立」。

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  name: 98-worker-nfs-rpc-slot-tables
  labels:
    machineconfiguration.openshift.io/role: worker
spec:
  config:
    ignition:
      version: 3.2.0
    storage:
      files:
        - contents:
            source: data:text/plain;charset=utf-
8;base64,b3B0aW9ucyBzdW5ycGMgdGNwX21heF9zbG90X3RhYmxlX2VudHJpZXM9MTI4Cg=
=
          filesystem: root
          mode: 420
          path: /etc/modprobe.d/sunrpc.conf
```

2. 建立 MCO 後，需要在所有工作節點上套用組態並逐一重新啟動。整個過程大約需要20到30分鐘。使用以下命令驗證機器配置是否已套用 `oc get mcp`並確保工人的機器配置池已更新。

```
[netapp-user@rhel7 openshift-deploy]$ oc get mcp
NAME     CONFIG                                             UPDATED   UPDATING
DEGRADED
master   rendered-master-a520ae930e1d135e0dee7168   True      False
False
worker   rendered-worker-de321b36eeba62df41feb7bc   True      False
False
```

iSCSI

要準備工作節點以允許透過 iSCSI 協定映射區塊儲存卷，您必須安裝必要的軟體包來支援該功能。

在 Red Hat OpenShift 中，這是透過在部署叢集後將 MCO（機器配置操作員）應用於叢集來處理的。

若要設定工作節點以執行 iSCSI 服務，請完成下列步驟：

1. 登入 OCP 網路控制台並導覽至 Compute > Machine Configs。點選建立機器配置。複製並貼上 YAML 文件，然後按一下「建立」。

   不使用多路徑時：

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 99-worker-element-iscsi
spec:
  config:
    ignition:
      version: 3.2.0
    systemd:
      units:
        - name: iscsid.service
          enabled: true
          state: started
  osImageURL: ""
```

   使用多路徑時：

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  name: 99-worker-ontap-iscsi
  labels:
    machineconfiguration.openshift.io/role: worker
spec:
  config:
    ignition:
      version: 3.2.0
    storage:
      files:
      - contents:
          source: data:text/plain;charset=utf-
8;base64,ZGVmYXVsdHMgewogICAgICAgIHVzZXJfZnJpZW5kbHlfbmFtZXMgbm8KICAgICA
gICBmaW5kX211bHRpcGF0aHMgbm8KfQoKYmxhY2tsaXN0X2V4Y2VwdGlvbnMgewogICAgICA
gIHByb3BlcnR5ICIoU0NTSV9JREVOVF98SURfV1dOKSIKfQoKYmxhY2tsaXN0IHsKfQoK
          verification: {}
        filesystem: root
        mode: 400
        path: /etc/multipath.conf
    systemd:
      units:
        - name: iscsid.service
          enabled: true
          state: started
        - name: multipathd.service
          enabled: true
          state: started
  osImageURL: ""
```

2. 配置建立後，大約需要 20 到 30 分鐘將配置套用到工作節點並重新載入它們。使用以下命令驗證機器配置是否已套用 `oc get mcp`並確保工人的機器配置池已更新。您也可以登入工作節點來確認 iscsid 服務正在執行（如果使用多路徑，則 multipathd 服務正在執行）。

```
[netapp-user@rhel7 openshift-deploy]$ oc get mcp
NAME     CONFIG                                             UPDATED   UPDATING
DEGRADED
master   rendered-master-a520ae930e1d135e0dee7168   True      False
False
worker   rendered-worker-de321b36eeba62df41feb7bc   True      False
False

[netapp-user@rhel7 openshift-deploy]$ ssh core@10.61.181.22 sudo
systemctl status iscsid
● iscsid.service - Open-iSCSI
   Loaded: loaded (/usr/lib/systemd/system/iscsid.service; enabled;
vendor preset: disabled)
   Active: active (running) since Tue 2021-05-26 13:36:22 UTC; 3 min ago
     Docs: man:iscsid(8)
           man:iscsiadm(8)
 Main PID: 1242 (iscsid)
   Status: "Ready to process requests"
    Tasks: 1
   Memory: 4.9M
      CPU: 9ms
   CGroup: /system.slice/iscsid.service
           └─1242 /usr/sbin/iscsid -f

[netapp-user@rhel7 openshift-deploy]$ ssh core@10.61.181.22 sudo
systemctl status multipathd
 ● multipathd.service - Device-Mapper Multipath Device Controller
   Loaded: loaded (/usr/lib/systemd/system/multipathd.service; enabled;
vendor preset: enabled)
   Active: active (running) since Tue 2021-05-26 13:36:22 UTC; 3 min ago
  Main PID: 918 (multipathd)
    Status: "up"
    Tasks: 7
    Memory: 13.7M
    CPU: 57ms
    CGroup: /system.slice/multipathd.service
            └─918 /sbin/multipathd -d -s
```

也可以透過執行以下命令來確認 MachineConfig 已成功應用且服務已如預期啟動 `oc debug` 帶有適當標誌的命令。

建立儲存系統後端

完成Trident Operator 安裝後，您必須為正在使用的特定NetApp儲存平台設定後端。按照下面的連結繼續設定並設定Trident。

- "NetApp ONTAP NFS"
- "NetApp ONTAP iSCSI"
- "NetApp Element iSCSI"

NetApp ONTAP NFS 配置

要實現Trident與NetApp ONTAP儲存系統的集成，您必須建立一個能夠與儲存系統通訊的後端。

1. 下載的安裝檔案中提供了範例後端文件 `sample-input`資料夾層次結構。對於服務 NFS 的NetApp ONTAP系統，複製 `backend-ontap-nas.json`文件到您的工作目錄並編輯該文件。

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/backends-
samples/ontap-nas/backend-ontap-nas.json ./
[netapp-user@rhel7 trident-installer]$ vi backend-ontap-nas.json
```

2. 編輯此檔案中的 backendName、managementLIF、dataLIF、svm、username 和 password 值。

```
{
    "version": 1,
    "storageDriverName": "ontap-nas",
    "backendName": "ontap-nas+10.61.181.221",
    "managementLIF": "172.21.224.201",
    "dataLIF": "10.61.181.221",
    "svm": "trident_svm",
    "username": "cluster-admin",
    "password": "password"
}
```

> 最佳做法是將自訂 backendName 值定義為 storageDriverName 和為 NFS 提供服務的 dataLIF 的組合，以便於識別。

3. 有了這個後端文件，運行以下命令來創建您的第一個後端。

```
[netapp-user@rhel7 trident-installer]$ ./tridentctl -n trident create
backend -f backend-ontap-nas.json
+------------------------+----------------
+--------------------------------------+--------+---------+
|          NAME          | STORAGE DRIVER |                 UUID
| STATE  | VOLUMES |
+------------------------+----------------
+--------------------------------------+--------+---------+
| ontap-nas+10.61.181.221 | ontap-nas      | be7a619d-c81d-445c-b80c-
5c87a73c5b1e | online |       0 |
+------------------------+----------------
+--------------------------------------+--------+---------+
```

4. 建立後端後，接下來必須建立儲存類別。與後端一樣，有一個範例儲存類別文件，可以根據 sample-inputs 資料夾中提供的環境進行編輯。將其複製到工作目錄並進行必要的編輯以反映已建立的後端。

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/storage-class-
samples/storage-class-csi.yaml.templ ./storage-class-basic.yaml
[netapp-user@rhel7 trident-installer]$ vi storage-class-basic.yaml
```

5. 對此文件唯一需要做的編輯是定義 `backendType`將值設定為新建立的後端的儲存驅動程式的名稱。還要注意名稱欄位值，該值必須在後續步驟中引用。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: basic-csi
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
```

有一個可選字段稱為 `fsType`這是在這個文件中定義的。可以在 NFS 後端刪除此行。

6. 運行 `oc`命令來建立儲存類別。

```
[netapp-user@rhel7 trident-installer]$ oc create -f storage-class-
basic.yaml
storageclass.storage.k8s.io/basic-csi created
```

7. 建立儲存類別後，您必須建立第一個持久化磁碟區宣告 (PVC)。有一個範例 `pvc-basic.yaml`也可用於執行位於 sample-inputs 中的此操作的檔案。

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/pvc-samples/pvc-
basic.yaml ./
[netapp-user@rhel7 trident-installer]$ vi pvc-basic.yaml
```

8. 對此文件唯一需要做的編輯是確保 `storageClassName`字段與剛剛建立的字段相符。 PVC 定義可以根據要設定的工作負載的需要進一步客製化。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: basic
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: basic-csi
```

9. 透過發出 `oc`命令。創建可能需要一些時間，具體取決於所創建的備份卷的大小，因此您可以在創建完成時觀察該過程。

```
[netapp-user@rhel7 trident-installer]$ oc create -f pvc-basic.yaml
persistentvolumeclaim/basic created

[netapp-user@rhel7 trident-installer]$ oc get pvc
NAME    STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
basic   Bound    pvc-b4370d37-0fa4-4c17-bd86-94f96c94b42d   1Gi
RWO            basic-csi      7s
```

NetApp ONTAP iSCSI 設定

要實現Trident與NetApp ONTAP儲存系統的集成，您必須建立一個能夠與儲存系統通訊的後端。

1. 下載的安裝檔案中提供了範例後端文件 `sample-input`資料夾層次結構。對於服務 iSCSI 的NetApp ONTAP系統，複製 `backend-ontap-san.json`文件到您的工作目錄並編輯該文件。

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/backends-
samples/ontap-san/backend-ontap-san.json ./
[netapp-user@rhel7 trident-installer]$ vi backend-ontap-san.json
```

2. 編輯此文件中的 managementLIF、dataLIF、svm、使用者名稱和密碼值。

```
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "managementLIF": "172.21.224.201",
  "dataLIF": "10.61.181.240",
  "svm": "trident_svm",
  "username": "admin",
  "password": "password"
}
```

3. 有了這個後端文件，運行以下命令來創建您的第一個後端。

```
[netapp-user@rhel7 trident-installer]$ ./tridentctl -n trident create
backend -f backend-ontap-san.json
+------------------------+----------------
+----------------------------------------+--------+---------+
|          NAME          | STORAGE DRIVER |                 UUID
| STATE  | VOLUMES |
+------------------------+----------------
+----------------------------------------+--------+---------+
| ontapsan_10.61.181.241 | ontap-san      | 6788533c-7fea-4a35-b797-
fb9bb3322b91 | online |       0 |
+------------------------+----------------
+----------------------------------------+--------+---------+
```

4. 建立後端後，接下來必須建立儲存類別。與後端一樣，有一個範例儲存類別文件，可以根據 sample-inputs 資料夾中提供的環境進行編輯。將其複製到工作目錄並進行必要的編輯以反映已建立的後端。

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/storage-class-
samples/storage-class-csi.yaml.templ ./storage-class-basic.yaml
[netapp-user@rhel7 trident-installer]$ vi storage-class-basic.yaml
```

5. 對此文件唯一需要做的編輯是定義 `backendType`將值設定為新建立的後端的儲存驅動程式的名稱。還要注意名稱欄位值，該值必須在後續步驟中引用。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: basic-csi
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
```

有一個可選字段稱為 `fsType`這是在這個文件中定義的。在 iSCSI 後端，可以將此值設定為特定的 Linux 檔案系統類型（XFS、ext4 等）或刪除以允許 OpenShift 決定使用哪種檔案系統。

6. 運行 `oc`命令來建立儲存類別。

```
[netapp-user@rhel7 trident-installer]$ oc create -f storage-class-
basic.yaml
storageclass.storage.k8s.io/basic-csi created
```

7. 建立儲存類別後，您必須建立第一個持久化磁碟區宣告 (PVC)。有一個範例 `pvc-basic.yaml`也可用於執行位於 sample-inputs 中的此操作的檔案。

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/pvc-samples/pvc-
basic.yaml ./
[netapp-user@rhel7 trident-installer]$ vi pvc-basic.yaml
```

8. 對此文件唯一需要做的編輯是確保 `storageClassName`字段與剛剛建立的字段相符。 PVC 定義可以根據要設定的工作負載的需要進一步客製化。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: basic
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: basic-csi
```

9. 透過發出 `oc`命令。創建可能需要一些時間，具體取決於所創建的備份卷的大小，因此您可以在創建完成時觀察該過程。

```
[netapp-user@rhel7 trident-installer]$ oc create -f pvc-basic.yaml
persistentvolumeclaim/basic created

[netapp-user@rhel7 trident-installer]$ oc get pvc
NAME    STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
basic   Bound    pvc-7ceac1ba-0189-43c7-8f98-094719f7956c   1Gi
RWO            basic-csi      3s
```

NetApp Element iSCSI 配置

要實現Trident與NetApp Element儲存系統的集成，您必須建立一個後端，以便使用 iSCSI 協定與儲存系統進行通訊。

1. 下載的安裝檔案中提供了範例後端文件 `sample-input`資料夾層次結構。對於服務於 iSCSI 的NetApp Element系統，複製 `backend-solidfire.json`文件到您的工作目錄並編輯該文件。

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/backends-
samples/solidfire/backend-solidfire.json ./
[netapp-user@rhel7 trident-installer]$ vi ./backend-solidfire.json
```

   a. 編輯使用者、密碼和 MVIP 值 `EndPoint`線。

   b. 編輯 `SVIP`價值。

```
 {
    "version": 1,
    "storageDriverName": "solidfire-san",
    "Endpoint": "https://trident:password@172.21.224.150/json-
rpc/8.0",
    "SVIP": "10.61.180.200:3260",
    "TenantName": "trident",
    "Types": [{"Type": "Bronze", "Qos": {"minIOPS": 1000, "maxIOPS":
2000, "burstIOPS": 4000}},
              {"Type": "Silver", "Qos": {"minIOPS": 4000, "maxIOPS":
6000, "burstIOPS": 8000}},
              {"Type": "Gold", "Qos": {"minIOPS": 6000, "maxIOPS":
8000, "burstIOPS": 10000}}]
}
```

2. 有了這個後端文件，運行以下命令來創建您的第一個後端。

```
[netapp-user@rhel7 trident-installer]$ ./tridentctl -n trident create
backend -f backend-solidfire.json
+------------------------+----------------
+--------------------------------------+--------+---------+
|          NAME          | STORAGE DRIVER |                 UUID
| STATE  | VOLUMES |
+------------------------+----------------
+--------------------------------------+--------+---------+
| solidfire_10.61.180.200 | solidfire-san  | b90783ee-e0c9-49af-8d26-
3ea87ce2efdf | online |       0 |
+------------------------+----------------
+--------------------------------------+--------+---------+
```

3. 建立後端後，接下來必須建立儲存類別。與後端一樣，有一個範例儲存類別文件，可以根據 sample-inputs 資料夾中提供的環境進行編輯。將其複製到工作目錄並進行必要的編輯以反映已建立的後端。

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/storage-class-
samples/storage-class-csi.yaml.templ ./storage-class-basic.yaml
[netapp-user@rhel7 trident-installer]$ vi storage-class-basic.yaml
```

4. 對此文件唯一需要做的編輯是定義 `backendType`將值設定為新建立的後端的儲存驅動程式的名稱。還要注意名稱欄位值，該值必須在後續步驟中引用。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: basic-csi
provisioner: csi.trident.netapp.io
parameters:
  backendType: "solidfire-san"
```

> 有一個可選字段稱為 `fsType`這是在這個文件中定義的。在 iSCSI 後端，可以將此值設定為特定的 Linux 檔案系統類型（XFS、ext4 等），或者可以刪除它以允許 OpenShift 決定使用哪種檔案系統。

5. 運行 `oc`命令來建立儲存類別。

```
[netapp-user@rhel7 trident-installer]$ oc create -f storage-class-
basic.yaml
storageclass.storage.k8s.io/basic-csi created
```

6. 建立儲存類別後，您必須建立第一個持久化磁碟區宣告 (PVC)。有一個範例 `pvc-basic.yaml`也可用於執行

位於 sample-inputs 中的此操作的檔案。

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/pvc-samples/pvc-
basic.yaml ./
[netapp-user@rhel7 trident-installer]$ vi pvc-basic.yaml
```

7. 對此文件唯一需要做的編輯是確保 `storageClassName`字段與剛剛建立的字段相符。 PVC 定義可以根據要設定的工作負載的需要進一步客製化。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: basic
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: basic-csi
```

8. 透過發出 `oc`命令。創建可能需要一些時間，具體取決於所創建的備份卷的大小，因此您可以在創建完成時觀察該過程。

```
[netapp-user@rhel7 trident-installer]$ oc create -f pvc-basic.yaml
persistentvolumeclaim/basic created

[netapp-user@rhel7 trident-installer]$ oc get pvc
NAME    STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
basic   Bound    pvc-3445b5cc-df24-453d-a1e6-b484e874349d   1Gi
RWO            basic-csi      5s
```

進階配置選項

探索負載平衡器選項

探索負載平衡器選項：**Red Hat OpenShift** 與**NetApp**

大多數情況下，Red Hat OpenShift 透過路由讓應用程式可供外界使用。透過為服務提供外部可存取的主機名稱來公開服務。 OpenShift 路由器可以使用定義的路由及其服務標識的端點來向外部用戶端提供這種命名連線。

然而在某些情況下，應用程式需要部署和配置客製化的負載平衡器來公開適當的服務。 NetApp Trident Protect 就是一個例子。為了滿足這項需求，我們評估了許多自訂負載平衡器選項。本節介紹它們的安裝和配置。

以下頁麵包含有關 Red Hat OpenShift 與NetApp解決方案中驗證的負載平衡器選項的更多資訊：

- "MetalLB"
- "F5 BIG-IP"

安裝 **MetalLB** 負載平衡器：**Red Hat OpenShift** 與**NetApp**

本頁列出了 MetalLB 負載平衡器的安裝和設定說明。

MetalLB 是安裝在 OpenShift 叢集上的自架網路負載平衡器，可在不在雲端供應商上執行的叢集中建立類型負載平衡器的 OpenShift 服務。 MetalLB 共同支撐 LoadBalancer 服務的兩個主要功能是地址分配和對外通告。

MetalLB 配置選項

根據 MetalLB 如何宣布分配給 OpenShift 叢集之外的 LoadBalancer 服務的 IP 位址，它以兩種模式運作：

- *第 2 層模式。 *在此模式下，OpenShift 叢集中的一個節點擁有該服務的所有權，並回應該 IP 的 ARP 請求，以使其在 OpenShift 叢集外部可存取。由於只有節點通告 IP，因此有頻寬瓶頸和故障轉移速度慢的限制。有關詳細信息，請參閱文檔"這裡"。
- *BGP 模式。 *在此模式下，OpenShift 叢集中的所有節點都會與路由器建立 BGP 對等會話，並通告路由以將流量轉送至服務 IP。實現此目的的先決條件是將 MetalLB 與該網路中的路由器整合。由於BGP中的哈希機制，當服務的IP到節點映射發生變化時，會受到一定的限制。欲了解更多信息，請參閱文檔"這裡"。

> 為了本文檔的目的，我們在第 2 層模式下配置 MetalLB。

安裝 **MetalLB** 負載平衡器

1. 下載 MetalLB 資源。

```
[netapp-user@rhel7 ~]$ wget
https://raw.githubusercontent.com/metallb/metallb/v0.10.2/manifests/name
space.yaml
[netapp-user@rhel7 ~]$ wget
https://raw.githubusercontent.com/metallb/metallb/v0.10.2/manifests/meta
llb.yaml
```

2. 編輯文件 `metallb.yaml`並刪除 `spec.template.spec.securityContext`來自控制器部署和揚聲器 DaemonSet。

   要刪除的行：

```
securityContext:
  runAsNonRoot: true
  runAsUser: 65534
```

3. 創建 `metallb-system`命名空間。

```
[netapp-user@rhel7 ~]$ oc create -f namespace.yaml
namespace/metallb-system created
```

4. 建立 MetalLB CR。

```
[netapp-user@rhel7 ~]$ oc create -f metallb.yaml
podsecuritypolicy.policy/controller created
podsecuritypolicy.policy/speaker created
serviceaccount/controller created
serviceaccount/speaker created
clusterrole.rbac.authorization.k8s.io/metallb-system:controller created
clusterrole.rbac.authorization.k8s.io/metallb-system:speaker created
role.rbac.authorization.k8s.io/config-watcher created
role.rbac.authorization.k8s.io/pod-lister created
role.rbac.authorization.k8s.io/controller created
clusterrolebinding.rbac.authorization.k8s.io/metallb-system:controller
created
clusterrolebinding.rbac.authorization.k8s.io/metallb-system:speaker
created
rolebinding.rbac.authorization.k8s.io/config-watcher created
rolebinding.rbac.authorization.k8s.io/pod-lister created
rolebinding.rbac.authorization.k8s.io/controller created
daemonset.apps/speaker created
deployment.apps/controller created
```

5. 在配置 MetalLB 揚聲器之前，請授予揚聲器 DaemonSet 提升的權限，以便它可以執行使負載平衡器工作所需的網路配置。

```
[netapp-user@rhel7 ~]$ oc adm policy add-scc-to-user privileged -n
metallb-system -z speaker
clusterrole.rbac.authorization.k8s.io/system:openshift:scc:privileged
added: "speaker"
```

6. 透過建立配置 MetalLB `ConfigMap`在 `metallb-system`命名空間。

```
[netapp-user@rhel7 ~]$ vim metallb-config.yaml

apiVersion: v1
kind: ConfigMap
metadata:
  namespace: metallb-system
  name: config
data:
  config: |
    address-pools:
    - name: default
      protocol: layer2
      addresses:
      - 10.63.17.10-10.63.17.200

[netapp-user@rhel7 ~]$ oc create -f metallb-config.yaml
configmap/config created
```

7. 現在，當建立負載平衡器服務時，MetalLB 會為服務指派一個 externalIP，並透過回應 ARP 請求來通告該 IP 位址。

> 如果您希望在 BGP 模式下配置 MetalLB，請跳過上面的步驟 6 並按照 MetalLB 文件中的步驟進行操作"這裡"。

安裝 **F5 BIG-IP** 負載平衡器

F5 BIG-IP 是一種應用交付控制器 (ADC)，它提供廣泛的高階生產級流量管理和安全服務，如 L4-L7 負載平衡、SSL/TLS 卸載、DNS、防火牆等。這些服務大大提高了應用程式的可用性、安全性和效能。

F5 BIG-IP 可以以多種方式部署和使用，可以在專用硬體上、在雲端或作為本地虛擬設備。請參閱此處的文檔，根據要求探索和部署 F5 BIG-IP。

為了將 F5 BIG-IP 服務與 Red Hat OpenShift 有效集成，F5 提供了 BIG-IP 容器入口服務 (CIS)。 CIS 作為控制器容器安裝，用於監控 OpenShift API 中的某些自訂資源定義 (CRD) 並管理 F5 BIG-IP 系統配置。 F5 BIG-IP CIS 可以設定為控制 OpenShift 中的服務類型 LoadBalancers 和 Routes。

此外，為了自動指派 IP 位址來為 LoadBalancer 類型提供服務，您可以利用 F5 IPAM 控制器。 F5 IPAM 控制器作為控制器 pod 安裝，它使用 ipamLabel 註解監視 LoadBalancer 服務的 OpenShift API，以從預先配置池中分配 IP 位址。

本頁列出了 F5 BIG-IP CIS 和 IPAM 控制器的安裝和設定說明。作為先決條件，您必須部署並獲得許可的 F5 BIG-IP 系統。它還必須獲得 SDN 服務的許可，該服務預設包含在 BIG-IP VE 基本許可證中。

> F5 BIG-IP 可以獨立或叢集模式部署。為了進行此驗證，F5 BIG-IP 以獨立模式部署，但出於生產目的，最好使用 BIG-IP 叢集以避免單點故障。

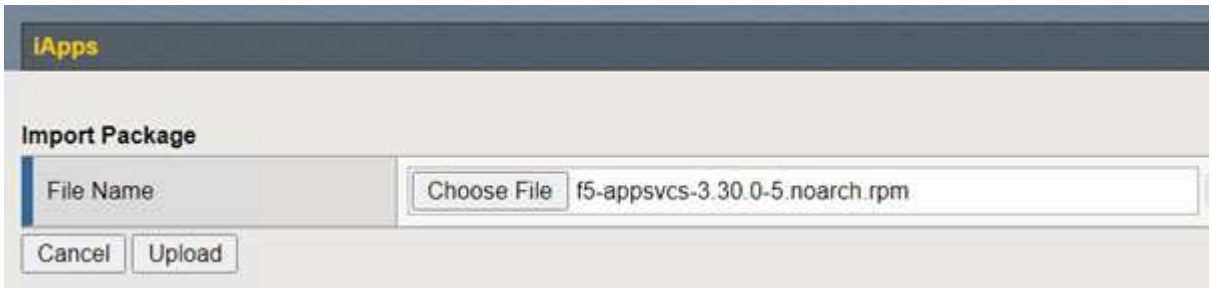
F5 BIG-IP 系統可以部署在專用硬體上、雲端或作為本地虛擬設備部署，版本高於 12.x，以便與 F5 CIS 整合。為了本文檔的目的，F5 BIG-IP 系統被驗證為虛擬設備，例如使用 BIG-IP VE 版本。

已驗證版本

科技	軟體版本
紅帽 OpenShift	4.6 EUS，4.7
F5 BIG-IP VE 版本	16.1.0
F5 容器入口服務	2.5.1
F5 IPAM 控制器	0.1.4
F5 AS3	3.30.0

安裝

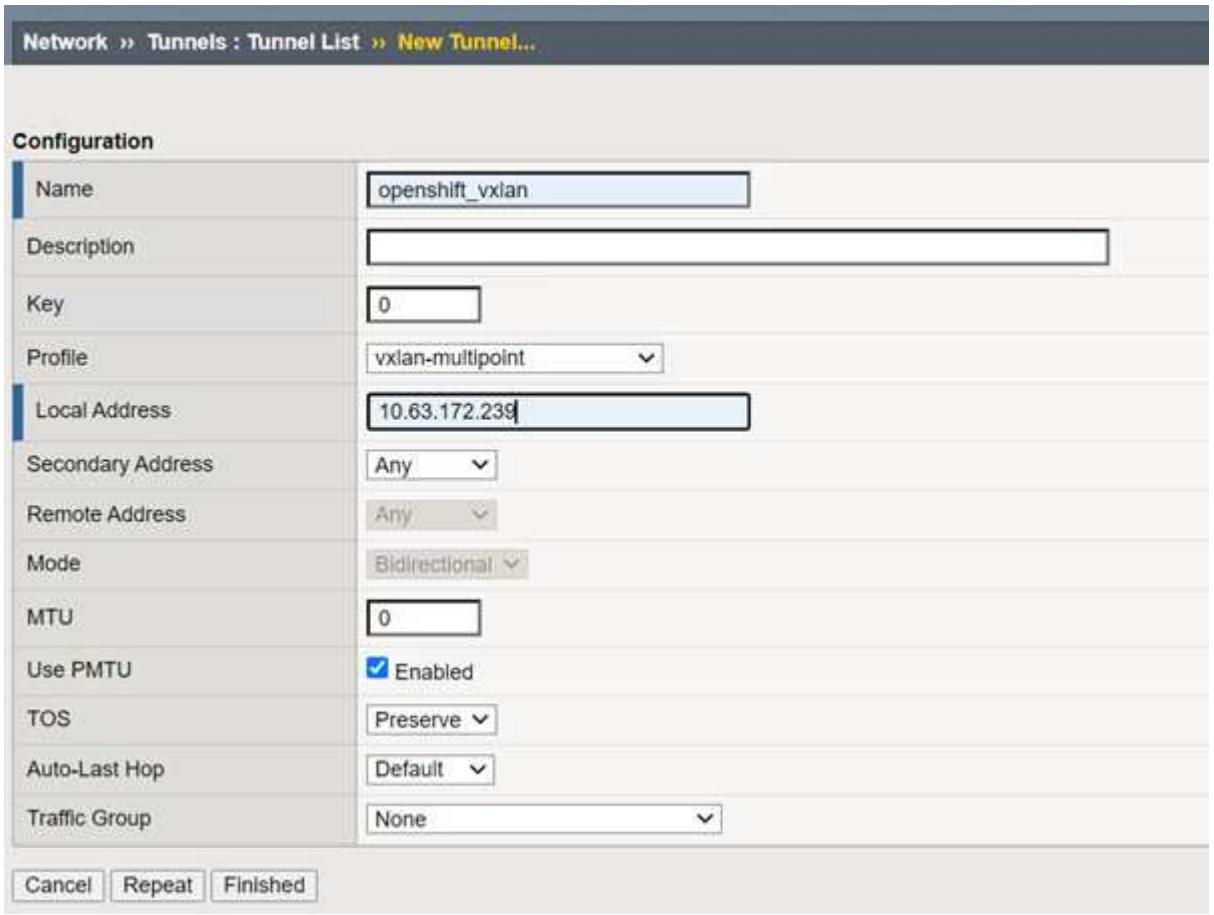
1. 安裝 F5 應用服務 3 擴充功能以允許 BIG-IP 系統接受 JSON 中的設定而不是命令。前往 "F5 AS3 GitHub 儲存庫"，並下載最新的RPM檔案。
2. 登入 F5 BIG-IP 系統，導覽至 iApps > Package Management LX 並點選 Import。
3. 點擊“選擇文件”並選擇下載的 AS3 RPM 文件，按一下“確定”，然後按一下“上傳”。



