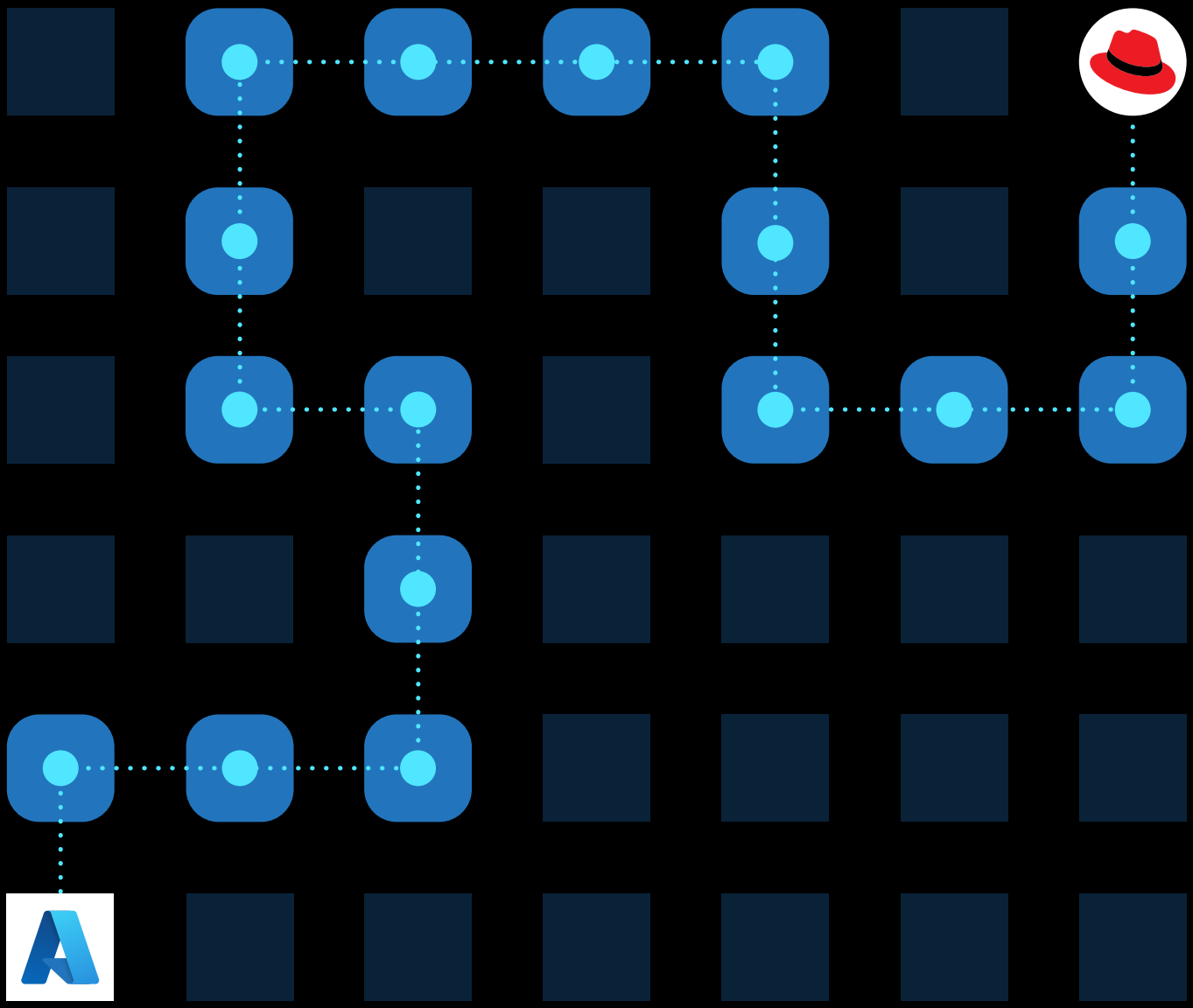
Serie di ebook

Microsoft Azure

Guida introduttiva ad Azure Red Hat OpenShift

Dalla Proof of Concept alla produzione



Guida introduttiva ad Azure Red Hat OpenShift

Capitolo 1

Contatta Microsoft e Red Hat

Ti interessa Azure Red Hat OpenShift e vuoi maggiori informazioni? Contattaci e parliamone insieme. Sebbene questa guida sia stata ideata per fornirti istruzioni sull'utilizzo della piattaforma, Red Hat e Microsoft mettono a tua disposizione una serie di risorse che vanno oltre la guida stessa, per garantirti un'esperienza ancora più avanzata. Insieme, vogliamo che Azure Red Hat OpenShift diventi una piattaforma applicativa in grado di soddisfare le tue esigenze di innovazione.

Azure Red Hat OpenShift è stata creata per andare incontro alle esigenze dei clienti che l'hanno chiesta. Il numero di clienti e aziende di Red Hat e Microsoft, così come di piccole organizzazioni che distribuiscono l'offerta di prodotti Red Hat in Microsoft Azure, non è mai stato così alto. OpenShift su Azure costituisce un'offerta autogestita totalmente supportata ormai da molti anni, ma le attività di installazione, deployment e gestione ordinaria del cluster richiedono esperienza in Kubernetes e tempo, che viene necessariamente sottratto ad altre attività cruciali per l'azienda.

Il numero di clienti che esegue il deployment con l'offerta gestita Azure Red Hat OpenShift senza alcuna difficoltà è sempre più alto; osserviamo perciò una tendenza a investire meno tempo nell'installazione e nelle attività ordinarie e a dedicarne di più alle applicazioni.

Questa guida, basata sulla nostra esperienza pratica, intende illustrare ai nostri clienti le procedure ottimali per creare applicazioni in Azure Red Hat OpenShift. È stata progettata come guida di supporto autonomo. Puoi leggere i capitoli in sequenza, dall'introduzione alla conclusione, oppure in base allo specifico argomento di tuo interesse.

Destinatari della guida

Questa guida è pensata per un pubblico specializzato (sviluppatori, operatori e architetti di piattaforme) che vogliono migliorare le proprie capacità di creazione e distribuzione di applicazioni avvalendosi di Azure e Red Hat OpenShift per il deployment a pieno servizio di cluster OpenShift totalmente gestiti.

Cosa troverai nella guida

La guida illustra gli argomenti necessari per comprendere e adottare Azure Red Hat OpenShift, dal Proof of Concept nell'organizzazione al deployment in produzione.

Il Capitolo 2, *Introduzione a Red Hat OpenShift,* si apre con una presentazione di Red Hat OpenShift e dei motivi che portano così tanti sviluppatori, operatori e architetti di piattaforma a scegliere questa soluzione come piattaforma applicativa e quali sono i vantaggi che ne ottengono. Vengono illustrate le ragioni del grande successo riscosso da Red Hat OpenShift.

Il Capitolo 3, *Azure Red Hat OpenShift,* spiega in dettaglio il servizio gestito e come viene erogato ai clienti.

Il Capitolo 4, *Domande sul pre-provisioning dell'architettura enterprise*, affronta importanti valutazioni di cui tener conto prima di effettuare il deployment di Azure Red Hat OpenShift e spiega alcune procedure ottimali che abbiamo appreso tramite i molti clienti che le hanno applicate in contesti reali.

Il Capitolo 5, *Provisioning di un cluster Azure Red Hat OpenShift,* illustra le principali risorse presenti nella documentazione ufficiale e necessarie per il deployment di un cluster Azure Red Hat OpenShift.

Il Capitolo 6, *Post-provisioning - Operazioni ordinarie,* esamina le attività di ordinaria gestione successive al provisioning. Dove possibile, questa guida evidenzia come tali attività siano semplificate dall'utilizzo del servizio gestito Azure Red Hat OpenShift.

Il Capitolo 7, *Deployment di applicazioni in Red Hat OpenShift,* è una breve guida alla distribuzione delle applicazioni nella piattaforma, un'attività simile a quella compiuta con Red Hat OpenShift in esecuzione in qualsiasi altro ambiente.

Il Capitolo 8, *Esplorazione della piattaforma applicativa*, contiene una panoramica guidata di alcune delle funzionalità chiave della piattaforma applicativa: registro dei container, pipeline, serverless e simili.

Il Capitolo 9, *Integrazione con altri servizi*, esamina l'integrazione della piattaforma tramite Azure Service Operator, Azure DevOps e servizi simili.

Il Capitolo 10, *Onboarding di carichi di lavoro e team,* chiude il documento fornendo alcune procedure ottimali e raccomandazioni sull'onboarding dei team che si occupano dell'applicazione e su come incentivare l'adozione del servizio nell'organizzazione.

Il Capitolo 11, *Conclusioni*, riassume il contenuto del documento.

Il Capitolo 12, *Glossario,* illustra alcuni dei termini utilizzati nel testo.

Capitolo 2

Introduzione a Red Hat OpenShift

Il capitolo offre una breve introduzione a Red Hat OpenShift illustrando cos'è, come viene utilizzato e i vantaggi che i clienti Red Hat ottengono dal servizio. Vengono presentati i concetti di base utili a chi si avvicina a OpenShift per la prima volta o come rapido riferimento per chi desidera approfondire la piattaforma.

Panoramica di Red Hat OpenShift

Red Hat OpenShift è una piattaforma applicativa enterprise-ready che offre operazioni automatizzate in tutto lo stack per gestire deployment di cloud ibridi e multicloud. È ottimizzato per migliorare la produttività degli sviluppatori e promuovere l'innovazione. Grazie all'automazione delle operazioni e alla gestione ottimizzata del ciclo di vita, Red Hat OpenShift offre agli sviluppatori gli strumenti necessari per creare e distribuire nuove applicazioni, semplificando ai team operativi le attività di provisioning, gestione e ridimensionamento della piattaforma Kubernetes.

I team di sviluppo possono accedere a immagini e soluzioni convalidate da centinaia di partner, scansionate e firmate per la sicurezza lungo l'intero processo di distribuzione. Un'unica piattaforma consente di accedere on-demand alle immagini e in modo nativo a un ampio ventaglio di servizi cloud di terzi.

Ovunque si trovino, i team operativi ottengono visibilità sui deployment e sui team, grazie ai registri e al monitoraggio integrati. Gli operatori Red Hat Kubernetes incorporano l'esclusiva logica applicativa che abilita le funzionalità del servizio, che viene non solo configurato ma anche ottimizzato per ottenere prestazioni eccellenti, aggiornato e con patch applicate da SO, il tutto con una sola operazione. In breve, Red Hat OpenShift costituisce un unico punto di riferimento che consente alle organizzazioni di sfruttare appieno tutte le potenzialità dei team IT e di sviluppo.

Servizi cloud OpenShift: risparmio sui costi e vantaggi per l'azienda

Migliaia di clienti si affidano a Red Hat OpenShift per distribuire in modo efficiente le applicazioni, migliorare le relazioni con i clienti e differenziarsi sul mercato per diventare leader nel proprio settore. Una ricerca di IDC, presentata nel white paper dal titolo "The business value of Red Hat OpenShift" del marzo 2021, sottolinea il maggior valore ottenuto dalle organizzazioni intervistate con la piattaforma Red Hat OpenShift, distribuendo nelle proprie aziende applicazioni e funzionalità di qualità più elevata in tempi più brevi e ottimizzando al tempo stesso i costi destinati a sviluppo e tecnologie IT e i tempi di lavoro del personale.

Dalla ricerca emergono alcuni dati chiave:

- Ritorno sugli investimenti (ROI) pari al 636% in 5 anni
- Recupero degli investimenti in 10 mesi
- Costi operativi ridotti del 54% in 5 anni
- Triplicano le funzionalità rilasciate nell'anno
- 20% di aumento della produttività degli sviluppatori
- Tempi di fermo non pianificati ridotti del 71%
- Efficienza dei team responsabili dell'infrastruttura IT migliorata del 21%

Fonte: white paper IDC sul valore aziendale, sponsorizzato da Red Hat: "*The Business Value of Red Hat OpenShift*. Doc. # US47539121, marzo 2021

Realizzati partendo da queste fondamenta di Red Hat OpenShift, i servizi cloud Red Hat OpenShift, che includono Azure Red Hat OpenShift, offrono all'azienda ulteriori vantaggi. Nello studio di Forrester dal titolo "The Total Economic Impact™ of Red Hat OpenShift Cloud Services - Cost Savings and Business Benefits", vengono evidenziati i numerosi e importanti vantaggi economici.

I principali sono elencati di seguito:

- Ritorno sugli investimenti (ROI) pari al 468%
- Valore attuale netto (VAN) pari a 4,08 milioni di dollari
- Recupero degli investimenti in 6 mesi

Al di là dei vantaggi economici, il report di Forrester quantifica anche altri benefici, tra cui:

- **Velocità di sviluppo aumentata**: l'impiego dei servizi cloud di Red Hat OpenShift consente alle aziende di ridurre il ciclo di sviluppo fino al 70%.
- **Il 20% del tempo degli sviluppatori viene recuperato dal lavoro di manutenzione dell'infrastruttura**: gli intervistati hanno notato che i servizi cloud di Red Hat OpenShift evitano agli sviluppatori di dover gestire l'infrastruttura di sviluppo delle applicazioni, consentendo loro di concentrarsi al 100% sulla creazione del prodotto o della soluzione. Nel corso di 3 anni, il tempo guadagnato e reinvestito dagli sviluppatori è equivale a oltre 2,3 milioni di dollari.
- **Efficienza operativa raddoppiata**: trattandosi di un servizio gestito, gli intervistati hanno notato che utilizzando i servizi cloud di Red Hat OpenShift possono riassegnare il 50% del personale DevOps, prima responsabile della gestione dell'infrastruttura, ad altre attività più produttive. In 3 anni, questa maggiore efficienza operativa ha un valore pari a oltre 1,3 milioni di dollari.*

Il report individua anche i seguenti vantaggi non quantificati:

- **Soddisfazione e fidelizzazione degli sviluppatori**: gli intervistati sottolineano i vantaggi garantiti agli sviluppatori dai servizi cloud Red Hat OpenShift, come la possibilità di suddividere gli aggiornamenti in componenti più piccoli, evitando di eseguire i test in tempi brevi e di rispondere a un fuoco incrociato di richieste in fase di produzione.
- **Sicurezza e riduzione dei rischi**: gli intervistati affermano che i servizi cloud Red Hat OpenShift hanno consentito l'automazione di alcuni aggiornamenti di funzionalità e sicurezza, eliminando la necessità di interventi manuali ma garantendo al tempo stesso la sicurezza degli ambienti.
- **Affidabilità**: gli intervistati segnalano che l'impiego dei servizi cloud Red Hat OpenShift ha reso la piattaforma applicativa più affidabile nel lungo termine, poiché si verificano meno interruzioni o guasti di sistema anche se l'ambiente è in espansione.
- **Portabilità e continuità operativa**: gli intervistati sottolineano come i servizi cloud di Red Hat OpenShift garantiscono la continuità operativa e sono di supporto alla strategia di ripristino di emergenza, grazie alla portabilità, scalabilità e flessibilità della soluzione.

Fonte: Forrester: "*The Total Economic Impact of Red Hat OpenShift Cloud Services*", dicembre 2021

***Per approfondire:** Le imprese ottengono una maggiore agilità grazie ai servizi cloud

"Red Hat OpenShift o Kubernetes su bare metal?": quanto costa realizzare una piattaforma applicativa Kubernetes proprietaria

Ci si riferisce spesso a Red Hat OpenShift come al "Kubernetes enterprise", ma a prima vista può essere difficile capire cosa ciò significhi. Di frequente i clienti chiedono se sia meglio OpenShift o Kubernetes su bare metal. È tuttavia importante capire che **OpenShift utilizza già Kubernetes**. Nell'architettura generale di OpenShift, è su Kubernetes che poggiano tanto le fondamenta della piattaforma OpenShift, quanto gran parte della strumentazione per l'esecuzione della stessa.

Kubernetes è un progetto open source di straordinaria rilevanza, uno dei principali dell'iniziativa Cloud Native Computing Foundation e una tecnologia essenziale per l'esecuzione dei container.

Tuttavia, la domanda concreta che i potenziali utenti di OpenShift si fanno è: "**Posso eseguire le mie applicazioni utilizzando solo Kubernetes**?" Molte organizzazioni completano il deployment di Kubernetes e scoprono di poter eseguire un container e perfino un'applicazione enterprise in pochissimi giorni. Non appena passano alle attività ordinarie, emergono però le esigenze di sicurezza. Con l'aumento delle applicazioni distribuite, alcune organizzazioni si vedono costrette a realizzare il proprio servizio **Platform-as-a-Service (PaaS)** con la tecnologia Kubernetes. Aggiungono un controller del traffico in ingresso open source, scrivono qualche script per connettersi alle pipeline di **integrazione e distribuzione continue (CI/CD)**, e quindi provano a distribuire un'applicazione più complessa... ed è qui che iniziano i problemi. Se il deployment di Kubernetes fosse la punta di un iceberg, la complessità delle attività ordinarie corrisponderebbe alla grande parte nascosta e sommersa nell'acqua, potenzialmente in grado di creare enormi danni.

Risolvere questi ostacoli e problemi è possibile, ma per farlo serve un team operativo formato da numerose persone, impegnate per settimane e mesi nel realizzare e poi gestire un servizio "PaaS personalizzato e basato su Kubernetes". Ciò genera inefficienze organizzative, difficoltà nell'offerta di supporto dal punto di vista della sicurezza e della certificazione, e la necessità di dover sviluppare ogni componente da zero al momento dell'onboarding dei team di sviluppo. In un ipotetico elenco è possibile raggruppare come segue tutte le attività necessarie per realizzare e gestire questa piattaforma Kubernetes personalizzata, con tutti i componenti extra essenziali per eseguire correttamente i container:

- **Gestione del cluster**: include installazione dei SO, applicazione di eventuali patch agli stessi, installazione di Kubernetes, configurazione della rete CNI, integrazione dell'autenticazione, configurazione del traffico in ingresso e in uscita, configurazione dello storage permanente, consolidamento dei nodi, patching di sicurezza e configurazione dell'ambiente cloud/multicloud sottostante.
- **Servizi applicativi**: include aggregazione dei registri, controlli di integrità, monitoraggio delle prestazioni, patching di sicurezza, registro dei container, configurazione del processo di staging dell'applicazione.
- **Integrazione degli sviluppatori**: include integrazione delle pipeline CI/CD, integrazione degli strumenti di sviluppo/IDE, integrazione del framework, compatibilità del middleware, realizzazione di dashboard per le performance delle applicazioni e RBAC.

Sebbene sia possibile aggiungere all'elenco molte altre azioni e tecnologie (molte delle quali essenziali a qualsiasi organizzazione per un uso efficace dei container), la complessità, il tempo e l'impegno necessario per organizzarle tutte è insignificante rispetto alla gestione continua di quei singoli componenti. Ogni integrazione deve essere accuratamente testata e ogni componente e attività avrà cicli di rilascio, criteri di sicurezza e patch differenti.

Quali sono i vantaggi di Red Hat OpenShift rispetto a Kubernetes su bare metal?

Quando un'organizzazione esegue i container in produzione, realizzati solo su Kubernetes bare metal, i componenti citati nella sezione precedente sono installati e integrati insieme per creare la piattaforma applicativa prevista dal progetto.

Combinando tutti gli elementi citati in un'unica piattaforma, Azure Red Hat OpenShift semplifica le attività dei team IT fornendo tutti gli strumenti necessari per operare ai team che si occupano delle applicazioni. Questi argomenti verranno approfonditi più avanti nella guida. Ora analizziamo alcune differenze chiave tra le due soluzioni.

- **Deployment semplice:** il deployment di un'applicazione in Kubernetes può richiedere molto tempo. Significa infatti trasferire il codice GitHub in un sistema, avviare un container, ospitarlo in un registro come Docker Hub e infine comprendere la pipeline CI/CD, un'attività potenzialmente complicata. OpenShift, invece, automatizza le attività più complesse e quelle di back-end, perché richiede allo sviluppatore solo di creare il progetto e caricare il codice.
- **Sicurezza:** oggi sulla maggior parte dei progetti Kubernetes lavorano team formati da più sviluppatori e operatori. Anche se Kubernetes ora supporta metodi di controllo quali RBAC e IAM, richiede comunque dispendiose attività manuali di installazione e configurazione. Dopo anni di esperienza d'utilizzo, Red Hat e OpenShift hanno identificato le procedure di sicurezza ottimali, ora direttamente disponibili agli utenti. È sufficiente aggiungere i nuovi utenti e OpenShift gestirà aspetti quali lo spazio dei nomi e la creazione di diversi criteri di sicurezza.
- **Flessibilità:** Azure Red Hat OpenShift rende disponibili procedure collaudate e ben note per il deployment, la gestione e gli aggiornamenti. Tutte le complesse attività di back-end vengono eseguite con un intervento minimo dell'operatore, che può così rilasciare le applicazioni più rapidamente. Sebbene questo metodo sia apprezzato dai team che preferiscono un approccio semplificato e ottimizzato, la piattaforma Kubernetes consente ai team DevOps di personalizzare manualmente le pipeline di flussi CI/CD, lasciando più spazio alla versatilità e alla creatività nello sviluppo dei processi.

- **Attività quotidiane:** i cluster sono composti da un gruppo di più VM; inevitabilmente, i team operativi dovranno avviare le nuove VM da aggiungere a un cluster. In Kubernetes il processo di configurazione può essere lungo e complesso, perché richiede la creazione degli script di configurazione di attività quali la registrazione automatica o l'automazione del cloud. In Azure Red Hat OpenShift le operazioni di provisioning, ridimensionamento e aggiornamento del cluster sono automatizzate e gestite dalla piattaforma.
- **Gestione**: benché sia possibile utilizzare le dashboard predefinite di Kubernetes incluse in qualsiasi distribuzione, la maggior parte degli sviluppatori esige una piattaforma più robusta. Azure Red Hat OpenShift fornisce un'ottima web console basata sull'API Kubernetes e funzionalità con le quali i team operativi possono gestire i propri carichi di lavoro.

Il *Capitolo 8, Esplorazione della piattaforma applicativa*, contiene una descrizione dettagliata di alcune delle funzionalità chiave a valore aggiunto di Azure Red Hat OpenShift.

Riepilogo

Azure Red Hat OpenShift è scelto da molti clienti comuni di Red Hat e Microsoft come piattaforma applicativa preferita per l'adozione di applicazioni containerizzate.

Nel capitolo successivo esamineremo Red Hat OpenShift come servizio cloud e ne approfondiremo gli aspetti legati all'architettura, all'integrazione e alla gestione.

Capitolo 3

Azure Red Hat OpenShift

Azure Red Hat OpenShift consente di distribuire cluster Red Hat OpenShift completamente gestiti senza doversi occupare di creare e gestire l'infrastruttura su cui eseguirli.

Azure Red Hat OpenShift è una piattaforma applicativa cloud native e on demand, progettata, gestita e supportata congiuntamente da Microsoft e Red Hat. Un team di **Site Reliability Engineering** (**SRE**) specializzato garantisce l'automazione, la scalabilità e la sicurezza dei cluster OpenShift e lavora fianco a fianco per fornire un'esperienza di supporto integrata. Non ci sono macchine virtuali da gestire e non è richiesta alcuna attività di patching. I nodi del piano di controllo e quelli di lavoro sono corretti, aggiornati e monitorati per tuo conto da Red Hat e Microsoft. I tuoi cluster Azure Red Hat OpenShift sono distribuiti nella tua sottoscrizione di Azure e inclusi nella tua fattura di Azure.

Accelera la distribuzione delle applicazioni con Azure Red Hat OpenShift:

- Fornisci agli sviluppatori gli strumenti per innovare in modo produttivo con pipeline CI/CD integrate, per poi collegare facilmente le applicazioni a centinaia di servizi Azure, come MySQL, PostgreSQL, Redis e Azure Cosmos DB.
- Elimina la complessità e gli ostacoli alla produttività automatizzando provisioning, configurazione e operazioni.
- Sfrutta la scalabilità alle tue condizioni: puoi avviare in pochi minuti un cluster altamente disponibile con tre nodi di lavoro e ridimensionarlo in funzione dell'evoluzione della domanda di applicativi; è inoltre disponibile una vasta gamma di nodi di lavoro standard, ad alta memoria o ad alta CPU.
- Ottieni operazioni, sicurezza e conformità di livello enterprise, con un'esperienza di supporto integrata.

Approfondiamo ora i dettagli tecnici che consentono di creare e far funzionare Red Hat OpenShift.

Architettura

Azure Red Hat OpenShift utilizza i servizi dell'infrastruttura Azure: macchine virtuali, gruppi di sicurezza di rete, account di storage e altri servizi di Azure, come base per l'installazione di Red Hat OpenShift. L'architettura di Azure è completamente distribuita nella tua sottoscrizione di Azure, il che ne facilita l'integrazione con gli altri servizi già in esecuzione nell'account.

Lo stesso OpenShift è basato su Red Hat Enterprise Linux CoreOS, che ospita l'architettura basata sui microservizi, piccole unità disaccoppiate che lavorano insieme. Su ogni macchina virtuale viene eseguito il servizio `kubelet`, che costituisce l'elemento di base del cluster Kubernetes. Il database dietro a questa architettura, eseguito nel piano di controllo, è **etcd**, un archivio affidabile di coppie chiave-valore in cluster.

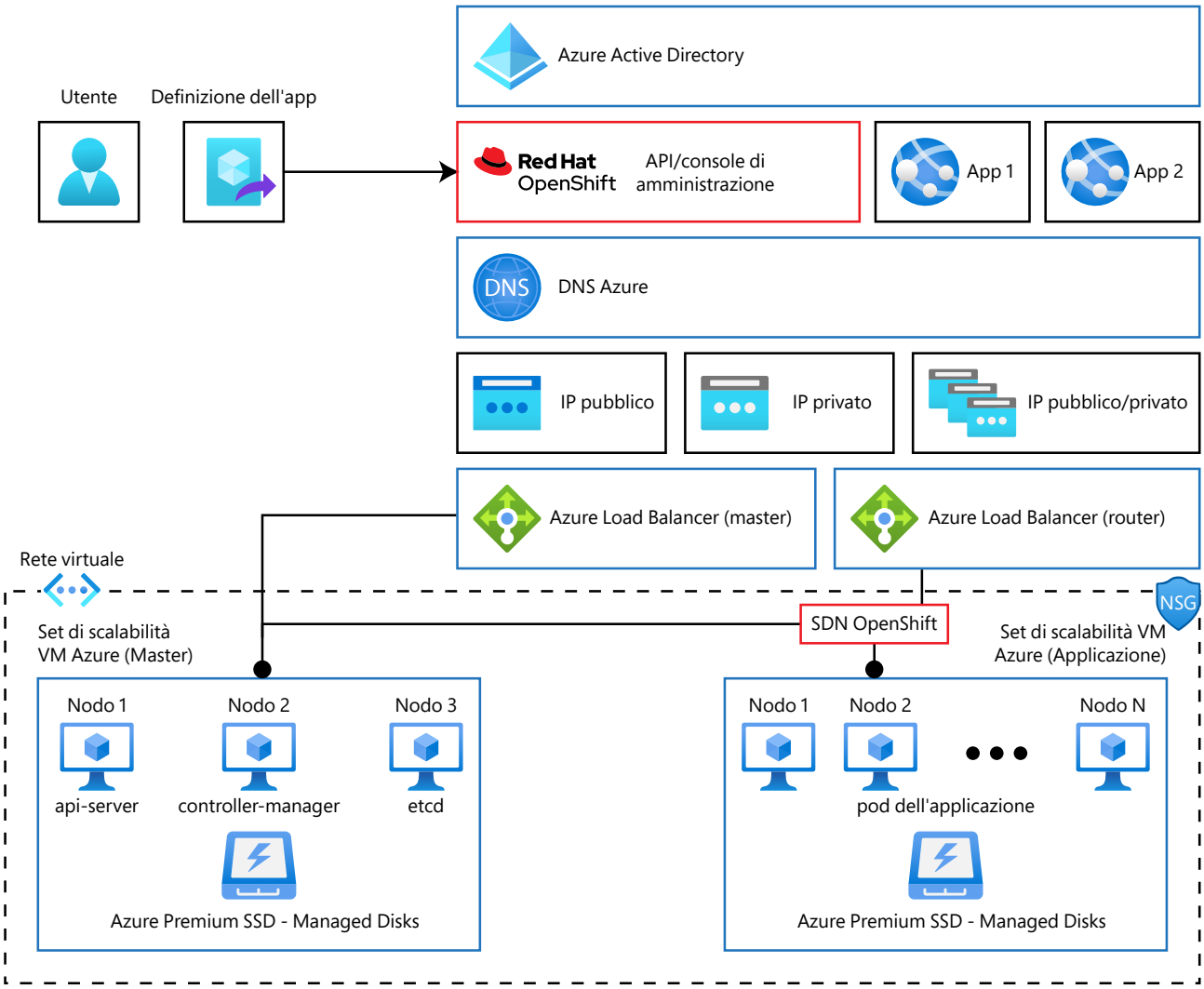


Figura 3.1: Architettura di Azure Red Hat OpenShift

Le due sezioni successive, *Elaborazione: nodi di controllo e di lavoro* e *Networking*, approfondiscono quanto riepilogato nell'immagine.

Claborazione: nodi di controllo e di lavoro

L'architettura di Red Hat OpenShift viene eseguita su macchine virtuali (nodi) che hanno uno di questi tre ruoli specifici: controllo, infrastruttura o applicazione. Poiché al momento della stesura di questo documento la versione 4 di Azure Red Hat OpenShift non supporta i nodi di infrastruttura, il testo illustra solo i nodi di controllo e di lavoro.

I nodi di **controllo** (chiamati anche nodi **master**) sono macchine virtuali che contengono i componenti fondamentali del cluster Kubernetes. Includono il server API, il server manager del controller, l'utilità di pianificazione ed etcd. Gestiscono i nodi del cluster Kubernetes e pianificano l'esecuzione dei pod nei nodi di lavoro.

- **Il server dell'API Kubernetes** convalida e configura i dati per i pod, i servizi e i controller di replica. Costituisce inoltre un punto focale per lo stato condiviso del cluster.
- **Il manager del controller di Kubernetes** osserva etcd per rilevare le modifiche agli oggetti, come replica, spazio dei nomi e oggetti del controller dell'account del servizio, e quindi utilizza l'API per applicare la condizione specificata. Diversi processi creano un cluster con un leader attivo alla volta.
- **L'unità di pianificazione di Kubernetes** monitora i pod appena creati a cui non è assegnato un nodo e sceglie il nodo migliore in cui ospitare il pod.
- **etcd** memorizza la condizione master permanente, mentre gli altri componenti monitorano etcd per rilevare le modifiche che consentono loro di passare alla condizione specificata.

I nodi dell'**applicazione** sono la posizione in cui le applicazioni vengono effettivamente eseguite.

Ciascun nodo di un cluster Kubernetes esegue un servizio chiamato `kubelet`, che gestisce l'**interfaccia di runtime del container** (**cri-o**), il proxy del servizio e altri servizi essenziali che vengono eseguiti su ciascun nodo. Tutti i nodi sono connessi tramite la tecnologia di networking software defined di OpenShift. In Azure Red Hat OpenShift, si tratta di una **Open Virtual Network (OVN)**, eseguita in una rete virtuale di Azure.

Azure Red Hat OpenShift crea nodi che vengono eseguiti su macchine virtuali Azure collegate a dischi di Azure per lo storage. Si tratta di dischi di grandi dimensioni, da 1 TB, necessari perché in Azure l'IOPS garantito per le prestazioni di storage è collegato alle dimensioni del disco. 1 TB di dimensione garantisce la larghezza di banda sufficiente per il disco di base utilizzato dal database etcd.

Networking

Azure Red Hat OpenShift richiede una singola rete virtuale Azure con due subnet configurate, una per i nodi del piano di controllo e una per i nodi di lavoro. La dimensione minima di entrambe le reti è /27 (32 indirizzi). È tuttavia importante non utilizzare una dimensione troppo piccola se si prevede di dimensionare successivamente il cluster.

I due Azure Load Balancer (indicati come **Master** e **Router** nell'immagine) indirizzano il traffico come indicato di seguito:

- Load balancer del piano di controllo/Master: indirizza il traffico in ingresso per gli utenti all'API OpenShift. Fornisce anche la connessione in uscita per i nodi del piano di controllo
- Load balancer dell'applicazione/Router: indirizza il traffico in ingresso alle applicazioni eseguite in OpenShift, tramite un "router" OpenShift o controller del traffico in ingresso. Fornisce anche la connessione in uscita per i nodi del piano di lavoro

Un eccellente articolo sulla configurazione, i requisiti e i limiti del networking è reperibile nella relativa **documentazione**.

Integrazione con altri servizi di Azure

In quanto servizio nativo su Azure, Azure Red Hat OpenShift può essere distribuito e integrato con numerosi servizi in uso su Azure. Di seguito si trova una panoramica dettagliata di alcuni dei servizi dell'infrastruttura Azure per i quali sono disponibili integrazioni comuni.

La *Figura 3.2* mostra molti dei punti di integrazione dei servizi Azure comuni per OpenShift su Azure:

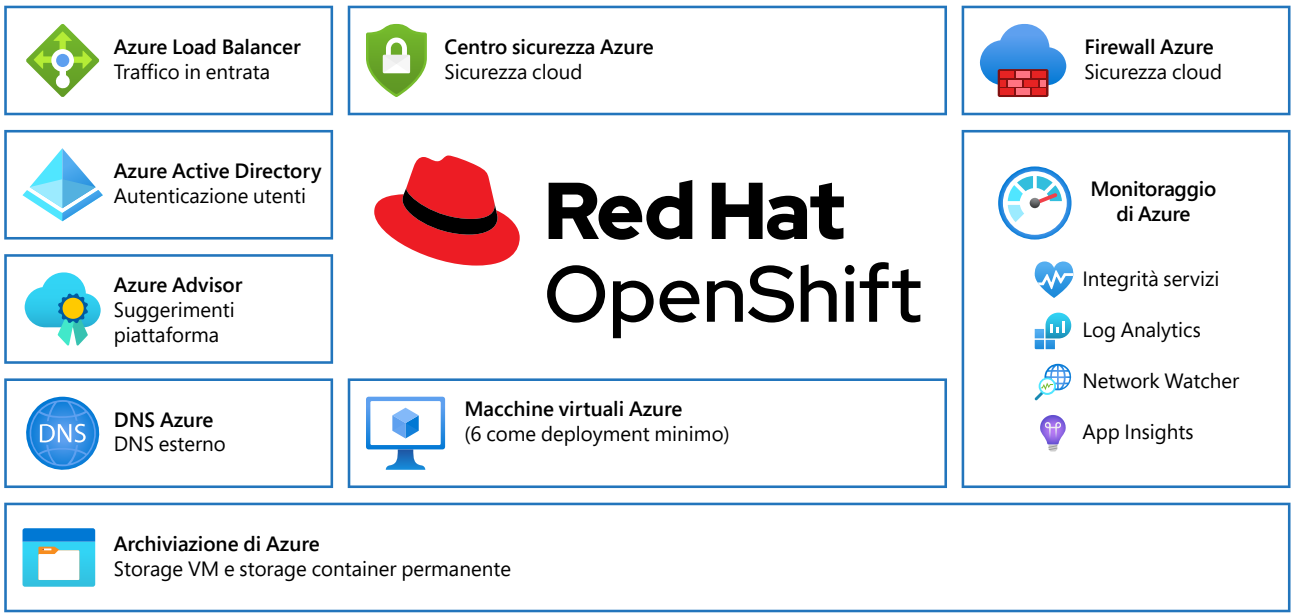


Figura 3.2: Punti di integrazione dei servizi Azure comuni

Inoltre, le applicazioni eseguite su OpenShift possono essere strettamente integrate con i servizi Azure utilizzando l'operatore del servizio Azure. Questo argomento viene approfondito nel *Capitolo 9, Integrazione con altri servizi*.

Gestione

Per comprendere quale sia la parte gestita del servizio Azure Red Hat OpenShift si può pensare che ogni elemento, dal data center agli operatori del cluster, sia un elemento "gestito". Gli operatori del cluster sono servizi che vengono eseguiti sui nodi del piano di controllo e che osservano il monitoraggio, gli aggiornamenti e l'integrità del cluster. Ciò significa che Microsoft e Red Hat monitorano, supportano e gestiscono quei componenti per mantenere il cluster online. La responsabilità di tutto ciò che si trova al di sopra degli operatori del cluster è del cliente.

I clienti di Azure Red Hat OpenShift hanno l'accesso completo con il ruolo **cluster-admin** al cluster, il che significa che devono anche condividere la responsabilità di non suddividere i componenti del cluster. È importante comprendere la Policy sul supporto per capire cosa si può e cosa non si può fare con il cluster.

Una descrizione dettagliata di quanto compete a Microsoft e Red Hat relativamente al servizio cloud è inclusa nella matrice di responsabilità di Azure Red Hat OpenShift.

Autenticazione e autorizzazione

Azure Active Directory è un metodo diffuso per fornire l'autenticazione ai cluster Azure Red Hat OpenShift. Non è tuttavia obbligatorio ed è quindi possibile avvalersi di altri meccanismi di autenticazione alternativi, come l'accesso tramite GitHub o un semplice "file password".

Se si utilizza Azure Active Directory, Azure Red Hat OpenShift e l'API Kubernetes API inoltreranno le richieste di autenticazione. Gli utenti inseriranno le proprie credenziali e verranno autorizzati in base al loro ruolo.

In primo luogo, il livello di autenticazione identifica l'utente associato alle richieste all'API Azure Red Hat OpenShift, quindi utilizza le informazioni sull'utente che invia la richiesta per determinare se concedere o meno l'autorizzazione.

Le istruzioni di configurazione per Azure Active Directory sono descritte nella sezione *Autenticazione - Azure Active Directory*. Dal punto di vista funzionale, questa procedura è simile anche se viene utilizzato un altro provider di autenticazione al posto di Azure Active Directory.

L'autorizzazione è gestita nel motore delle policy di Azure Red Hat OpenShift, che definisce azioni come "create pod" o "list services" e le raggruppa in ruoli in un documento di policy. I ruoli sono vincolati agli utenti o ai gruppi in base all'identificatore dell'utente o del gruppo. Quando un utente o un account di un servizio tenta di eseguire un'azione, il motore delle policy verifica uno o più dei ruoli assegnati all'utente (ad esempio, amministratore del cliente o amministratore del progetto in uso) prima di consentire all'utente di continuare.

Le relazioni tra ruoli del cluster, ruoli locali, associazioni dei ruoli del cluster, associazioni dei ruoli locali, utenti, gruppi e account di servizio sono illustrate nella *Figura 3.3*.

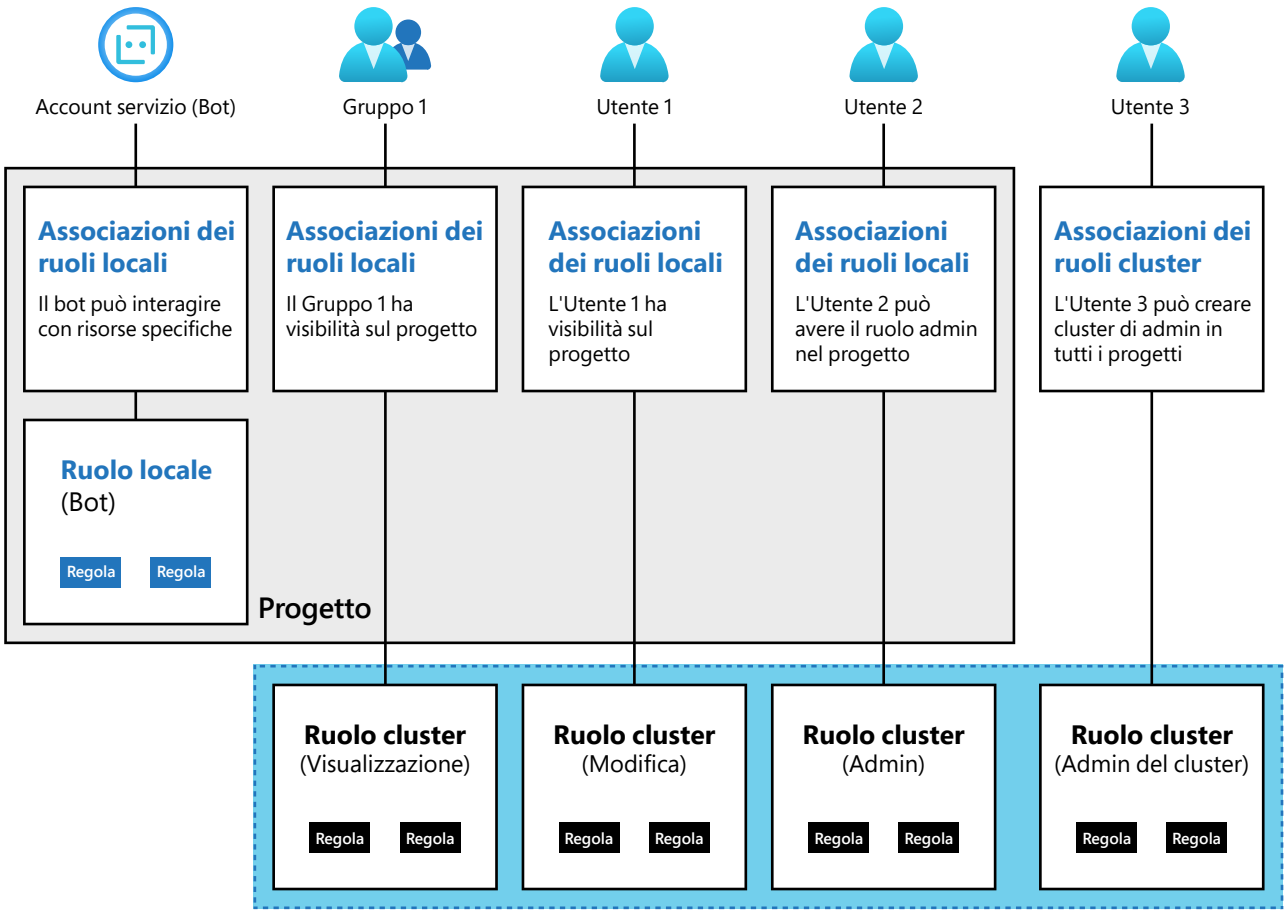


Figura 3.3: Relazioni tra ruoli

Nella documentazione di Red Hat OpenShift è disponibile un elenco completo dei provider di autenticazione.

Supporto

Le modalità di gestione del supporto di Azure Red Hat OpenShift sono esclusive. I team del supporto di Microsoft e Red Hat lavorano insieme e collaborano anche con il team di Site Reliability Engineering (SRE), per agevolare il funzionamento del servizio.

I clienti inviano le richieste di supporto nel portale di Azure; tali richieste vengono smistate e indirizzate dai tecnici di Microsoft e Red Hat, in modo da poter essere soddisfatte in tempi rapidi, indipendentemente dal fatto che siano a livello della piattaforma Azure o di quella OpenShift.

Di seguito uno schema del processo di supporto integrato:

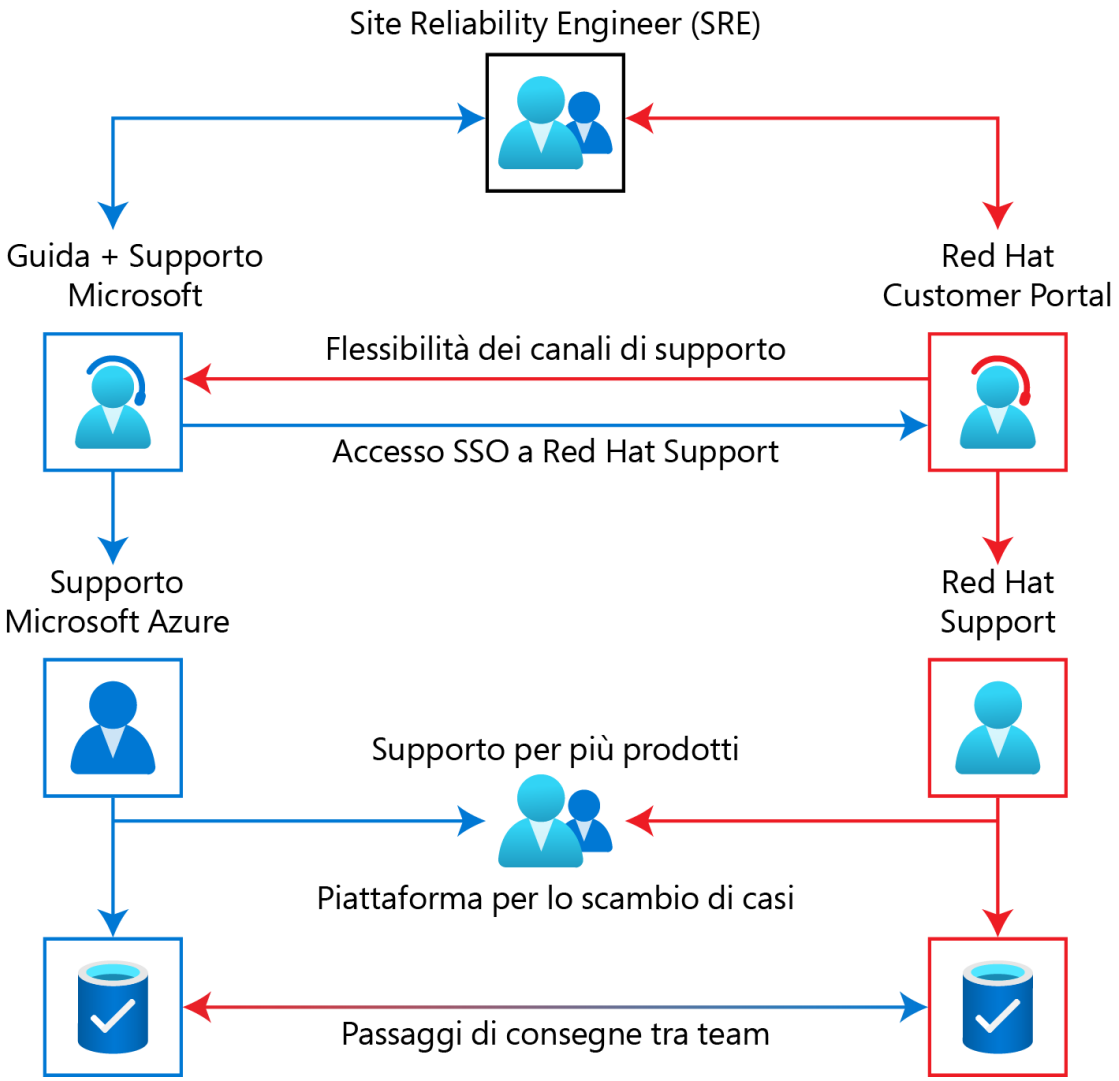


Figura 3.4: Processo di supporto integrato

La Figura 3.4 mostra che il cliente può generare un ticket di supporto nel portale del supporto Microsoft o in quello del supporto Red Hat. Per accedere al portale del supporto Red Hat, il cluster deve essere registrato in OpenShift Cluster Manager.

I tecnici del supporto Microsoft possono collaborare direttamente con Red Hat in qualsiasi fase, con il consenso del cliente, tramite la piattaforma per lo scambio di casi. Lo stesso vale per i tecnici del supporto Red Hat che richiedono la collaborazione con Microsoft. I tecnici del supporto di entrambe le aziende possono rivolgersi al team SRE che può intraprendere azioni di riparazione dei cluster se necessario.

Prezzi e sottoscrizioni

Uno dei principali vantaggi di Azure Red Hat OpenShift rispetto a un'installazione fai da te, è che l'elaborazione, la rete, lo storage e l'infrastruttura di Azure, così come le sottoscrizioni di OpenShift, vengono fatturati tramite la sottoscrizione di Azure e non separatamente. Per un'idea indicativa dei costi della soluzione, è sufficiente aggiungere Azure Red Hat OpenShift al generatore di quote nel [calcolatore dei prezzi di Azure](#).

Di seguito una guida per comprendere la pagina di definizione dei prezzi:

1. Digitare *openshift* nella casella di ricerca e aggiungere il prodotto a un nuovo preventivo.

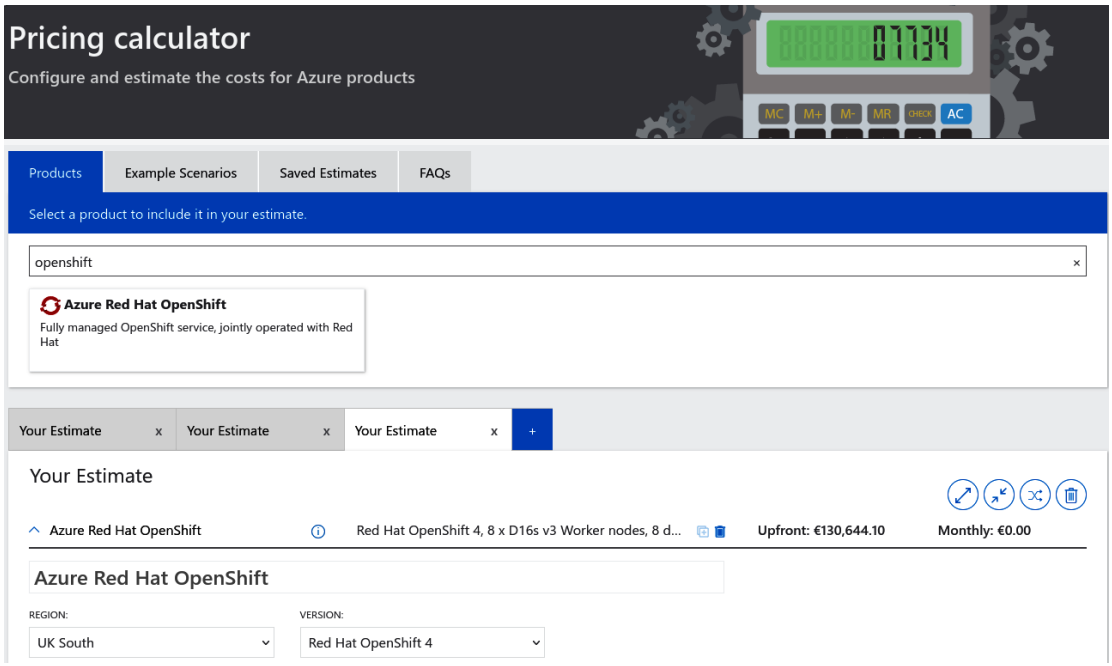


Figura 3.5: Riquadro del calcolatore dei prezzi

2. Nella parte inferiore della pagina è visibile una casella combinata nella quale è possibile indicare la valuta locale per unità di prezzo. Nell'esempio viene utilizzata la sterlina inglese (GBP).
3. Impostare l'area di Azure per il deployment di Azure Red Hat OpenShift. Il prezzo potrà variare leggermente tra le regioni perché tiene conto dei vari costi di elaborazione delle aree di Azure.