4. 確認 AS3 擴充功能已成功安裝。

5. 接下來配置OpenShift和BIG-IP系統之間通訊所需的資源。首先透過在 BIG-IP 系統上為 OpenShift SDN 建立 VXLAN 隧道介面來在 OpenShift 和 BIG-IP 伺服器之間建立隧道。導航至網路 > 隧道 > 設定文件，按一下創建，然後將父設定檔設定為 vxlan，將泛洪類型設定為多播。輸入設定檔的名稱並按一下“完成”。

6. 導覽至網路 > 隧道 > 隧道列表，按一下創建，然後輸入隧道的名稱和本機 IP 位址。選擇上一個步驟建立的隧道設定文件，然後按一下「完成」。



7. 使用叢集管理員權限登入 Red Hat OpenShift 叢集。
8. 在 OpenShift 上為 F5 BIG-IP 伺服器建立一個 hostsubnet，將子網路從 OpenShift 叢集擴展到 F5 BIG-IP 伺服器。下載主機子網路 YAML 定義。

```
wget https://github.com/F5Networks/k8s-bigip-
ctlr/blob/master/docs/config_examples/openshift/f5-kctlr-openshift-
hostsubnet.yaml
```

9. 編輯主機子網路檔案並為 OpenShift SDN 新增 BIG-IP VTEP（VXLAN 隧道）IP。

```
apiVersion: v1
kind: HostSubnet
metadata:
  name: f5-server
  annotations:
    pod.network.openshift.io/fixed-vnid-host: "0"
    pod.network.openshift.io/assign-subnet: "true"
# provide a name for the node that will serve as BIG-IP's entry into the
cluster
host: f5-server
# The hostIP address will be the BIG-IP interface address routable to
the
# OpenShift Origin nodes.
# This address is the BIG-IP VTEP in the SDN's VXLAN.
hostIP: 10.63.172.239
```

根據您環境的需求變更 hostIP 和其他詳細資訊。

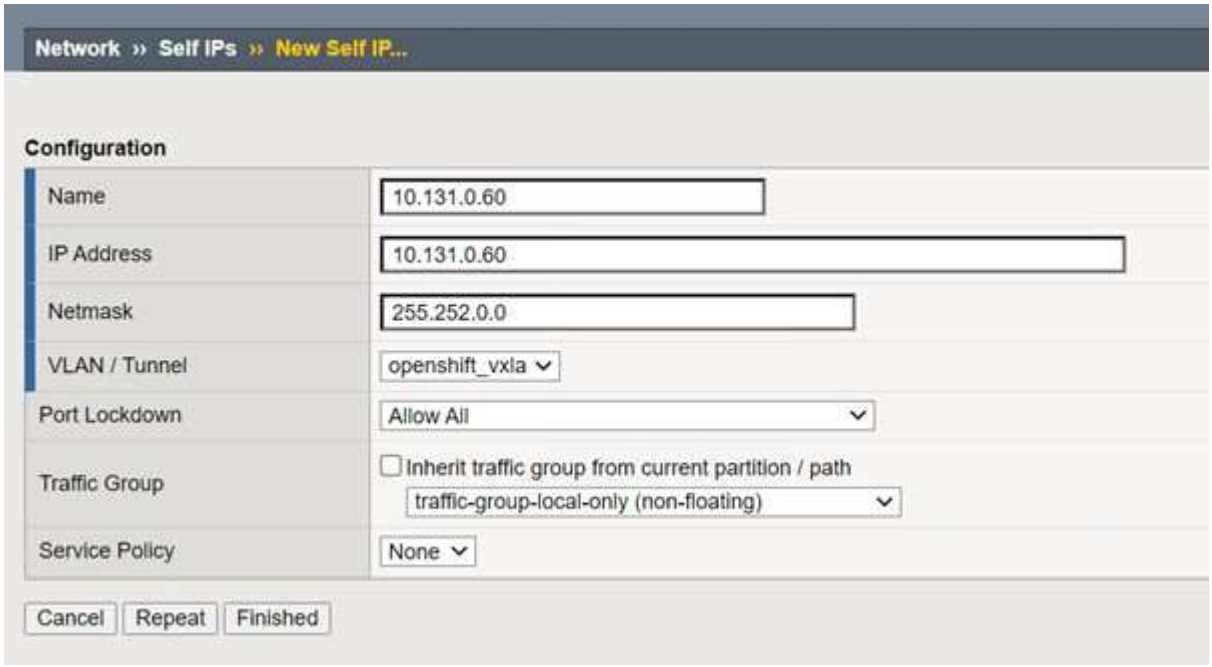
10. 建立 HostSubnet 資源。

```
[admin@rhel-7 ~]$ oc create -f f5-kctlr-openshift-hostsubnet.yaml

hostsubnet.network.openshift.io/f5-server created
```

11. 取得為 F5 BIG-IP 伺服器所建立的主機子網路的叢集 IP 子網路範圍。

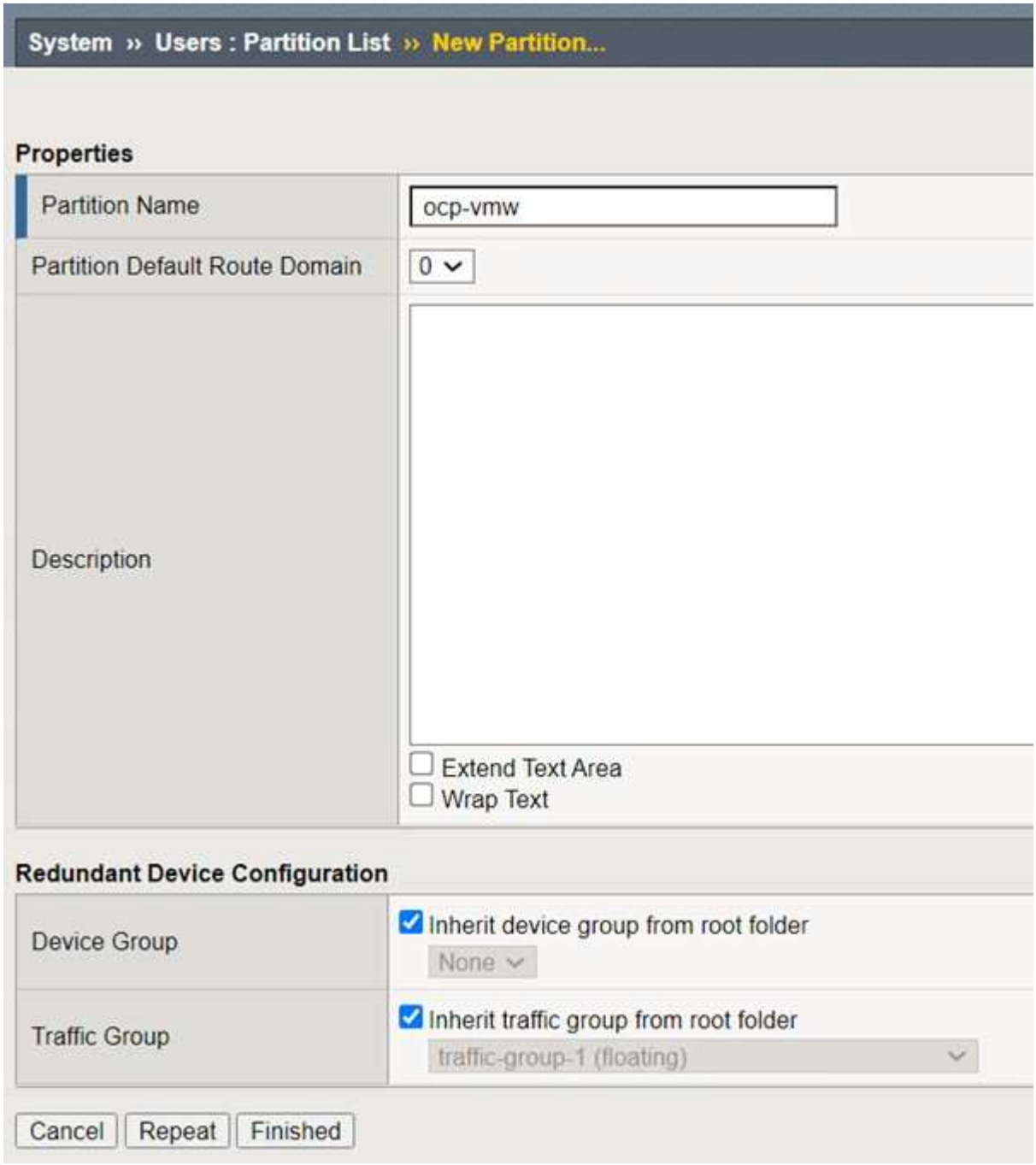
```
[admin@rhel-7 ~]$ oc get hostsubnet

NAME                           HOST                           HOST IP
SUBNET          EGRESS CIDRS   EGRESS IPS
f5-server                      f5-server                      10.63.172.239
10.131.0.0/23
ocp-vmw-nszws-master-0         ocp-vmw-nszws-master-0         10.63.172.44
10.128.0.0/23
ocp-vmw-nszws-master-1         ocp-vmw-nszws-master-1         10.63.172.47
10.130.0.0/23
ocp-vmw-nszws-master-2         ocp-vmw-nszws-master-2         10.63.172.48
10.129.0.0/23
ocp-vmw-nszws-worker-r8fh4     ocp-vmw-nszws-worker-r8fh4     10.63.172.7
10.130.2.0/23
ocp-vmw-nszws-worker-tvr46     ocp-vmw-nszws-worker-tvr46     10.63.172.11
10.129.2.0/23
ocp-vmw-nszws-worker-wdxhg     ocp-vmw-nszws-worker-wdxhg     10.63.172.24
10.128.2.0/23
ocp-vmw-nszws-worker-wg8r4     ocp-vmw-nszws-worker-wg8r4     10.63.172.15
10.131.2.0/23
ocp-vmw-nszws-worker-wtgfw     ocp-vmw-nszws-worker-wtgfw     10.63.172.17
10.128.4.0/23
```

12. 在 OpenShift VXLAN 上建立一個自身 IP，其 IP 位於與 F5 BIG-IP 伺服器對應的 OpenShift 主機子網路範圍內。登入 F5 BIG-IP 系統，導覽至網路 > 自有 IP，然後按一下建立。輸入為 F5 BIG-IP 主機子網路所建立的叢集 IP 子網路中的 IP，選擇 VXLAN 隧道，然後輸入其他詳細資料。然後按一下“完成”。



13. 在 F5 BIG-IP 系統中建立一個分區，以便與 CIS 一起設定和使用。導覽至系統 > 使用者 > 分區列表，按一

下創建，然後輸入詳細資訊。然後按一下“完成”。



F5 建議不要對 CIS 管理的分區進行手動設定。

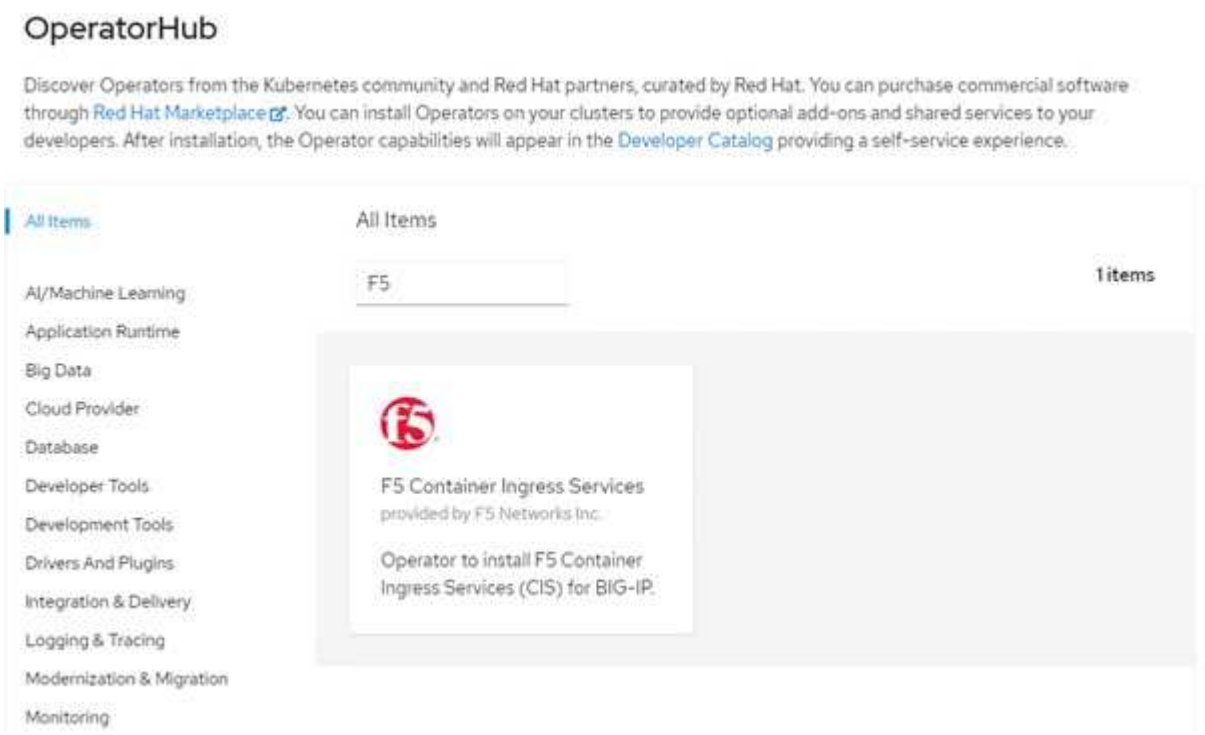
14. 使用來自 OperatorHub 的操作員安裝 F5 BIG-IP CIS。使用 cluster-admin 權限登入 Red Hat OpenShift 集群，並使用 F5 BIG-IP 系統登入憑證建立一個 secret，這是操作員的先決條件。

```
[admin@rhel-7 ~]$ oc create secret generic bigip-login -n kube-system
--from-literal=username=admin --from-literal=password=admin

secret/bigip-login created
```

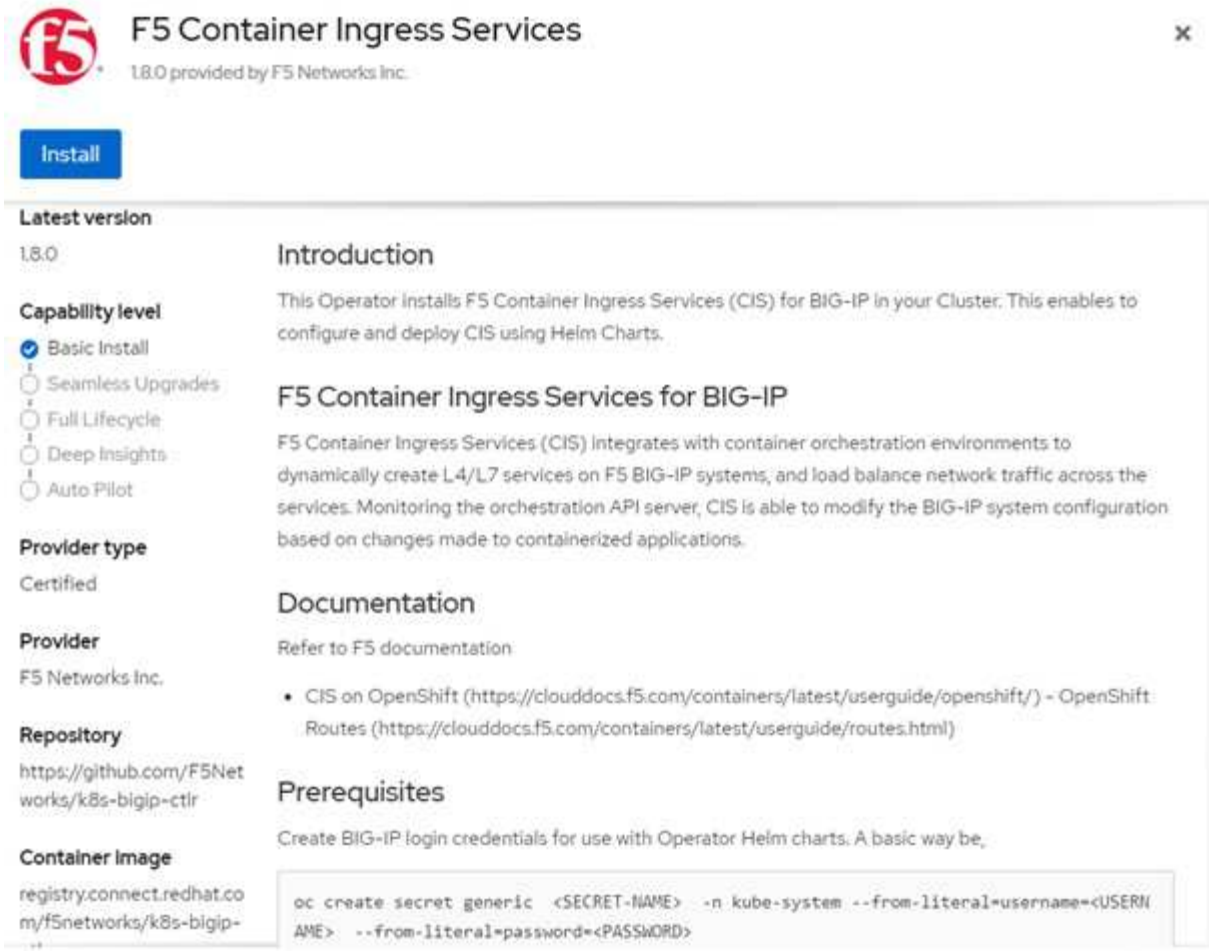
15. 安裝 F5 CIS CRD。

```
[admin@rhel-7 ~]$ oc apply -f
https://raw.githubusercontent.com/F5Networks/k8s-bigip-
ctlr/master/docs/config_examples/crd/Install/customresourcedefinitions.y
ml

customresourcedefinition.apiextensions.k8s.io/virtualservers.cis.f5.com
created
customresourcedefinition.apiextensions.k8s.io/tlsprofiles.cis.f5.com
created
customresourcedefinition.apiextensions.k8s.io/transportservers.cis.f5.co
m created
customresourcedefinition.apiextensions.k8s.io/externaldnss.cis.f5.com
created
customresourcedefinition.apiextensions.k8s.io/ingresslinks.cis.f5.com
created
```

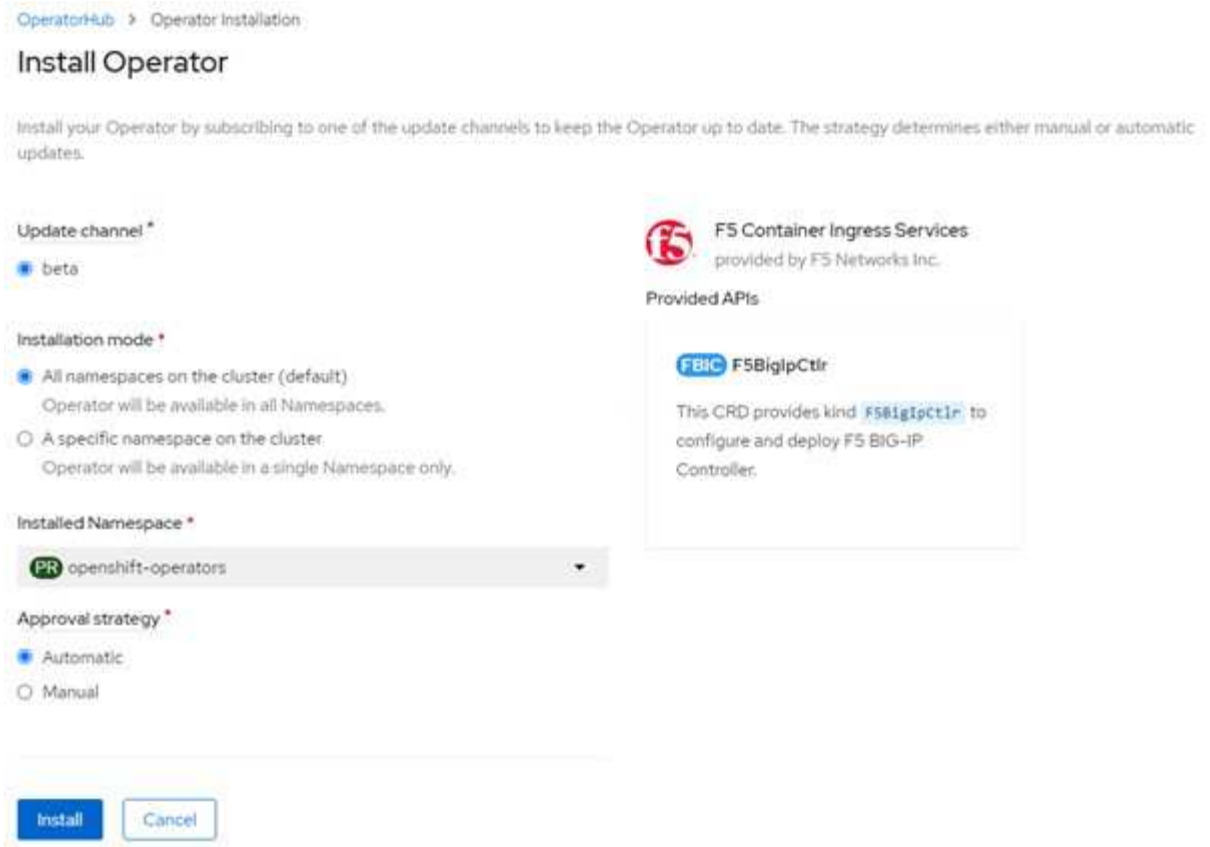
16. 導覽至 Operators > OperatorHub，搜尋關鍵字 F5，然後按一下 F5 Container Ingress Service 磁貼。



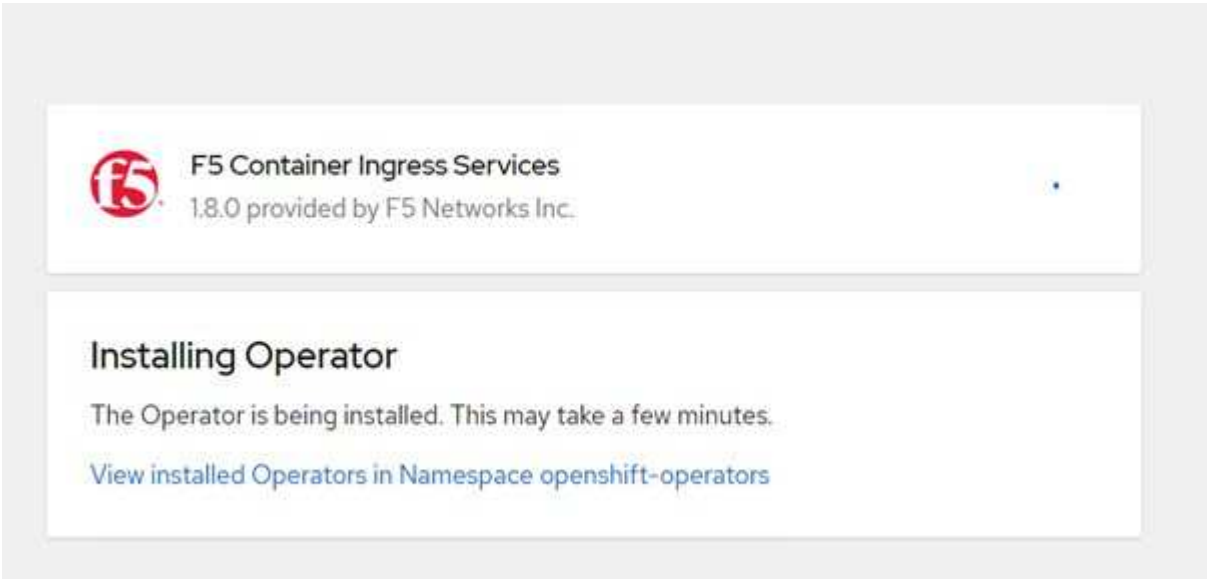
17. 閱讀操作員資訊並點擊“安裝”。



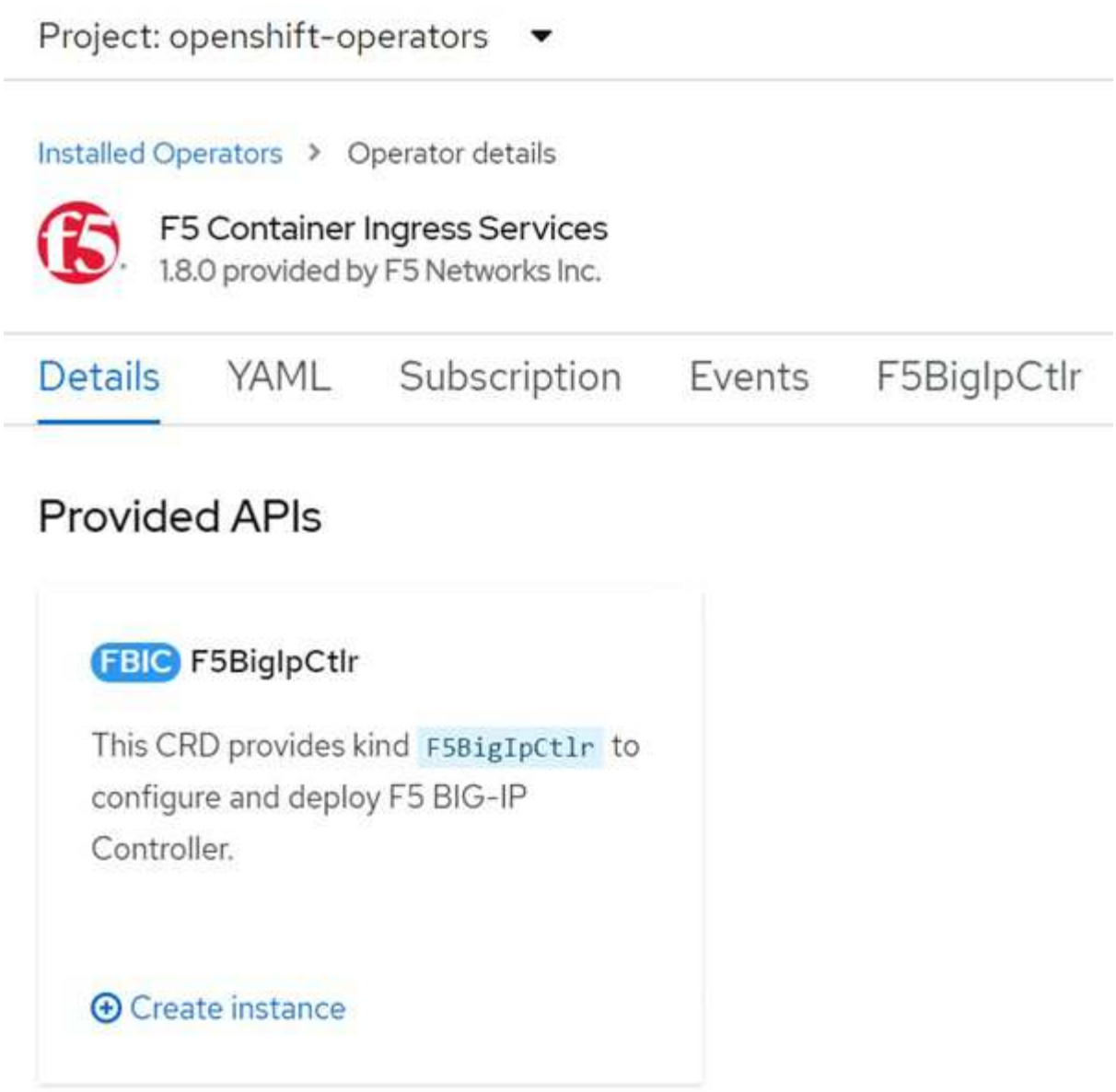
18. 在安裝操作員畫面上，保留所有預設參數，然後按一下安裝。



19. 安裝操作員需要一段時間。



20. 操作員安裝完成後，將顯示「安裝成功」訊息。
21. 導覽至 Operators > Installed Operators，按一下 F5 Container Ingress Service，然後按一下 F5BigIpCtlr 圖塊下的 Create Instance。



22. 點選 YAML View，更新必要的參數後貼上以下內容。

> 更新參數 `bigip_partition`，` openshift_sdn_name`， `bigip_url`和 `bigip_login_secret` 在複製內容之前，請先查看以下設定的值。

```
apiVersion: cis.f5.com/v1
kind: F5BigIpCtlr
metadata:
  name: f5-server
  namespace: openshift-operators
spec:
  args:
    log_as3_response: true
    agent: as3
    log_level: DEBUG
    bigip_partition: ocp-vmw
    openshift_sdn_name: /Common/openshift_vxlan
    bigip_url: 10.61.181.19
    insecure: true
    pool-member-type: cluster
    custom_resource_mode: true
    as3_validation: true
    ipam: true
    manage_configmaps: true
  bigip_login_secret: bigip-login
  image:
    pullPolicy: Always
    repo: f5networks/cntr-ingress-svcs
    user: registry.connect.redhat.com
  namespace: kube-system
  rbac:
    create: true
  resources: {}
  serviceAccount:
    create: true
  version: latest
```

23. 貼上此內容後，按一下「建立」。這會在 kube-system 命名空間中安裝 CIS pod。

預設情況下，Red Hat OpenShift 提供了一種透過路由公開服務以實現 L7 負載平衡的方法。內建的 OpenShift 路由器負責宣傳和處理這些路由的流量。但是，您也可以設定 F5 CIS 以透過外部 F5 BIG-IP 系統支援路由，該系統可以作為輔助路由器運行，也可以作為自架 OpenShift 路由器的替代品運行。 CIS 在 BIG-IP 系統中建立一個虛擬伺服器，充當 OpenShift 路由的路由器，而 BIG-IP 負責處理廣告和流量路由。有關啟用此功能的參數的信息，請參閱此處的文檔。請注意，這些參數是在 apps/v1 API 中為 OpenShift Deployment 資源定義的。因此，當將這些與 F5BigIpCtlr 資源 cis.f5.com/v1 API 一起使用時，請將參數名稱中的連字符 (-) 替換為下劃線 (_)。

24. 傳遞給 CIS 資源所建立的參數包括 `ipam: true`和 `custom_resource_mode: true`。這些參數是啟用 CIS 與 IPAM 控制器整合所必需的。透過建立 F5 IPAM 資源來驗證 CIS 是否已啟用 IPAM 整合。

```
[admin@rhel-7 ~]$ oc get f5ipam -n kube-system

NAMESPACE   NAME                                AGE
kube-system   ipam.10.61.181.19.ocp-vmw       43s
```

25. 建立 F5 IPAM 控制器所需的服務帳戶、角色和角色綁定。建立一個 YAML 檔案並貼上以下內容。

```
[admin@rhel-7 ~]$ vi f5-ipam-rbac.yaml

kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: ipam-ctlr-clusterrole
rules:
  - apiGroups: ["fic.f5.com"]
    resources: ["ipams","ipams/status"]
    verbs: ["get", "list", "watch", "update", "patch"]
---
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: ipam-ctlr-clusterrole-binding
  namespace: kube-system
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: ipam-ctlr-clusterrole
subjects:
  - apiGroup: ""
    kind: ServiceAccount
    name: ipam-ctlr
    namespace: kube-system
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: ipam-ctlr
  namespace: kube-system
```

26. 創建資源。

```
[admin@rhel-7 ~]$ oc create -f f5-ipam-rbac.yaml

clusterrole.rbac.authorization.k8s.io/ipam-ctlr-clusterrole created
clusterrolebinding.rbac.authorization.k8s.io/ipam-ctlr-clusterrole-
binding created
serviceaccount/ipam-ctlr created
```

27. 建立一個 YAML 檔案並貼上下面提供的 F5 IPAM 部署定義。

更新下面的 spec.template.spec.containers[0].args 中的 ip-range 參數以反映與您的設定相對應的 ipamLabels 和 IP 位址範圍。

ipam標籤[`range1`和 `range2`在下方的範例中] 需要為 LoadBalancer 類型的服務進行註釋，以便 IPAM 控制器從定義的範圍內偵測並指派 IP 位址。

```
[admin@rhel-7 ~]$ vi f5-ipam-deployment.yaml

apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    name: f5-ipam-controller
  name: f5-ipam-controller
  namespace: kube-system
spec:
  replicas: 1
  selector:
    matchLabels:
      app: f5-ipam-controller
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: f5-ipam-controller
    spec:
      containers:
      - args:
        - --orchestration=openshift
        - --ip-range='{"range1":"10.63.172.242-10.63.172.249",
"range2":"10.63.170.111-10.63.170.129"}'
        - --log-level=DEBUG
        command:
        - /app/bin/f5-ipam-controller
        image: registry.connect.redhat.com/f5networks/f5-ipam-
controller:latest
        imagePullPolicy: IfNotPresent
        name: f5-ipam-controller
      dnsPolicy: ClusterFirst
      restartPolicy: Always
      schedulerName: default-scheduler
      securityContext: {}
      serviceAccount: ipam-ctlr
      serviceAccountName: ipam-ctlr
```

28. 建立 F5 IPAM 控制器部署。

```
[admin@rhel-7 ~]$ oc create -f f5-ipam-deployment.yaml

deployment/f5-ipam-controller created
```

29. 驗證 F5 IPAM 控制器 pod 是否正在運作。

```
[admin@rhel-7 ~]$ oc get pods -n kube-system

NAME                                      READY   STATUS    RESTARTS
AGE
f5-ipam-controller-5986cff5bd-2bvn6       1/1     Running   0
30s
f5-server-f5-bigip-ctlr-5d7578667d-qxdgj  1/1     Running   0
14m
```

30. 建立 F5 IPAM 模式。

```
[admin@rhel-7 ~]$ oc create -f
https://raw.githubusercontent.com/F5Networks/f5-ipam-
controller/main/docs/_static/schemas/ipam_schema.yaml

customresourcedefinition.apiextensions.k8s.io/ipams.fic.f5.com
```

確認

1. 建立 LoadBalancer 類型的服務

```
[admin@rhel-7 ~]$ vi example_svc.yaml

apiVersion: v1
kind: Service
metadata:
  annotations:
    cis.f5.com/ipamLabel: range1
  labels:
    app: f5-demo-test
  name: f5-demo-test
  namespace: default
spec:
  ports:
  - name: f5-demo-test
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: f5-demo-test
  sessionAffinity: None
  type: LoadBalancer
```

```
[admin@rhel-7 ~]$ oc create -f example_svc.yaml

service/f5-demo-test created
```

2. 檢查 IPAM 控制器是否為其指派了外部 IP。

```
[admin@rhel-7 ~]$ oc get svc

NAME           TYPE           CLUSTER-IP       EXTERNAL-IP
PORT(S)        AGE
f5-demo-test   LoadBalancer   172.30.210.108   10.63.172.242
80:32605/TCP   27s
```

3. 建立部署並使用建立的 LoadBalancer 服務。

```
[admin@rhel-7 ~]$ vi example_deployment.yaml

apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: f5-demo-test
  name: f5-demo-test
spec:
  replicas: 2
  selector:
    matchLabels:
      app: f5-demo-test
  template:
    metadata:
      labels:
        app: f5-demo-test
    spec:
      containers:
      - env:
        - name: service_name
          value: f5-demo-test
        image: nginx
        imagePullPolicy: Always
        name: f5-demo-test
        ports:
        - containerPort: 80
          protocol: TCP
```

```
[admin@rhel-7 ~]$ oc create -f example_deployment.yaml

deployment/f5-demo-test created
```

4. 檢查 pod 是否正在運作。

```
[admin@rhel-7 ~]$ oc get pods

NAME                            READY   STATUS    RESTARTS   AGE
f5-demo-test-57c46f6f98-47wwp   1/1     Running   0          27s
f5-demo-test-57c46f6f98-cl2m8   1/1     Running   0          27s
```

5. 檢查BIG-IP系統中是否為OpenShift中LoadBalancer類型的服務建立了對應的虛擬伺服器。導覽至本機流量 > 虛擬伺服器 > 虛擬伺服器清單。

建立私有鏡像倉庫

對於大多數 Red Hat OpenShift 部署，使用公開註冊表 "Quay.io"或者 "DockerHub"滿足大多數客戶的需求。然而，有時客戶可能希望託管他們自己的私人或定製圖像。

此程序記錄如何建立由Trident和NetApp ONTAP提供的持久磁碟區支援的私人映像註冊表。

Trident Protect 需要一個登錄機碼來託管Astra容器所需的映像。以下部分介紹在 Red Hat OpenShift 叢集上設定私人註冊表以及推送支援安裝Trident Protect 所需的映像的步驟。

建立私有鏡像倉庫

1. 從目前預設儲存類別中刪除預設註釋，並將 Trident 支援的儲存類別註釋為 OpenShift 叢集的預設儲存類別。

```
[netapp-user@rhel7 ~]$ oc patch storageclass thin -p '{"metadata":
{"annotations": {"storageclass.kubernetes.io/is-default-class":
"false"}}}'
storageclass.storage.k8s.io/thin patched

[netapp-user@rhel7 ~]$ oc patch storageclass ocp-trident -p
'{"metadata": {"annotations": {"storageclass.kubernetes.io/is-default-
class": "true"}}}'
storageclass.storage.k8s.io/ocp-trident patched
```

2. 透過在以下位置輸入以下儲存參數來編輯 imageregistry 操作符 `spec`部分。

```
[netapp-user@rhel7 ~]$ oc edit
configs.imageregistry.operator.openshift.io

storage:
  pvc:
    claim:
```

3. 在 `spec`用於建立具有自訂主機名稱的 OpenShift 路由的部分。儲存並退出。

```
routes:
  - hostname: astra-registry.apps.ocp-vmw.cie.netapp.com
    name: netapp-astra-route
```

當您想要為路由指定自訂主機名稱時，可以使用上述路由配置。如果您希望 OpenShift 建立具有預設主機名稱的路由，則可以將下列參數新增至 `spec`部分： `defaultRoute: true 。

自訂 TLS 證書

當您使用自訂主機名稱進行路由時，預設情況下，它會使用 OpenShift Ingress 操作員的預設 TLS 配置。但是，您可以向路由新增自訂 TLS 配置。為此，請完成以下步驟。

a. 使用路由的 TLS 憑證和金鑰建立一個秘密。

```
[netapp-user@rhel7 ~]$ oc create secret tls astra-route-tls -n
openshift-image-registry –cert/home/admin/netapp-astra/tls.crt
--key=/home/admin/netapp-astra/tls.key
```

b. 編輯 imageregistry 操作符，新增下列參數 `spec`部分。

```
[netapp-user@rhel7 ~]$ oc edit
configs.imageregistry.operator.openshift.io

routes:
  - hostname: astra-registry.apps.ocp-vmw.cie.netapp.com
    name: netapp-astra-route
    secretName: astra-route-tls
```

4. 再次編輯imageregistry Operator，將Operator的管理狀態改為 `Managed`狀態。儲存並退出。

```
oc edit configs.imageregistry/cluster

managementState: Managed
```

5. 如果滿足所有先決條件，則會為私有鏡像註冊表建立 PVC、pod 和服務。幾分鐘後，註冊表就會啟動。

```
[netapp-user@rhel7 ~]$oc get all -n openshift-image-registry

NAME                                                    READY   STATUS
```

```
RESTARTS   AGE
pod/cluster-image-registry-operator-74f6d954b6-rb7zr   1/1     Running
3          90d
pod/image-pruner-1627257600-f5cpj                      0/1     Completed
0          2d9h
pod/image-pruner-1627344000-swqx9                      0/1     Completed
0          33h
pod/image-pruner-1627430400-rv5nt                      0/1     Completed
0          9h
pod/image-registry-6758b547f-6pnj8                     1/1     Running
0          76m
pod/node-ca-bwb5r                                      1/1     Running
0          90d
pod/node-ca-f8w54                                      1/1     Running
0          90d
pod/node-ca-gjx7h                                      1/1     Running
0          90d
pod/node-ca-lcx4k                                      1/1     Running
0          33d
pod/node-ca-v7zmx                                      1/1     Running
0          7d21h
pod/node-ca-xpppp                                      1/1     Running
0          89d

NAME                              TYPE        CLUSTER-IP       EXTERNAL-
IP   PORT(S)     AGE
service/image-registry            ClusterIP   172.30.196.167   <none>
5000/TCP    15h
service/image-registry-operator   ClusterIP   None             <none>
60000/TCP   90d

NAME                     DESIRED   CURRENT   READY   UP-TO-DATE
AVAILABLE   NODE SELECTOR            AGE
daemonset.apps/node-ca   6         6         6       6            6
kubernetes.io/os=linux   90d

NAME                                           READY   UP-TO-DATE
AVAILABLE   AGE
deployment.apps/cluster-image-registry-operator   1/1     1            1
90d
deployment.apps/image-registry                    1/1     1            1
15h

NAME                                                      DESIRED
CURRENT   READY   AGE
replicaset.apps/cluster-image-registry-operator-74f6d954b6   1         1
```

```
1       90d
replicaset.apps/image-registry-6758b547f                    1         1
1       76m
replicaset.apps/image-registry-78bfbd7f59                   0         0
0       15h
replicaset.apps/image-registry-7fcc8d6cc8                   0         0
0       80m
replicaset.apps/image-registry-864f88f5b                    0         0
0       15h
replicaset.apps/image-registry-cb47fffb                     0         0
0       10h

NAME                                COMPLETIONS   DURATION   AGE
job.batch/image-pruner-1627257600   1/1           10s        2d9h
job.batch/image-pruner-1627344000   1/1           6s         33h
job.batch/image-pruner-1627430400   1/1           5s         9h

NAME                          SCHEDULE    SUSPEND   ACTIVE   LAST
SCHEDULE   AGE
cronjob.batch/image-pruner   0 0 * * *   False     0        9h
90d

NAME                                     HOST/PORT
PATH   SERVICES          PORT    TERMINATION   WILDCARD
route.route.openshift.io/public-routes   astra-registry.apps.ocp-
vmw.cie.netapp.com          image-registry   <all>   reencrypt     None
```

6. 如果您正在為入口操作員 OpenShift 登錄路由使用預設 TLS 證書，則可以使用下列命令取得 TLS 證書。

```
[netapp-user@rhel7 ~]$ oc extract secret/router-ca --keys=tls.crt -n
openshift-ingress-operator
```

7. 為了允許 OpenShift 節點存取並從登錄中提取映像，請將憑證新增至 OpenShift 節點上的 docker 用戶端。在 `openshift-config`命名空間使用 TLS 憑證並將其修補到叢集映像配置以使憑證受信任。

```
[netapp-user@rhel7 ~]$ oc create configmap astra-ca -n openshift-config
--from-file=astra-registry.apps.ocp-vmw.cie.netapp.com=tls.crt

[netapp-user@rhel7 ~]$ oc patch image.config.openshift.io/cluster
--patch '{"spec":{"additionalTrustedCA":{"name":"astra-ca"}}}'
--type=merge
```

8. OpenShift 內部註冊表由身份驗證控制。所有OpenShift使用者都可以存取OpenShift登錄表，但登入使用者可以執行的操作取決於使用者權限。

a. 若要允許使用者或一組使用者從登錄中提取圖像，必須為使用者指派 registry-viewer 角色。

```
[netapp-user@rhel7 ~]$ oc policy add-role-to-user registry-viewer
ocp-user

[netapp-user@rhel7 ~]$ oc policy add-role-to-group registry-viewer
ocp-user-group
```

b. 若要允許使用者或使用者群組寫入或推送映像，必須為使用者指派註冊表編輯器角色。

```
[netapp-user@rhel7 ~]$ oc policy add-role-to-user registry-editor
ocp-user

[netapp-user@rhel7 ~]$ oc policy add-role-to-group registry-editor
ocp-user-group
```

9. 為了讓 OpenShift 節點存取登錄機碼並推送或拉取映像，您需要設定拉取金鑰。

```
[netapp-user@rhel7 ~]$ oc create secret docker-registry astra-registry-
credentials --docker-server=astra-registry.apps.ocp-vmw.cie.netapp.com
--docker-username=ocp-user --docker-password=password
```

10. 然後可以將這個拉取機密修補到服務帳戶或在對應的 pod 定義中引用。

a. 若要將其修補到服務帳戶，請執行下列命令。

```
[netapp-user@rhel7 ~]$ oc secrets link <service_account_name> astra-
registry-credentials --for=pull
```

b. 若要在 pod 定義中引用 pull secret，請將下列參數新增至 `spec`部分。

```
imagePullSecrets:
  - name: astra-registry-credentials
```

11. 若要從 OpenShift 節點以外的工作站推送或拉取映像，請完成下列步驟。

a. 將 TLS 憑證新增至 docker 用戶端。

```
[netapp-user@rhel7 ~]$ sudo mkdir /etc/docker/certs.d/astra-
registry.apps.ocp-vmw.cie.netapp.com

[netapp-user@rhel7 ~]$ sudo cp /path/to/tls.crt
/etc/docker/certs.d/astra-registry.apps.ocp-vmw.cie.netapp.com
```

b. 使用 oc login 指令登入 OpenShift。

```
[netapp-user@rhel7 ~]$ oc login --token=sha256~D49SpB_lesSrJYwrM0LIO
-VRcjWHu0a27vKa0 --server=https://api.ocp-vmw.cie.netapp.com:6443
```

c. 使用 OpenShift 使用者憑證透過 podman/docker 指令登入註冊表。

podman

```
[netapp-user@rhel7 ~]$ podman login astra-registry.apps.ocp-
vmw.cie.netapp.com -u kubeadmin -p $(oc whoami -t) --tls
-verify=false
```

+ 注意：如果您使用 `kubeadmin`使用者登入私有註冊中心，然後使用令牌而不是密碼。

碼頭工人

```
[netapp-user@rhel7 ~]$ docker login astra-registry.apps.ocp-
vmw.cie.netapp.com -u kubeadmin -p $(oc whoami -t)
```

+ 注意：如果您使用 `kubeadmin`使用者登入私有註冊中心，然後使用令牌而不是密碼。

d. 推送或拉取映像。

podman

```
[netapp-user@rhel7 ~]$ podman push astra-registry.apps.ocp-
vmw.cie.netapp.com/netapp-astra/vault-controller:latest
[netapp-user@rhel7 ~]$ podman pull astra-registry.apps.ocp-
vmw.cie.netapp.com/netapp-astra/vault-controller:latest
```

碼頭工人

```
[netapp-user@rhel7 ~]$ docker push astra-registry.apps.ocp-
vmw.cie.netapp.com/netapp-astra/vault-controller:latest
[netapp-user@rhel7 ~]$ docker pull astra-registry.apps.ocp-
vmw.cie.netapp.com/netapp-astra/vault-controller:latest
```

解決方案驗證和用例

解決方案驗證與用例：**Red Hat OpenShift** 與**NetApp**

本頁提供的範例是 Red Hat OpenShift 與NetApp的解決方案驗證和用例。

- "使用持久性儲存部署 Jenkins CI/CD 管道"
- "使用NetApp在 Red Hat OpenShift 上設定多租戶"
- "搭載NetApp ONTAP 的Red Hat OpenShift 虛擬化"
- "NetApp協助 Red Hat OpenShift 上 Kubernetes 的高階叢集管理"

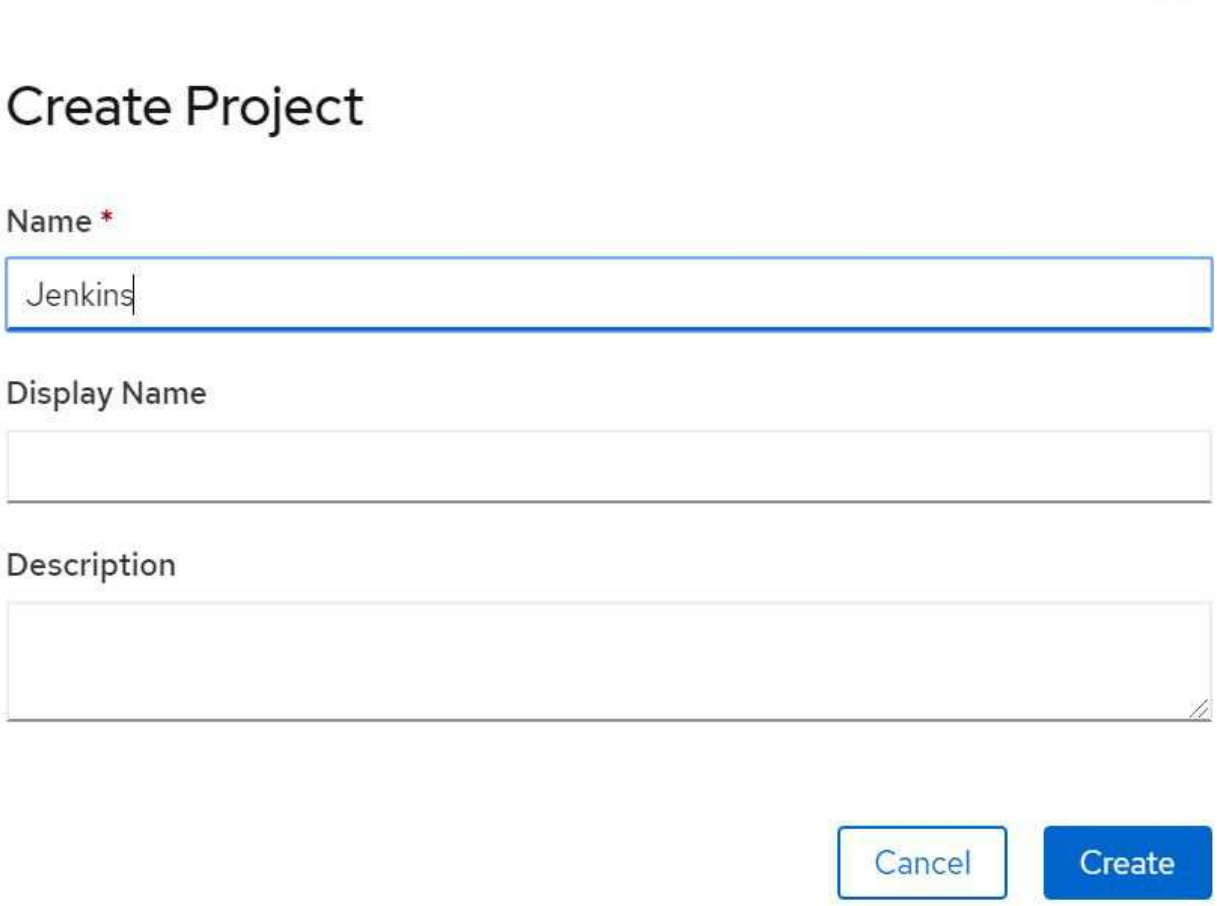
使用持久性儲存部署 **Jenkins CI/CD** 管道：**Red Hat OpenShift** 與**NetApp**