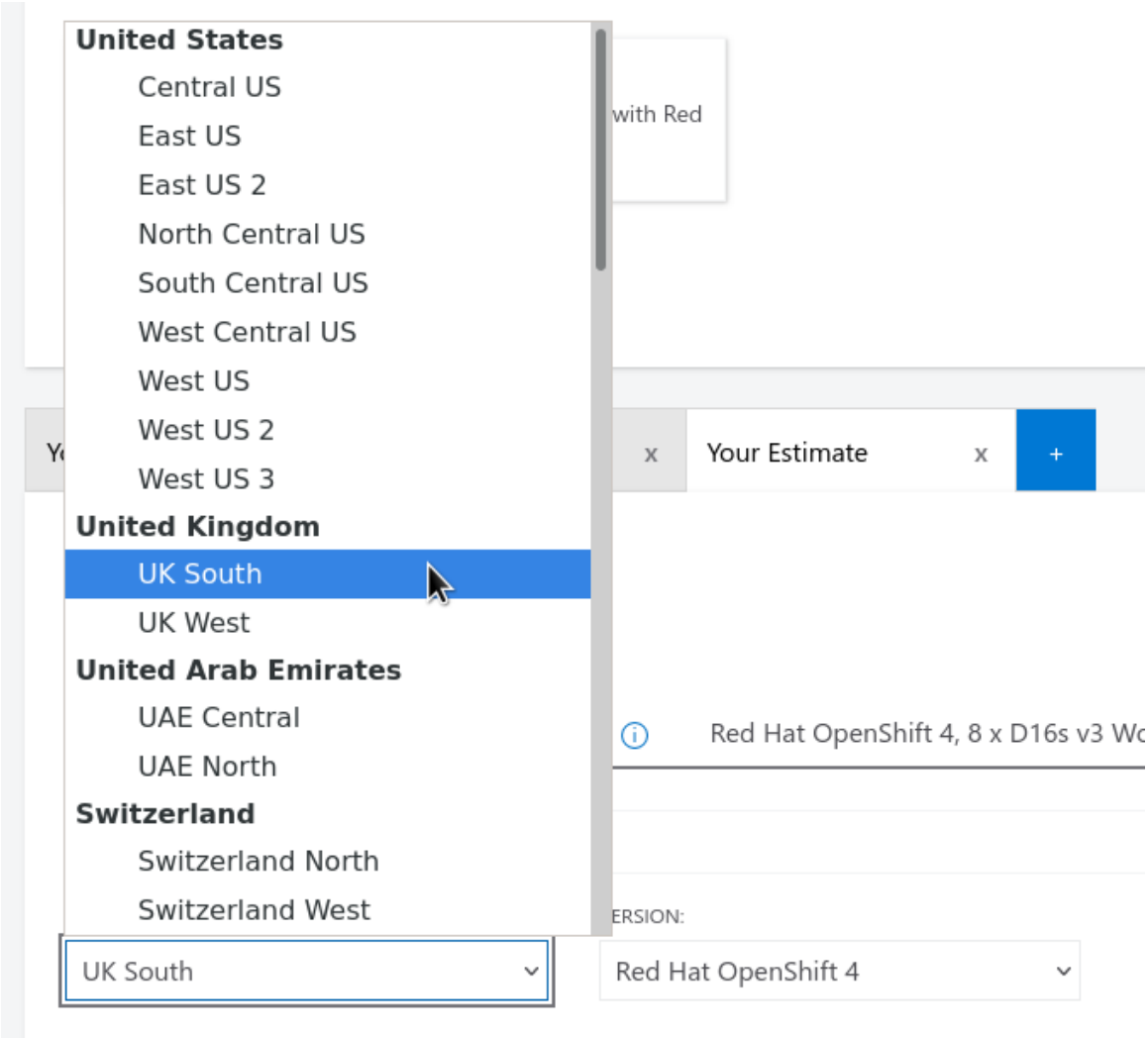


Figura 3.6: Selezione dell'area di Azure

4. **I nodi di lavoro** sono la posizione in cui le applicazioni vengono effettivamente eseguite. In **Savings Options** si trovano due sezioni. **License** indica il costo delle sottoscrizioni di OpenShift, mentre la sezione **Virtual Machine** fa riferimento al costo dei servizi di elaborazione necessari per eseguire il cluster. Il numero massimo di nodi di lavoro supportati in un cluster è 100.

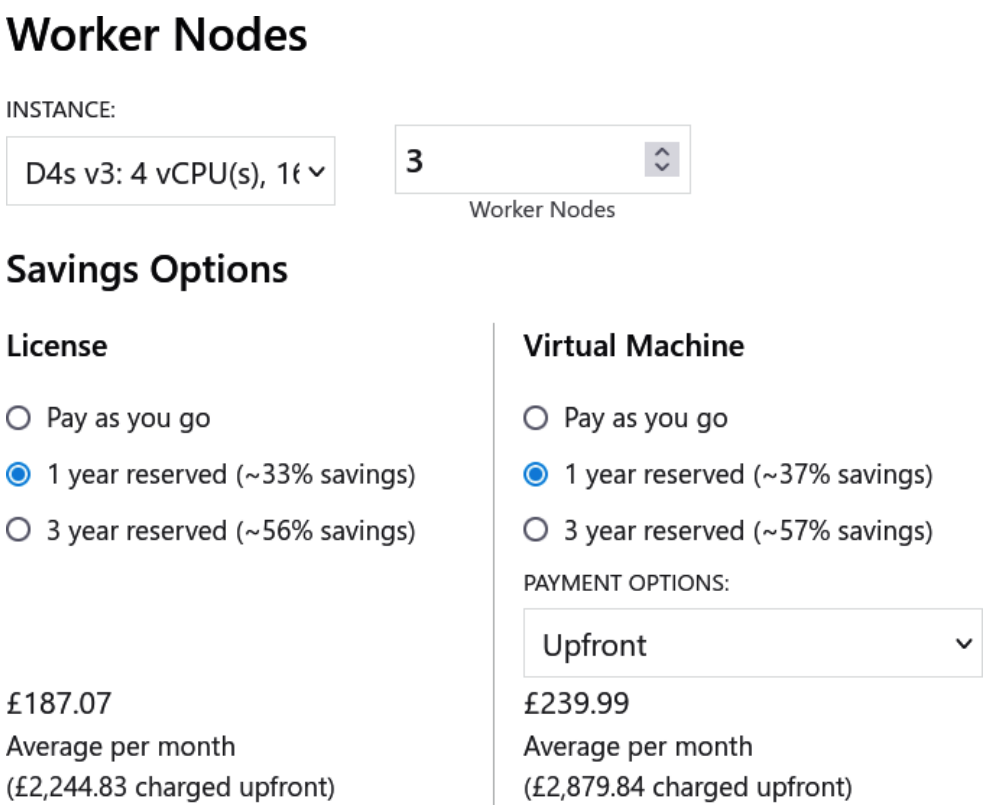


Figura 3.7: Dettagli dei nodi di lavoro

5. **Managed OS Disks** indica la dimensione dei dischi utilizzati da Red Hat CoreOS per le partizioni del sistema operativo. Non si riferisce allo storage utilizzato dai volumi permanenti dell'applicazione il cui provisioning viene eseguito separatamente.

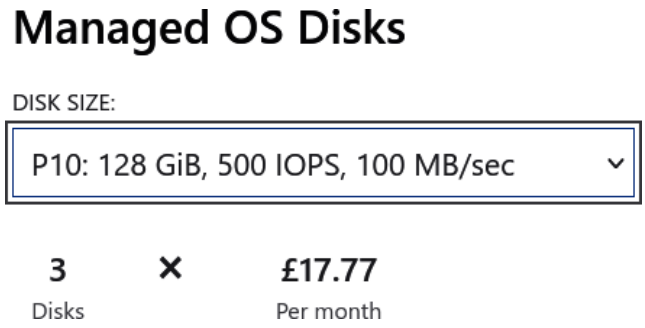


Figura 3.8: Managed OS Disks

6. È presente una sezione simile per i **Master Nodes**, che vengono comunemente definiti nodi di controllo. Ai nodi di controllo è associato un disco molto più grande rispetto a quello dei nodi di lavoro, perché richiedono IOPS e throughput più elevati, forniti in Azure con dimensioni di disco più grandi. Per una maggiore stabilità del cluster, il numero esatto di nodi del piano di controllo deve essere sempre pari a tre.

Il calcolatore è progettato per mostrare il prezzo indicativo; il prezzo effettivo varia infatti in base all'utilizzo.

Istanze di macchine virtuali riservate di Azure

Con **istanza riservata** si definisce un impegno a consumare la risorsa in un periodo di tempo prolungato, in genere uno o tre anni. Sono disponibili opzioni di istanze riservate per le macchine virtuali di Azure Red Hat OpenShift.

Le organizzazioni scelgono in genere le istanze riservate per ottenere un forte sconto sul servizio IaaS, quando sono certe che un cluster sarà in esecuzione per un periodo di tempo più lungo. Vengono spesso eseguite, ad esempio, negli ambienti di produzione. È bene sottolineare che le istanze riservate non modificano il livello di servizio né l'architettura del cluster.

Per approfondire le istanze riservate di Azure

Riepilogo

In questo capitolo sono state illustrate alcune specifiche del servizio cloud gestito Azure Red Hat OpenShift. Dopo aver esaminato l'architettura in modo dettagliato, è stata delineata l'integrazione con altri servizi Azure (che verranno ulteriormente approfonditi nel *Capitolo 9, Integrazione con altri servizi*), e sono state accennate alcune considerazioni relative a gestione, autenticazione, supporto e prezzi.

Il capitolo successivo si incentrerà sulle domande e sulle decisioni che deve affrontare un'organizzazione nella fase di pre-provisioning del deployment di Azure Red Hat OpenShift.

Capitolo 4

Domande sul pre-provisioning dell'architettura enterprise

Visualizzando Azure Red Hat OpenShift nel portale di Azure e leggendo la relativa documentazione, è possibile distribuire un cluster con Red Hat OpenShift con un impegno minimo. Dedicando tuttavia più tempo alla pianificazione dei deployment e ponendosi in anticipo alcune domande, le organizzazioni possono risparmiare tempo evitando l'eliminazione e il reprovisioning dei cluster in futuro.

Questo capitolo si basa sull'esperienza pratica maturata lavorando con numerosi clienti in contesti reali. Prende in esame molte delle domande a cui, in generale, è necessario rispondere prima di iniziare. Il capitolo esamina:

- Numero di cluster necessari, incluso per lo staging, la produzione e simili
- Visibilità delle reti pubbliche e private
- Connettività ibrida, come la connessione alle soluzioni on premise

Andiamo quindi a illustrare come calcolare il numero di cluster necessario.

Quanti cluster sono necessari?

Esistono numerosi modelli di deployment per OpenShift, ma una domanda comune è: "Quanti cluster servono alla mia organizzazione?" Naturalmente questa decisione dipende dall'organizzazione, ma i paragrafi seguenti forniscono alcune linee guida per capire qual è il numero di cluster più adatto.

Fasi del ciclo di vita: sviluppo, test, produzione

La maggior parte delle organizzazioni, indipendentemente dalla dimensione, distribuisce i propri sistemi IT seguendo le fasi del ciclo di vita. A volte questo approccio viene chiamato schema di gestione. Le fasi più comuni del ciclo di vita sono sviluppo, test e produzione. La presenza di più fasi consente di testare le modifiche e i deployment delle applicazioni in un ambiente sicuro, prima che le modifiche arrivino nell'ambiente di produzione. Lo schema di gestione più comune e raccomandato prevede almeno tre cluster di Azure Red Hat OpenShift distinti:

- **Sviluppo:** è il cluster in cui qualsiasi sviluppatore e operatore può di mettere alla prova ogni elemento. Può trattarsi di un unico cluster "sandbox" di grandi dimensioni, ma è più normale e anche più sicuro disporre di cluster più piccoli e con durata inferiore, in cui la frequenza di esecuzione ed eliminazione dei test è molto elevata.
- **Test:** è il cluster in cui vengono testate e convalidate le modifiche programmate al cluster, ad esempio patch o modifiche alla configurazione, prima di essere inviate in produzione. In alcune organizzazioni viene chiamato cluster di "pre produzione", benché il termine possa indicare anche un intero ambiente a sé stante.
- **Produzione:** è il cluster in cui viene eseguita l'applicazione reale.

Oltre a quelli precedenti, alcune organizzazioni predispongono ambienti aggiuntivi, come quelli di test dell'integrazione. Il numero di ambienti effettivamente necessari è noto solo all'organizzazione che li utilizzerà. In situazioni di incertezza, è possibile confrontare applicazioni enterprise simili per individuare modelli di deployment comuni.

Negli scenari in cui Azure Red Hat OpenShift viene utilizzato per applicazioni non critiche, possono essere sufficienti due soli cluster (accorpando sviluppo e test), o perfino un singolo cluster che contenga sviluppo, test e produzione. I vantaggi in questo caso sono il contenimento dei costi del cloud e la riduzione del numero complessivo di cluster da gestire. Quando operano con un solo cluster, gli amministratori possono scegliere di utilizzare spazi dei nomi distinti per lo sviluppo, il test e la produzione. Eseguire un singolo cluster ha tuttavia degli svantaggi:

- Le modifiche che interessano l'intero cluster (le patch software, ad esempio) possono introdurre nella fase di produzione problemi che avrebbero potuto essere individuati ed evitati se le modifiche fossero state eseguite in un ambiente di test.
- Se un'applicazione ha un comportamento imprevisto in un ambiente di test o sviluppo, ad esempio genera numerosi container o utilizza tutto lo spazio su disco disponibile, può provocare problemi nell'ambiente di produzione.

Questo testo non può indicare il numero esatto di cluster che garantiscono il perfetto funzionamento del tuo scenario; come detto prima, è consigliabile considerare le applicazioni enterprise simili già presenti nell'organizzazione per un'idea del numero di fasi del ciclo di vita impiegate.

Continuità operativa, ripristino di emergenza e failover

Oltre agli ambienti di staging standard, molte organizzazioni cercheranno anche di creare almeno un ambiente di produzione di failover, che viene utilizzato qualora si verifichi un guasto critico che blocca l'intero cluster o l'area di Azure. Viene chiamato anche **Disaster Recovery (DR)**. Viene distribuito in genere in un'area di Azure diversa da quella del cluster di produzione.

Il servizio cloud gestito è associato a un **contratto di servizio** del 99,95%, una percentuale accettabile per molte applicazioni business-critical. Altre applicazioni tuttavia richiedono SLA più elevati. È importante capire che con un singolo cluster non è possibile superare questo livello di SLA. Occorre quindi stabilire se la propria applicazione richiede un livello di servizio superiore al 99,95% o se questo è sufficiente.

In caso contrario, è possibile ottenere livelli più alti di disponibilità del servizio (anche del 99,999%) calcolando uno SLA composito prodotto dall'esecuzione simultanea di più cluster. I cluster possono trovarsi tutti nella stessa area (ad esempio, `westeurope`) o su più aree (ad esempio `westeurope` e `northeurope`) per livelli di disponibilità ancora più elevati. La documentazione di Azure indicata di seguito illustra come calcolare SLA compositi quando si eseguono più cluster.

[Documentazione di Azure sugli SLA compositi]

I deployment di Azure Red Hat OpenShift multicluster e su più aree esulano dall'ambito del presente documento, perché quando si realizzano architetture più complesse occorre risolvere numerose problematiche associate, ad esempio, all'indirizzamento del traffico verso il cluster e alla scelta di quali dati condividere e come nel cluster. Queste tematiche richiedono valutazioni più approfondite.

Aree e zone di disponibilità

Azure Red Hat OpenShift è progettato per utilizzare tre zone di disponibilità in ogni regione in cui viene distribuito. In Azure, una [zona di disponibilità] corrisponde a un datacenter autonomo ubicato in un'area e dotato di alimentazione, tecnologie di raffreddamento e connettività alla rete. Analizzando un deployment di Azure Red Hat OpenShift è possibile vedere un singolo nodo di controllo (macchina virtuale) e un nodo applicazione in ogni zona di disponibilità.

La *Figura 4.1* mostra come sono distribuiti i nodi di controllo e i nodi applicazione (di lavoro) in tre zone di disponibilità di una singola area di Azure:

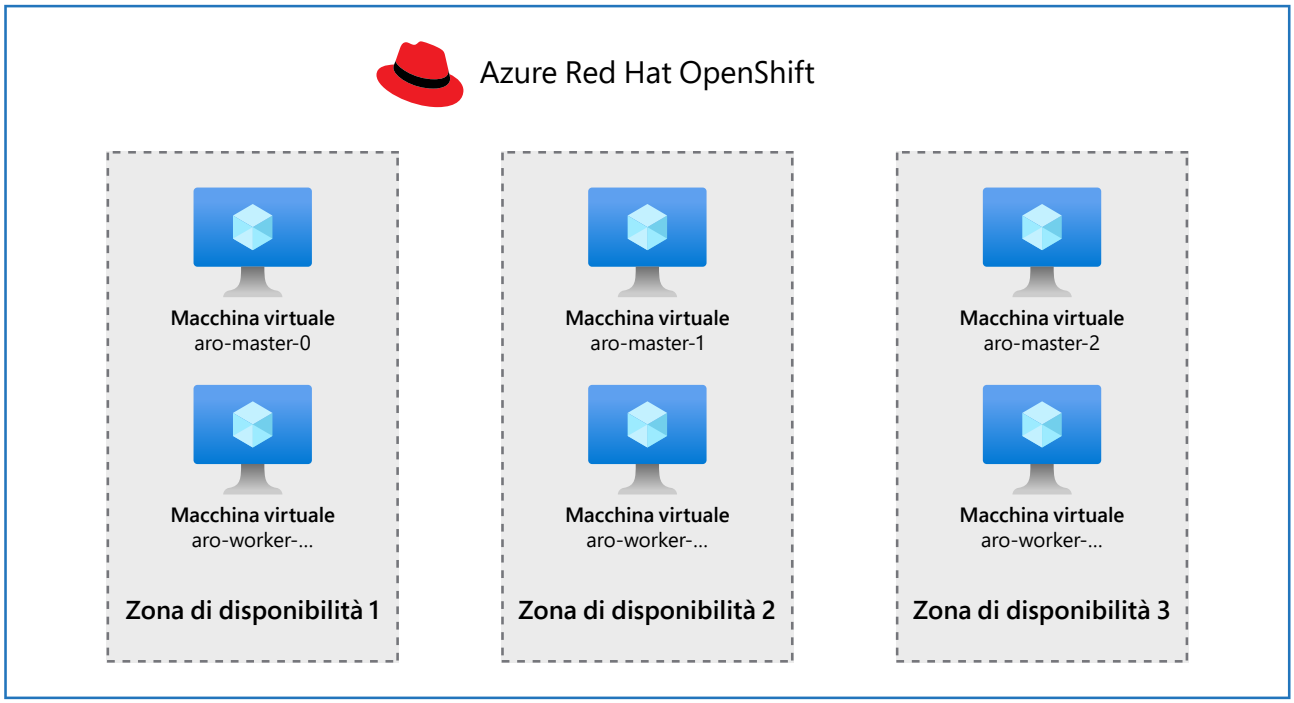


Figura 4.1: Distribuzione dei nodi di controllo e dei nodi applicazione in tre zone di disponibilità di una singola area di Azure

Azure Red Hat OpenShift è progettato per essere offerto come un servizio completamente gestito ad alta disponibilità. Non è pertanto possibile distribuire meno di tre nodi di controllo e di tre nodi di lavoro.

Concetti relativi alla rete

Questa eccellente pagina della documentazione, già citata nell'introduzione ad Azure Red Hat OpenShift, illustra i concetti di base relativi alla rete:

- Concetti relativi alla rete per Azure Red Hat OpenShift

La pagina include una descrizione dettagliata di ogni componente della rete in Azure Red Hat OpenShift: i servizi di bilanciamento del carico, gli indirizzi IP pubblici e privati, i gruppi di sicurezza di rete e altro ancora.

Alcuni punti chiave da tenere a mente:

- Azure Red Hat OpenShift viene distribuito in una rete virtuale nuova o esistente. Supporta una singola rete virtuale, ma va detto che più reti non apporterebbero ulteriori vantaggi, perché OpenShift dispone nel primo livello la propria rete **SDN (Software Defined Network)**, definita OVS.
- La dimensione minima delle subnet del nodo master e dell'applicazione è `/27`.
- Il CIDR del pod predefinito è `10.128.0.0/14`.
- Il CIDR del servizio predefinito è `172.30.0.0/16`.
- Ad ogni nodo è allocata una subnet `/23` (512 indirizzi IP) per i propri pod. Non è possibile modificare questo valore.
- Al momento non sono supportati gli indirizzi IP in uscita.
- È possibile controllare l'istradamento del traffico in uscita, nello specifico per inviarlo tramite Azure Firewall. Al momento della stesura di questo documento, questa funzionalità è in anteprima pubblica e la relativa documentazione è disponibile qui: Limitare il traffico in uscita.

Visibilità della rete privata o pubblica

Si legge spesso che Azure Red Hat OpenShift può essere distribuito con deployment su rete pubblica o privata. Sebbene ciò sia vero, è utile comprendere la differenza tra rendere pubblico o privato il piano di controllo e rendere pubbliche o private le applicazioni nel cluster.

Per quanto riguarda la visibilità del server delle API, questa decisione va presa al momento del provisioning, perché non è possibile modificare la visibilità da pubblica a privata o viceversa dopo aver eseguito il provisioning del cluster.

Più avanti in questo capitolo procederemo alla creazione del cluster Azure Red Hat OpenShift e verrà eseguito il comando `az aro create`, che accetta argomenti relativi alla visibilità del cluster. L'esempio seguente mostra come regolare la visibilità:

Entrambe private

```
az aro create .... --apiserver-visibility Private --ingress-visibility Private
```

Privata per il server API e pubblica per il nodo di lavoro in ingresso

```
az aro create .... --apiserver-visibility Private --ingress-visibility Public
```

La visibilità di `apiserver` e del traffico in ingresso per l'applicazione trovano corrispondenza nel diagramma dell'architettura di Azure nella *Figura 4.2*, che mostra come Azure Red Hat OpenShift utilizza il bilanciamento del carico interno e pubblico di Azure, in cui l'API e il router vengono distribuiti in funzione delle opzioni di visibilità selezionate.

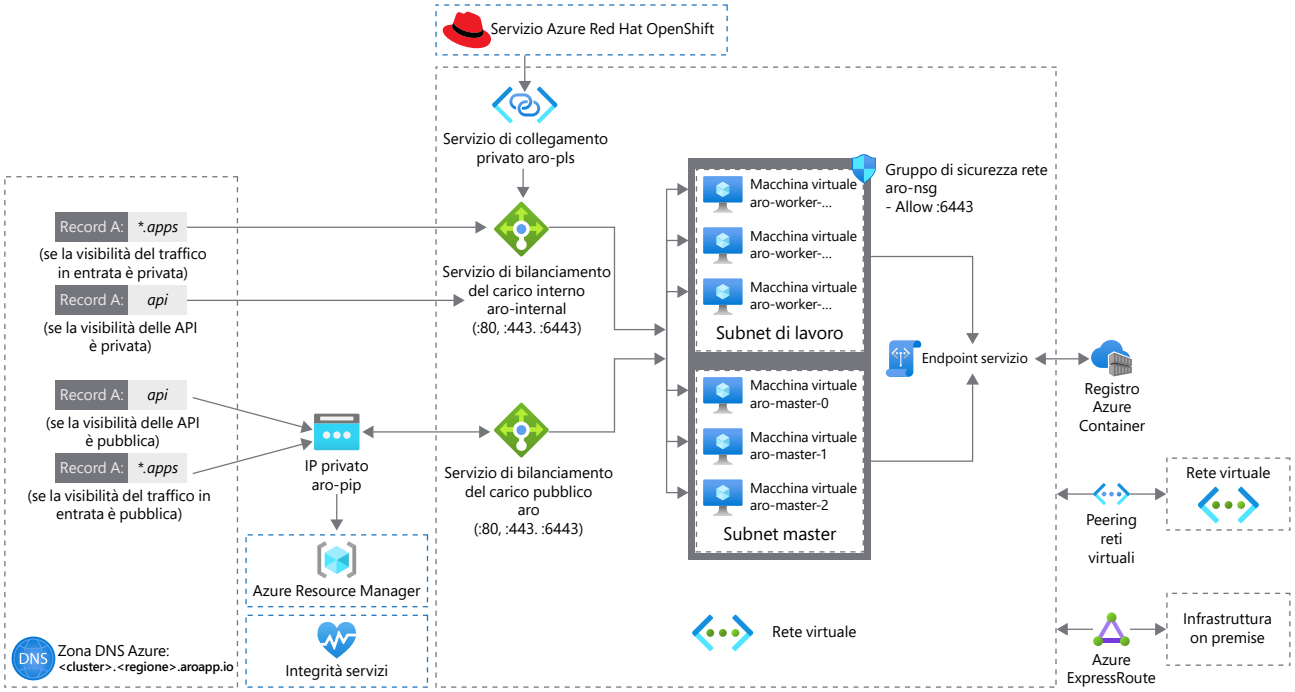


Figura 4.2: Utilizzo del bilanciamento del carico di Azure interno e pubblico in Azure Red Hat OpenShift

Di seguito viene descritta in maggior dettaglio la visibilità del server API e del traffico in ingresso.