本節提供使用 Jenkins 部署持續整合/持續交付或部署 (CI/CD) 管道以驗證解決方案運作的步驟。

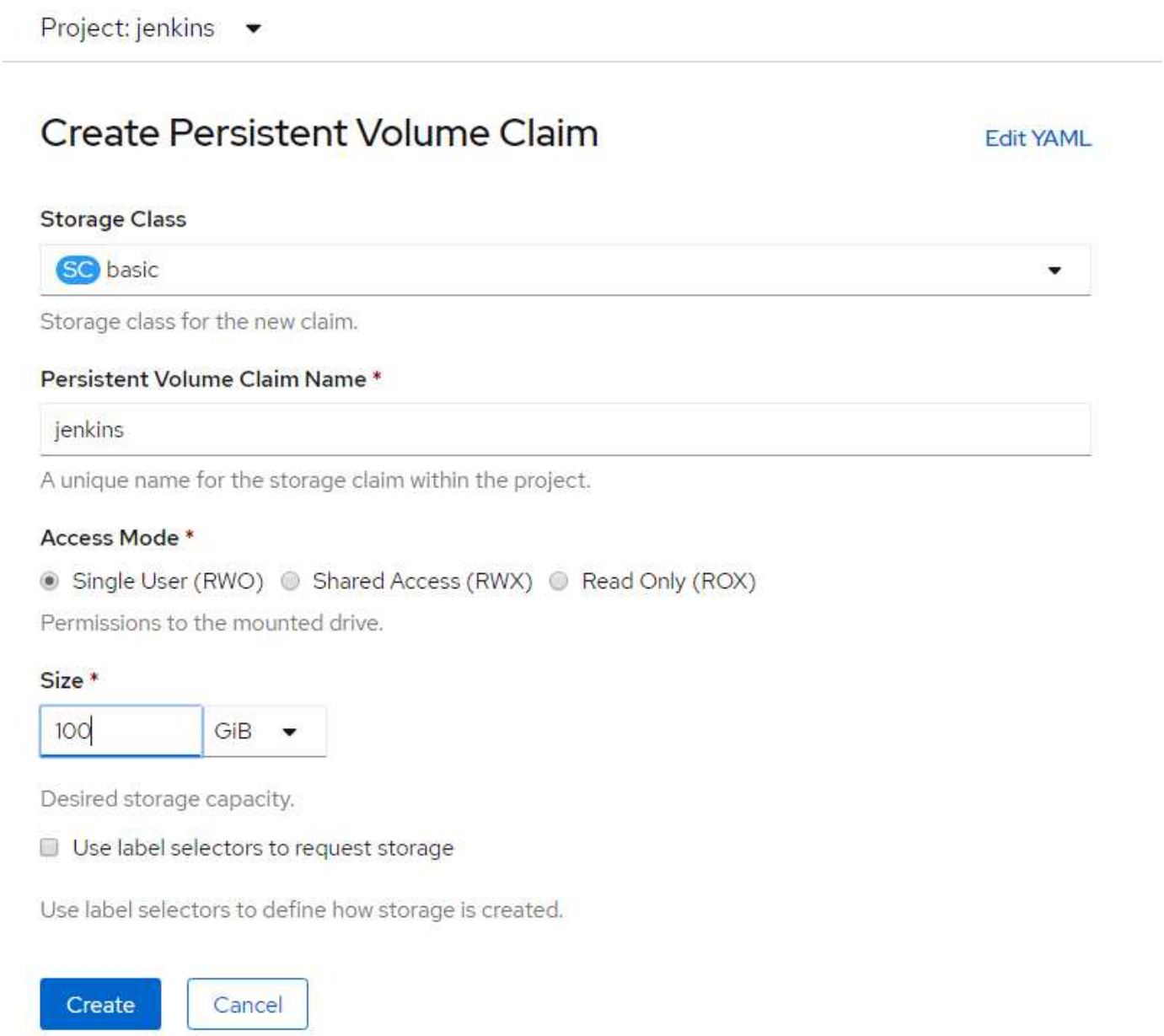
建立**Jenkins**部署所需的資源

若要建立部署 Jenkins 應用程式所需的資源，請完成以下步驟：

1. 建立一個名為 Jenkins 的新專案。



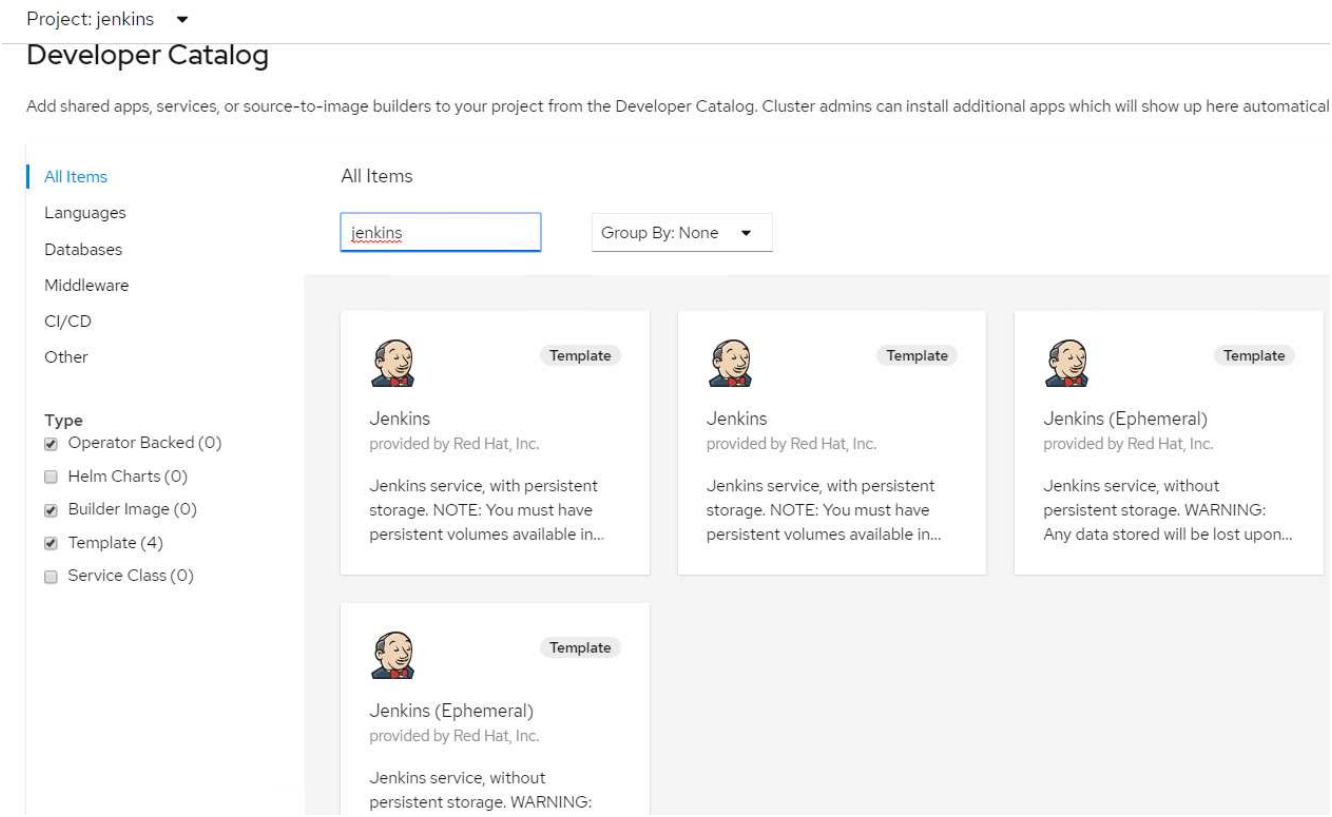
2. 在這個例子中，我們部署了具有持久性儲存的 Jenkins。為了支持 Jenkins 構建，請建立 PVC。導覽至儲存 > 持久性卷聲明，然後按一下建立持久性卷聲明。選擇剛剛建立的儲存類，確保持久性磁碟區聲明名稱為jenkins，選擇適當的大小和存取模式，然後按一下建立。



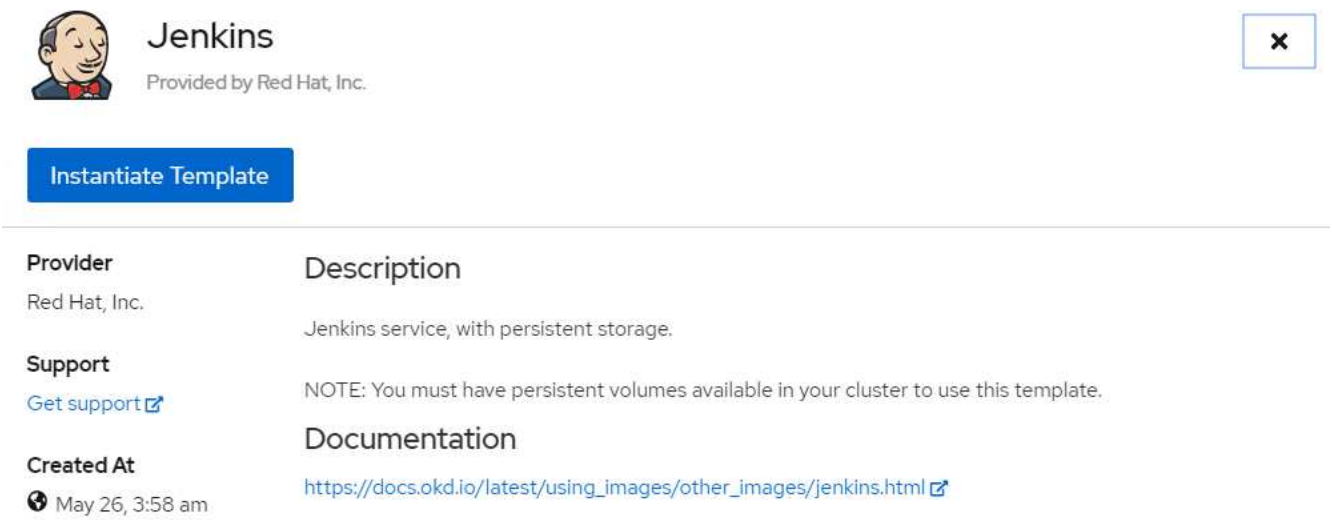
使用持久性儲存部署 **Jenkins**

若要使用持久性儲存部署 Jenkins，請完成下列步驟：

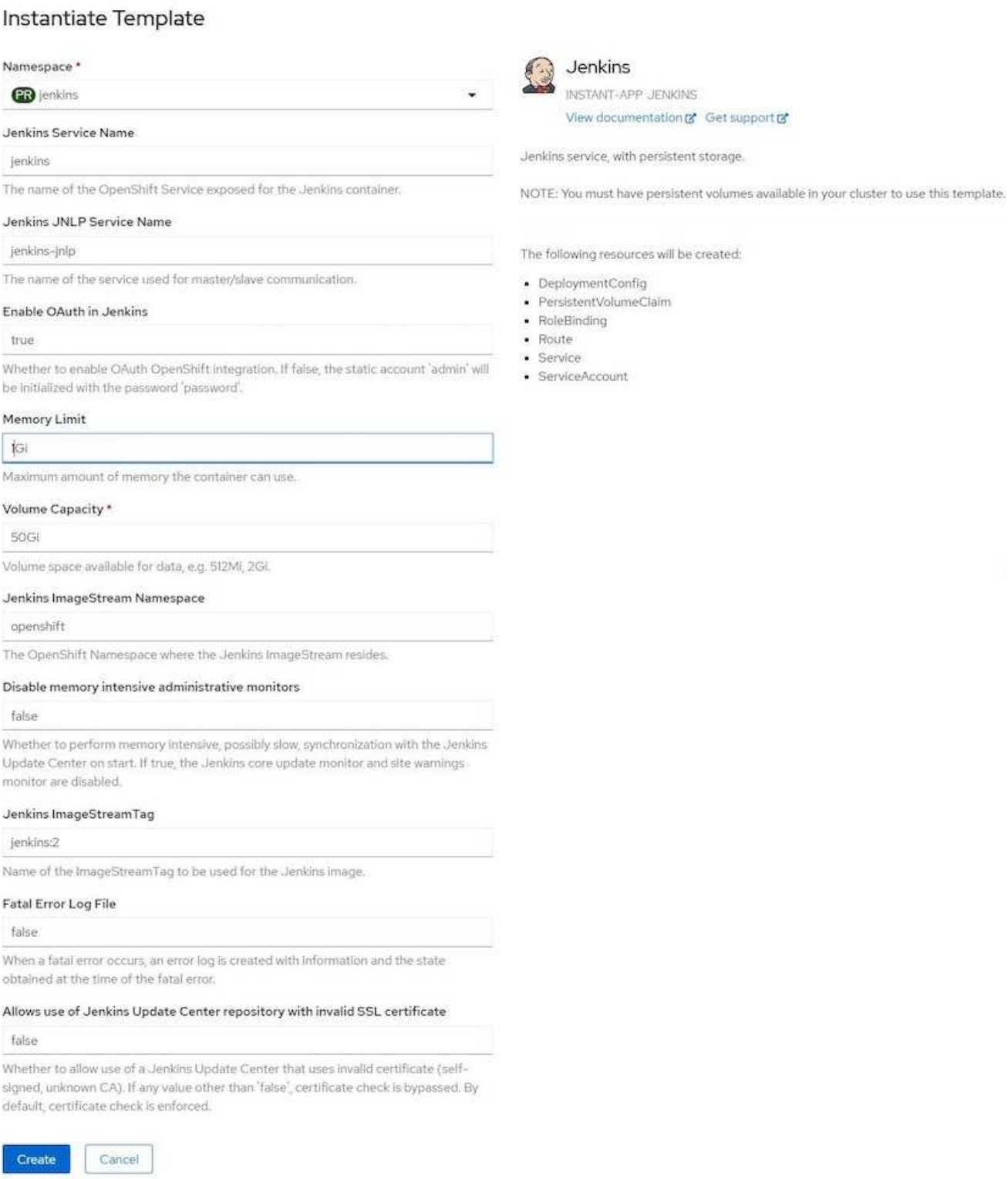
1. 在左上角，將角色從管理員變更為開發人員。點擊 +新增並選擇來自目錄。在按關鍵字過濾欄中，搜尋 jenkins。選擇具有持久性儲存的 Jenkins 服務。



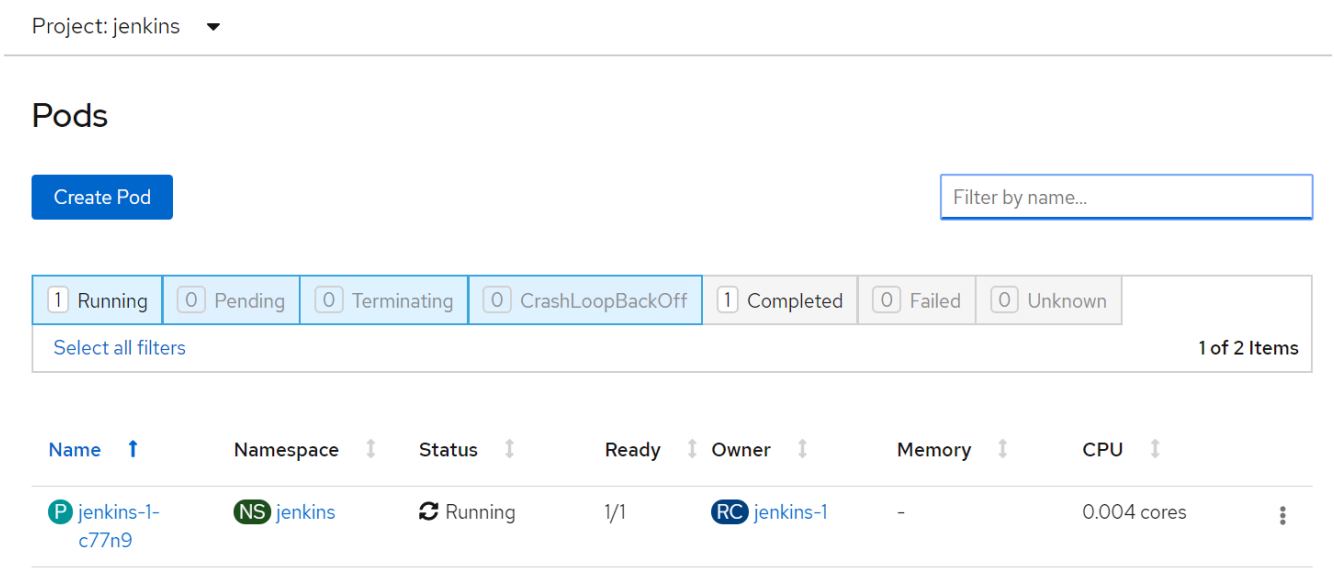
2. 點選 `Instantiate Template`。



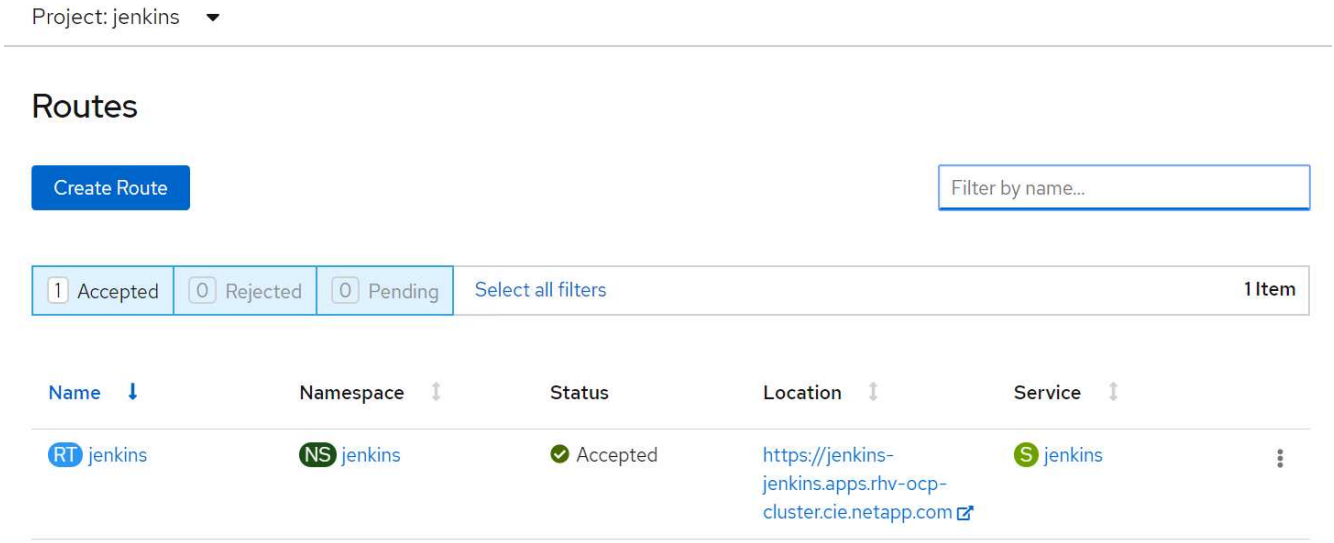
3. 預設情況下，會填入 Jenkins 應用程式的詳細資訊。根據您的需求，修改參數並點擊"建立"。此過程創建了在 OpenShift 上支援 Jenkins 所需的所有資源。



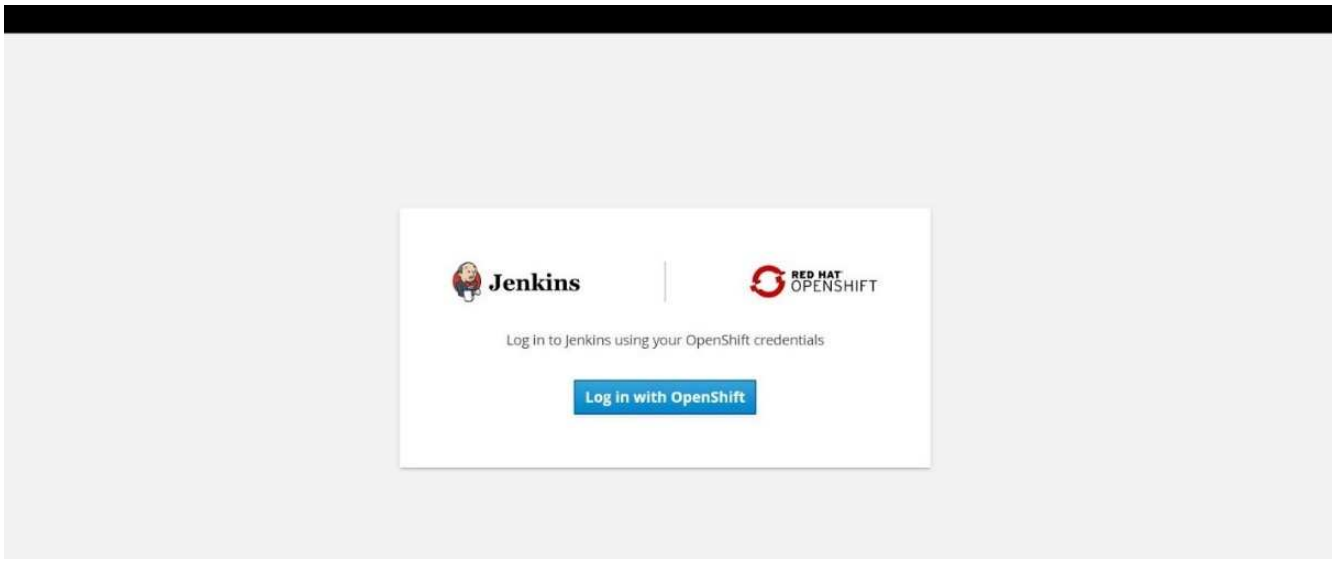
4. Jenkins pod 大約需要 10 到 12 分鐘才能進入就緒狀態。



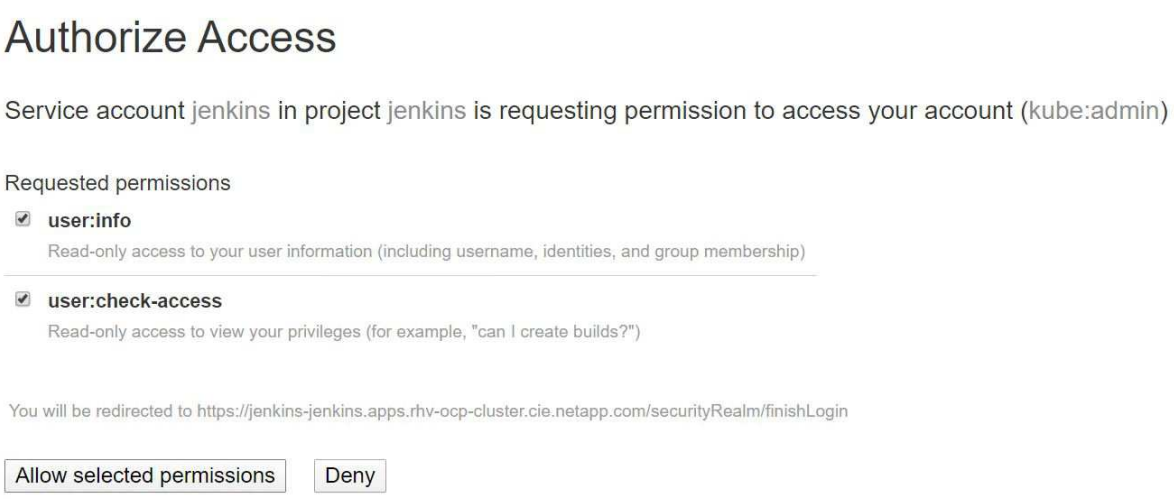
5. 實例化 Pod 後，導覽至「網路」>「路由」。若要開啟 Jenkins 網頁，請點選 jenkins 路由提供的 URL。



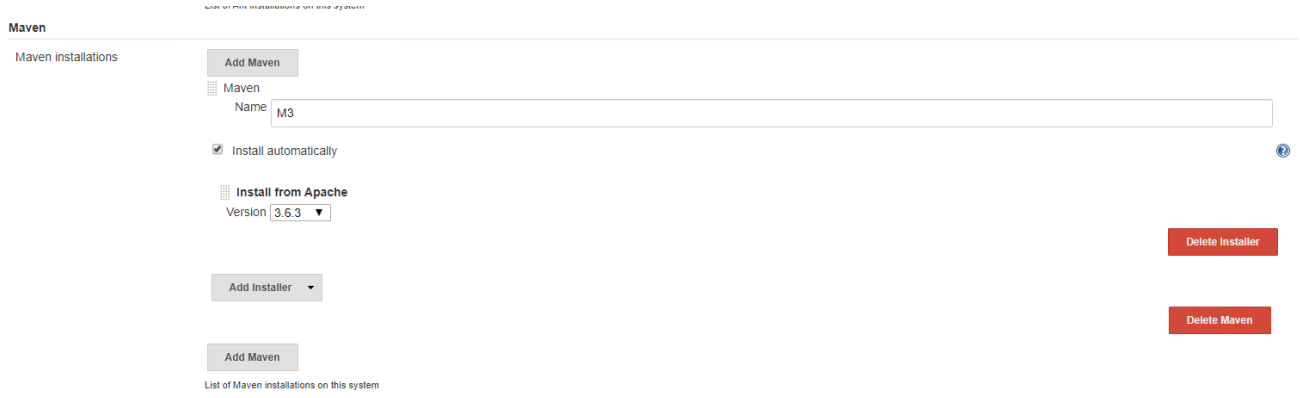
6. 由於在建立 Jenkins 應用程式時使用了 OpenShift OAuth，因此請按一下使用 OpenShift 登入。



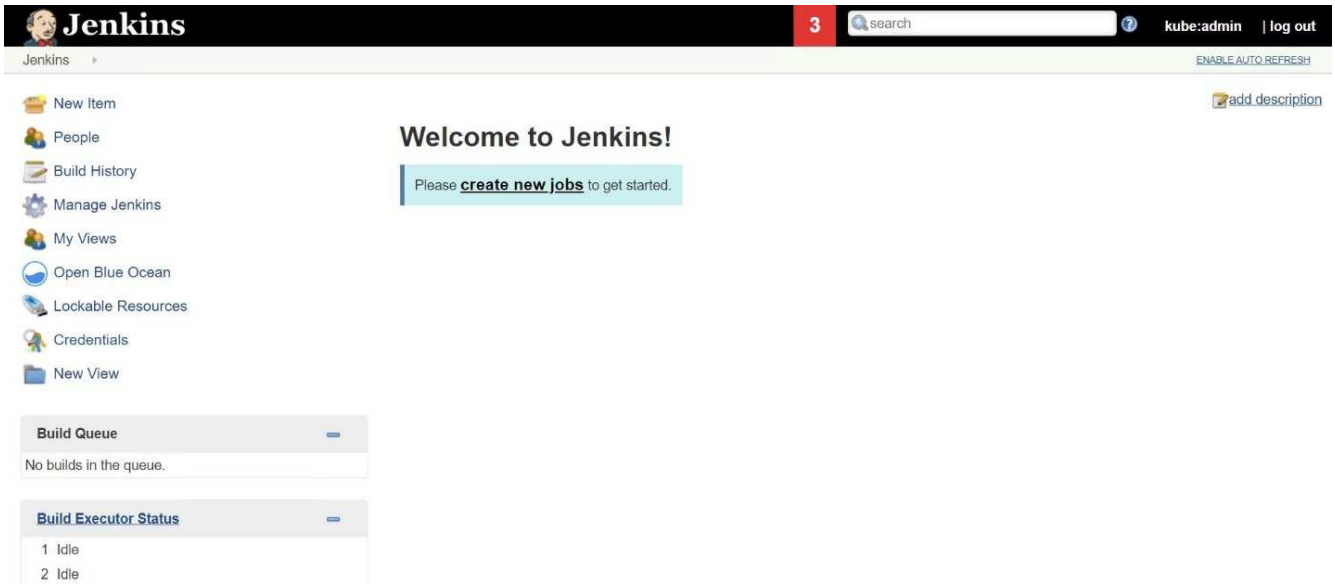
7. 授權 Jenkins 服務帳戶存取 OpenShift 使用者。



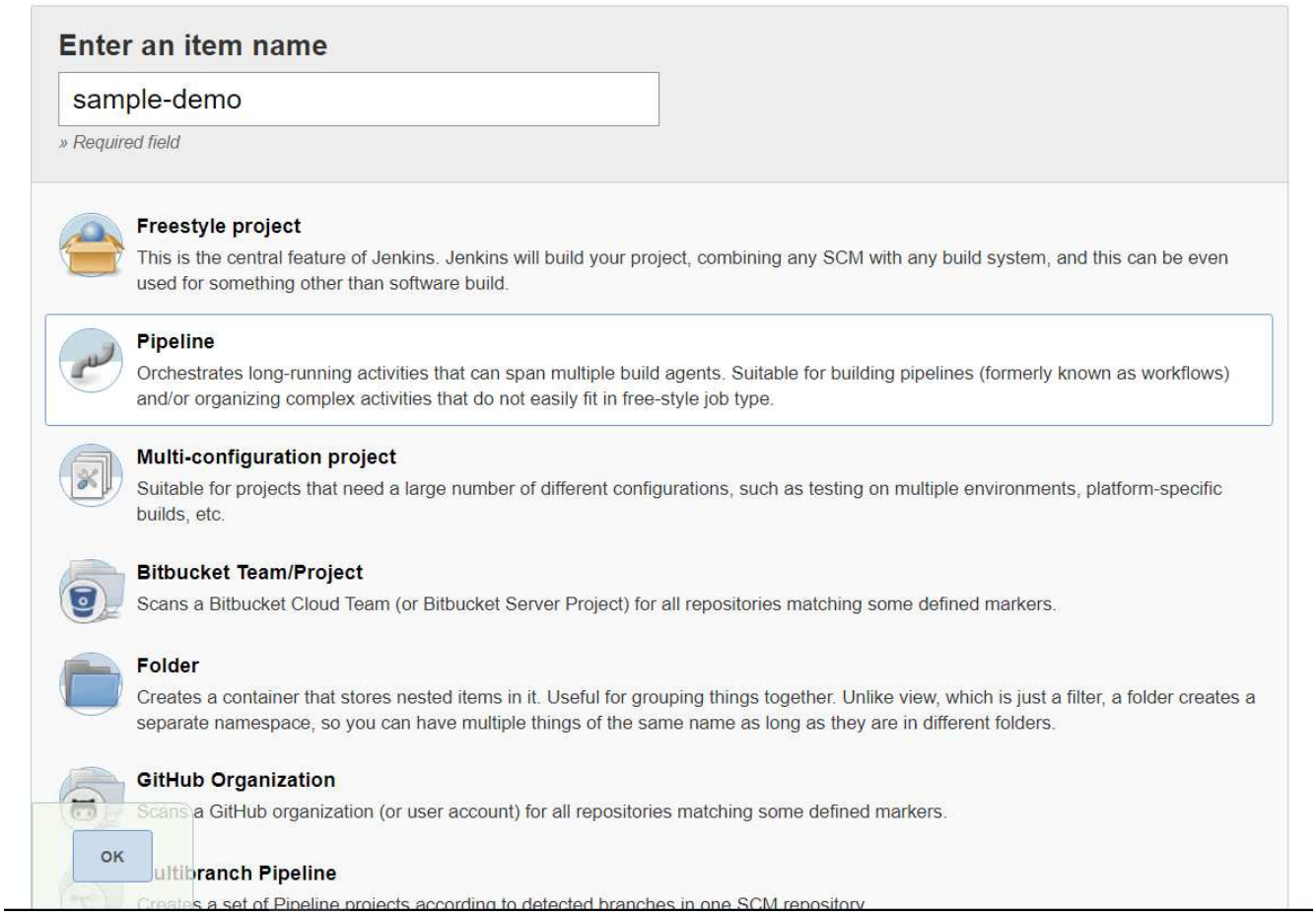
8. 將顯示 Jenkins 歡迎頁面。因為我們使用的是 Maven 構建，所以請先完成 Maven 安裝。導航至“管理 Jenkins”>“全域工具配置”，然後在“Maven”子標題中按一下“新增 Maven”。輸入您選擇的名稱並確保選擇了“自動安裝”選項。按一下「Save（儲存）」。