Visibilità del server API (piano di controllo)

`--apiserver-visibility` può essere **pubblica** o **privata**:

- **Privata** significa che il piano di controllo di Red Hat OpenShift (anche noto come "nodo master") in cui viene eseguita l'API Kubernetes non è accessibile dalla rete Internet pubblica. Questo piano di controllo serve a sviluppatori e operatori per controllare il cluster e può essere utilizzato per distribuire, eliminare o dimensionare le applicazioni e il cluster stesso. Nella maggior parte dei casi, le aziende con connessioni ExpressRoute ad Azure, o che utilizzano una rete **VPN (Virtual Private Network)** per accedere alle risorse di elaborazione, dovranno scegliere l'opzione privata.
- **Pubblica** significa che il piano di controllo del cluster è accessibile dalla rete Internet pubblica. In questo caso, il cluster è esposto al rischio di attacco, anche se l'accesso deve essere autenticato. Impostare questa opzione su `public` è tuttavia utile per gli ambienti di laboratorio o di test in cui non è possibile controllare la rete di origine degli utenti. Per gli ambienti in cui vengono archiviati dati reali o per qualsiasi tipo di ambiente di produzione, è fortemente raccomandato impostare la visibilità del server API su `privata`.

L'elenco seguente fornisce alcuni esempi in cui è necessaria la connessione del server API, un aspetto di cui tenere conto quando si sceglie tra pubblica e privata:

- Sviluppatori che utilizzano script e strumenti dal proprio ambiente IDE (ad esempio `kubectl rollout`)
- Operatori che ispezionano lo stato del proprio cluster (ad esempio `kubectl get nodes`)
- Server CI/CD che devono ispezionare o modificare lo stato dei deployment (ad esempio Azure DevOps)
- Strumenti di sicurezza del cluster che si connettono al cluster per esaminare lo stato
- Strumenti di monitoraggio che si affidano all'API cluster

Nella maggior parte dei casi, è possibile configurare la connettività della rete affinché avvenga su connessioni **private**, ma l'elenco precedente potrebbe includere esempi aggiuntivi specifici dell'organizzazione; potrebbero anche emergere esigenze non previste che impongono di impostare la visibilità del server API su **pubblica**.

Visibilità del traffico in ingresso (applicazioni)

`--ingress-visibility` può essere pubblica o privata:

- **Privata** significa che i servizi esposti (relativi alle applicazioni in esecuzione sul cluster) non consentono le connessioni dirette dalla rete Internet pubblica. È comunque possibile configurare il routing della rete in modo che gli utenti debbano passare attraverso il firewall di Azure o di un'applicazione web, eseguito facoltativamente su una rete Azure separata, prima di connettersi alle applicazioni aziendali. L'opzione **privata** funziona anche per le applicazioni nel cluster che verranno utilizzate solo all'interno dell'organizzazione, come quelle per l'elaborazione dei pagamenti, l'analisi dei dati o altre applicazioni web interne.
- **Pubblica** significa che le applicazioni nel cluster sono raggiungibili dalla rete Internet pubblica. Sarà comunque necessario configurare un ingresso o una risorsa di routing per poter accedere a tali applicazioni. È ad esempio il caso dell'hosting di un sito web di e-commerce pubblico o di altre applicazioni pubbliche su Hat OpenShift.

In alcuni casi con "ingresso" si indica il "router" OpenShift.

Raccomandazioni per la produzione

È fortemente raccomandato:

- Accedere al cluster tramite una connessione privata e non dalla rete Internet pubblica. Azure ExpressRoute è la scelta migliore quando è necessaria la connessione permanente al cluster da una rete aziendale on premise o da una sede aziendale. La connessione privata può anche essere fornita da una VPN in Azure. Nella sezione **Connettività ibrida** questi aspetti vengono ulteriormente approfonditi.
- Impostare la visibilità del server API (piano di controllo) su `privata` e, facoltativamente, limitare ulteriormente l'accesso con i firewall.
- Impostare la visibilità del traffico in ingresso (applicazioni) su `privata` per le applicazioni eseguite entro l'organizzazione. Nel caso di uno scenario di utilizzo per applicazioni su Azure Red Hat OpenShift che devono essere ospitate sulla rete Internet pubblica, configurare un cluster Azure Red Hat OpenShift distinto: almeno un cluster per le app interne e un altro per le app esterne con la visibilità impostata su pubblica. È tuttavia possibile combinare l'ingresso `pubblico` e `privato` nello stesso cluster.

Naturalmente la maggior parte delle organizzazioni farà riferimento ai propri team di rete e della sicurezza per determinare eventuali controlli aggiuntivi necessari. Ad esempio, in alcune organizzazioni le applicazioni web devono essere protette da un firewall per applicazioni web, uno schema di deployment comune anche quando si distribuiscono applicazioni su Azure Red Hat OpenShift.

Connettività ibrida

La maggior parte delle organizzazioni che distribuisce Azure Red Hat OpenShift esegue applicazioni che devono riconnettersi ai servizi di supporto in esecuzione negli ambienti on premise. Quando la riconnessione va da Azure ai siti on premise, si parla di connettività ibrida o di architettura di cloud ibrido. Questo tipo di connessione può essere fornita in diversi modi. Le due opzioni principali sono:

- **VPN**, adatta per una connettività a bassa complessità verso Azure. In genere le VPN si connettono tramite il Gateway VPN di Azure. Per altre informazioni, leggi l'articolo Che cos'è un Gateway VPN?
- **I circuiti Azure ExpressRoute**, adatti a connessioni dedicate, robuste e permanenti ad Azure. Per approfondire l'argomento, leggi l'articolo Che cos'è Azure ExpressRoute?

Indipendentemente dal metodo di connessione scelto, entrambe le opzioni consentono alle applicazioni on premise di connettersi alle applicazioni eseguite su Azure Red Hat OpenShift e viceversa.

Quando la connessione avviene da un ambiente on premise verso Azure Red Hat OpenShift, si avvale spesso di una rete virtuale hub. La *Figura 4.3* illustra il funzionamento della connessione con Azure ExpressRoute:

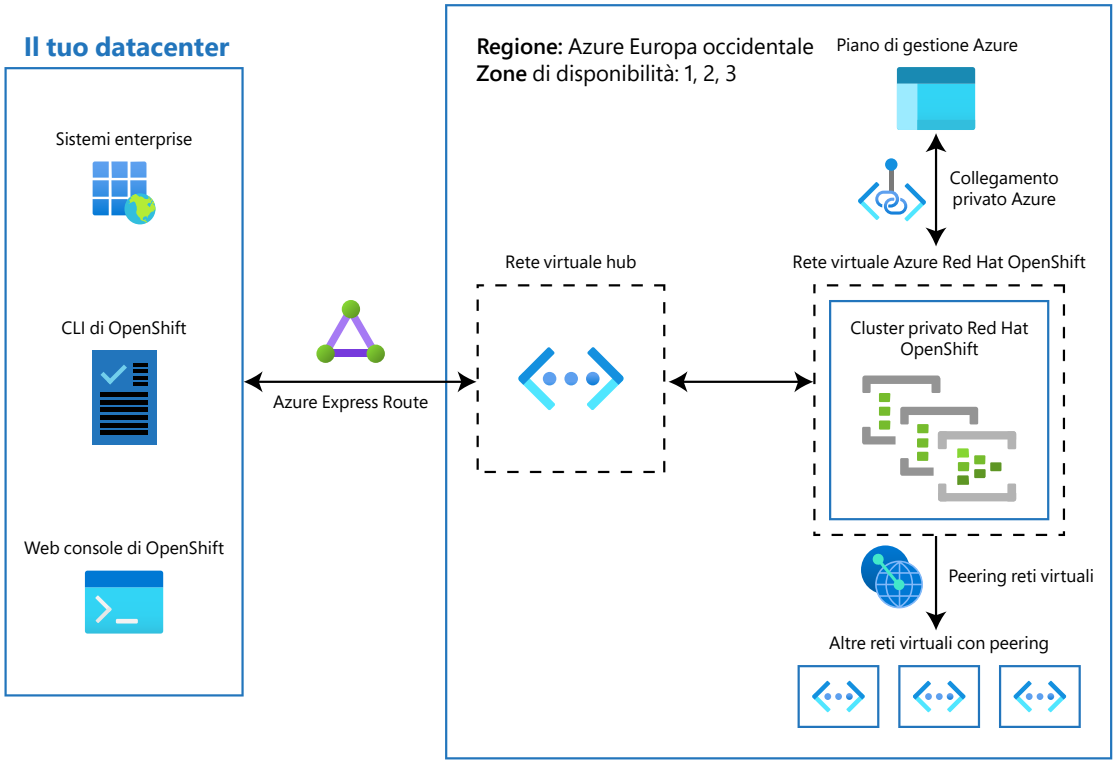


Figura 4.3: Connessione con Azure ExpressRoute

L'immagine precedente mostra l'architettura ad alto livello su cui si basa la connessione tra Azure ExpressRoute e Azure Red Hat OpenShift. Sebbene l'immagine rappresenti il funzionamento di Azure ExpressRoute, dal punto di vista concettuale la connessione tramite una VPN è molto simile.

Considerazioni per applicazioni eseguite in un'architettura ibrida

Quando su Azure Red Hat OpenShift si eseguono applicazioni che si riconnettono agli ambienti on premise, i proprietari dell'applicazione devono tenere conto di alcuni aspetti.

- **Latenza della connessione:** Azure ExpressRoute offre una connessione verso gli ambienti on premise caratterizzata da una latenza relativamente bassa, mentre le reti VPN che viaggiano sulla rete Internet pubblica possono avere una latenza considerevolmente elevata. È raro che questo aspetto causi problemi per applicazioni come i web server, in cui la connessione si limita a trasmettere il traffico HTTP. Se, tuttavia, un'applicazione lato server eseguita su tale web server deve connettersi a un database eseguito on premise, l'incidenza sulle prestazioni può essere notevole.
- **Costo del traffico in ingresso/in uscita:** per alcuni tipi di connessione vengono addebitati i costi del traffico in ingresso/in uscita, sia tramite Azure o dal provider della connessione. Per evitare sorprese, è consigliabile misurare preventivamente tale spesa in un ambiente di test, per poi prevedere i possibili costi nei momenti di picco del carico.
- **Scenari con errori o guasti:** mentre le connessioni con Azure ExpressRoute sono progettate per essere robuste e permanenti, le connessioni VPN sono soggette a interruzioni o a connessioni e disconnessioni frequenti. Essendo eseguite sulla rete Internet pubblica, la latenza e la qualità del servizio di connessione VPN possono subire forti variazioni durante la giornata. È importante testare le prestazioni dell'applicazione sia con collegamenti ibridi in cattive condizioni, sia con prestazioni ottimali della connessione. Occorre infine verificare se, nel caso in cui la connessione subisca un'interruzione di diverse ore, l'applicazione eseguita su Azure Red Hat OpenShift registrerà solo una diminuzione delle prestazioni o se la sua disponibilità è totalmente dipendente dalla connessione ibrida.

Ogni organizzazione dovrà tenere conto di altri aspetti specifici, ma i punti sopra delineati costituiscono un buon punto di partenza per capire se l'architettura ibrida è funzionale alle tue esigenze.

Riepilogo

In questo capitolo sono state esaminate alcune delle frequenti domande sul pre-provisioning che occorre approfondire prima di distribuire Azure Red Hat OpenShift. Il capitolo successivo presenta alcune risorse che saranno utili nella fase di distribuzione effettiva del cluster.

Capitolo 5

Provisioning di un cluster Azure Red Hat OpenShift

Ricapitoliamo gli argomenti illustrati finora.

- Nel *Capitolo 2*, *Introduzione a Red Hat OpenShift*, abbiamo brevemente presentato Red Hat OpenShift e illustrato i vantaggi che offre rispetto a Kubernetes su bare metal.
- Nel *Capitolo 3*, *Azure Red Hat OpenShift*, abbiamo descritto le caratteristiche specifiche del servizio cloud gestito Azure Red Hat OpenShift e analizzato alcuni aspetti chiave tra cui architettura, gestione, autenticazione, supporto e definizione dei prezzi.
- Nel *Capitolo 4*, *Domande sul pre-provisioning dell'architettura enterprise*, sono state esaminate le domande comuni a cui un'organizzazione deve rispondere prima di distribuire Azure Red Hat OpenShift.

Tutto è pronto per il provisioning e la distribuzione dei cluster. Le istruzioni ufficiali per il deployment sono reperibili nel sito della documentazione. Il provisioning è una parte del deployment; eseguire il provisioning di un deployment significa accertarsi di disporre dell'infrastruttura IT necessaria per supportare un deployment.

[Tutorial – Creare un cluster di Azure Red Hat OpenShift 4]

Questo capitolo non descrive i singoli comandi che è necessario digitare per creare un cluster, perché cambiano nel tempo e sono già illustrati in dettaglio nella relativa documentazione.

Vengono invece fornite delle linee guida relative al processo nel suo complesso e a quanto è necessario considerare quando si esegue il deployment di un cluster.

Tempistiche dei deployment manuali

Alcune organizzazioni devono sapere quanto tempo è necessario per il provisioning di un cluster. Di seguito cercheremo di capirlo in dettaglio.

In linea di massima, i prerequisiti per il deployment sono i seguenti:

- Riga di comando `az` installata nella workstation dell'amministratore di sistema.
- Provider di risorse registrati per essere utilizzati con la tua sottoscrizione di Azure.
- Segreto pull di Red Hat (facoltativo).
- Dominio per il cluster (facoltativo).
- Rete virtuale con due subnet vuote, una per i nodi del piano di controllo e una per i nodi applicazione.

Per quanto riguarda il tempo necessario per configurare questi prerequisiti, probabilmente un amministratore di Azure e Red Hat OpenShift qualificato completa queste attività in 30 minuti circa la prima volta e in 10 minuti se le procedure vengono eseguite ripetutamente nell'ambito di un processo manuale.

Una volta predisposti i prerequisiti, il processo di provisioning automatizzato richiede in genere tra i 25 e i 40 minuti, a seconda dell'attività nell'area di Azure.

Automazione del deployment

Acquisita la consapevolezza che il processo di provisioning di Azure Red Hat OpenShift è automatizzato, un'organizzazione proverà a utilizzare strumenti per automatizzare anche i prerequisiti, ovvero la creazione di reti, subnet, entità di servizio, ecc.

Esistono molti strumenti in grado di automatizzare questi passaggi. Di seguito un elenco degli strumenti raccomandati:

- Strumento a riga di comando az: quando automatizzato, questo strumento viene installato di norma in un container, o simili, come parte di un processo CI/CD. Gli strumenti più utilizzati in questo ambito sono Jenkins, Azure DevOps o verosimilmente Ansible. Lo strumento az deve essere installato una sola volta, ma potrebbero essere necessari cluster aggiuntivi per impostare l'ID della sottoscrizione di Azure in modo da rispecchiare parti diverse dell'organizzazione.
- I provider di risorse registrati per essere utilizzati con la sottoscrizione di Azure: come visto in precedenza, è un passaggio della configurazione dello strumento a riga di comando az.
- Un segreto pull di Red Hat (facoltativo): Red Hat rende disponibile un'API REST supportata e documentata per ottenere un segreto pull, con istruzioni reperibili in questo **articolo**.
- Un dominio per il cluster (facoltativo): dipende da come vengono creati i record DNS, ma se si utilizza Azure DNS, sono strumenti adatti Terraform, Ansible o gli altri strumenti di automazione di Azure più diffusi.
- Una rete virtuale con due subnet vuote, una per i nodi (master) del piano di controllo e una per i nodi applicazione (di lavoro): questa fase è automatizzata dagli strumenti di automazione di Azure quali Terraform, Ansible o simili, che possono creare la rete e le subnet per tuo conto.

Una volta automatizzati questi prerequisiti, il tempo necessario (da 30 a 10 minuti) nel caso del processo manuale potrà essere ridotto ad appena un minuto o due. Non è invece possibile accelerare il deployment del cluster (che richiede sempre tra i 25 e i 40 minuti), ma questo potrebbe essere un processo di distribuzione end-to-end molto veloce rispetto a quello on premise.

Oltre ad accelerare il processo, l'automazione del deployment ha altri vantaggi; i clienti che la utilizzano ad esempio, ottengono un processo consolidato e ripetibile, facile da registrare e verificare. Per i clienti è inoltre molto semplice realizzare cataloghi self-service nei propri portali, per facilitare ai team il provisioning e il deprovisioning dei cluster di Azure Red Hat OpenShift, senza l'intervento dei team della piattaforma cloud.

Accesso al cluster

Questa sezione fornisce indicazioni utili per accedere al cluster Azure Red Hat OpenShift dopo averne eseguito il provisioning.

Tramite l'interfaccia utente web

Da una shell Bash se è stata installata la CLI, o dalla sessione Azure Cloud Shell (Bash) nel portale di Azure, recuperare l'URL di accesso al cluster eseguendo il comando:

```
az aro show -n $CLUSTER_NAME -g $RG_NAME --query "consoleProfile.url" -o tsv
```

Il risultato ottenuto dovrebbe essere simile a `openshift.xxxxxxxxxxxxxxxxxxx.eastus.aroapp.io`. L'URL di accesso al cluster sarà `https://` seguito dal valore `consoleProfile.url`; ad esempio, `https://openshift.xxxxxxxxxxxxxxxxxxx.eastus.aroapp.io`.

Aprire questo URL nel browser. Il sistema chiede di accedere come utente `kubeadmin`. Utilizzare il nome utente e la password fornite dal programma di installazione.

Dopo aver effettuato l'accesso, verrà visualizzata la web console di Azure Red Hat OpenShift.

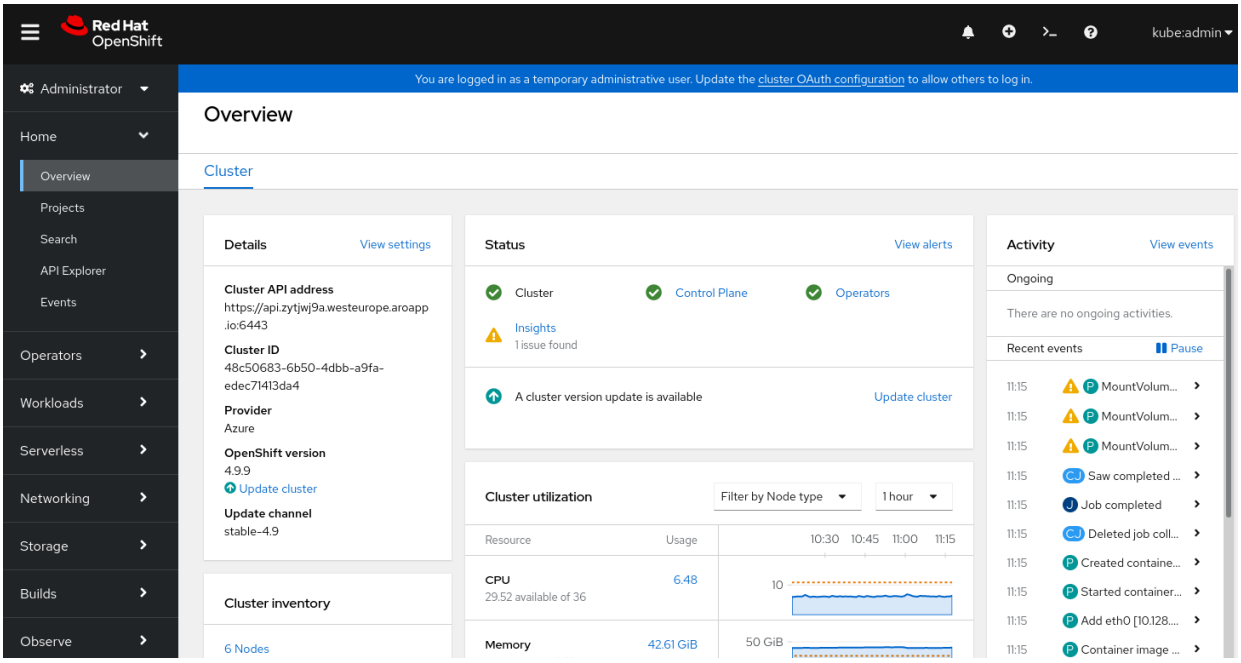


Figura 5.1: Web console di Azure Red Hat OpenShift

È bene acquisire familiarità con la web console. È inoltre necessario eseguire una versione recente di OpenShift e disporre di componenti che siano integri e funzionanti, o che lo diventino immediatamente dopo l'installazione.

Tramite l'interfaccia a riga di comando (CLI) di OpenShift (oc)

È necessario scaricare la versione più recente della CLI di OpenShift (oc). Per farlo, è sufficiente accedere e visualizzare la pagina all'indirizzo https://console.redhat.com/openshift/downloads.

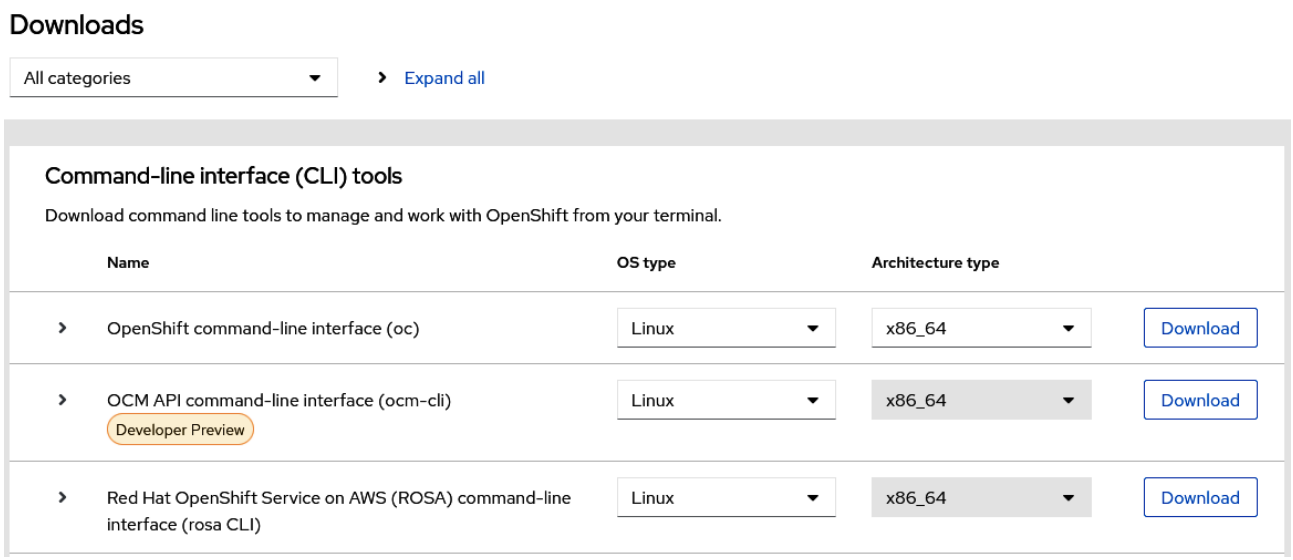


Figura 5.2: Download della CLI di OpenShift

Estrarre l'archivio (`.tar.gz` o `.zip`) nel sistema e posizionare il comando `oc` in un punto qualsiasi del percorso. In Linux, il comando `oc` viene in genere collocato in `/usr/local/sbin/`.

Eseguire la CLI di OpenShift e accedere al cluster

Per autenticarsi al cluster dalla riga di comando è necessario recuperare il comando e il token di accesso dalla web console. Accedere alla web console con il browser, fare clic sul nome utente in alto a destra, e quindi su **Copy login command**.

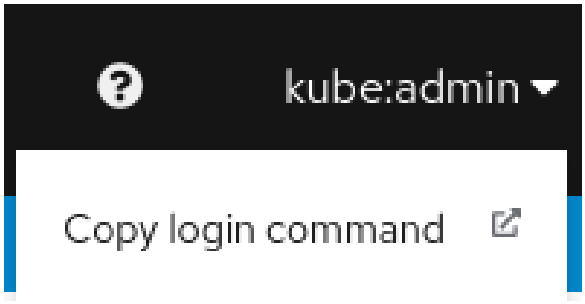


Figura 5.3: Schermata Copy login command

Viene visualizzata una nuova pagina, simile alla seguente:

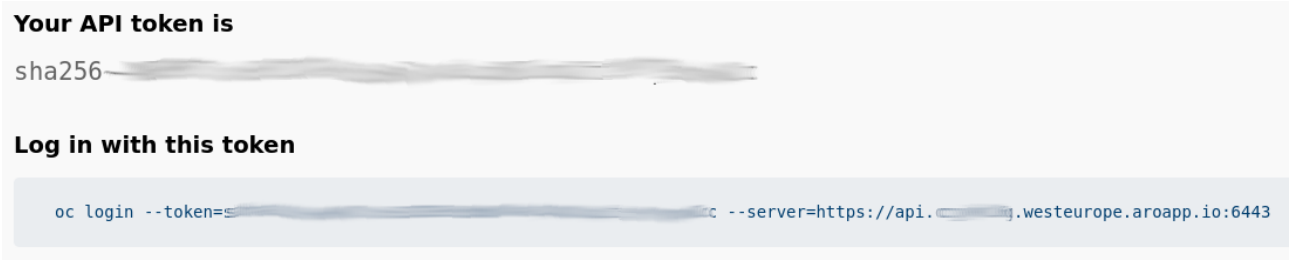


Figura 5.4: Accesso con il token API

Copiare il comando e incollarlo in un terminale per accedere al cluster Azure Red Hat OpenShift.

Se, ad esempio, si sta utilizzando la sessione Azure Cloud Shell (Bash) nel portale di Azure, incollare il comando di accesso; a questo punto il comando `oc status` sarà simile al seguente:

```
user@Azure: oc status
In project default on server https://api.cyki1k6g.westeurope.aroapp.io:6443

svc/openshift - kubernetes.default.svc.cluster.local
svc/kubernetes - 172.30.0.1:443 -> 6443

View details with 'oc describe <resource>/<name>' or list everything with 'oc get all'.
```

Potrebbe essere utile approfondire ulteriormente la riga di comando `oc` e familiarizzare con il nuovo ambiente Azure Red Hat OpenShift. Se hai già esperienza con OpenShift 4, questo servizio cloud di Azure Red Hat OpenShift ti sembrerà familiare in quanto funziona in modo simile ad altri ambienti OpenShift 4 che potresti aver utilizzato in passato.

Riepilogo

Questo capitolo è molto breve perché le istruzioni ufficiali per il provisioning sono esaustive e devono essere considerate la guida definitiva per l'esecuzione di Azure Red Hat OpenShift nella sottoscrizione di Azure. Avrai notato che le istruzioni per il provisioning sono molto più semplici di quelle necessarie per il provisioning di un ambiente OpenShift autogestito. Infatti, il servizio Azure Red Hat OpenShift è altamente ingegnerizzato e ben integrato con Azure; inoltre viene distribuito con un'architettura prescrittiva e universale, testata e supportata in modo ottimale. Nel complesso, offre innumerevoli vantaggi alle organizzazioni, che potranno dedicare meno tempo al provisioning di Azure Red Hat OpenShift e concentrarsi sul deployment delle applicazioni.

Capitolo 6

Post-provisioning - Operazioni ordinarie

Completato il deployment di Azure Red Hat OpenShift, occorre completare alcune attività di post-provisioning necessarie prima che il cluster sia pronto per la fase di produzione. Questo capitolo ne illustra alcune, tra le quali:

- Autenticazione – Azure Active Directory: come sincronizzare i gruppi di utenti con un operatore della community
- Operatori: OperatorHub
- Funzionamento dei registri: inoltro dei registri
- Funzionamento del monitoraggio: monitoraggio dei container OpenShift
- Aggiornamenti e patch: le versioni supportate del ciclo di vita
- Scalabilità del cluster: dimensionamento manuale e automatico del cluster
- Scalabilità delle applicazioni: breve nota su come dimensionare le applicazioni
- Configurazione di un segreto pull: come eseguire la registrazione in OpenShift Cluster Manager
- Intervalli limite: dove applicare un intervallo limite
- Storage permanente: classi di storage disponibili e supportate
- Sicurezza e conformità: una nota sui controlli di sicurezza

Queste sezioni, ognuna delle quali affronta una diversa attività di post-provisioning, ti consentiranno di comprendere meglio come gestire un cluster appena installato e come prepararlo per le applicazioni di produzione.

Autenticazione – Azure Active Directory

Azure Red Hat OpenShift supporta tutti i provider di autenticazione elencati nella documentazione di OpenShift; qui è disponibile l'elenco completo dei provider di autenticazione. La maggior parte dei clienti utilizzerà Azure Active Directory, che è già disponibile in Azure, per fornire autenticazione e accesso SSO ad Azure Red Hat OpenShift.

La configurazione dell'integrazione con Azure Active Directory è un'attività di ordinaria amministrazione, perché in alcuni casi le organizzazioni preferiscono utilizzare un provider di configurazione alternativo. La procedura di configurazione e installazione di Azure Active Directory richiede circa 15 minuti la prima volta, ma una volta acquisita familiarità con il processo potrà essere eseguita in meno di cinque minuti. Molte organizzazioni scelgono di automatizzare parte di questo provisioning avvalendosi di strumenti quali i modelli ARM o i playbook Ansible.

- Configurare Azure Active Directory per Azure Red Hat OpenShift (portale)
- Configurare Azure Active Directory per Azure Red Hat OpenShift (interfaccia a riga di comando)

Utilizzo dei gruppi di utenti di Active Directory

La configurazione dell'autenticazione di Azure Active Directory consente agli utenti di accedere solo con le credenziali esistenti, senza trasferire i gruppi di utenti già presenti per un utente in Red Hat OpenShift. È frequente che un'organizzazione importi i gruppi di utenti da Active Directory in modo che possano essere utilizzati per configurare le autorizzazioni utente in OpenShift. A questo scopo è disponibile un operatore distinto, supportato dalla community, denominato operatore Group Sync.

- Operatore Group Sync su GitHub

Supportato dalla community significa che al momento della stesura di questo documento l'operatore non è ufficialmente supportato da Red Hat o da Microsoft. Si consiglia di seguire le dettagliate istruzioni di configurazione e installazione che accompagnano il file README del progetto per questo operatore.

Operatori

In OpenShift 4, gli operatori sono uno dei componenti fondamentali della piattaforma e assicurano un importante valore aggiunto per i clienti di Red Hat OpenShift. Sono creati da codice ed eseguono un servizio in un container nel cluster. Alcuni operatori sono responsabili della gestione di elementi quali la rete del cluster, la configurazione dei sistemi e gli aggiornamenti del cluster. Sono spesso chiamati operatori del cluster; nella sezione **Administration** della console di OpenShift console ne è visibile un elenco.

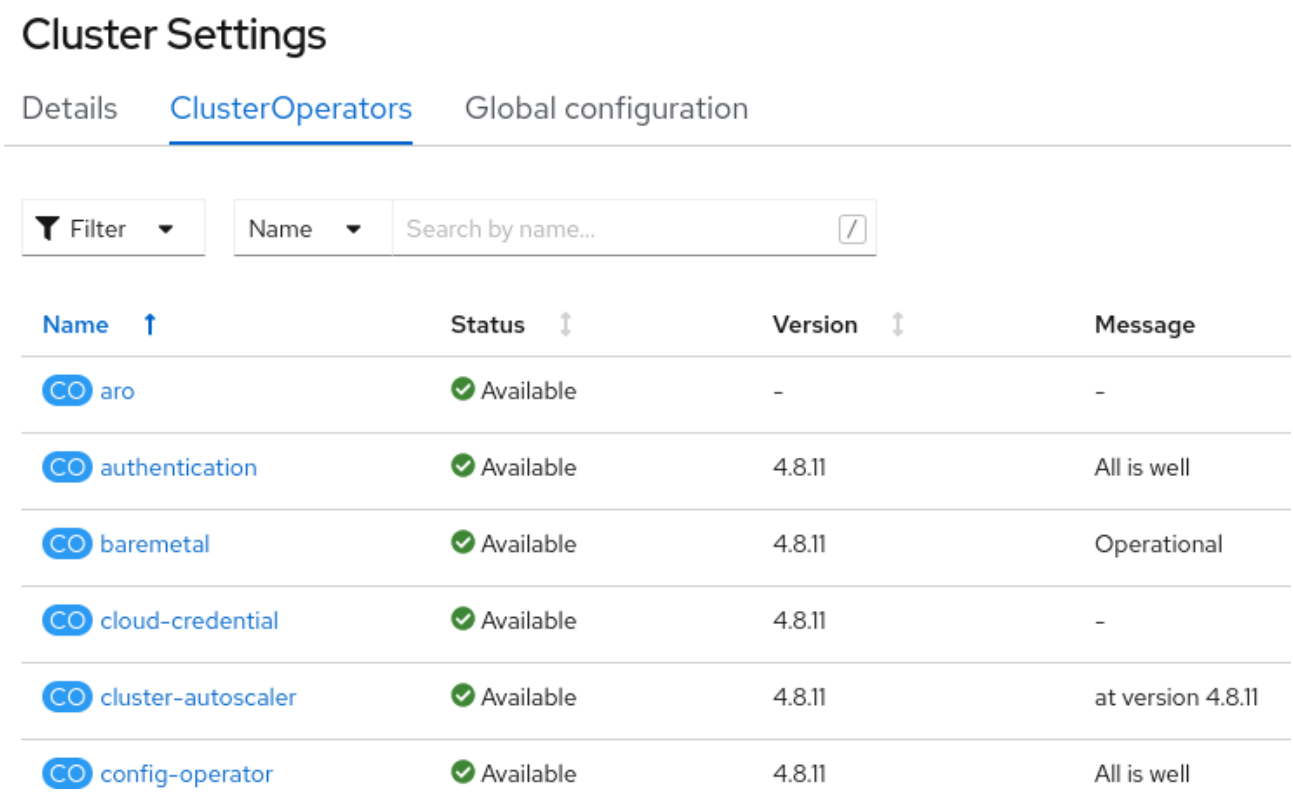


Figura 6.1: Impostazioni del cluster

Nella schermata precedente è visibile l'operatore specifico per Azure Red Hat OpenShift, che gestisce parti del servizio nell'intento di garantire che il cluster mantenga lo stato di configurazione supportato.

Gli operatori possono gestire vari elementi, tra cui l'integrità di un servizio, gli aggiornamenti, le patch, la scalabilità e diverse altre funzioni.

OperatorHub

Oltre agli operatori standard del cluster eseguiti in background in qualsiasi cluster, gli amministratori possono accedere ad altri operatori tramite OperatorHub. OperatorHub facilita l'individuazione degli operatori più diffusi, sia della community sia di livello enterprise, che un amministratore può voler rendere disponibili durante l'installazione di Red Hat OpenShift. OperatorHub è reperibile nella barra laterale di qualsiasi cluster OpenShift.

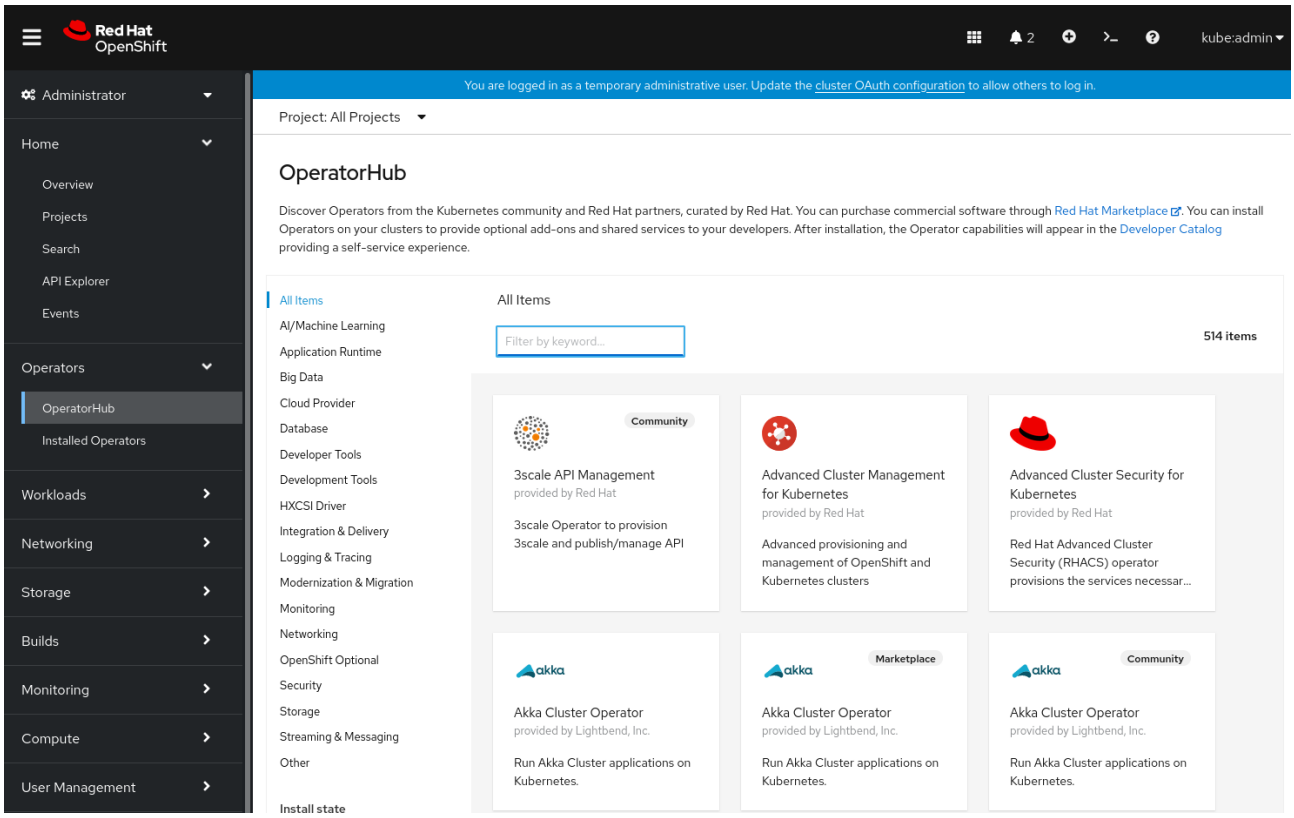


Figura 6.2: OperatorHub

Tramite gli operatori, gli amministratori possono distribuire e gestire in modo rapido e semplice i software più diffusi, senza doversi occupare della configurazione e del codice Kubernetes. Molti prodotti Red Hat sono raggruppati in pacchetti come operatori, ad esempio Advanced Cluster Management, Red Hat OpenShiftService Mesh e Red Hat OpenShift Pipelines. Esiste inoltre un'ampia libreria di operatori per software non Red Hat, come il database MariaDB, Couchbase o i driver per lo storage a blocchi di IBM.

Per ulteriori informazioni sugli operatori sono utili i seguenti link:

- Operatorhub.io
- Operatori in OpenShift

Funzionamento dei registri

Azure Red Hat OpenShift utilizza la stessa architettura di logging e monitoraggio di Red Hat OpenShift. Entrambe le offerte utilizzano gli stessi operatori per il logging.

I registri sono suddivisi in tre categorie:

- **Applicazione:** registri dei container generati dalle applicazioni utente in esecuzione nel cluster, ad eccezione delle applicazioni di container dell'infrastruttura.
- **Infrastruttura:** registri generati dai componenti dell'infrastruttura in esecuzione nel cluster e dai nodi di OpenShift Container Platform, ad esempio i registri di journaling. I componenti dell'infrastruttura sono pod eseguiti nei progetti `openshift*`, `kube*` o predefiniti.
- **Audit:** registri generati da auditd, il sistema di audit dei nodi, archiviati nel file `/var/log/audit/audit.log`, e registri di audit del server API Kubernetes API e del server API OpenShift.

Per una descrizione più approfondita dei registri di OpenShift, leggi l'articolo specifico sui registri di Red Hat OpenShift nella documentazione del prodotto.

Utilizzo dell'inoltro dei registri del cluster ad Azure Monitor

Un'integrazione comune è quella che prevede l'invio dei registri di Azure Red Hat OpenShift al servizio di analisi del container di Azure Monitor, che viene a volte denominato Azure Log Analytics. Il servizio consente di conservare i registri in Azure in modo permanente e a costo contenuto.

L'architettura di logging di OpenShift esegue Fluent Bit su ogni nodo; il primo approccio di un operatore dovrebbe essere quello di adattare la configurazione del servizio Fluent Bit affinché invii i registri direttamente. L'adattamento della configurazione di logging in Azure Red Hat OpenShift è tuttavia un'attività che esula dalla politica di supporto. Per rimanere nell'ambito del supporto, OpenShift prevede un meccanismo integrato per l'inoltro dei log chiamato `ClusterLogForwarder`.

Utilizzando `ClusterLogForwarder` è possibile inoltrare i log ad Elasticsearch, Flutentd e syslog; va sottolineato che Azure Monitor non supporta direttamente nessuno di questi protocolli. Una soluzione alternativa è l'aggiunta di un'istanza intermedia di Fluent Bit, che riceve i registri da OpenShift e li inoltra ad Azure Monitor. Questo approccio è descritto in dettaglio nella documentazione Microsoft e anche nella documentazione della community.

Funzionamento del monitoraggio

In quanto servizio gestito, non è necessario definire una complessa attività di monitoraggio personalizzato per Azure Red Hat OpenShift perché è già prevista nel servizio a pagamento.

Può però essere utile monitorare i container di OpenShift tramite lo strumento di analisi Azure Container. Si tratta di una funzionalità al momento in anteprima pubblica, descritta in questa documentazione.

Aggiornamenti e patch

Durante il provisioning di un cluster Azure Red Hat OpenShift, non è selezionato alcun canale di aggiornamento; ciò significa che, per impostazione predefinita, il cluster non riceve gli aggiornamenti.

Per visualizzare il canale di aggiornamento, passare alla sezione **Administration** → **Cluster Settings** tramite la barra di navigazione laterale. Nell'immagine sono visibili le impostazioni del cluster subito dopo il provisioning di un cluster Azure Red Hat OpenShift:

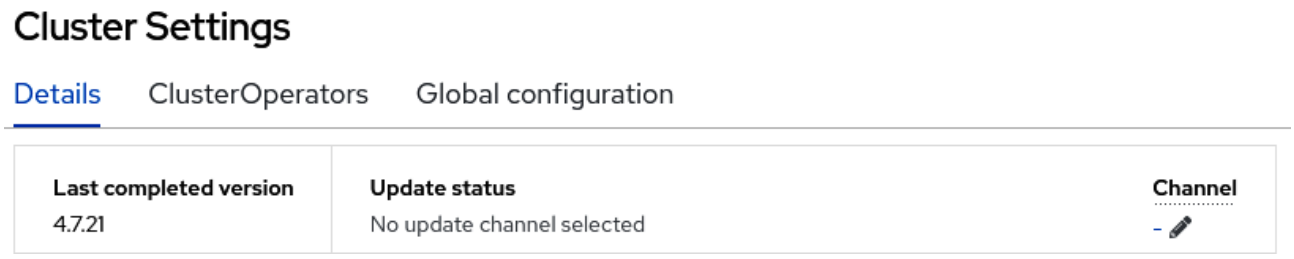


Figura 6.3: Schermata Cluster Settings dopo il provisioning di un cluster

Per selezionare un canale di aggiornamento, selezionare il collegamento "-" e visualizzare le opzioni disponibili:

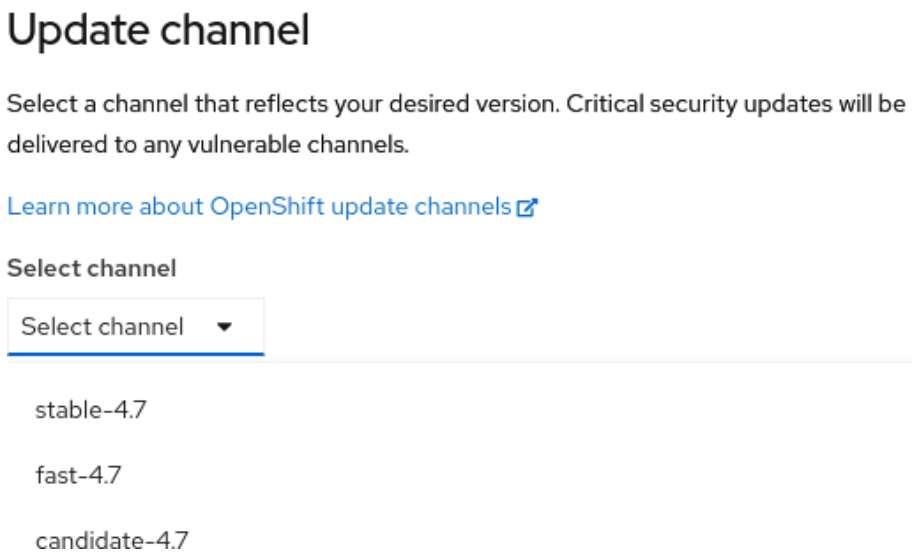


Figura 6.4: Selezione di un canale di aggiornamento

Per comprendere le differenze tra le tipologie `stable`, `fast` e `candidate`, vedere la pagina della documentazione su canali e release di aggiornamento.

È consigliabile eseguire tutti i cluster in produzione in un canale **stable**.

Ciclo di vita della versione supportata

Per pianificare il deployment in produzione, è fondamentale conoscere quali versioni di Azure Red Hat OpenShift sono supportate. Nella pagina ufficiale del ciclo di vita sono disponibili informazioni utili a comprendere quali versioni siano supportate e quali no.

Pagina dei cicli di vita supportati di Azure Red Hat OpenShift

Per semplificare al massimo le informazioni contenute nella pagina:

Azure Red Hat OpenShift supporta due versioni secondarie **generalmente disponibili (GA, Generally Available)** di Red Hat OpenShift Container Platform:

- La versione secondaria GA più recente rilasciata in Azure Red Hat OpenShift (che definiremo con N)
- Una versione secondaria precedente (N-1)

Red Hat OpenShift Container Platform utilizza la gestione delle versioni semantica, che impiega differenti livelli di numeri di versione per specificare i diversi livelli di gestione della versione. La tabella seguente illustra le parti di un numero di versione semantico, in questo caso la versione di esempio 4.9.3:

Versione principale (x)	Versione secondaria (y)	Patch (z)
4	9	3

Ogni numero nella versione indica la compatibilità generale con la versione precedente:

- **Versione principale:** al momento non sono pianificate release di versioni principali. Le versioni principali cambiano quando vengono modificate API incompatibili o quando si interrompe la retrocompatibilità.
- **Versione secondaria:** rilasciata approssimativamente ogni tre mesi. Gli aggiornamenti delle versioni secondarie possono includere aggiunte, migliorie, disattivazioni e rimozioni di funzioni, correzione di bug e ottimizzazioni della sicurezza.
- **Patch:** vengono in genere rilasciate ogni settimana o secondo necessità. Gli aggiornamenti di versione delle patch possono includere correzioni di bug e avanzamenti della sicurezza.

Altre informazioni sul funzionamento delle versioni supportate sono disponibili nella pagina relativa al ciclo di vita del supporto per Azure Red Hat OpenShift.

Scalabilità del cluster

Red Hat OpenShift Container Platform e, per estensione, Azure Red Hat OpenShift sono progettati per un'architettura scalabile. Quando pianificano l'ampliamento nel contesto di OpenShift, amministratori e proprietari delle applicazioni devono considerare la scalabilità del cluster e quella dell'applicazione come due argomenti distinti.

In questa sezione affronteremo la scalabilità del cluster, ma è presente una piccola nota separata che riguarda la scalabilità delle applicazioni.

Massimi supportati

Con scalabilità del cluster si fa riferimento all'aggiunta di ulteriori nodi di lavoro al cluster, che forniranno altra capacità di elaborazione alle applicazioni in esecuzione. Occorre inoltre verificare la necessità di rendere scalabile anche il piano di controllo. Azure Red Hat OpenShift presenta già tre nodi del piano di controllo, che in linea di principio dovrebbero fornire la capacità sufficiente per poter raggiungere il numero massimo di nodi di lavoro supportati in un cluster di Azure Red Hat OpenShift, che al momento è 60.

Al momento della stesura di questo documento, sono supportate tre dimensioni di istanza dei nodi del piano di controllo: `Standard_D8s_v3`, `Standard_D16s_v3` e `Standard_D16s_v3`.

Esistono altri tipi di istanze di Azure supportate per i nodi di lavoro: con ottimizzazione per il calcolo (serie F), con ottimizzazione per la memoria (serie E) e per utilizzo generico (serie D).

Un elenco dei tipi di istanza di Azure correntemente supportati per Azure Red Hat OpenShift è disponibile nella **pagina dei criteri di supporto**.

Deployment minimo e scalabilità a zero

In alcuni casi, le organizzazioni potrebbero voler distribuire cluster più piccoli per finalità di test e sviluppo, o in altre situazioni in cui una disponibilità ridotta è comunque accettabile. Al momento la dimensione minima del cluster è di tre nodi del piano di controllo e tre nodi applicazione e non è supportata alcuna dimensione inferiore a questa.

Scalabilità manuale del cluster

Gli operatori che visitano il portale di Azure potrebbero essere tentati di aggiungere ulteriori nodi di lavoro al cluster Azure Red Hat OpenShift creando direttamente una macchina virtuale Azure. Questa operazione non è però possibile, perché il gruppo di risorse che contiene Azure Red Hat OpenShift è "bloccato" dal punto di vista dell'amministratore di Azure, e indipendentemente dalle autorizzazioni: neanche gli utenti con l'autorizzazione **Proprietario** possono modificare i contenuti di un gruppo di risorse Azure Red Hat OpenShift. Pertanto, se si tenta di creare manualmente una macchina virtuale da questo gruppo si riceveranno messaggi di errore che segnalano l'autorizzazione negata.