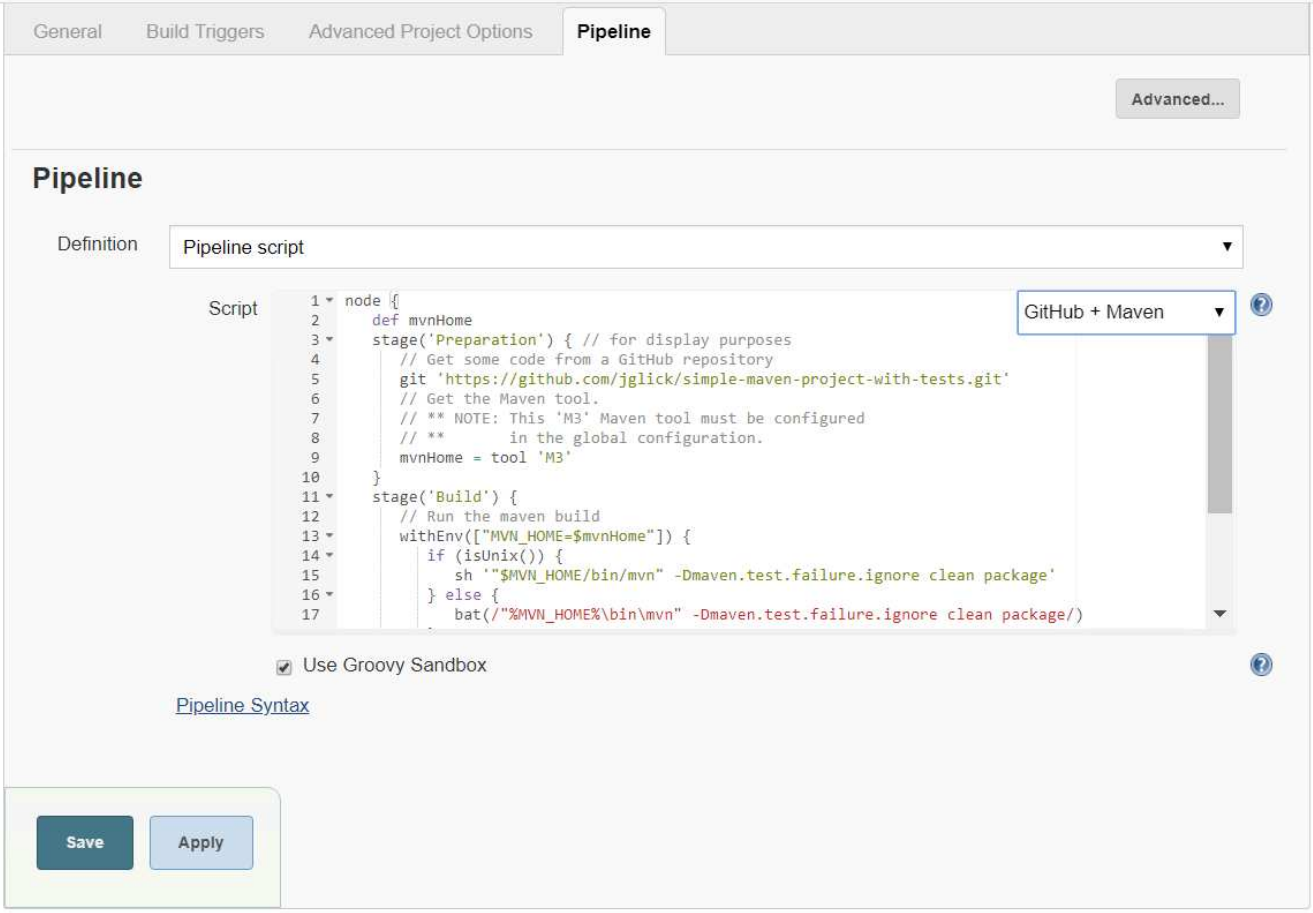
9. 現在您可以建立一個管道來示範 CI/CD 工作流程。在主頁上，按一下左側功能表中的「建立新作業」或「新專案」。



10. 在建立專案頁面上，輸入您選擇的名稱，選擇管道，然後按一下確定。



11. 選擇管道選項卡。從嘗試範例管道下拉選單中，選擇 Github + Maven。程式碼會自動填入。按一下「Save（儲存）」。



12. 按一下「立即建置」以透過準備、建置和測試階段觸發開發。完成整個建置過程並顯示建置結果可能需要幾分鐘。

13. 每當程式碼發生任何變化時，都可以重建管道來修補新版本的軟體，從而實現持續整合和持續交付。按一下「最近變更」可追蹤自上一版本以來的變更。

配置多租戶

使用**NetApp**在 **Red Hat OpenShift** 上設定多租戶

許多在容器上執行多個應用程式或工作負載的組織傾向於為每個應用程式或工作負載部署一個 Red Hat OpenShift 叢集。這使得他們能夠對應用程式或工作負載實施嚴格隔離，優化效能並減少安全漏洞。但是，為每個應用程式部署單獨的 Red Hat OpenShift 叢集會帶來一系列問題。它增加了必須自行監控和管理每個叢集的營運開銷，由於不同應用程式的專用資源而增加了成本，並阻礙了有效的可擴展性。

為了克服這些問題，可以考慮在單一 Red Hat OpenShift 叢集中執行所有應用程式或工作負載。但在這樣的架構下，資源隔離和應用安全漏洞是主要挑戰之一。一個工作負載中的任何安全漏洞都可能自然蔓延到另一個工作負載，從而擴大影響範圍。此外，由於預設沒有資源分配策略，因此一個應用程式突然不受控制地利用資源都會影響另一個應用程式的效能。

因此，組織尋求能夠兼顧兩方面優勢的解決方案，例如，允許他們在單一叢集中運行所有工作負載，同時為每個工作負載提供專用叢集的優勢。

一個有效的解決方案是在 Red Hat OpenShift 上設定多租用戶。多租戶是一種允許多個租戶在同一個叢集上共存的架構，對資源、安全性等進行適當的隔離。在此上下文中，租用戶可被視為叢集資源的子集，配置為由特定使

用者群組用於專用目的。在 Red Hat OpenShift 叢集上配置多租戶具有以下優勢：

- 透過共享集群資源來減少資本支出和營運支出
- 降低營運和管理開銷
- 保護工作負載免受安全漏洞的交叉污染
- 保護工作負載免受資源爭用導致的意外效能下降

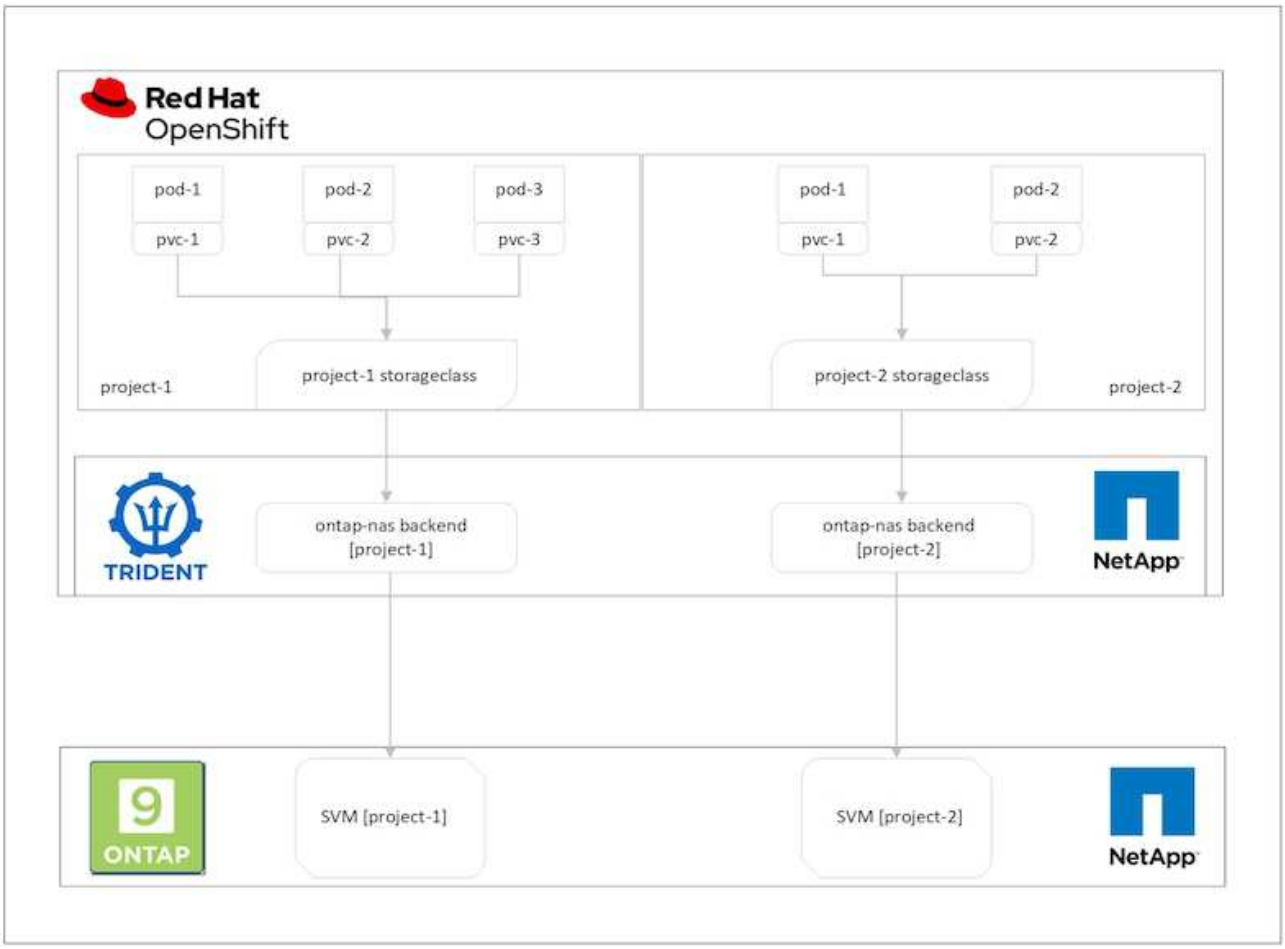
對於完全實現的多租戶 OpenShift 集群，必須為屬於不同資源桶的集群資源配置配額和限制：運算、儲存、網路、安全性等。雖然我們在此解決方案中涵蓋了所有資源桶的某些方面，但我們專注於透過在由 NetApp ONTAP 支援的TridentNetApp分配的儲存資源上配置多租戶來隔離和保護同一 Red Hat OpenShift 叢集上多個工作負載所服務或使用的資料的ONTAP實踐。

架構

儘管NetApp ONTAP支援的 Red Hat OpenShift 和Trident預設不提供工作負載之間的隔離，但它們提供了可用於配置多租用戶的廣泛功能。為了更好地理解在由NetApp ONTAP支援的Trident的 Red Hat OpenShift 叢集上設計多租用戶解決方案，讓我們考慮一個具有一組需求的範例並概述圍繞它的配置。

假設一個組織在 Red Hat OpenShift 叢集上執行兩個工作負載，這是兩個不同團隊正在進行的兩個專案的一部分。這些工作負載的資料駐留在由Trident在NetApp ONTAP NAS 後端動態配置的 PVC 上。該組織需要為這兩個工作負載設計一個多租戶解決方案，並隔離用於這些專案的資源，以確保維護安全性和效能，主要專注於為這些應用程式提供服務的資料。

下圖描述了由NetApp ONTAP支援的具有Trident的 Red Hat OpenShift 叢集上的多租用戶解決方案。



技術要求

1. NetApp ONTAP儲存叢集
2. Red Hat OpenShift 叢集
3. Trident

Red Hat OpenShift – 叢集資源

從 Red Hat OpenShift 叢集的角度來看，首先要啟動的頂層資源是專案。 OpenShift 專案可以看作是一個叢集資源，它將整個 OpenShift 叢集劃分為多個虛擬叢集。因此，專案層級的隔離為配置多租戶提供了基礎。

接下來是在叢集中配置 RBAC。最佳做法是讓所有開發人員在身分識別提供者 (IdP) 中將單一項目或工作負載配置到單一使用者群組。 Red Hat OpenShift 允許 IdP 整合和使用者群組同步，從而允許將來自 IdP 的使用者和群組匯入到叢集中。這有助於叢集管理員將專用於某個專案的叢集資源的存取隔離給從事該專案的使用者群組或群組，從而限制對任何叢集資源的未經授權的存取。要了解有關 IdP 與 Red Hat OpenShift 整合的更多信息，請參閱文檔 "這裡"。

NetApp ONTAP

隔離作為 Red Hat OpenShift 叢集的持久性儲存提供者的共用儲存非常重要，以確保在每個專案的儲存上建立的磁碟區對於主機來說就像是在單獨的儲存空間上建立的一樣。為此，請在NetApp ONTAP上建立與專案或工作負載數量相同的 SVM（儲存虛擬機器），並將每個 SVM 專用於一個工作負載。

Trident

在NetApp ONTAP上為不同的專案建立不同的 SVM 後，必須將每個 SVM 對應到不同的Trident後端。 Trident上的後端配置驅動持久性儲存到 OpenShift 叢集資源的分配，並且需要對應到 SVM 的詳細資訊。這至少應該是後端的協定驅動程式。或者，它允許您定義如何在儲存上配置卷，並為磁碟區的大小或聚合的使用等設定限制。關於Trident後端定義的詳細資訊可以參見 "這裡"。

Red Hat OpenShift – 儲存資源

設定Trident後端後，下一步是設定 StorageClasses。配置與後端數量相同的儲存類，為每個儲存類提供僅在一個後端啟動磁碟區的存取權限。我們可以在定義儲存類別時使用 storagePools 參數將 StorageClass 對應到特定的Trident後端。定義儲存類別的詳細資訊可以在這裡找到 "這裡"。因此，從 StorageClass 到Trident後端存在一對一映射，該映射指向一個 SVM。這可確保透過指派給該專案的 StorageClass 提出的所有儲存聲明均由專用於該專案的 SVM 提供服務。

由於儲存類別不是命名空間資源，我們如何確保另一個命名空間或專案中的 pod 對一個專案的儲存類別的儲存聲明被拒絕？答案是使用 ResourceQuotas。 ResourceQuotas 是控制每個專案資源總使用量的物件。它可以限制專案中物件可消耗的資源數量和總量。幾乎所有專案的資源都可以使用 ResourceQuotas 來限制，有效使用 ResourceQuotas 可以幫助組織削減因資源過度配置或過度消耗而導致的成本和中斷。請參閱文檔 "這裡"了解更多。

對於這種用例，我們需要限制特定專案中的 pod 從非專用於其專案的儲存類別中聲明儲存。為此，我們需要透過設定來限制其他儲存類別的持久性卷聲明 `<storage-class-name>.storageclass.storage.k8s.io/persistentvolumeclaims`為 0。此外，叢集管理員必須確保專案中的開發人員無權修改資源配額。

配置

對於任何多租戶解決方案，任何使用者都不能存取超出所需的叢集資源。因此，作為多租戶配置的一部分而要配置的整個資源集被劃分到叢集管理員、儲存管理員和每個專案的開發人員之間。

下表概述了不同使用者需要執行的不同任務：

角色	任務
集群管理員	為不同的應用程式或工作負載建立項目
	為 storage-admin 建立 ClusterRoles 和 RoleBindings
	為開發人員建立角色和角色綁定，並分配對特定專案的存取權限
	[可選] 配置項目以在特定節點上調度 Pod
儲存管理	在NetApp ONTAP上建立 SVM
	建立Trident後端
	建立儲存類別
	建立儲存資源配額
開發人員	驗證在指定專案中建立或修補 PVC 或 Pod 的權限
	驗證在另一個專案中建立或修補 PVC 或 Pod 的權限
	驗證檢視或編輯項目、資源配額和儲存類別的權限

配置

以下是使用NetApp在 Red Hat OpenShift 上設定多租用戶的先決條件。

先決條件

- NetApp ONTAP叢集
- Red Hat OpenShift 叢集
- 在叢集上安裝的Trident
- 安裝了 tridentctl 和 oc 工具並新增至 $PATH 的管理員工作站
- ONTAP管理員存取權限
- 叢集管理員存取 OpenShift 叢集
- 集群與身分提供者集成
- 配置身份提供者以有效區分不同團隊中的用戶

配置：叢集管理任務

Red Hat OpenShift 叢集管理員執行下列任務：

1. 以叢集管理員身分登入 Red Hat OpenShift 叢集。
2. 建立兩個項目，分別對應不同的工程。

```
oc create namespace project-1
oc create namespace project-2
```

3. 為 project-1 建立開發人員角色。

```
cat << EOF | oc create -f -
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: project-1
  name: developer-project-1
rules:
  - verbs:
      - '*'
    apiGroups:
      - apps
      - batch
      - autoscaling
      - extensions
      - networking.k8s.io
      - policy
```

```
      - apps.openshift.io
      - build.openshift.io
      - image.openshift.io
      - ingress.operator.openshift.io
      - route.openshift.io
      - snapshot.storage.k8s.io
      - template.openshift.io
    resources:
      - '*'
  - verbs:
      - '*'
    apiGroups:
      - ''
    resources:
      - bindings
      - configmaps
      - endpoints
      - events
      - persistentvolumeclaims
      - pods
      - pods/log
      - pods/attach
      - podtemplates
      - replicationcontrollers
      - services
      - limitranges
      - namespaces
      - componentstatuses
      - nodes
  - verbs:
      - '*'
    apiGroups:
      - trident.netapp.io
    resources:
      - tridentsnapshots
EOF
```

本節提供的角色定義只是一個範例。必須根據最終使用者的要求來定義開發人員的角色。

1. 同樣，為 project-2 建立開發人員角色。
2. 所有 OpenShift 和NetApp儲存資源通常由儲存管理員管理。儲存管理員的存取權限由安裝Trident時建立的 trident 操作員角色控制。除此之外，儲存管理員還需要存取 ResourceQuotas 來控制儲存的消耗方式。
3. 建立一個用於管理叢集中所有專案的 ResourceQuotas 的角色，並將其附加到儲存管理員。

```
cat << EOF | oc create -f -
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: resource-quotas-role
rules:
  - verbs:
      - '*'
    apiGroups:
      - ''
    resources:
      - resourcequotas
  - verbs:
      - '*'
    apiGroups:
      - quota.openshift.io
    resources:
      - '*'
EOF
```

4. 確保叢集與組織的身分提供者集成，並且使用者群組與叢集群組同步。以下範例顯示身分提供者已與叢集整合並與使用者群組同步。

```
$ oc get groups
NAME                          USERS
ocp-netapp-storage-admins     ocp-netapp-storage-admin
ocp-project-1                 ocp-project-1-user
ocp-project-2                 ocp-project-2-user
```

1. 為儲存管理員配置 ClusterRoleBindings。

```
cat << EOF | oc create -f -
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: netapp-storage-admin-trident-operator
subjects:
  - kind: Group
    apiGroup: rbac.authorization.k8s.io
    name: ocp-netapp-storage-admins
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: trident-operator
---
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: netapp-storage-admin-resource-quotas-cr
subjects:
  - kind: Group
    apiGroup: rbac.authorization.k8s.io
    name: ocp-netapp-storage-admins
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: resource-quotas-role
EOF
```

對於儲存管理員，必須綁定兩個角色：trident-operator 和 resource-quotas。

1. 為開發者建立 RoleBindings，將developer-project-1 角色綁定到 project-1 中對應的群組（ocp-project-1）。

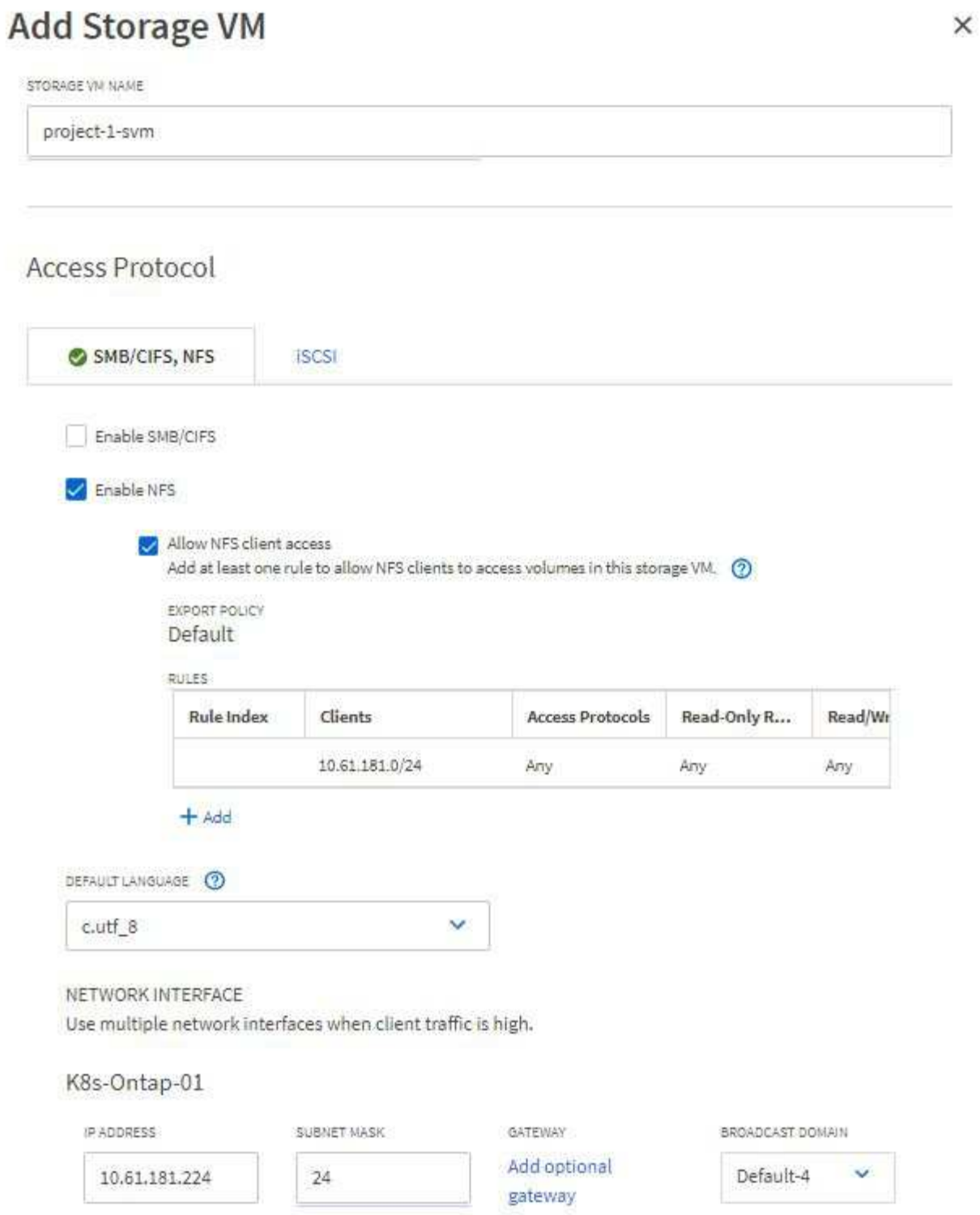
```
cat << EOF | oc create -f -
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: project-1-developer
  namespace: project-1
subjects:
  - kind: Group
    apiGroup: rbac.authorization.k8s.io
    name: ocp-project-1
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: developer-project-1
EOF
```

2. 同樣的，在project-2中為開發者建立RoleBindings，將開發者角色綁定到對應的使用者群組。

配置：儲存管理任務

儲存管理員必須配置以下資源：

1. 以管理員身分登入NetApp ONTAP叢集。
2. 導航到儲存>儲存虛擬機，然後按一下新增。透過提供所需的詳細信息，建立兩個 SVM，一個用於專案 1，另一個用於專案 2。也要建立一個 vsadmin 帳戶來管理 SVM 及其資源。



1. 以儲存管理員身分登入 Red Hat OpenShift 叢集。
2. 為 project-1 建立後端並將其對應到專用於該專案的 SVM。 NetApp建議使用 SVM 的 vsadmin 帳戶將後端連接到 SVM，而不是使用ONTAP叢集管理員。

```
cat << EOF | tridentctl -n trident create backend -f
{
    "version": 1,
    "storageDriverName": "ontap-nas",
    "backendName": "nfs_project_1",
    "managementLIF": "172.21.224.210",
    "dataLIF": "10.61.181.224",
    "svm": "project-1-svm",
    "username": "vsadmin",
    "password": "NetApp123"
}
EOF
```