Il meccanismo sottostante alla scalabilità di Azure Red Hat OpenShift è la funzione di OpenShift `MachineSets`, in base alla quale OpenShift distribuirà un nuovo nodo applicazione quando indicato. Gli utenti con privilegi di amministrazione del cluster potranno visualizzare **MachineSets** in **Compute**.

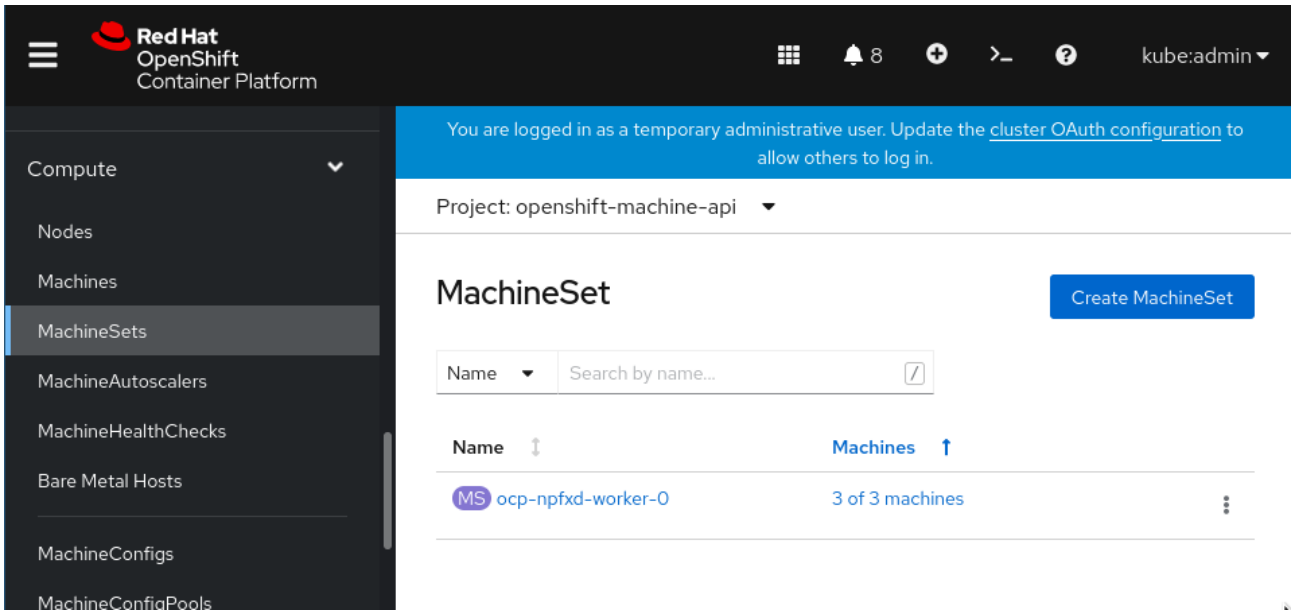


Figura 6.5: Accesso come amministratore alla funzione MachineSets

Facendo clic sul menu "edit" di un nodo applicazione `MachineSet`, sarà possibile eseguire in modo semplice il provisioning di una nuova macchina virtuale per il nodo applicazione, selezionando l'opzione **Edit Machine Count**.

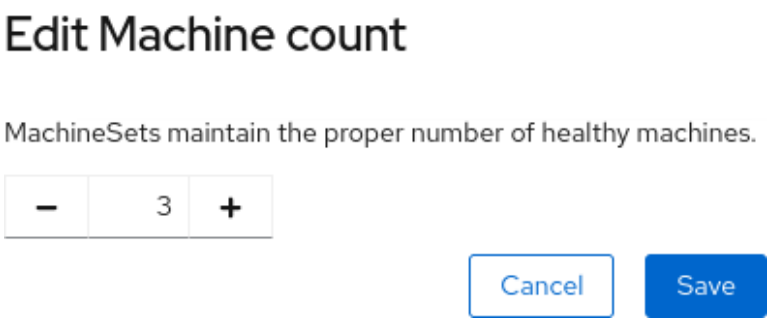


Figura 6.6: Modifica del conteggio computer

Dopo aver aggiornato il conteggio, gli amministratori possono passare al portale di Azure o alla vista **Compute** → **Machines** per visualizzare il provisioning in corso della nuova macchina virtuale del nodo applicazione.

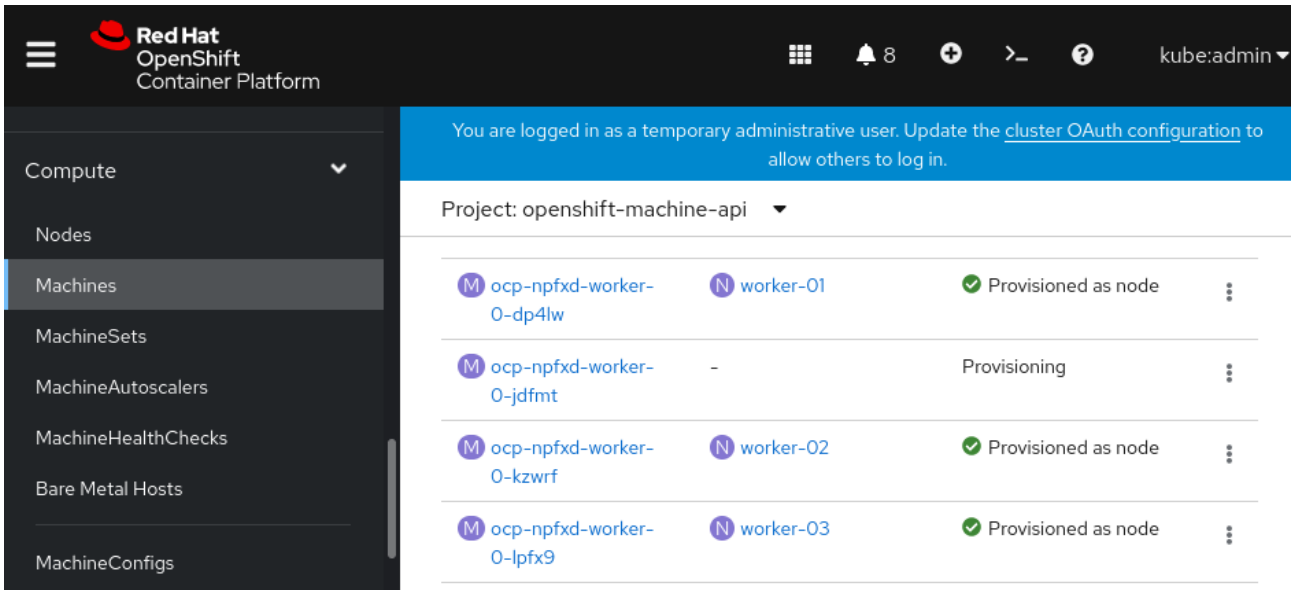


Figura 6.7: Vista dei computer

Nella maggior parte delle aree di Azure sono in genere necessari circa cinque minuti per il provisioning di una macchina virtuale aggiuntiva.

Per portare online questa nuova capacità non servono attività di post provisioning, in quanto OpenShift la renderà automaticamente disponibile nel cluster e le applicazioni potranno utilizzarla quando necessario.

Scalabilità automatica del cluster

Nel deployment predefinito di Azure Red Hat OpenShift la funzionalità di scalabilità automatica non è configurata, ma attivarla è davvero semplice. L'amministratore dovrà semplicemente creare una risorsa `MachineAutoscaler`, che si trova nel menu laterale **Compute** di Azure Red Hat OpenShift.

La risorsa `MachineAutoscaler` opera su un `MachineSet`, che a sua volta crea o elimina la capacità della macchina virtuale del nodo applicazione, secondo necessità. L'esempio seguente mostra come sia possibile gestire un numero massimo o minimo di computer in un `MachineSet`:

```
apiVersion: autoscaling.openshift.io/v1beta1
kind: MachineAutoscaler
metadata:
    name: worker-us-east-1a
    namespace: openshift-machine-api
spec:
    minReplicas: 1
    maxReplicas: 12
    scaleTargetRef:
        apiVersion: machine.openshift.io/v1beta1
        kind: MachineSet
        name: worker
```

OpenShift prevede anche uno strumento per la scalabilità automatica, che consente la scalabilità in base alla disponibilità di una quantità stabilita di RAM, CPU o simili. La scelta tra `MachineAutoscaler` o `ClusterAutoscaler` dipende da come si intende aggiungere e rimuovere la capacità nel cluster.

Documentazione di Red Hat OpenShift su MachineAutoscalers e ClusterAutoscalers

Scalabilità delle applicazioni

La scalabilità delle applicazioni basate su microservizi è un argomento complesso che esula dall'ambito di questo documento. Di seguito segnaliamo due utili pagine introduttive su questo tema:

- HorizontalPodAutoscaler: per specificare il numero minimo e massimo di pod da eseguire, oltre all'utilizzo di CPU o memoria previsto per i pod.
- Serverless e scalabilità a zero con Knative.

Il cluster monitora i container in esecuzione e la capacità rimanente. Nel caso in cui Knative o `HorizontalPodAutoscaler` richiedano più risorse di quelle disponibili al momento nel cluster, `ClusterAutoscaler` o `MachineSet` interverranno per aggiungere le risorse necessarie.

Con questo approccio è possibile dimensionare sia le applicazioni che il cluster, insieme e in qualsiasi direzione, assecondando le esigenze.

Configurazione di un segreto pull (registrazione in OpenShift Cluster Manager)

In un deployment recente di Azure Red Hat OpenShift non è ancora configurato un "segreto pull" per `cloud.redhat.com`. Ciò significa che, per impostazione predefinita, i cluster non saranno visibili nella console cloud di Red Hat Hybrid (http://console.redhat.com) ma in OpenShift Cluster Manager.

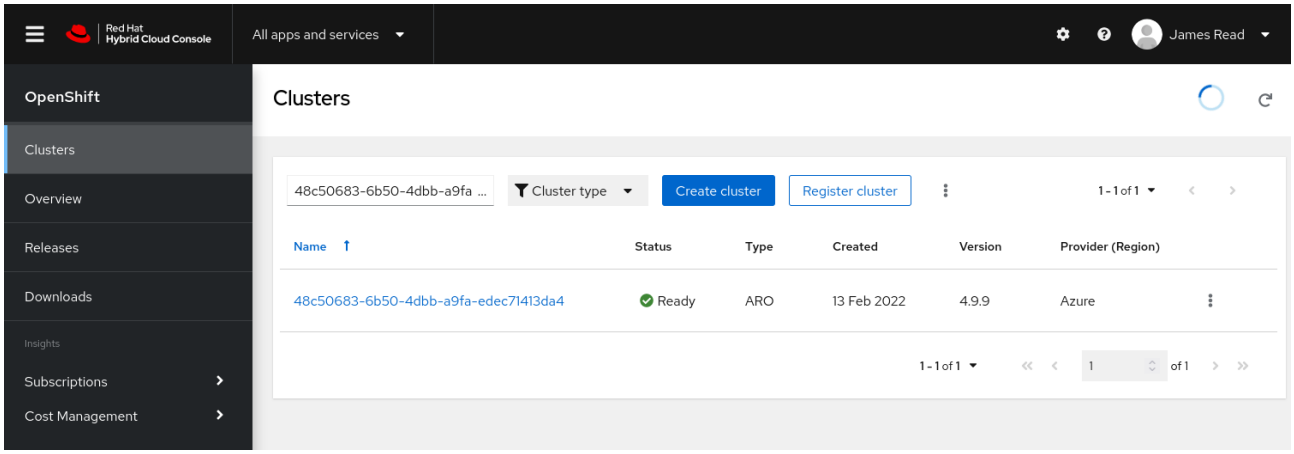


Figura 6.8: OpenShift Cluster Manager che visualizza un cluster Azure Red Hat OpenShift

La configurazione di un segreto pull per `cloud.redhat.com` è molto semplice e oltre a visualizzare il cluster nel portale OpenShift Cluster Manager all'indirizzo http://console.redhat.com, consente di generare ticket di supporto direttamente in Red Hat, tramite il sistema di generazione dei ticket di supporto standard.

Qui si trovano le istruzioni per configurare un segreto pull:

- Come aggiungere o aggiornare un segreto pull

Un ulteriore vantaggio derivante dalla configurazione di un segreto pull è che il cliente può generare i ticket di supporto direttamente con Red Hat. Il segreto pull garantisce un diritto nell'account Red Hat che rende visibile il cluster al personale del supporto. Per aprire un ticket di supporto per Azure Red Hat OpenShift si seguirà quindi la stessa procedura utilizzata per qualsiasi altro prodotto Red Hat.

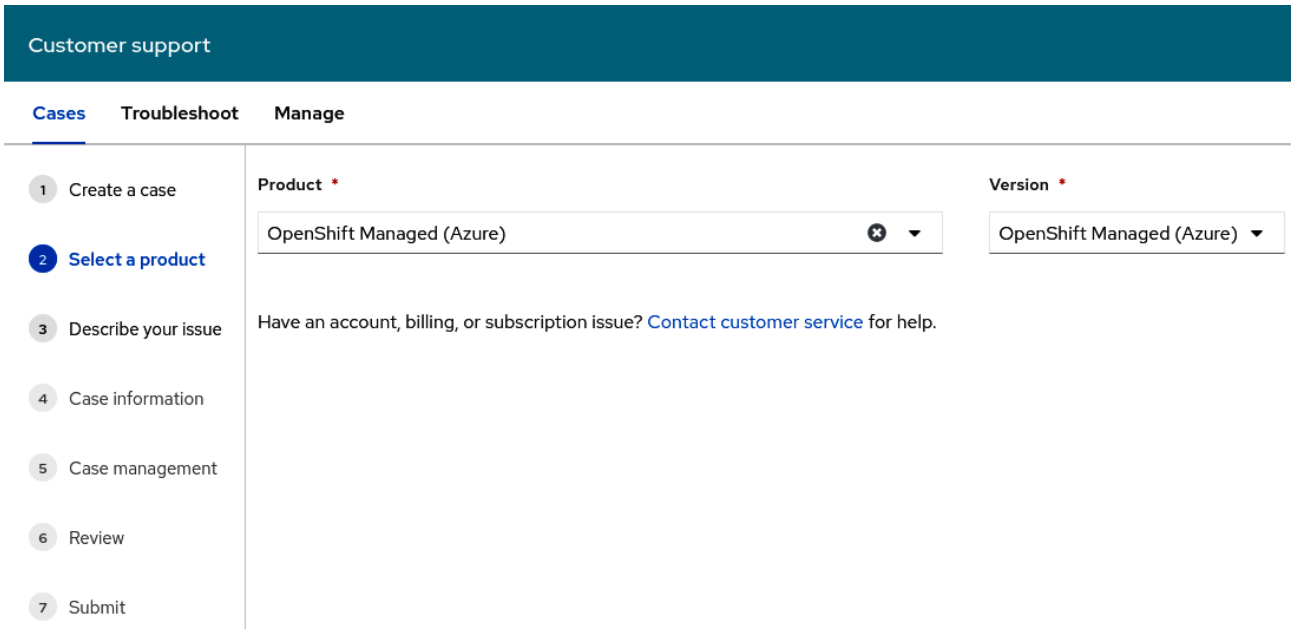


Figura 6.9: Creazione di un caso per il supporto

Intervalli limite

In genere, poco tempo dopo la distribuzione di applicazioni containerizzate da parte di un team di sviluppo o che gestisce le applicazioni, queste iniziano a comportarsi in modo errato e a consumare risorse non necessarie nel cluster. Ne è un esempio un'applicazione difettosa con una perdita di memoria, o una che sia stata erroneamente impostata per dimensionarsi dopo aver consumato il 10% della CPU invece del 100%. Per evitare situazioni in cui la scalabilità delle applicazioni va fuori controllo e consuma troppe risorse, è consigliabile utilizzare `LimitRange`.

`LimitRange` consente di limitare il consumo di risorse per alcuni oggetti specifici di un progetto. È possibile applicare `LimitRange` a:

- Pod e container: è possibile definire i requisiti minimi e massimi di CPU e memoria per i pod e i rispettivi container.
- Flussi di immagini: è possibile limitare il numero di immagini e tag in un oggetto `ImageStream`.
- Immagini: è possibile limitare la dimensione delle immagini che possono essere inviate a un registro interno.
- **Richieste di volumi permanenti (Persistent Volume Claim, PVC):** è possibile limitare la dimensione delle PVC che possono essere inviate.

Un esempio di intervallo limite applicabile ai container, che regola il livello minimo, massimo e predefinito di richieste di CPU e memoria consentite, è mostrato di seguito:

```
apiVersion: "v1"
kind: "LimitRange"
metadata:
  name: "resource-limits"
spec:
  limits:
    - type: "Container"
      max:
        cpu: "2"
        memory: "1Gi"
      min:
        cpu: "100m"
        memory: "4Mi"
      default:
        cpu: "300m"
        memory: "200Mi"
      defaultRequest:
        cpu: "200m"
        memory: "100Mi"
      maxLimitRequestRatio:
        cpu: "10"
```

Questo snippet di codice `LimitRange` può essere applicato copiando e incollando il codice YAML direttamente nell'editor della console Red Hat OpenShift o da un file con `oc apply -f limitrange.yaml`.

[Documentazione sugli intervalli limite]

Storage permanente

Alle macchine virtuali distribuite da Azure Red Hat OpenShift sono collegati dischi di Azure che servono per installare Red Hat CoreOS e per eseguire il servizio Azure Red Hat OpenShift. Questi dischi non devono essere utilizzati dalle applicazioni ma solo dal cluster OpenShift.

Per le applicazioni che richiedono lo storage permanente è indicata la funzione `PersistentVolume` di Kubernetes; nella documentazione di OpenShift è disponibile una pagina esaustiva che descrive in dettaglio i concetti di base dello storage permanente. Mentre tecnicamente Azure Red Hat OpenShift supporta tutti i provider di `PersistentVolume` come installazione OpenShift autogestita, in Azure gli storage permanenti più utilizzati sono:

Nome	Tipo	Modalità di accesso	Classe di storage
Azure Files	File system, non conforme a POSIX	ReadWriteOnce	Azure File
Azure Disk	Blocco	ReadWriteOnce	Azure Disk
Red Hat OpenShift Data Foundation	File system, Blocco, Oggetto	(Numerosi)	Documenti OCS/ODF

Sicurezza e conformità

La documentazione di Red Hat OpenShift include una notevole quantità di contenuti utili a comprendere come integrare controlli di sicurezza e come gestire la conformità.

Documentazione di OpenShift su sicurezza e conformità

Questa sezione della documentazione include:

- Una descrizione dettagliata del funzionamento della sicurezza dei container in OpenShift. In OpenShift sono disponibili molti controlli di sicurezza out-of-the-box per container, non reperibili in altri servizi basati su Kubernetes; questi controlli aiutano a mantenere protette organizzazioni e applicazioni.
- Scansione dei pod alla ricerca di vulnerabilità
- Accesso ai registri di audit
- Configurazione di certificati

Sono spiegati anche alcuni operatori relativi alla sicurezza, quali:

- **Operatore di conformità:** esegue scansioni e fornisce raccomandazioni per la soluzione di problematiche di sicurezza comuni.
- **Controllo dell'integrità dei file:** controlla in modo continuativo che siano apportate modifiche ai file, soprattutto a quelli di configurazione della sicurezza dei dati sensibili.

Riepilogo

In questo capitolo abbiamo esaminato la maggior parte delle attività e degli argomenti di cui deve tener conto un'organizzazione per adattare un cluster appena distribuito alle applicazioni di produzione. Non è necessario affrontare questi temi a ogni successivo deployment, ma per il primo cluster distribuito devono essere considerati lo storage permanente, i limiti, l'autenticazione e gli altri vari argomenti citati.

In genere un'organizzazione tenterà di automatizzare il più possibile le attività di post-provisioning. Ad esempio, è possibile procedere all'installazione e alla configurazione di Azure Active Directory utilizzando uno script di PowerShell o qualche modello ARM. Sarà così possibile ridurre il tempo dedicato alle attività ripetitive nel caso in cui si esegua il deployment di molti cluster.

Il capitolo successivo di questo documento illustra il deployment di un'applicazione di esempio in un cluster production-ready di Azure Red Hat OpenShift.

Capitolo 7

Deployment di un'applicazione di esempio

Il contenuto di questa guida è destinato a un pubblico di tecnici, principalmente ai team di sviluppo e operativi, che desidera approfondire gli aspetti fondamentali della fase di produzione di Azure Red Hat OpenShift. La guida presuppone che il lettore abbia già familiarità con i concetti di base di Red Hat OpenShift, perché gran parte dell'architettura, dei portali di gestione e dell'esperienza utente sono identici a quelli di Azure Red Hat OpenShift.

Questo capitolo costituisce un punto di partenza per illustrare come eseguire il deployment di una semplice applicazione di esempio chiamata "Fruit Smoothies", che è intesa come introduzione e promemoria per comprendere meglio come utilizzare Azure Red Hat OpenShift. Si tratta di un'applicazione di esempio gestita per Azure Kubernetes Service il cui deployment in Azure Red Hat OpenShift viene effettuato per dimostrare come OpenShift sia totalmente compatibile con Kubernetes.

Il capitolo si basa sui contenuti presenti in http://aroworkshop.io, ai quali sono stati aggiunti descrizioni e contesto.

Panoramica dell'applicazione di rating

L'architettura dell'applicazione è semplice, come indicato nell'illustrazione:

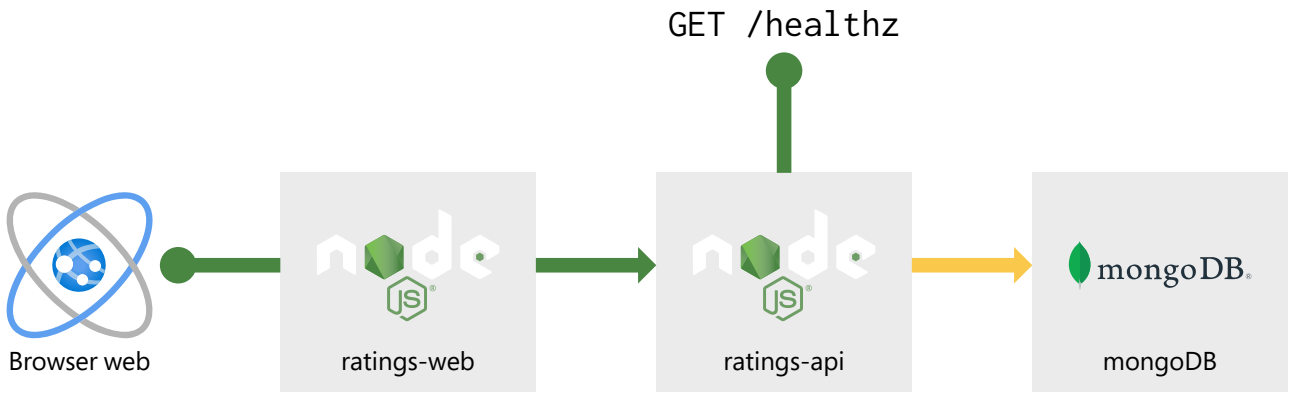


Figura 7.1: Architettura dell'applicazione Fruit Smoothies

Come mostra lo schema precedente, l'applicazione è composta da tre servizi e due endpoint pubblici, illustrati anche nella tabella seguente. Il primo endpoint a sinistra dell'immagine rappresenta il browser web di un utente che visualizza l'applicazione web HTML pubblica. Il secondo endpoint, `healthz`, è un controllo integrità nel servizio `ratings-api`. La tabella sottostante contiene le descrizioni dei singoli servizi e i collegamenti ai relativi repository:

Componente	Descrizione	Repository GitHub
`rating-web`	Un front end web pubblico: il "sito web"	Repository GitHub
`rating-api`	Il servizio acquisisce l'input dall'IU web e lo archivia nel database. Restituisce inoltre i risultati dal database all'applicazione web, sulla porta 3000.	Repository GitHub
`mongodb`	Un database NoSQL con dati precaricati.	Dati

Per approfondire e comprendere queste applicazioni, puoi visitare i collegamenti ai repository GitHub. Le istruzioni che seguono mostrano tutti i passaggi necessari per eseguire il deployment di ogni applicazione.

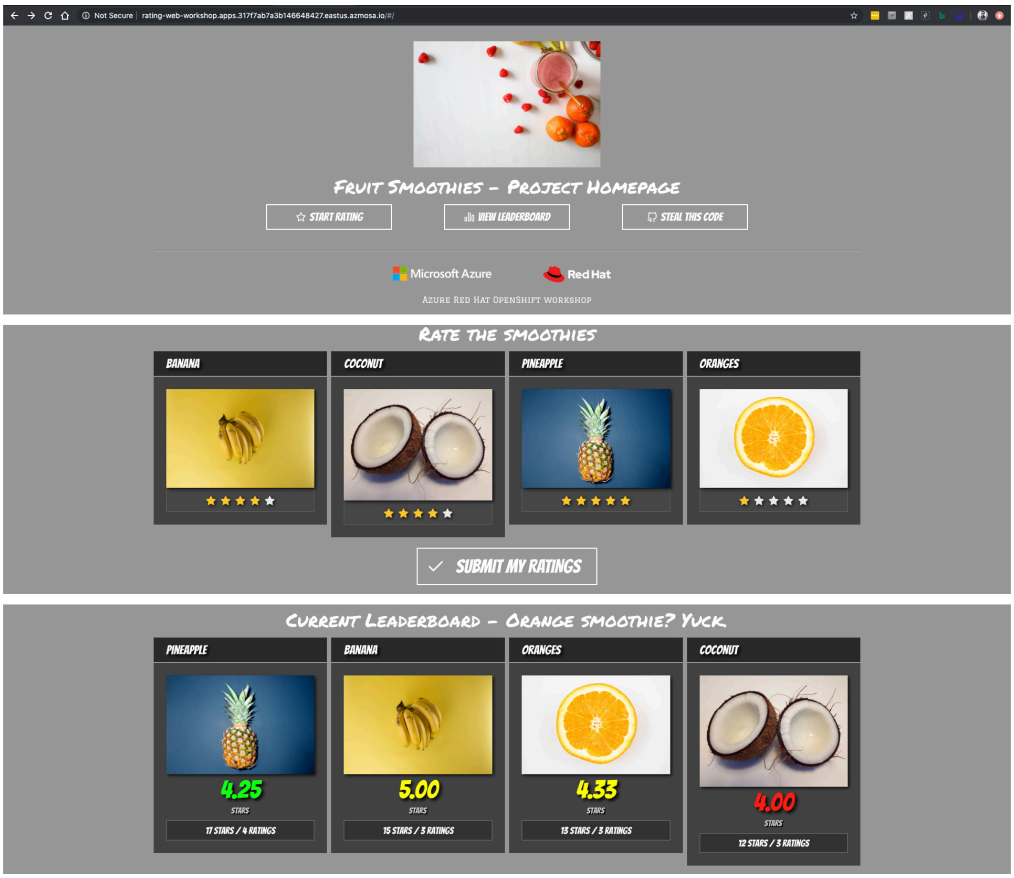


Figura 7.2: Panoramica dell'applicazione

Al termine, otterrai un'applicazione web che avrà un aspetto simile a quello mostrato nelle immagini precedenti. Avrai anche compreso meglio i passaggi necessari ai team di sviluppo e operativi per eseguire il deployment delle applicazioni in Azure Red Hat OpenShift, che ti saranno utili per un corretto processo decisionale quando dovrai eseguire il deployment in autonomia.

Creare e connettere il cluster

Per il resto del capitolo si presuppone l'esistenza di un ambiente Azure Red Hat OpenShift funzionante nel quale operare. Questa applicazione di rating non presenta requisiti specifici e verrà distribuita in un ambiente di Azure Red Hat OpenShift vuoto e appena creato. Se non hai mai eseguito il provisioning di un cluster, rivedi i capitoli seguenti:

- *Capitolo 4, Domande sul pre-provisioning dell'architettura enterprise*
- *Capitolo 5, Provisioning di un cluster Azure Red Hat OpenShift*

La sezione *Accesso al cluster* nel *Capitolo 5, Provisioning di un cluster Azure Red Hat OpenShift,* è un ottimo promemoria su come accedere a un cluster di cui è già stato eseguito il provisioning.

Accedere alla web console

Ogni cluster Azure Red Hat OpenShift dispone di un indirizzo DNS per la web console OpenShift. Per visualizzare i cluster nella sottoscrizione Azure corrente, è possibile utilizzare il comando `az aro list`:

```
az aro list -o table
```

Viene visualizzato l'URL della web console del cluster. Aprire il collegamento in una nuova scheda del browser e accedere come utente `kubeadmin` o con un altro account utente con autorizzazioni per la creazione di progetti.

Dopo aver effettuato l'accesso, verrà visualizzata la web console di Azure Red Hat OpenShift.

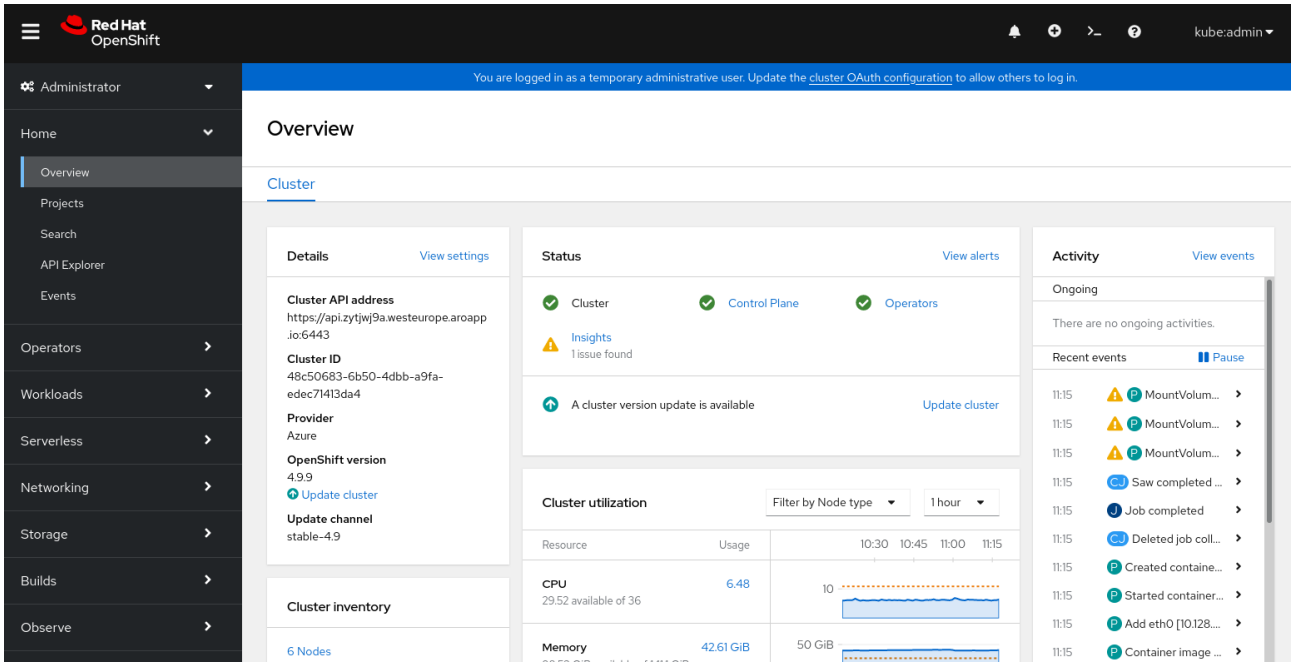


Figura 7.3: Web console di Azure Red Hat OpenShift

Installare il client OpenShift

Aprire Azure Cloud Shell o utilizzare il terminal Linux locale e installare il client da riga di comando OpenShift, che sarà necessario per accedere al cluster dalla riga di comando. Di seguito le istruzioni per questa operazione:

```
cd ~
curl https://mirror.openshift.com/pub/openshift-v4/clients/ocp/latest/openshift-client-linux.tar.gz > openshift-client-linux.tar.gz

mkdir openshift

tar -zxvf openshift-client-linux.tar.gz -C openshift

echo 'export PATH=$PATH:~/openshift' >> ~/.bashrc && source ~/.bashrc
```

In alternativa, per Windows o Mac, i collegamenti per il download sono i seguenti:

- https://mirror.openshift.com/pub/openshift-v4/clients/ocp/latest/openshift-client-windows.zip
- https://mirror.openshift.com/pub/openshift-v4/clients/ocp/latest/openshift-client-mac.tar.gz

Sarà possibile eseguire il comando oc da uno dei pacchetti scaricati.

Recuperare il comando e il token di accesso

Dopo aver installato il client è necessario ottenere un token per accedere al cluster. Accedere alla web console di OpenShift, fare clic sul nome utente in alto a destra, e quindi fare clic su **Copy login command**.

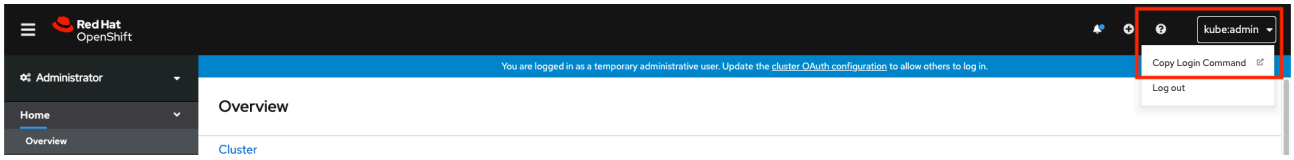


Figura 7.4: Copy Login Command per accedere al cluster

Incollare il comando di accesso nella shell (terminale Linux locale o Azure Cloud Shell). A questo punto è possibile connettersi al cluster.

Figura 7.5: Utilizzare il comando di accesso per connettersi al cluster

Una volta effettuata la connessione, si procede con la creazione del progetto.

Creazione di un progetto

In OpenShift un progetto è simile a una cartella logica che contiene l'applicazione di rating. In Red Hat OpenShift, tutti i container e le applicazioni devono trovarsi in un progetto di qualche tipo. È possibile utilizzare i progetti per separare le applicazioni e perfino i reparti. Le istruzioni seguenti illustrano come creare un progetto.

Sebbene sia possibile creare un progetto dall'interfaccia web, queste istruzioni utilizzano la riga di comando:

```
oc new-project workshop
```

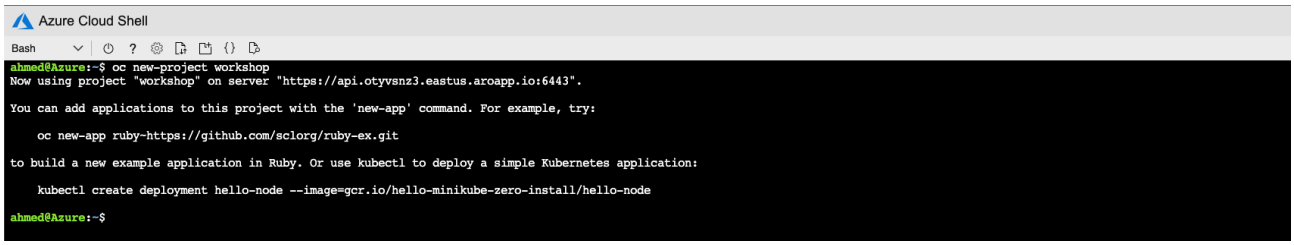


Figura 7.6: Creare un nuovo progetto workshop in Azure Cloud Shell

Una volta creato il progetto, sarà possibile aprirlo utilizzando `oc project workshop`. Il passaggio successivo consiste nel deployment del primo dei tre microservizi che sono stati presentati all'inizio di questo capitolo, MongoDB, che verrà eseguito nel progetto appena creato.

Risorse

- Documentazione di Azure Red Hat OpenShift – Introduzione alla CLI
- Documentazione Azure Red Hat OpenShift – Progetti

Deployment di MongoDB

Azure Red Hat OpenShift fornisce un modello e un'immagine di container che facilitano la creazione di un nuovo servizio database di MongoDB. Il modello fornisce dei campi parametrici nei quali definire tutte le variabili di ambiente obbligatorie (utente, password, nome del database, ecc.) con valori predefiniti, inclusa la generazione automatica dei valori delle password. Vengono inoltre definiti una configurazione del deployment e un servizio.

L'istanza di MongoDB viene distribuita direttamente come immagine del container da Docker Hub. OpenShift consente di eseguire questa operazione direttamente tramite la web console. Passare alla visualizzazione per gli sviluppatori, in alto nel menu, quindi aprire la pagina **Add** e selezionare **Container images**.

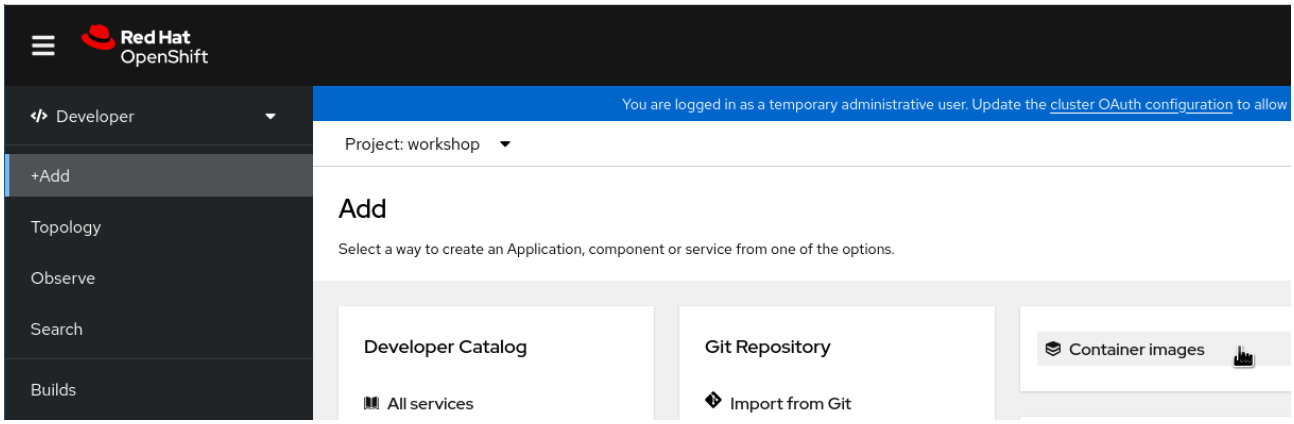


Figura 7.7: Aggiunta di un'immagine del container

Viene visualizzata la pagina **Deploy Image**. Compilare il form come indicato di seguito:

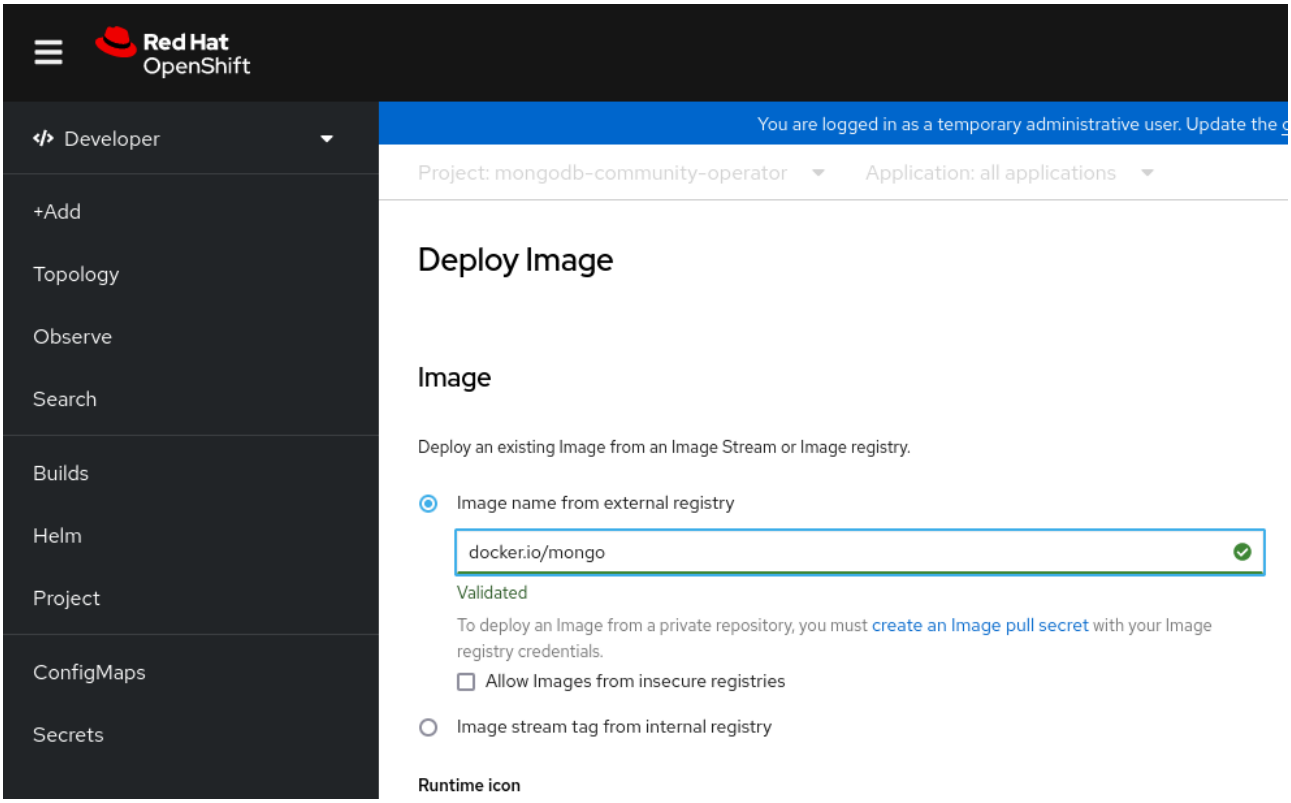


Figura 7.8: Form da compilare per il deployment dell'immagine

Accertarsi di impostare i valori nel form come segue:

Campo	Valore
Image name from external registry	`docker.io/mongo`
Runtime icon	`mongodb`
Application name	`ratings`
Nome	`mongodb`
Create route to application	Non selezionato - una route consentirebbe l'accesso al database dall'esterno, il che non è necessario per questa applicazione
Resource type	Deployment

In fondo al form selezionare il collegamento Deployment per espandere il form e consentire l'impostazione delle variabili di ambiente.

La seguente variabile di ambiente funge da valore predefinito per inizializzare il database al primo avvio:

```
Name                    Value
MONGO_INITDB_DATABASE   ratingsdb
```

Per impostazione predefinita, per il database MongoDB nome utente e password non vengono impostati e l'autenticazione è disabilitata.

Controllare di nuovo la variabile di ambiente e quindi fare clic sul pulsante **Create** per proseguire e creare il container MongoDB.

Dopo poco, l'istanza di MongoDB sarà presente e in esecuzione nel progetto `workspace`. Questo deployment è visualizzabile passando alla vista **Topology**.

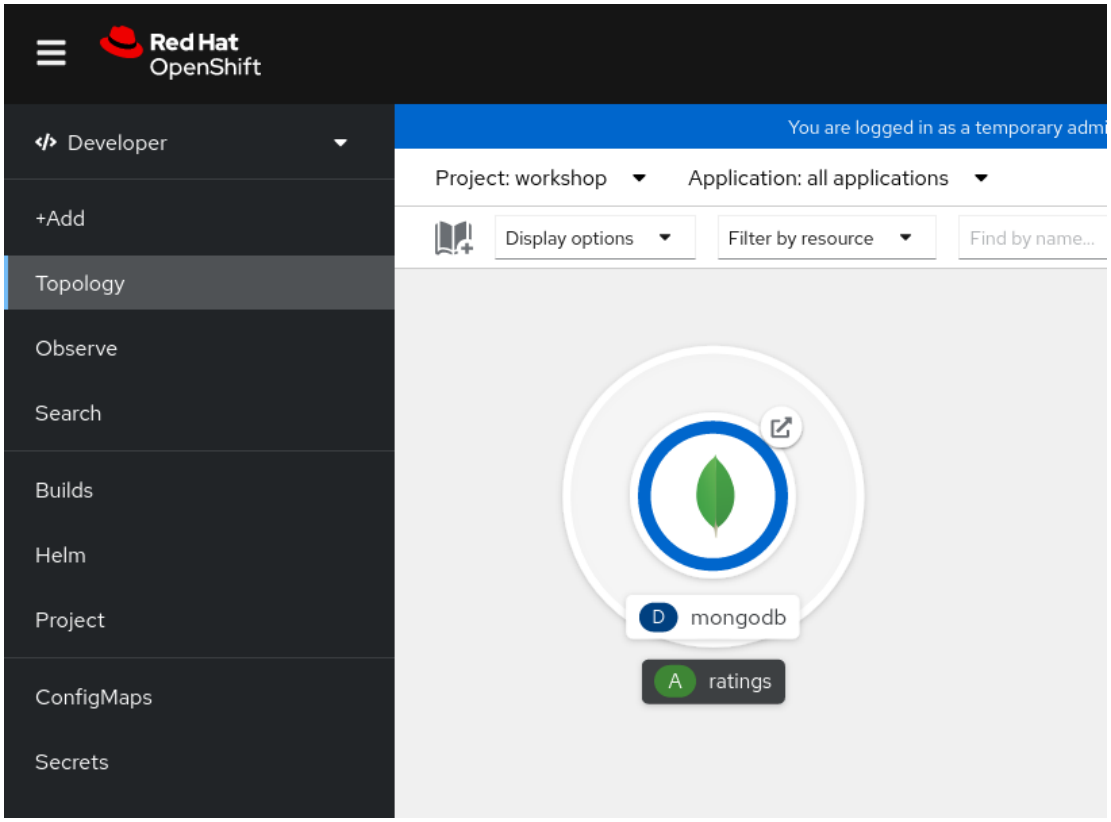


Figura 7.9: Istanza di MongoDB attiva e in esecuzione

Eseguire il comando `oc get all` per visualizzare lo stato della nuova applicazione e verificare che il deployment del modello MongoDB sia riuscito. Di seguito un esempio di output del comando `oc get all`:

```
user@host: oc get all
NAME                              READY    STATUS     RESTARTS    AGE
pod/mongo-6c6fcb45b8-8wpdm    1/1      Running    0           29s

NAME             TYPE         CLUSTER-IP       EXTERNAL-IP    PORT(S)      AGE
service/mongo    ClusterIP    172.30.88.119    <none>         27017/TCP    30s

NAME                      READY    UP-TO-DATE    AVAILABLE    AGE
deployment.apps/mongo     1/1      1             1            30s

NAME                                  DESIRED    CURRENT    READY    AGE
replicaset.apps/mongo-6c6fcb45b8    1          1          1        30s

NAME                                        IMAGE REPOSITORY
TAGS      UPDATED
imagestream.image.openshift.io/mongo    image-registry.openshift-image-registry.svc:5000/workshop/mongo
latest    30 seconds ago
```

Se tutto funziona correttamente, la colonna `STATUS` conferma che il container è in esecuzione (Running).

Recuperare il nome host del servizio MongoDB

Completato il deployment, è necessario individuare il servizio creato per poter accedere al database dal cluster. `svc` è l'abbreviazione di `services`:

```
user@host: oc get svc mongodb
NAME     TYPE         CLUSTER-IP       EXTERNAL-IP    PORT(S)      AGE
mongo    ClusterIP    172.30.88.119    <none>         27017/TCP    77s
```

Il servizio sarà accessibile al seguente nome DNS: `mongodb.workshop.svc.cluster.local`, formato da `[nome servizio].[nome progetto].svc.cluster.local`. Il nome viene risolto solo all'interno del cluster.

Deployment di Ratings API

Si procede ora con il deployment della seconda applicazione, chiamata `rating-api`. Si tratta di un'applicazione node.js che si connette a un'istanza di MongoDB per recuperare e classificare gli elementi. Di seguito sono fornite alcune delle informazioni necessarie per il deployment dell'applicazione.

- `rating-api` in GitHub
- Il container espone la porta `3000`
- La connessione MongoDB viene configurata utilizzando una variabile di ambiente denominata `MONGODB_URI`

Annotare queste informazioni poiché verranno utilizzate nelle sezioni successive.

Biforcare l'applicazione nel repository GitHub personale

Per poter apportare modifiche, ad esempio aggiungere webhook CI/CD, è necessario disporre di una copia personale del codice di `ratings-api`. In Git, questa copia è anche chiamata "fork" o biforcazione. È possibile biforcare la copia dell'applicazione nel proprio repository GitHub. Aprire il repository GitHub precedente e fare clic sul pulsante **Fork**.

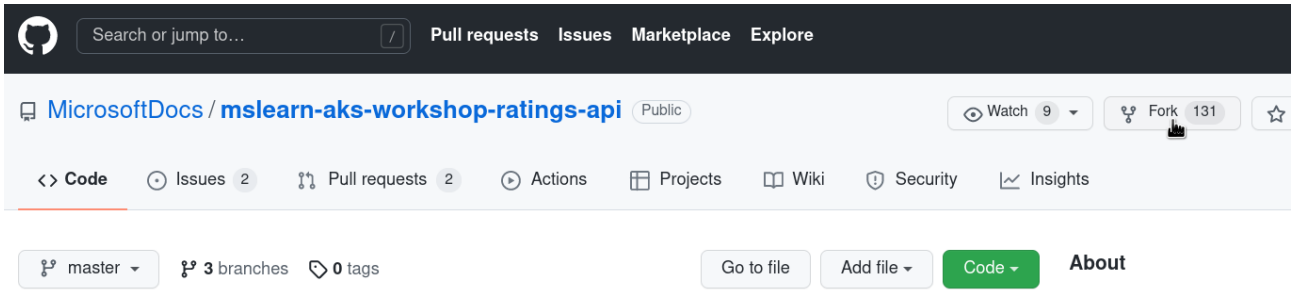


Figura 7.10: Biforcazione dell'applicazione nel repository GitHub

Annotare il nuovo indirizzo del repository, che verrà utilizzato nelle istruzioni seguenti.

Utilizzare la CLI di OpenShift per il deployment di rating-api

OpenShift consente il deployment del codice direttamente da un repository Git mediante la ricerca dei contenuti e la selezione di un generatore di immagini basato sui contenuti Java, PHP, Perl, Python o simili. Nel nostro caso, `rating-api` è un'applicazione JavaScript. Il generatore di immagini esegue il download delle dipendenze JavaScript mediante npm; ne risulterà una nuova immagine del container archiviata nel registro interno dei container di OpenShift. Questa strategia di build è denominata **Source 2 Image (S2I)** ed è descritta in dettaglio nel glossario.