我們在此範例中使用 ontap-nas 驅動程式。根據用例建立後端時使用適當的驅動程式。

我們假設Trident已安裝在 trident 專案中。

1. 類似地為 project-2 建立Trident後端並將其對應到專用於 project-2 的 SVM。
2. 接下來，建立儲存類別。為 project-1 建立儲存類，並透過設定 storagePools 參數將其配置為使用專用於 project-1 的後端儲存池。

```
cat << EOF | oc create -f -
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: project-1-sc
provisioner: csi.trident.netapp.io
parameters:
  backendType: ontap-nas
  storagePools: "nfs_project_1:.*"
EOF
```

3. 同樣，為 project-2 建立儲存類別並將其配置為使用專用於 project-2 的後端儲存池。
4. 建立 ResourceQuota 來限制 project-1 中的資源從專用於其他專案的儲存類別中請求儲存。

```
cat << EOF | oc create -f -
kind: ResourceQuota
apiVersion: v1
metadata:
  name: project-1-sc-rq
  namespace: project-1
spec:
  hard:
    project-2-sc.storageclass.storage.k8s.io/persistentvolumeclaims: 0
EOF
```

5. 類似地，建立一個 ResourceQuota 來限制 project-2 中的資源從專用於其他專案的儲存類別中請求儲存。

驗證

若要驗證前面步驟中配置的多租用戶架構，請完成下列步驟：

驗證在指定專案中建立 **PVC** 或 **Pod** 的權限

1. 以 ocp-project-1-user、project-1 中的開發人員身分登入。
2. 檢查建立新項目的權限。

```
oc create ns sub-project-1
```

3. 使用指派給 project-1 的儲存類別在 project-1 中建立 PVC。

```
cat << EOF | oc create -f -
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: test-pvc-project-1
  namespace: project-1
  annotations:
    trident.netapp.io/reclaimPolicy: Retain
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: project-1-sc
EOF
```

4. 檢查與 PVC 關聯的 PV。

```
oc get pv
```

5. 驗證 PV 及其磁碟區是否在NetApp ONTAP上專用於 project-1 的 SVM 中建立。

```
volume show -vserver project-1-svm
```

6. 在 project-1 中建立一個 pod，並掛載上一個步驟所建立的 PVC。

```
cat << EOF | oc create -f -
kind: Pod
apiVersion: v1
metadata:
  name: test-pvc-pod
  namespace: project-1
spec:
  volumes:
    - name: test-pvc-project-1
      persistentVolumeClaim:
       claimName: test-pvc-project-1
  containers:
    - name: test-container
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - mountPath: "/usr/share/nginx/html"
          name: test-pvc-project-1
EOF
```

7. 檢查 pod 是否正在運作以及是否安裝了磁碟區。

```
oc describe pods test-pvc-pod -n project-1
```

驗證在另一個專案中建立 **PVC** 或 **Pod** 或使用專用於另一個專案的資源的存取權限

1. 以 ocp-project-1-user、project-1 中的開發人員身分登入。
2. 使用指派給 project-2 的儲存類別在 project-1 中建立 PVC。

```
cat << EOF | oc create -f -
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: test-pvc-project-1-sc-2
  namespace: project-1
  annotations:
    trident.netapp.io/reclaimPolicy: Retain
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: project-2-sc
EOF
```

3. 在 project-2 中建立 PVC。

```
cat << EOF | oc create -f -
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: test-pvc-project-2-sc-1
  namespace: project-2
  annotations:
    trident.netapp.io/reclaimPolicy: Retain
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: project-1-sc
EOF
```

4. 確保 PVC `test-pvc-project-1-sc-2`和 `test-pvc-project-2-sc-1`沒有創造。

```
oc get pvc -n project-1
oc get pvc -n project-2
```

5. 在 project-2 中建立一個 pod。

```
cat << EOF | oc create -f -
kind: Pod
apiVersion: v1
metadata:
  name: test-pvc-pod
  namespace: project-1
spec:
  containers:
    - name: test-container
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
EOF
```

驗證檢視和編輯項目、資源配額和儲存類別的權限

1. 以 ocp-project-1-user、project-1 中的開發人員身分登入。
2. 檢查建立新項目的權限。

```
oc create ns sub-project-1
```

3. 驗證查看項目的存取權限。

```
oc get ns
```

4. 檢查使用者是否可以檢視或編輯 project-1 中的 ResourceQuotas。

```
oc get resourcequotas -n project-1
oc edit resourcequotas project-1-sc-rq -n project-1
```

5. 驗證使用者是否有權查看儲存類別。

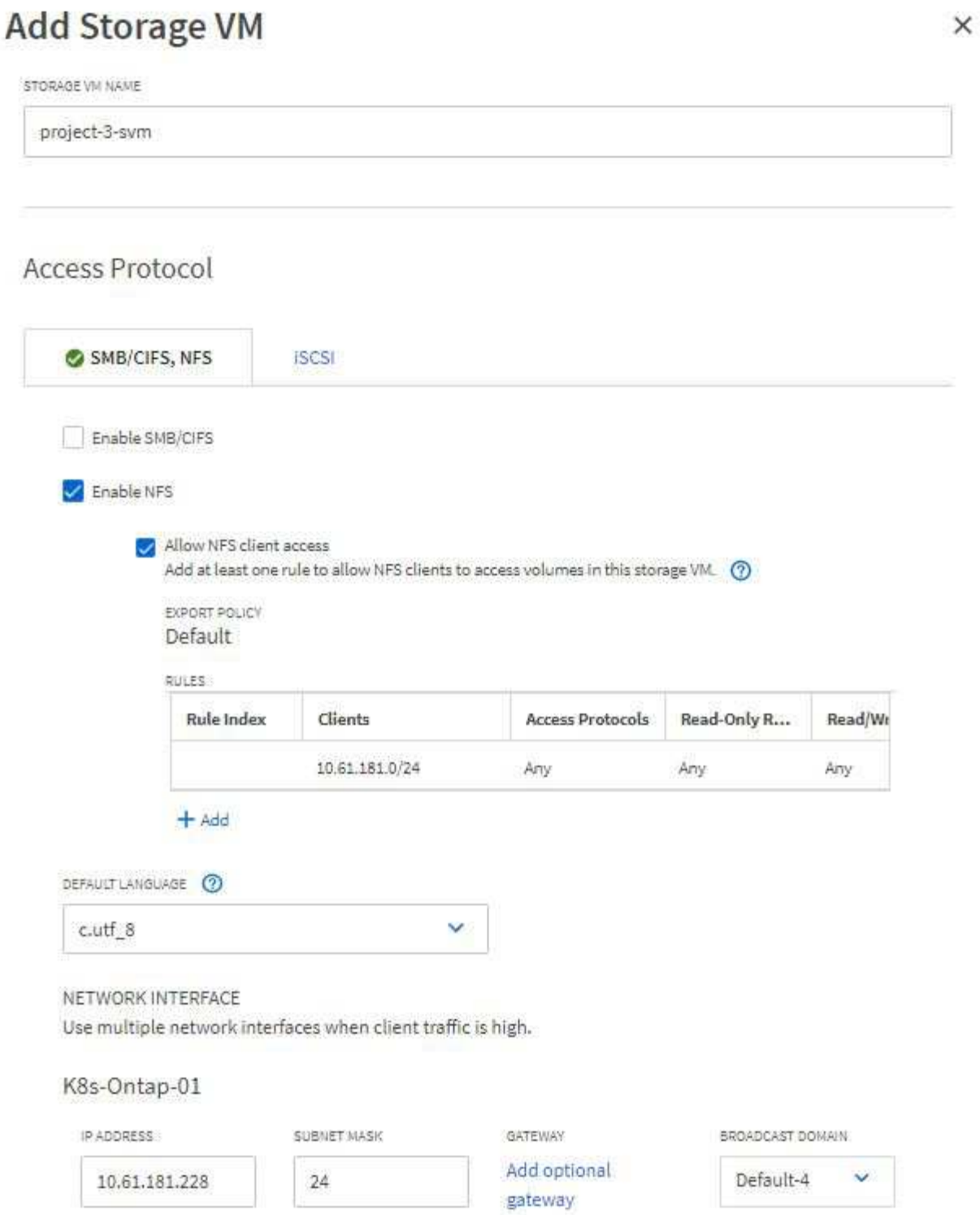
```
oc get sc
```

6. 檢查存取權限以描述儲存類別。
7. 驗證使用者編輯儲存類別的權限。

```
oc edit sc project-1-sc
```

在多租用戶配置中，新增具有儲存資源的新項目需要額外的配置以確保不違反多租用戶原則。若要在多租戶叢集中新增更多項目，請完成以下步驟：

1. 以儲存管理員身分登入NetApp ONTAP叢集。
2. 導航至 `Storage → Storage VMs`並點擊 `Add`。建立一個專用於 project-3 的新 SVM。也要建立一個 vsadmin 帳戶來管理 SVM 及其資源。



1. 以叢集管理員身分登入 Red Hat OpenShift 叢集。
2. 建立新項目。

```
oc create ns project-3
```

3. 確保在 IdP 上建立了 project-3 的使用者群組並與 OpenShift 叢集同步。

```
oc get groups
```

4. 為 project-3 建立開發人員角色。

```
cat << EOF | oc create -f -
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: project-3
  name: developer-project-3
rules:
  - verbs:
      - '*'
    apiGroups:
      - apps
      - batch
      - autoscaling
      - extensions
      - networking.k8s.io
      - policy
      - apps.openshift.io
      - build.openshift.io
      - image.openshift.io
      - ingress.operator.openshift.io
      - route.openshift.io
      - snapshot.storage.k8s.io
      - template.openshift.io
    resources:
      - '*'
  - verbs:
      - '*'
    apiGroups:
      - ''
    resources:
      - bindings
      - configmaps
      - endpoints
      - events
      - persistentvolumeclaims
      - pods
      - pods/log
      - pods/attach
      - podtemplates
```

```
      - replicationcontrollers
      - services
      - limitranges
      - namespaces
      - componentstatuses
      - nodes
  - verbs:
      - '*'
    apiGroups:
      - trident.netapp.io
    resources:
      - tridentsnapshots
EOF
```

本節提供的角色定義只是一個範例。必須根據最終使用者的要求來定義開發人員的角色。

1. 為 project-3 中的開發人員建立 RoleBinding，將 development-project-3 角色綁定到 project-3 中對應的群組（ocp-project-3）。

```
cat << EOF | oc create -f -
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: project-3-developer
  namespace: project-3
subjects:
  - kind: Group
    apiGroup: rbac.authorization.k8s.io
    name: ocp-project-3
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: developer-project-3
EOF
```

2. 以儲存管理員身分登入 Red Hat OpenShift 集群
3. 建立Trident後端並將其對應到專用於 project-3 的 SVM。 NetApp建議使用 SVM 的 vsadmin 帳戶將後端連接到 SVM，而不是使用ONTAP叢集管理員。

```
cat << EOF | tridentctl -n trident create backend -f
{
    "version": 1,
    "storageDriverName": "ontap-nas",
    "backendName": "nfs_project_3",
    "managementLIF": "172.21.224.210",
    "dataLIF": "10.61.181.228",
    "svm": "project-3-svm",
    "username": "vsadmin",
    "password": "NetApp!23"
}
EOF
```

我們在此範例中使用 ontap-nas 驅動程式。根據用例使用適當的驅動程式建立後端。

我們假設Trident已安裝在 trident 專案中。

1. 為 project-3 建立儲存類別並將其配置為使用專用於 project-3 的後端儲存池。

```
cat << EOF | oc create -f -
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: project-3-sc
provisioner: csi.trident.netapp.io
parameters:
  backendType: ontap-nas
  storagePools: "nfs_project_3:.*"
EOF
```

2. 建立 ResourceQuota 來限制 project-3 中的資源從專用於其他專案的儲存類別中請求儲存。

```
cat << EOF | oc create -f -
kind: ResourceQuota
apiVersion: v1
metadata:
  name: project-3-sc-rq
  namespace: project-3
spec:
  hard:
    project-1-sc.storageclass.storage.k8s.io/persistentvolumeclaims: 0
    project-2-sc.storageclass.storage.k8s.io/persistentvolumeclaims: 0
EOF
```

3. 修補其他專案中的 ResourceQuotas，以限制這些專案中的資源存取專用於 project-3 的儲存類別的儲存。

```
oc patch resourcequotas project-1-sc-rq -n project-1 --patch
'{"spec":{"hard":{ "project-3-
sc.storageclass.storage.k8s.io/persistentvolumeclaims": 0}}}'
oc patch resourcequotas project-2-sc-rq -n project-2 --patch
'{"spec":{"hard":{ "project-3-
sc.storageclass.storage.k8s.io/persistentvolumeclaims": 0}}}'
```

Kubernetes 的高階叢集管理

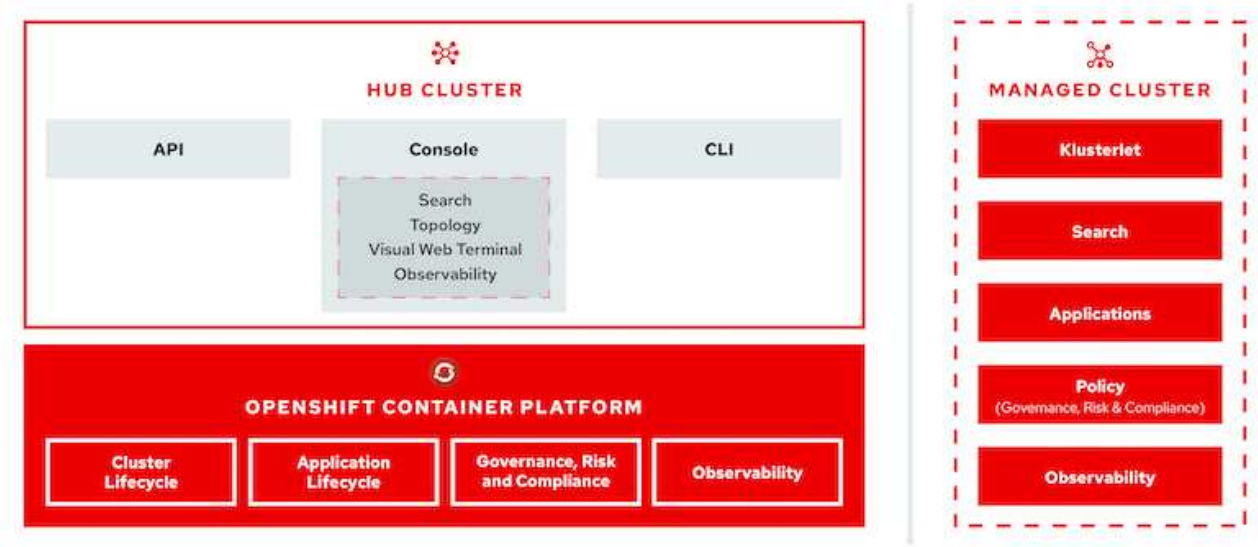
Kubernetes 的高階叢集管理：Red Hat OpenShift 與NetApp - 概述

隨著容器化應用程式從開發過渡到生產，許多組織需要多個 Red Hat OpenShift 叢集來支援該應用程式的測試和部署。同時，組織通常在 OpenShift 叢集上託管多個應用程式或工作負載。因此，每個組織最終都會管理一組集群，而 OpenShift 管理員必須面對額外的挑戰，即管理和維護跨多個本地資料中心和公有雲的一系列環境中的多個集群。為了因應這些挑戰，Red Hat 推出了針對 Kubernetes 的高階叢集管理。

Red Hat Advanced Cluster Management for Kubernetes 讓您能夠執行以下任務：

1. 跨資料中心和公有雲建立、匯入和管理多個集群
2. 從單一控制台部署和管理多個叢集上的應用程式或工作負載
3. 監控和分析不同集群資源的健康和狀態
4. 監控並強制執行跨多個叢集的安全合規性

Red Hat Advanced Cluster Management for Kubernetes 作為 Red Hat OpenShift 叢集的附加元件安裝，並使用該叢集作為其所有操作的中央控制器。此集群稱為中心集群，它為使用者公開了一個管理平面以連接到高階集群管理。透過進階叢集管理主控台匯入或建立的所有其他 OpenShift 叢集均由中心叢集管理，並稱為託管叢集。它在託管集群上安裝了一個名為 Klusterlet 的代理，將它們連接到中心集群，並滿足與集群生命週期管理、應用程式生命週期管理、可觀察性和安全合規性相關的不同活動的請求。



有關詳細信息，請參閱文檔 "這裡"。

為 Kubernetes 部署 ACM

為 Kubernetes 部署高階叢集管理

本節介紹使用NetApp在 Red Hat OpenShift 上對 Kubernetes 進行進階叢集管理。

先決條件

1. 用於中心叢集的 Red Hat OpenShift 叢集（高於 4.5 版本）
2. 用於託管叢集的 Red Hat OpenShift 叢集（高於 4.4.3 版本）
3. 叢集管理員存取 Red Hat OpenShift 叢集
4. Red Hat 訂閱 Kubernetes 高階叢集管理

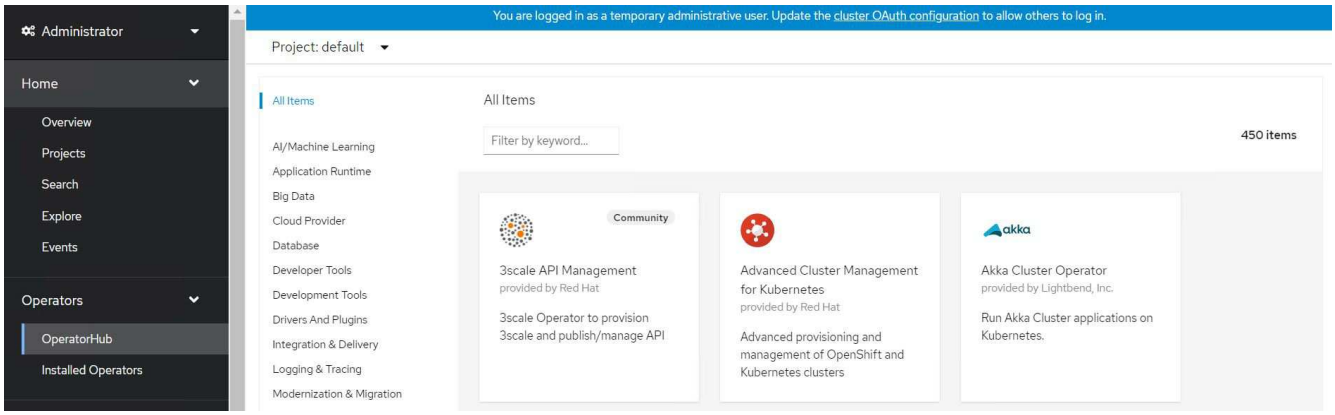
高階叢集管理是 OpenShift 叢集的一個附加元件，因此根據中心和管理叢集中使用的功能，對硬體資源有一定的要求和限制。在確定集群大小時需要考慮這些問題。查看文件 "這裡"了解更多詳情。

或者，如果中心叢集具有用於託管基礎架構元件的專用節點，並且您只想在這些節點上安裝高階叢集管理資源，則需要相應地向這些節點新增容忍度和選擇器。更多詳細資訊請參閱文檔 "這裡"。

為 Kubernetes 部署高階叢集管理

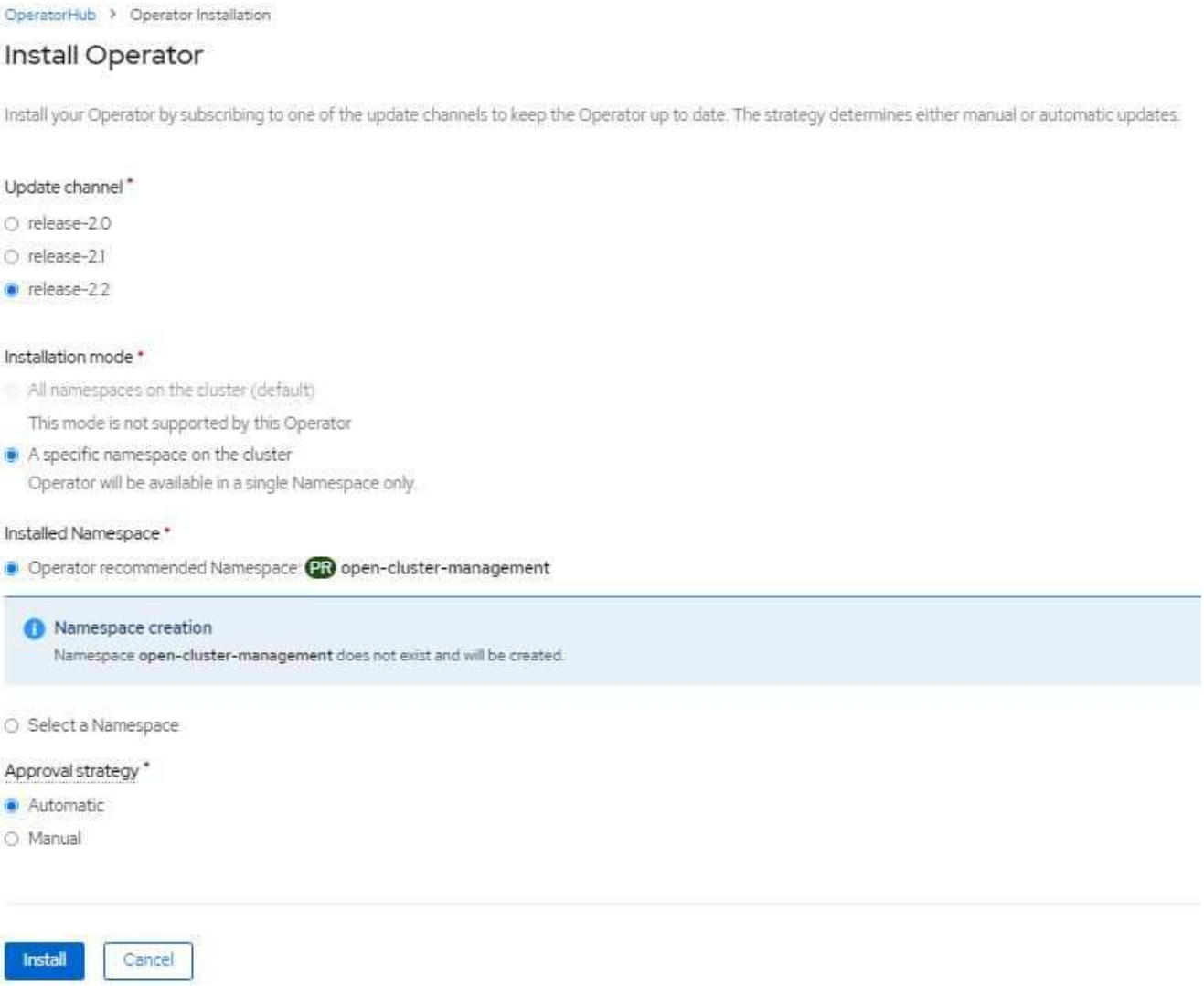
若要在 OpenShift 叢集上安裝 Kubernetes 進階叢集管理，請完成下列步驟：

1. 選擇一個 OpenShift 集群作為中心集群，並使用集群管理員權限登入。
2. 導覽至 Operators > Operators Hub 並蒐索 Kubernetes 的高階叢集管理。

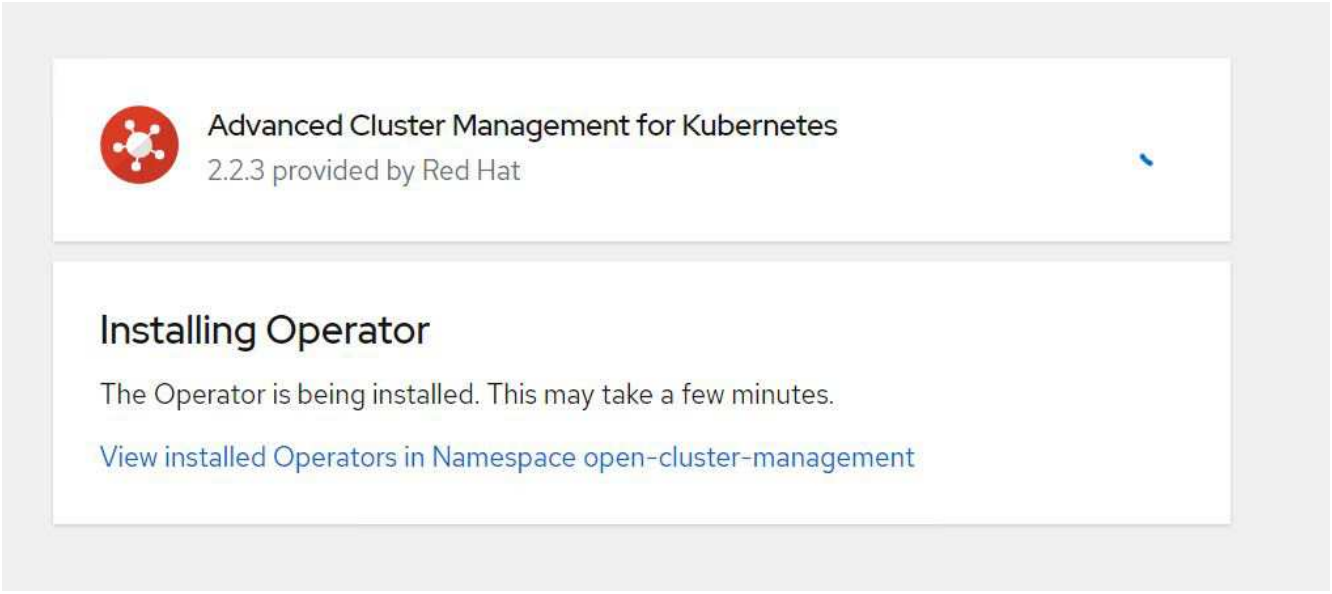


3. 選擇 Kubernetes 的高階叢集管理，然後按一下安裝。

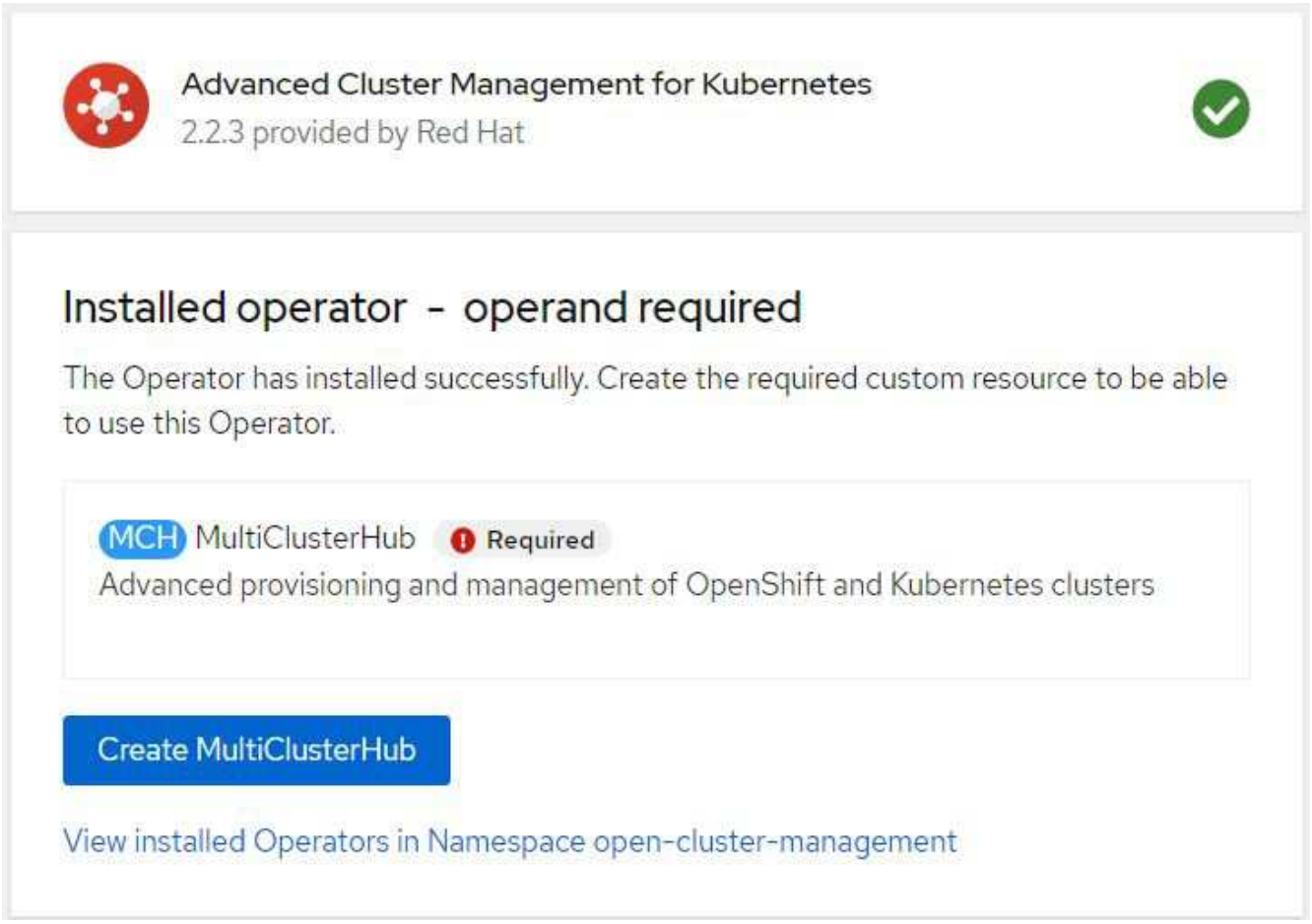
4. 在「安裝操作員」畫面上，提供必要的詳細資訊（NetApp建議保留預設參數）並按一下「安裝」。



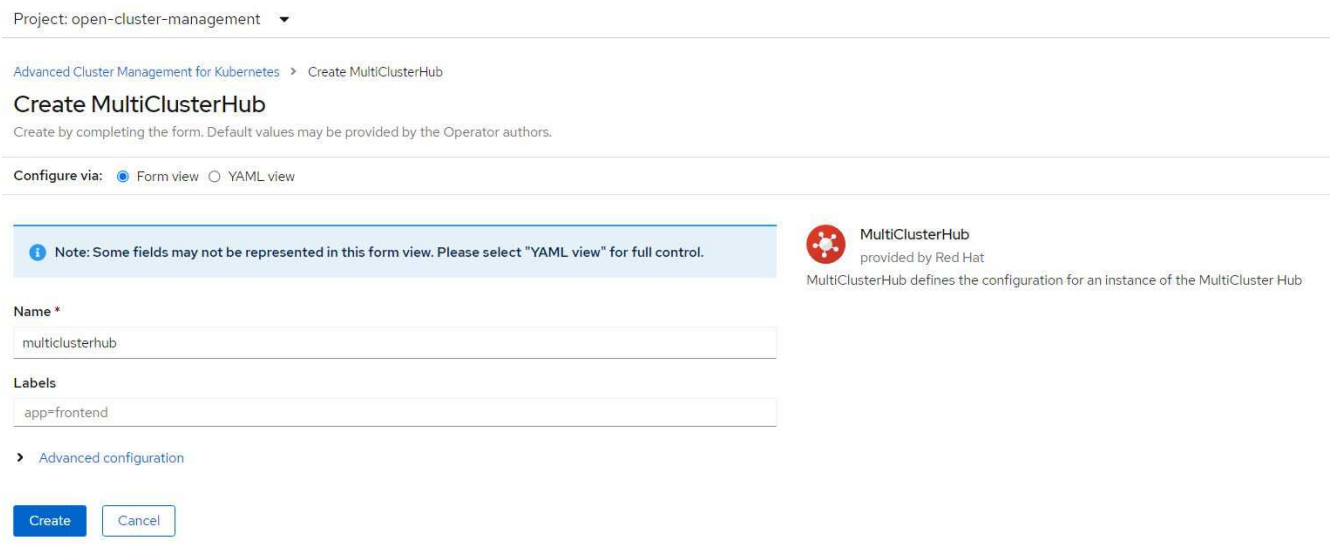
5. 等待操作員安裝完成。



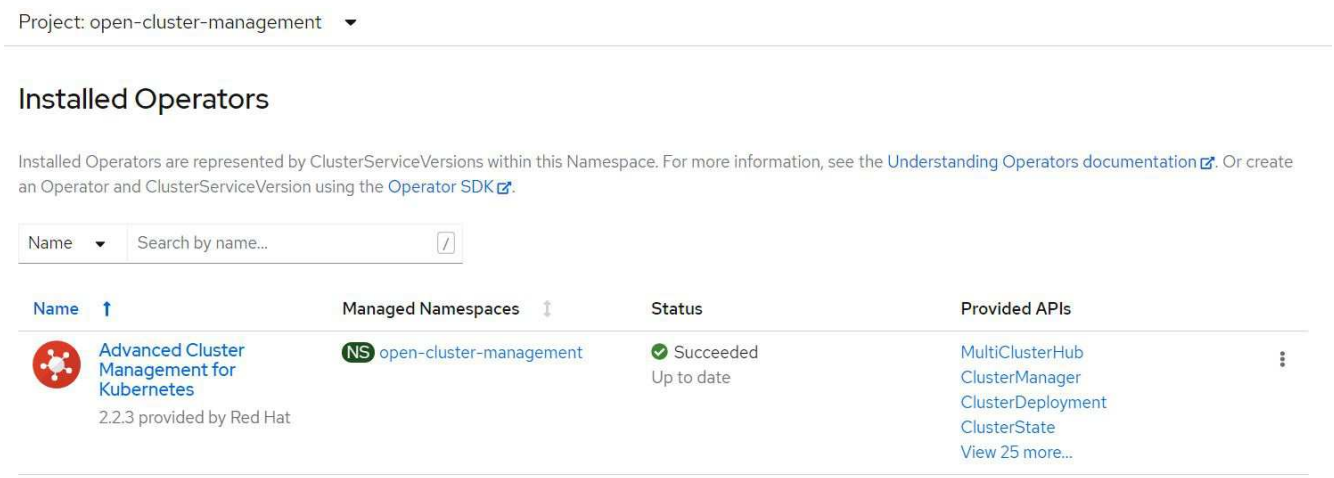
6. 安裝操作員後，按一下建立 MultiClusterHub。



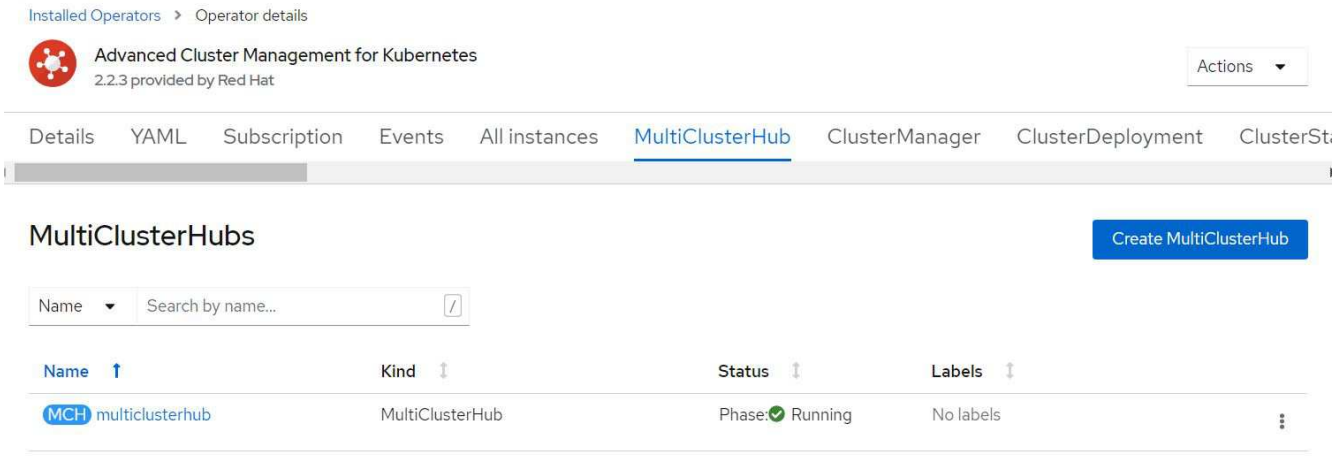
7. 在建立 MultiClusterHub 畫面上，提供詳細資訊後按一下建立。這將啟動多集群集線器的安裝。



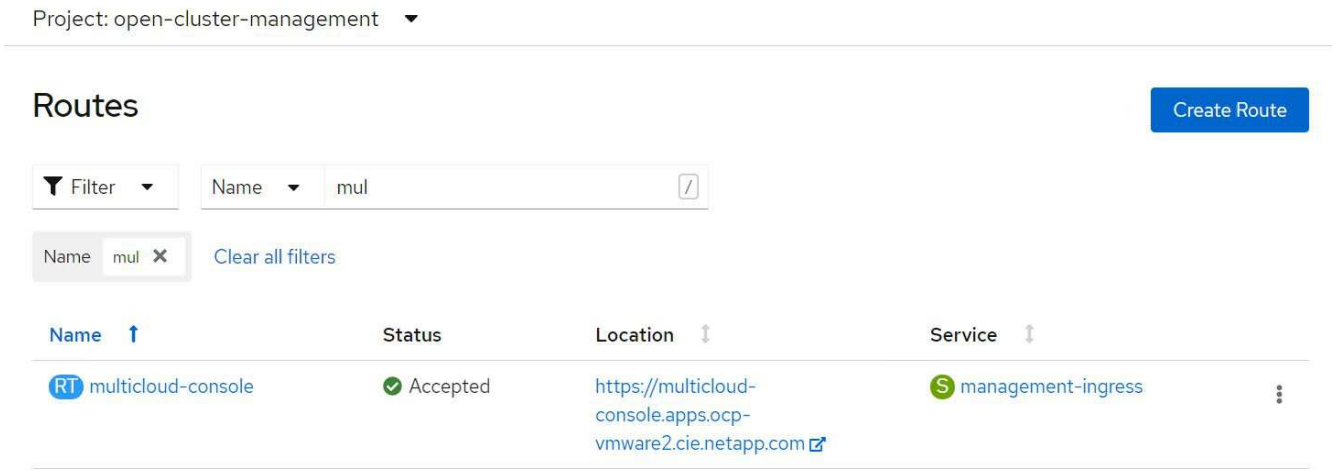
8. 當 open-cluster-management 命名空間中的所有 pod 都轉為 Running 狀態，並且操作員轉為 Succeeded 狀態後，Kubernetes 高階叢集管理就安裝完成了。



9. 完成集線器安裝需要一些時間，完成後，多集群集線器將進入運作狀態。

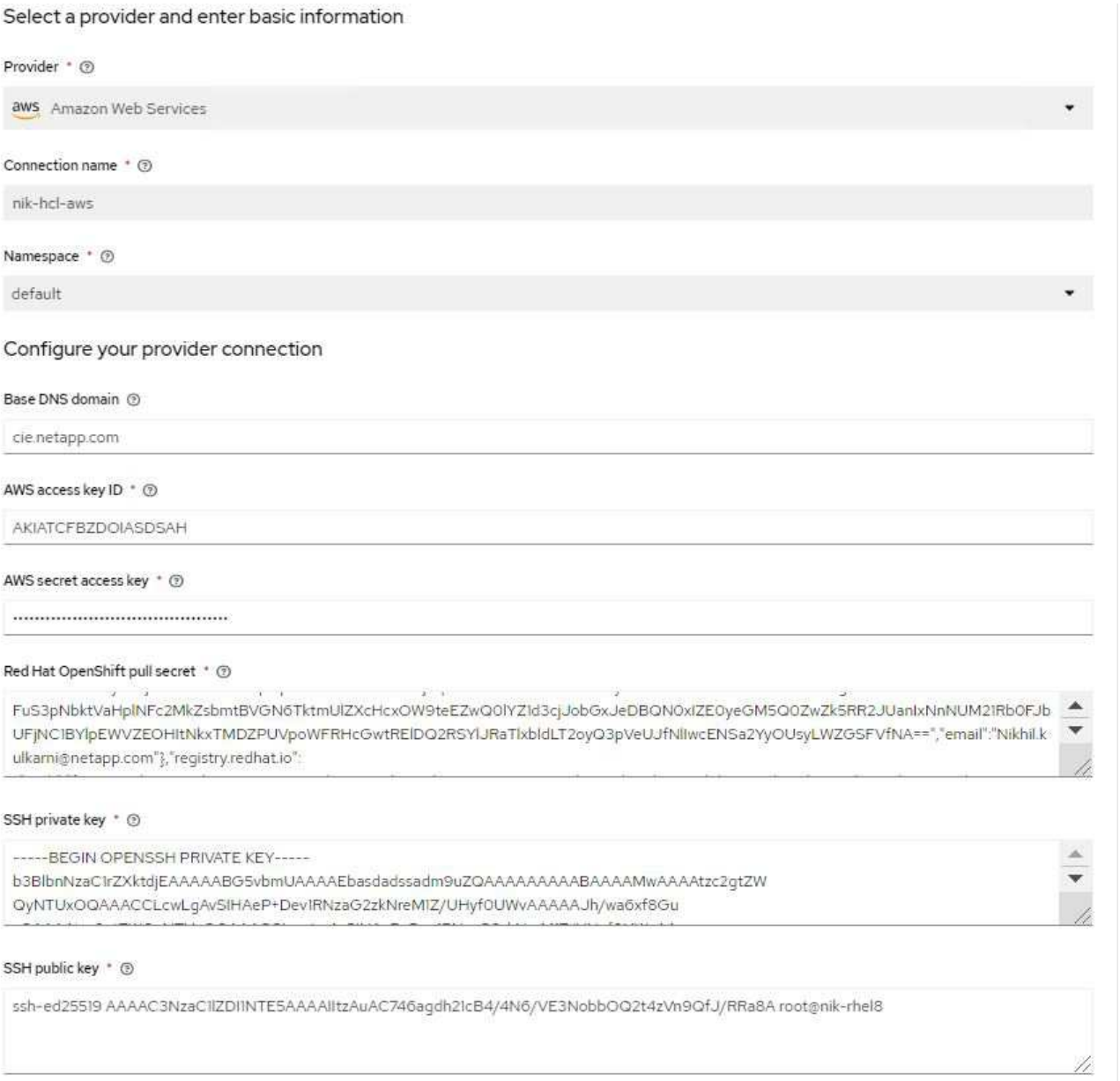


10. 它在 open-cluster-management 命名空間中建立一條路由。連接到路由中的URL，存取高階叢集管理控制台。

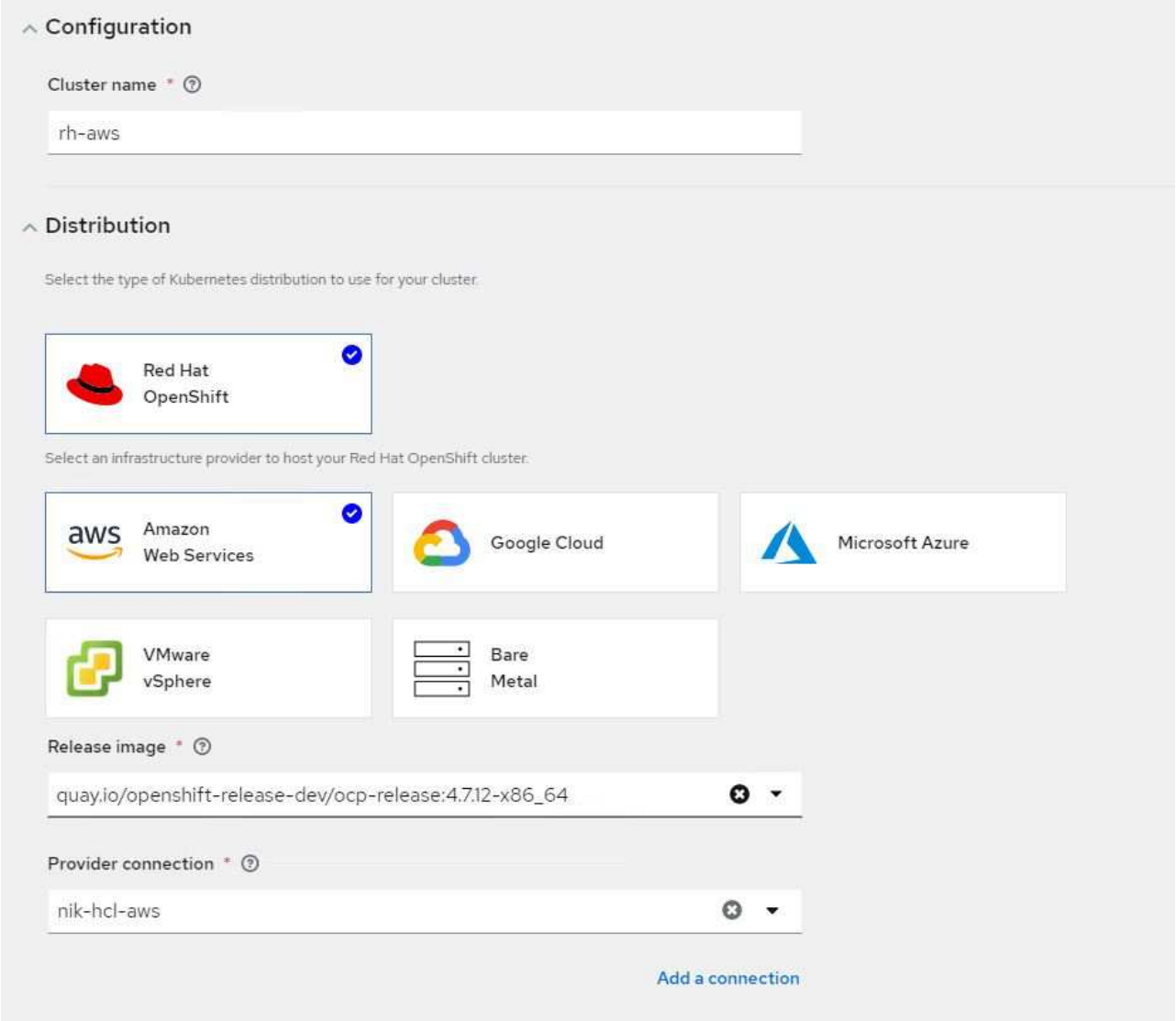


若要管理不同的 OpenShift 集群，您可以建立它們或將它們匯入進階集群管理。

1. 首先導航到自動化基礎設施>叢集。
2. 若要建立新的 OpenShift 集群，請完成以下步驟：
   a. 建立提供者連接：導航至提供者連接並按一下新增連接，提供與所選提供者類型相對應的所有詳細信息，然後按一下新增。



   b. 若要建立新集群，請導航至“集群”，然後按一下“新增集群”>“建立集群”。提供集群和相應提供者的詳細信息，然後按一下「建立」。



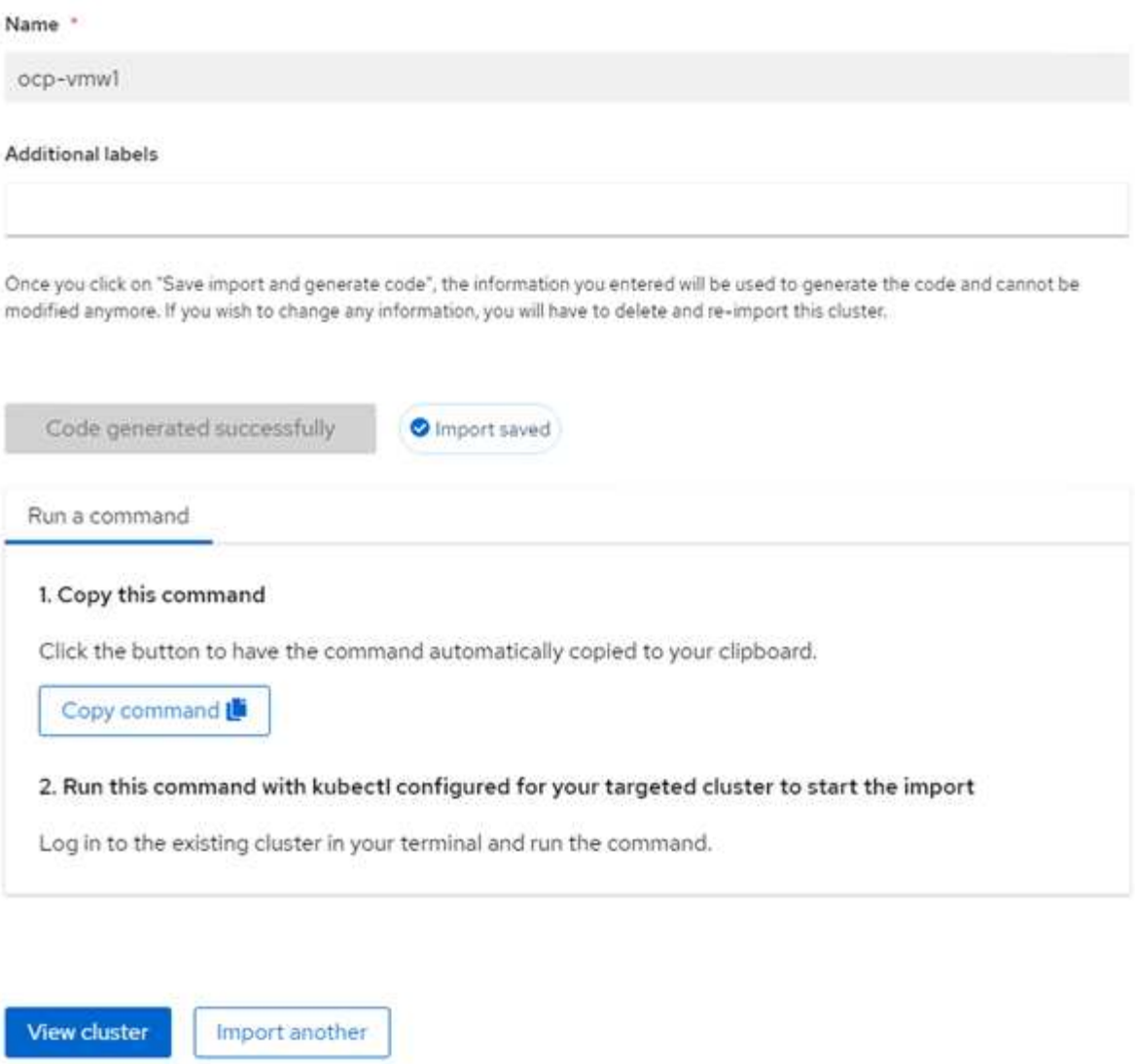
c. 叢集建立完成後，出現在叢集清單中，狀態為Ready。

3. 若要匯入現有集群，請完成以下步驟：

a. 導航至集群並點擊新增集群>匯入現有集群。

b. 輸入叢集名稱，點選儲存匯入並產生程式碼。顯示新增現有叢集的命令。

c. 按一下“複製命令”，在要新增到中心叢集的叢集上執行該命令。這將啟動叢集上必要代理的安裝，並且此過程完成後，叢集將以「就緒」狀態出現在叢集清單中。

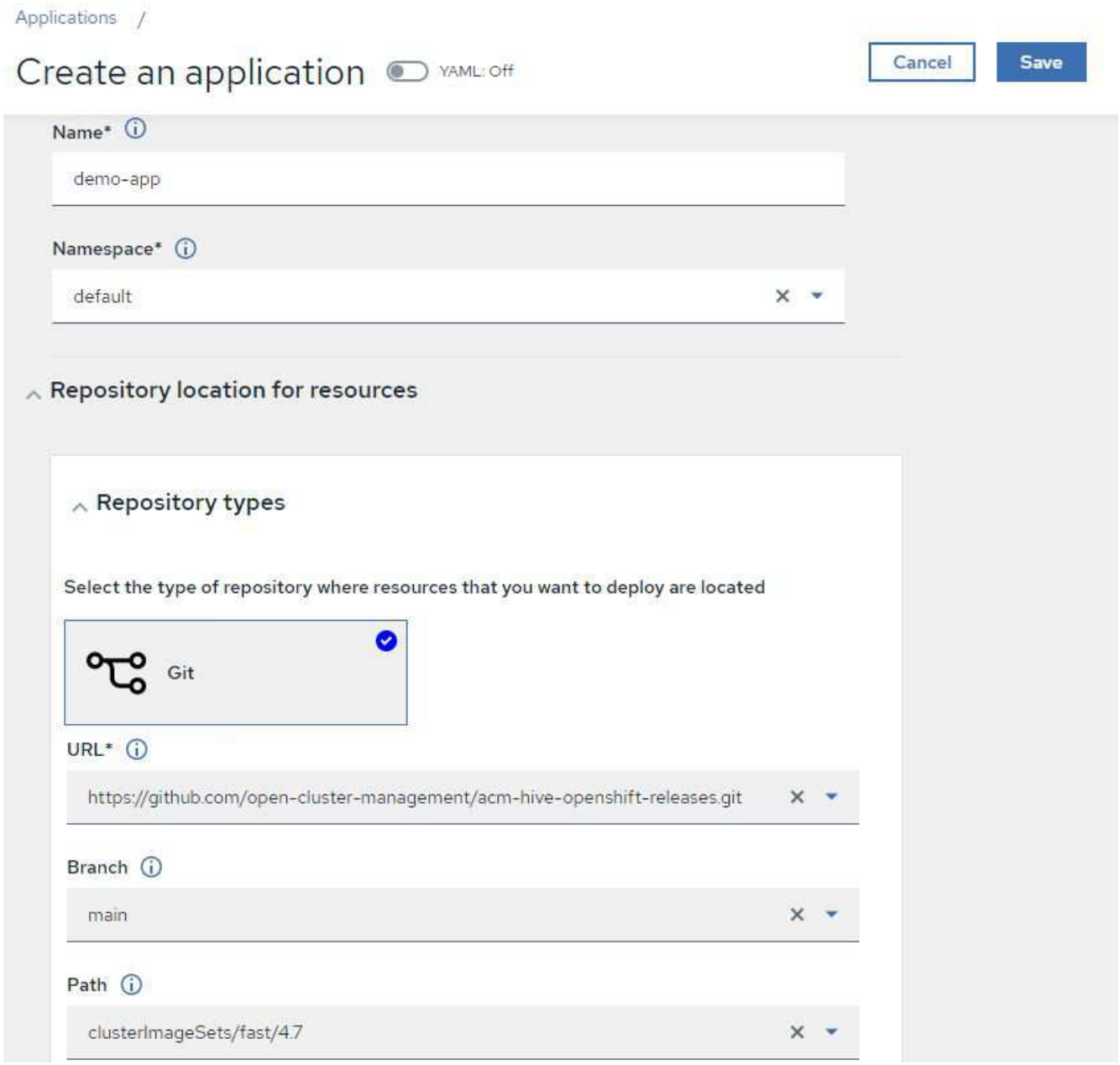


4. 建立並匯入多個叢集後，您可以從單一控制台監控和管理它們。

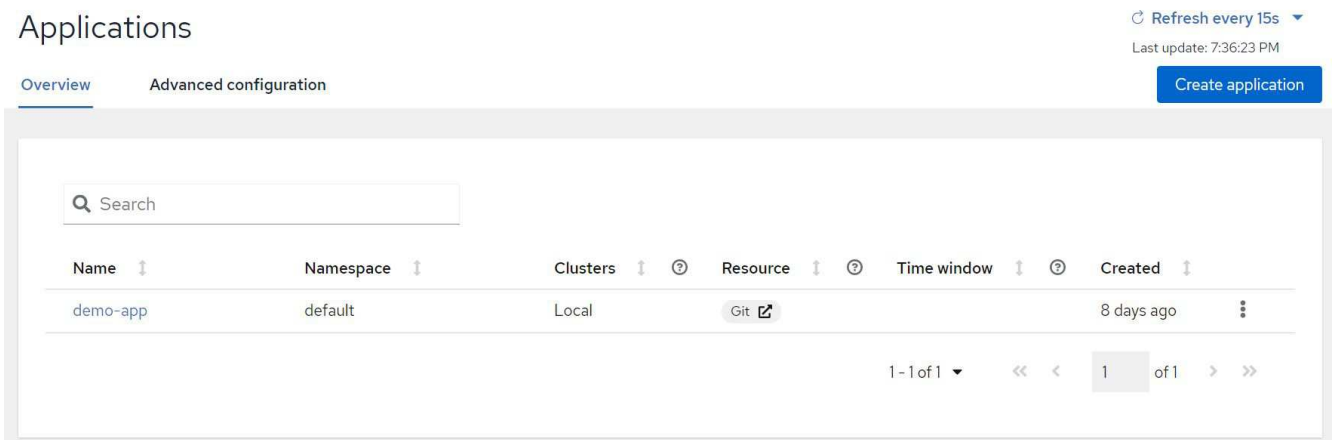
應用程式生命週期管理

要建立一個應用程式並跨一組叢集管理它，

1. 從側邊欄導航到“管理應用程式”，然後按一下“建立應用程式”。提供您想要建立的應用程式的詳細信息，然後按一下「儲存」。



2. 應用程式元件安裝完成後，該應用程式就會出現在清單中。

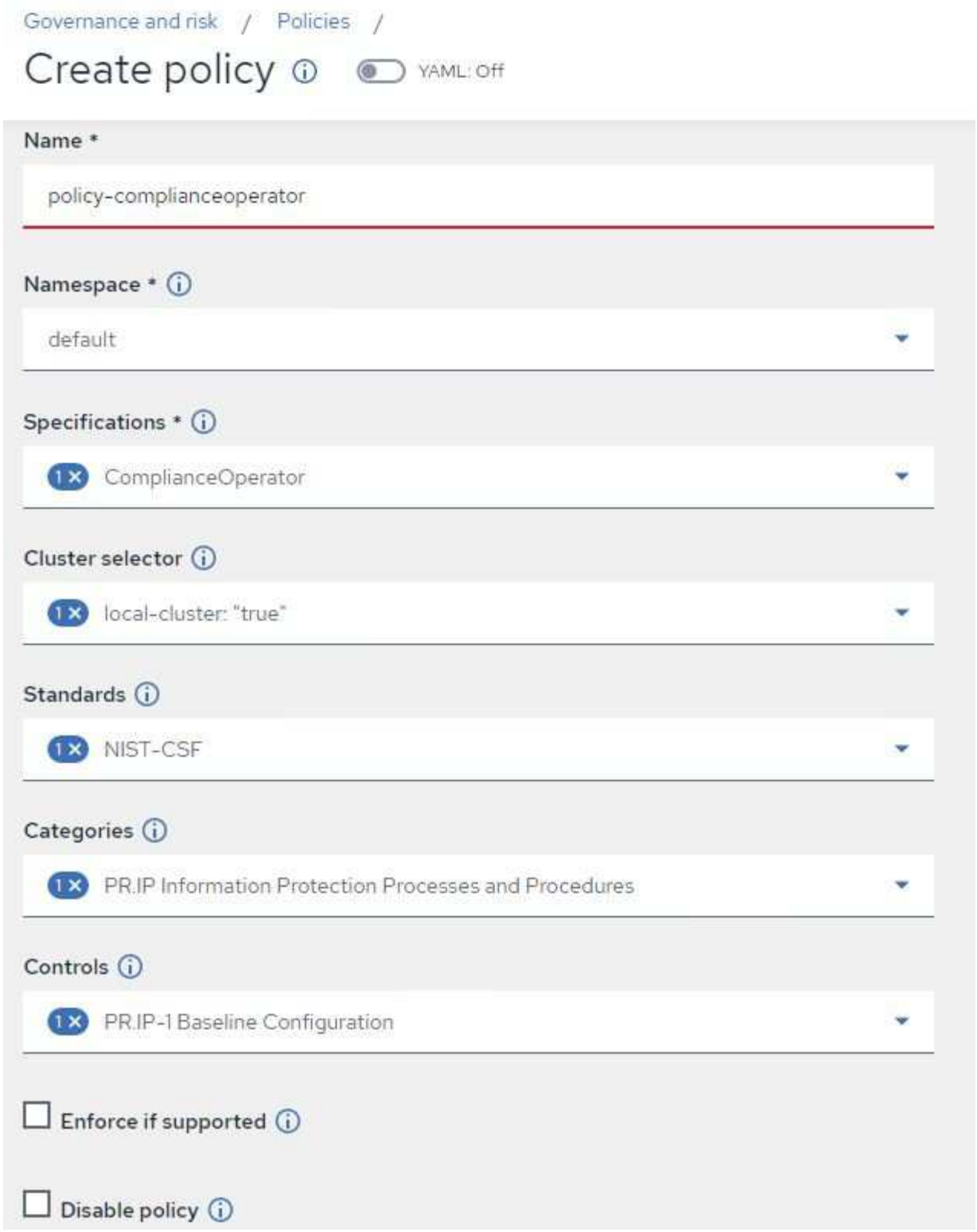


3. 現在可以從控制台監控和管理該應用程式。

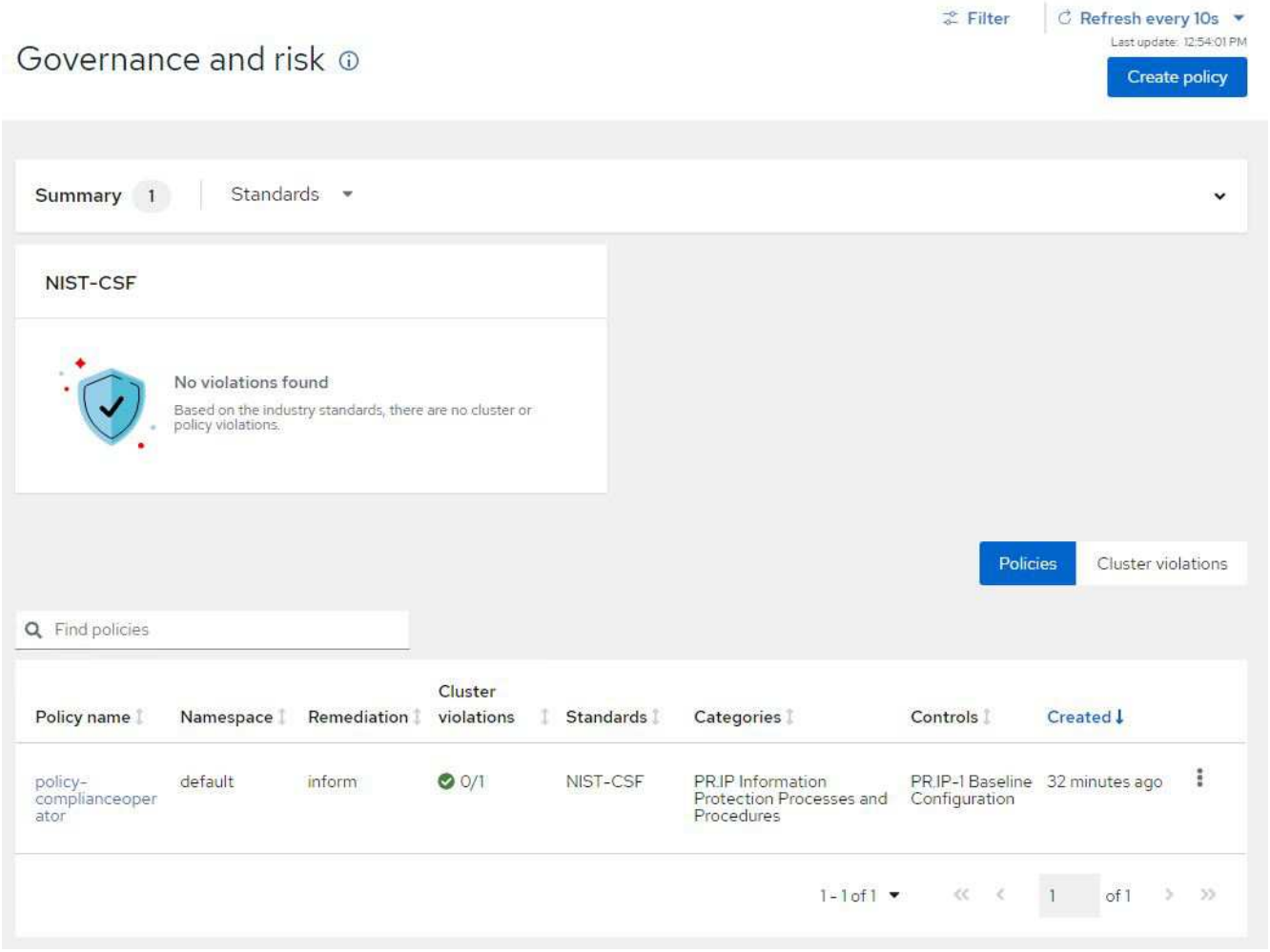
治理與風險

此功能可讓您為不同的叢集定義合規策略並確保叢集遵守該策略。您可以設定策略來通知或補救任何偏離或違反規則的行為。

1. 從側邊欄導覽至「治理和風險」。
2. 若要建立合規性策略，請按一下建立策略，輸入策略標準的詳細信息，然後選擇應遵守此策略的叢集。如果您想自動修正違反此策略的行為，請選取「如果支援則強制執行」複選框，然後按一下「建立」。



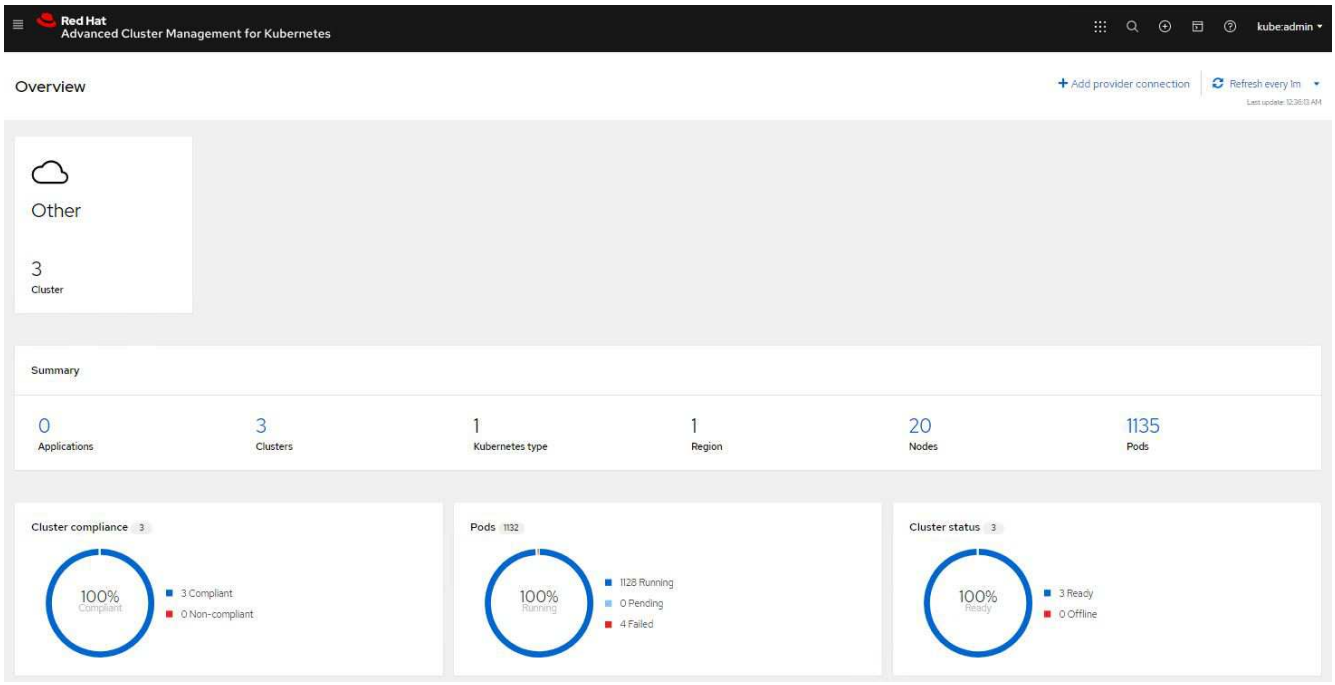
3. 配置所有必要的策略後，可以從進階叢集管理中監控和修復任何策略或叢集違規行為。



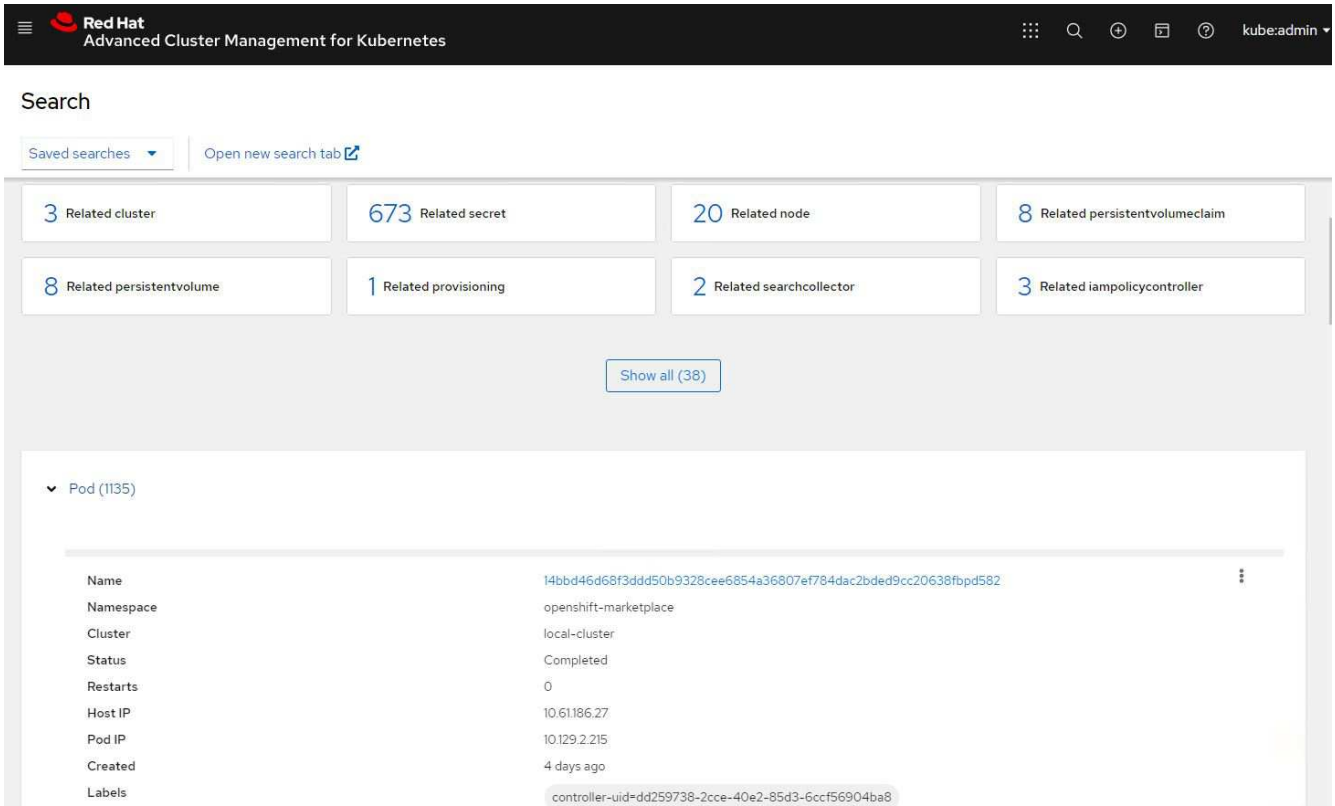
可觀察性

Kubernetes 的高階叢集管理提供了一種監控所有叢集中的節點、pod、應用程式和工作負載的方法。

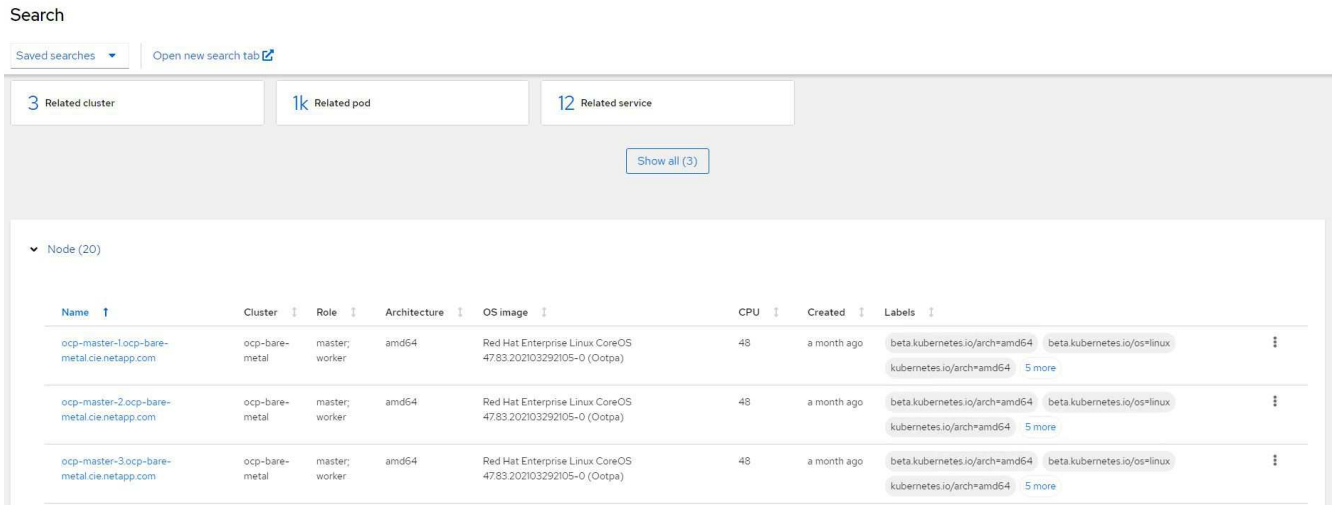
1. 導覽至觀察環境 > 概覽。



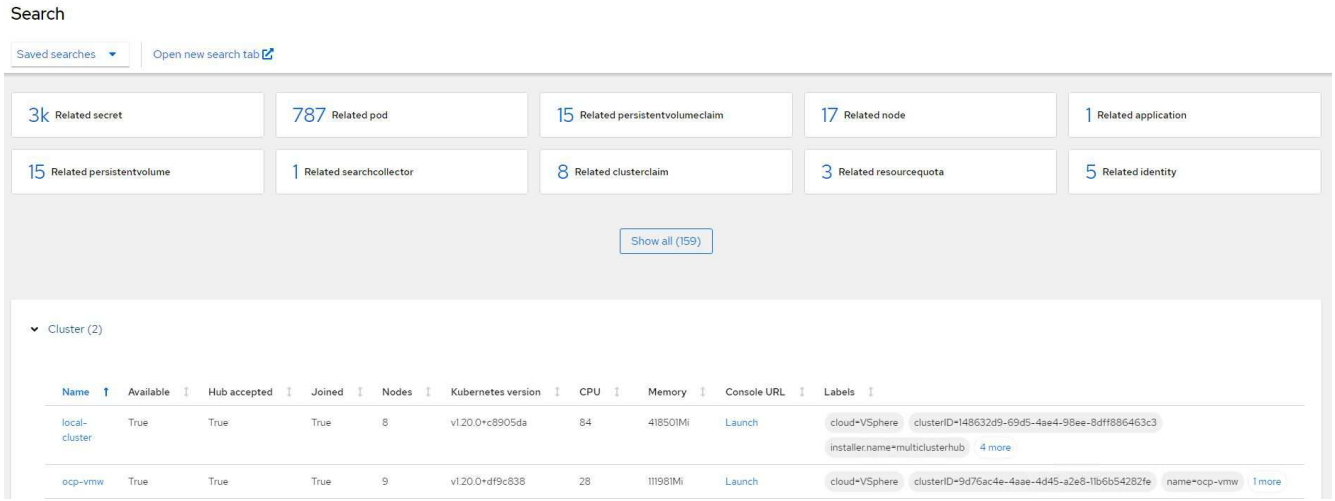
2. 所有叢集中的所有 pod 和工作負載都根據各種過濾器進行監控和排序。點擊Pods即可查看對應數據。



3. 基於各種資料點對叢集中的所有節點進行監控和分析。點擊節點可以深入了解相應的詳細資訊。

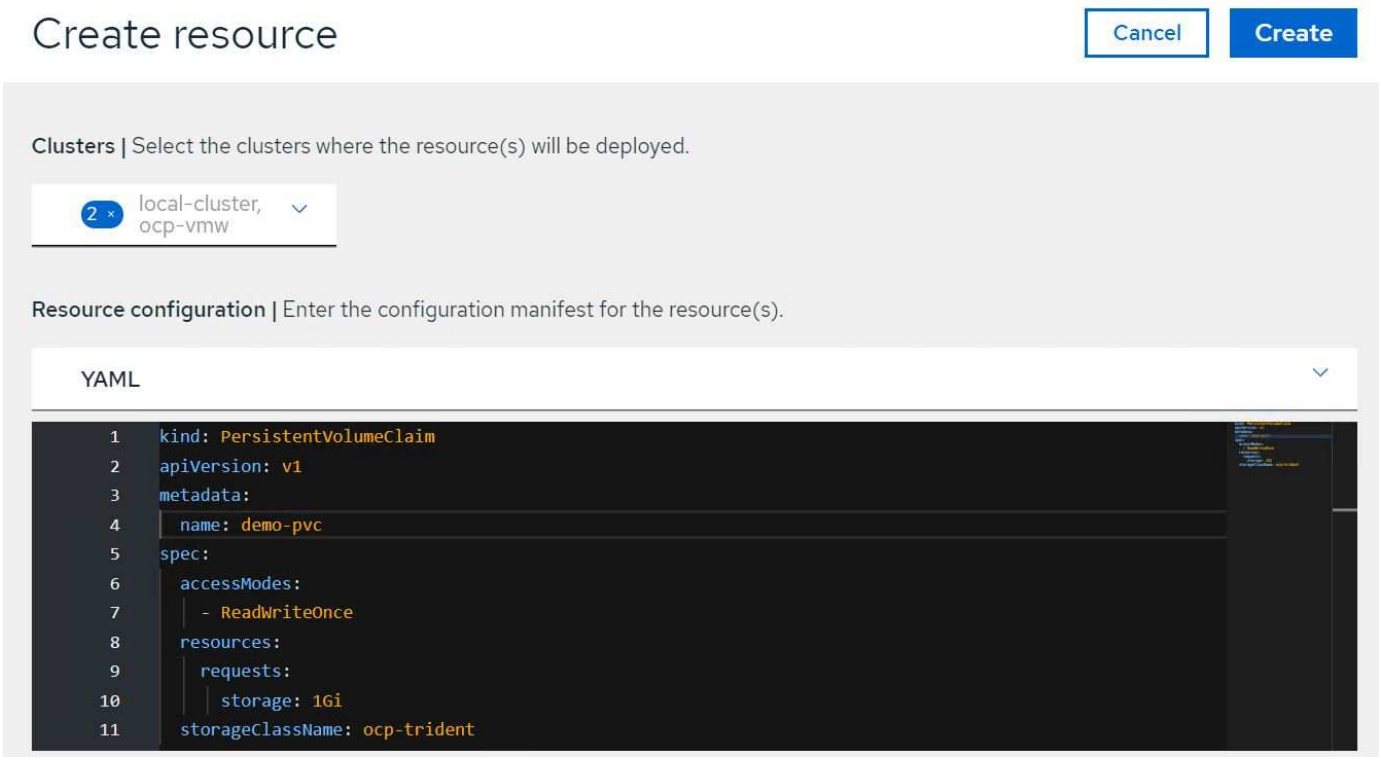


4. 所有叢集都根據不同的叢集資源和參數進行監控和組織。按一下「集群」可查看集群詳細資訊。



在多個集群上建立資源

Kubernetes 的高階叢集管理可讓使用者從控制台同時在一個或多個託管叢集上建立資源。例如，如果您在不同的站點擁有 OpenShift 集群，且這些集群由不同的NetApp ONTAP集群支持，並且想要在兩個站點配置 PVC，則可以單擊頂部欄上的 (+) 號。然後選擇要建立 PVC 的集群，貼上資源 YAML，然後按一下建立。



使用Trident Protect 為容器應用程式和虛擬機器提供資料保護

此解決方案展示如何使用Trident Protect 對容器和虛擬機器執行資料保護作業。

1. 有關在 OpenShift Container 平台中為容器應用程式建立快照和備份以及從中復原的詳細信息，請參閱"這裡"。
2. 有關在 OpenShift Container 平台上部署的 OpenShift Virtualization 中建立和恢復虛擬機備份的詳細信息，請參閱"這裡"。

使用第三方工具對容器應用程式和虛擬機器進行資料保護

此解決方案展示如何使用與 Red Hat OpenShift Container 平台中的 OADP 操作員整合的 Velero 對容器和虛擬機器執行資料保護操作。

1. 有關在 OpenShift Container 平台中建立和恢復容器應用程式備份的詳細信息，請參閱"這裡"。
2. 有關在 OpenShift Container 平台上部署的 OpenShift Virtualization 中建立和恢復虛擬機備份的詳細信息，請參閱"這裡"。

了解有關 Red Hat OpenShift 虛擬化與NetApp儲存整合的其他資源

存取其他資源，這些資源提供有關在各種平台和技術上支援使用ONTAP部署、管理和最佳化 Red Hat OpenShift Virtualization 的更多資訊。

- NetApp文檔

"https://docs.netapp.com/"

- Trident文檔

  "https://docs.netapp.com/us-en/trident/index.html"

- Red Hat OpenShift 文檔

  "https://access.redhat.com/documentation/en-us/openshift_container_platform/4.7/"

- Red Hat OpenStack 平台文檔

  "https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/16.1/"

- Red Hat 虛擬化文檔

  "https://access.redhat.com/documentation/en-us/red_hat_virtualization/4.4/"

- VMware vSphere 文檔

  "https://docs.vmware.com/"