È possibile avviare una nuova build S2I con il comando `oc new-app`:

```
user@host: oc new-app https://github.com/<your GitHub username>/mslearn-aks-workshop-ratings-api
--strategy=source --name=rating-api

--> Found image 0aea15f (3 weeks old) in image stream "openshift/nodejs" under tag "14-ubi8" for
"nodejs"

    Node.js 14
    ----------
     Node.js 14 available as container is a base platform for building and running various Node.
js 14 applications and frameworks. Node.js is a platform built on Chrome's JavaScript runtime
for easily building fast, scalable network applications. Node.js uses an event-driven, non-
blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time
applications that run across distributed devices.

    Tags: builder, nodejs, nodejs14

    * The source repository appears to match: nodejs
    * A source build using source code from https://github.com/MicrosoftDocs/mslearn-aks-
workshop-ratings-api will be created
      * The resulting image will be pushed to image stream tag "rating-api:latest"
      * Use 'oc start-build' to trigger a new build
```

Passare alla vista **Topology** nella web console. Viene visualizzato l'avvio della build dell'applicazione e dopo qualche minuto il deployment.

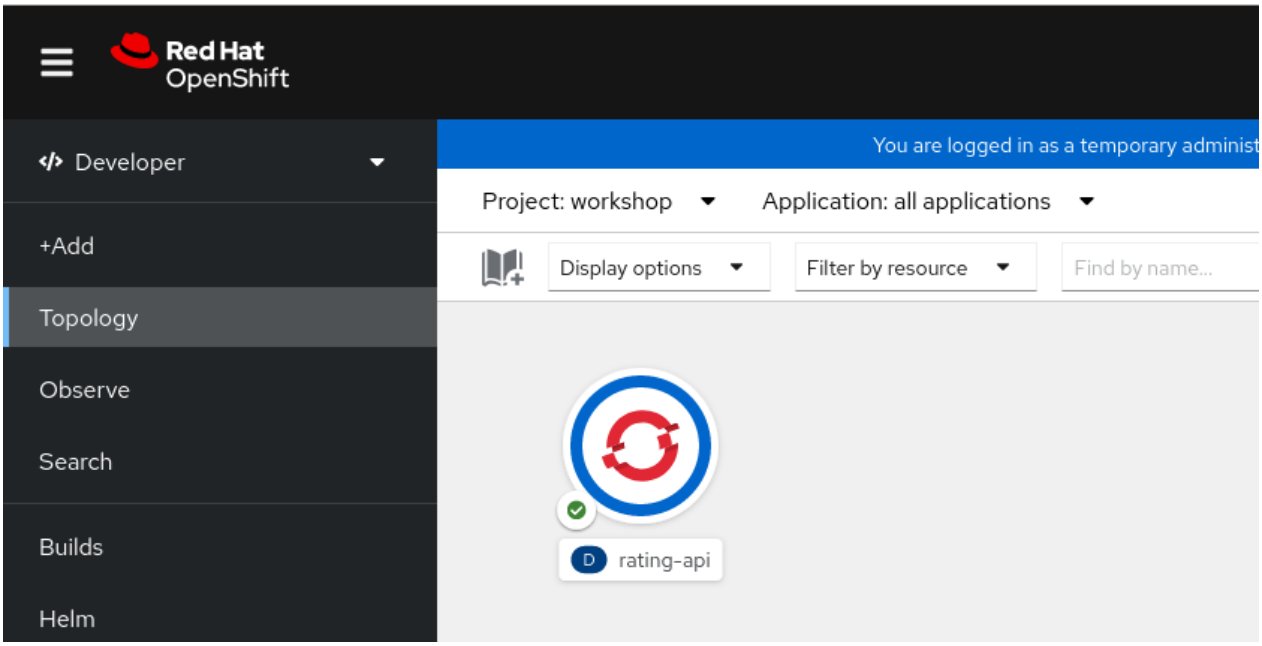


Figura 7.11: Vista Topology

La build e l'avvio del pod dovrebbero richiedere solo un minuto o due.

Configurare le variabili di ambiente necessarie

A questo punto, il deployment del database e dell'API di rating dovrebbe essere completato. Occorre indicare all'API come connettersi al database. La configurazione delle applicazioni containerizzate viene in genere eseguita utilizzando variabili di ambiente.

Fare clic sul deployment e modificarlo. Creare la seguente variabile di ambiente:

```
Name          Value
MONGODB_URI   mongodb://mongodb.workshop.svc.cluster.local:27017/ratingsdb
```

Dopo averla salvata, la variabile attiverà la ridistribuzione del servizio `ratings-api` affinché utilizzi la nuova variabile di ambiente.

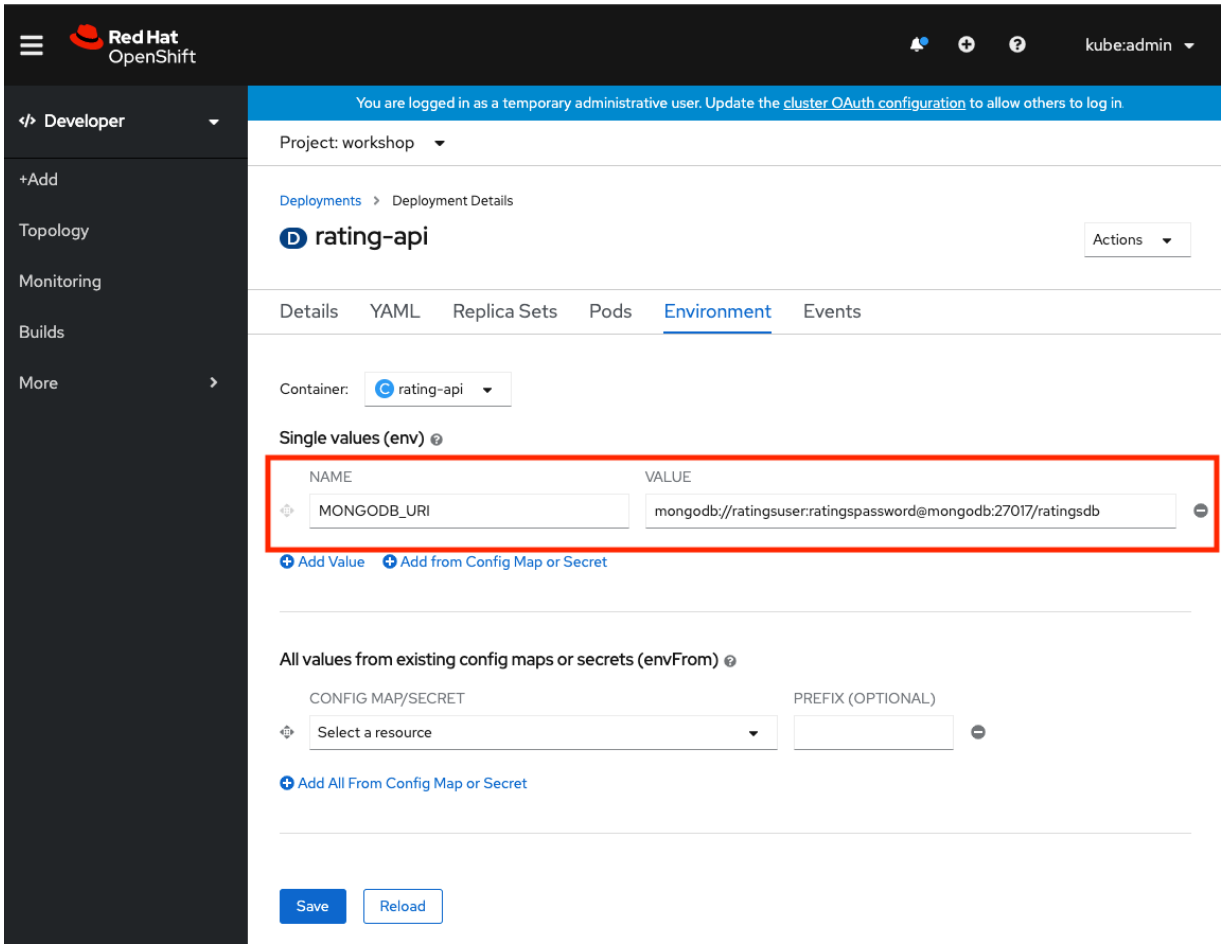


Figura 7.12: Impostazione della variabile di ambiente MONGODB_URI tramite la web console

È anche possibile utilizzare la riga di comando, come mostrato di seguito:

```
oc set env deploy/rating-api MONGODB_URI=mongodb://mongodb.workshop.svc.cluster.local:27017/ratingsdb
```

Qualunque sia il metodo utilizzato, OpenShift dovrà riavviare il container per poter utilizzare le nuove variabili di ambiente.

Verificare che il servizio sia in esecuzione

Se si passa ai registri del deployment di `rating-api` dovrebbe essere visualizzato un messaggio di registro che conferma la riuscita della connessione del codice a MongoDB. A tal fine, nella schermata dei dettagli del deployment, fare clic sulla scheda **Pods** e quindi su uno dei pod.

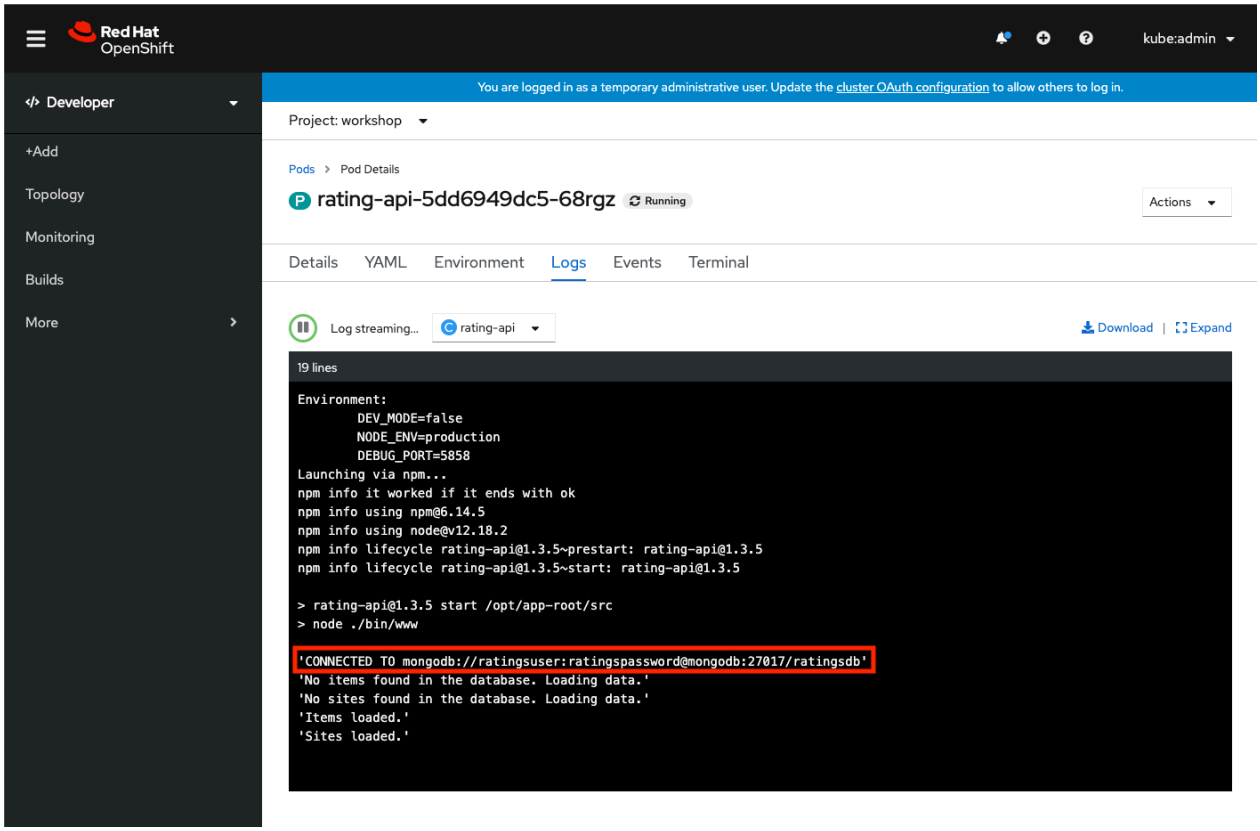


Figura 7.13: Messaggio del registro che conferma che il codice è stato connesso a MongoDB

Modificare la porta del servizio rating-api

OpenShift crea un servizio utilizzando la porta `8080`. A causa di un aggiornamento della libreria, tuttavia, questo servizio viene eseguito sulla porta `3000`. È necessario modificare il servizio predefinito.

Passare al menu **Networking** → **Services** e, dall'elenco dei servizi, modificare il servizio `rating-api` come indicato di seguito (sostituire `8080` con `3000`):

```
  ports:
    - name: 3000-tcp
      protocol: TCP
      port: 3000
      targetPort: 3000
```

Riavviare il servizio affinché utilizzi la nuova porta:

```
user@host: oc rollout restart deploy/rating-api
```

A questo punto il servizio sarà vincolato alla porta corretta, con il numero `3000` invece di `8080`.

Recuperare il nome host del servizio rating-api

È necessario convalidare la presenza del servizio `rating-api`, perché verrà utilizzato nella sezione seguente per il deployment dell'applicazione `rating-web`:

```
oc get service rating-api
```

Il servizio sarà accessibile al nome DNS `rating-api.workshop.svc.cluster.local:3000`, sulla porta `3000`, formato da `[nome servizio].[nome progetto].svc.cluster.local`. Il nome viene risolto solo all'interno del cluster.

Deployment del front end di rating

`rating-web` è un'applicazione Node.js che si connette a `rating-api`. Di seguito sono fornite alcune delle informazioni necessarie per il deployment dell'applicazione.

- `rating-web` in GitHub
- Il container espone la porta `8080`
- L'app web si connette all'API sul DNS interno del cluster, utilizzando un proxy tramite una variabile di ambiente denominata `API`

Utilizzare la CLI di OpenShift per il deployment di rating-web

Come con l'applicazione `rating-api`, è possibile distribuire anche questa applicazione con S2I mediante il comando `oc new-app`. Questa volta l'applicazione viene creata con una strategia Dockerfile, mediante un Dockerfile già presente nel repository Git. Non va quindi specificato l'argomento `--strategy`:

```
user@host: oc new-app https://github.com/<your GitHub username>/mslearn-aks-workshop-ratings-web
--name rating-web

--> Found container image e1495e4 (2 years old) from Docker Hub for "node:13.5-alpine"

    * An image stream tag will be created as "node:13.5-alpine" that will track the source image
    * A Docker build using source code from https://github.com/MicrosoftDocs/mslearn-aks-
workshop-ratings-web will be created
      * The resulting image will be pushed to image stream tag "rating-web:latest"
      * Every time "node:13.5-alpine" changes a new build will be triggered

--> Creating resources ...
    imagestream.image.openshift.io "node" created
    imagestream.image.openshift.io "rating-web" created
    buildconfig.build.openshift.io "rating-web" created
    deployment.apps "rating-web" created
    service "rating-web" created
--> Success
```

Entro poco tempo le dipendenze vengono compilate in un container; sono necessari circa 2 o 3 minuti per completare la compilazione prima di poter tornare alla vista **Topology** per visualizzare il servizio online.

Configurare le variabili di ambiente necessarie

Creare la variabile di ambiente `API` per la configurazione del deployment di `rating-web`. Il valore di questa variabile rappresenta il nome host/la porta del servizio `rating-api`.

Invece di impostare la variabile di ambiente tramite la web console di Azure Red Hat OpenShift, è possibile farlo tramite la CLI di OpenShift:

```
oc set env deploy rating-web API=http://rating-api:3000
```

Esporre il servizio rating-web tramite una route

Esporre il servizio significa creare un URL pubblicamente accessibile, utilizzabile dagli utenti per accedere al servizio. Se il servizio non è esposto, l'accesso è possibile esclusivamente tramite il cluster:

```
user@host: oc expose svc/rating-web
route.route.openshift.io/rating-web exposed
```

Infine, ottenere l'URL del servizio appena esposto:

```
user@host: oc get route rating-web
NAME          HOST/PORT                                                PATH    SERVICES      PORT
TERMINATION    WILDCARD
rating-web    rating-web-workshop.apps.zytjwj9a.westeurope.aroapp.io             rating-web    8080-tcp
None
```

Notare che per impostazione predefinita il **nome di dominio completo (FQDN)** è composto dal nome dell'applicazione e dal nome del progetto. La parte rimanente è il sottodominio, ovvero il sottodominio delle app specifiche del cluster Azure Red Hat OpenShift.

Provare il funzionamento del servizio

Aprire il nome host nel browser. Viene visualizzata la pagina dell'app di rating. A questo punto è possibile provare l'app, inviare alcuni voti e controllare la tabella dei punteggi.

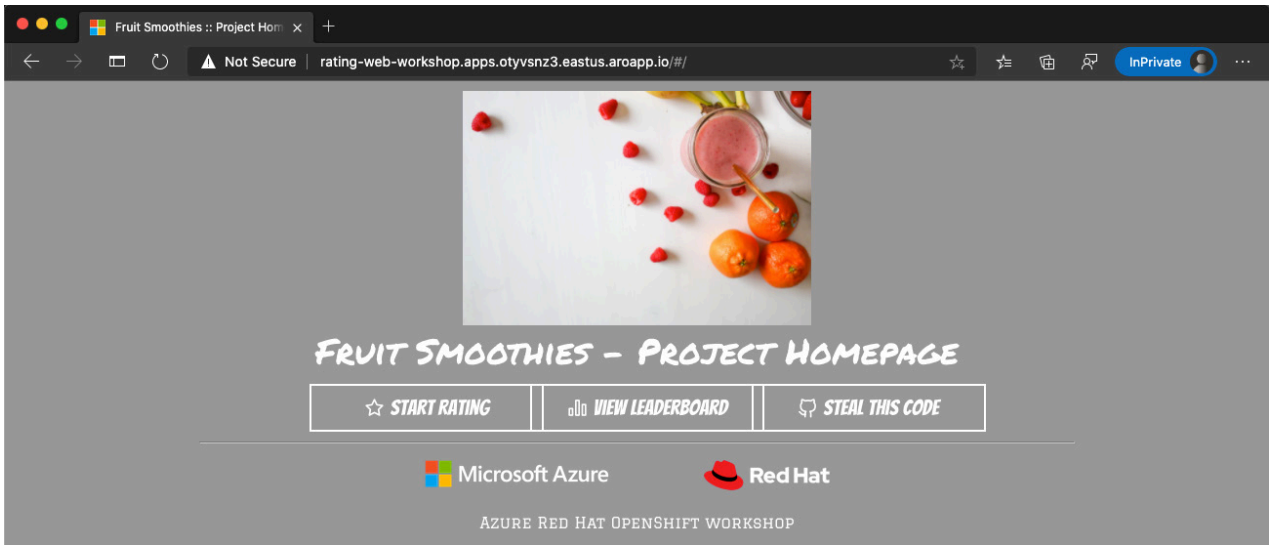


Figura 7.14: Provare il servizio dell'app di rating

Configurare il webhook di GitHub

Per attivare le build di S2I quando si invia il codice nel repository GitHub, è necessario impostare il webhook GitHub:

1. Recuperare il segreto del trigger del webhook GitHub, che verrà utilizzato nell'URL del webhook GitHub:

```
user@host: oc get bc/rating-web -o=jsonpath='{.spec.triggers..github.secret}'
3ffcc8d5-a243
```

   Annotare il codice del segreto per utilizzarlo più avanti.

2. Recuperare l'URL del trigger del webhook GitHub dalla configurazione della build:

```
user@host: oc describe bc/rating-web
...
Webhook GitHub:
        URL:    https://api.quwhfg7o.westeurope.aroapp.io:6443/apis/build.openshift.io/v1/
namespaces/workshop/buildconfigs/rating-web/webhooks/<secret>/github
...
```

3. Sostituire il segnaposto `<secret>` con il segreto recuperato nel primo passaggio. Nel nostro caso il segreto è `3ffcc8d5-a243`, quindi l'URL risultante sarà:

```
https://api.quwhfg7o.westeurope.aroapp.io:6443/apis/build.openshift.io/v1/namespaces/workshop/
buildconfigs/rating-web/webhooks/3ffcc8d5-a243/github
```

   Questo URL verrà utilizzato per configurare il webhook nel repository GitHub personale.

4. Passare al repository GitHub personale. Selezionare **Add Webhook** da **Settings** → **Webhooks**.
5. Nel campo **Payload URL** incollare l'URL GitHub modificando il valore `<secret>` con il segreto specifico appena recuperato.
6. Nel campo **Content type**, modificare l'indirizzo predefinito `application/x-www-form-urlencoded` in `application/json`.

7. Fare clic su **Add webhook**.

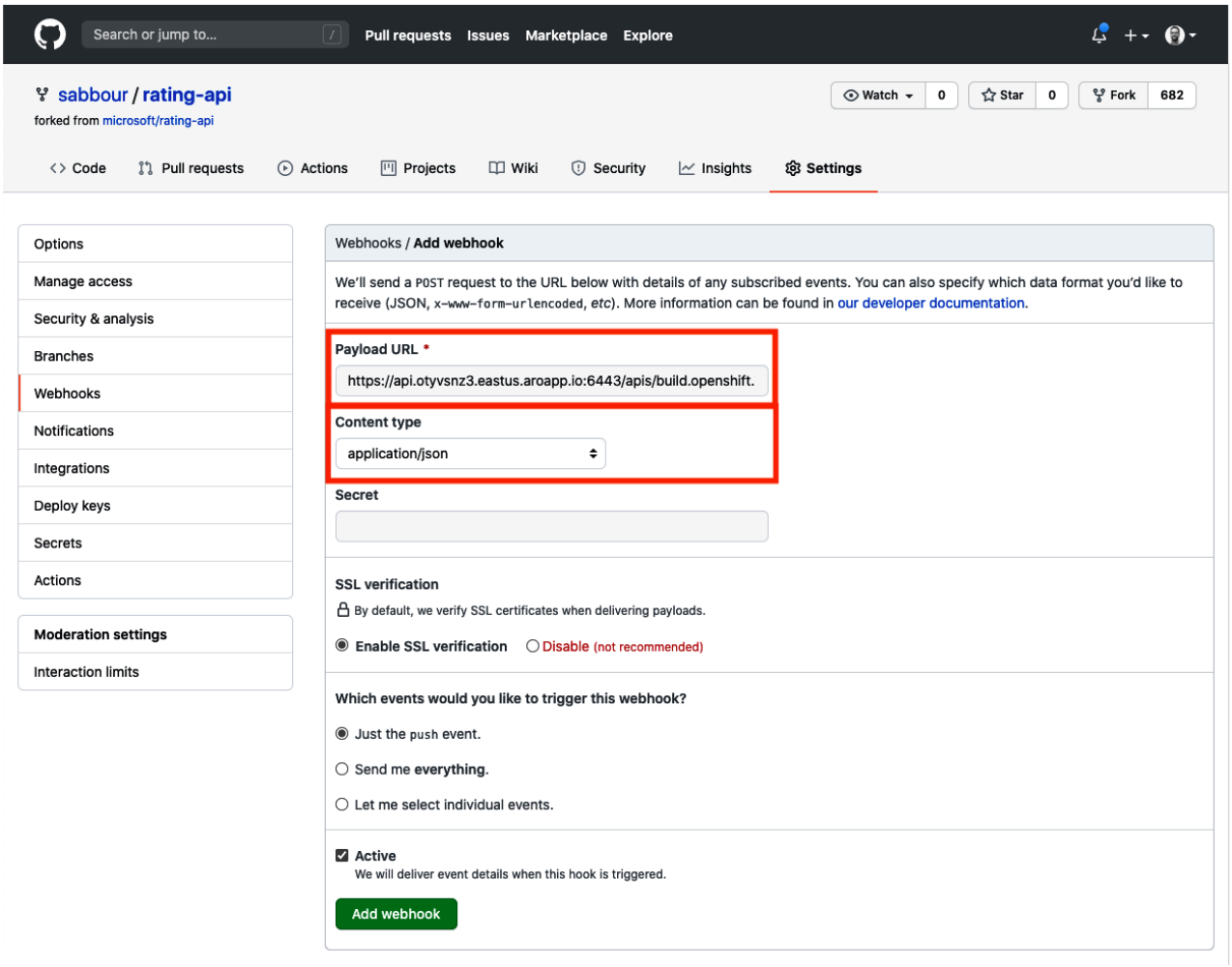


Figura 7.15: Aggiunta di un webhook

Viene visualizzato un messaggio di GitHub che conferma la riuscita della configurazione del webhook.

A questo punto, ogni volta che viene apportata una modifica al repository GitHub, viene automaticamente avviata una nuova build che, se corretta, avvia a sua volta un nuovo deployment.

Apportare una modifica all'app del sito web e visualizzare l'esecuzione dell'aggiornamento

Aprire il file `https://github.com/<nome utente GitHub>/rating-web/blob/master/src/App.vue` nel repository personale in GitHub.

Modificare il file cambiando ad esempio la riga `background-color: #999;` in `background-color: #0071c5`.

Eseguire il commit delle modifiche apportate al file nel branch `master`.

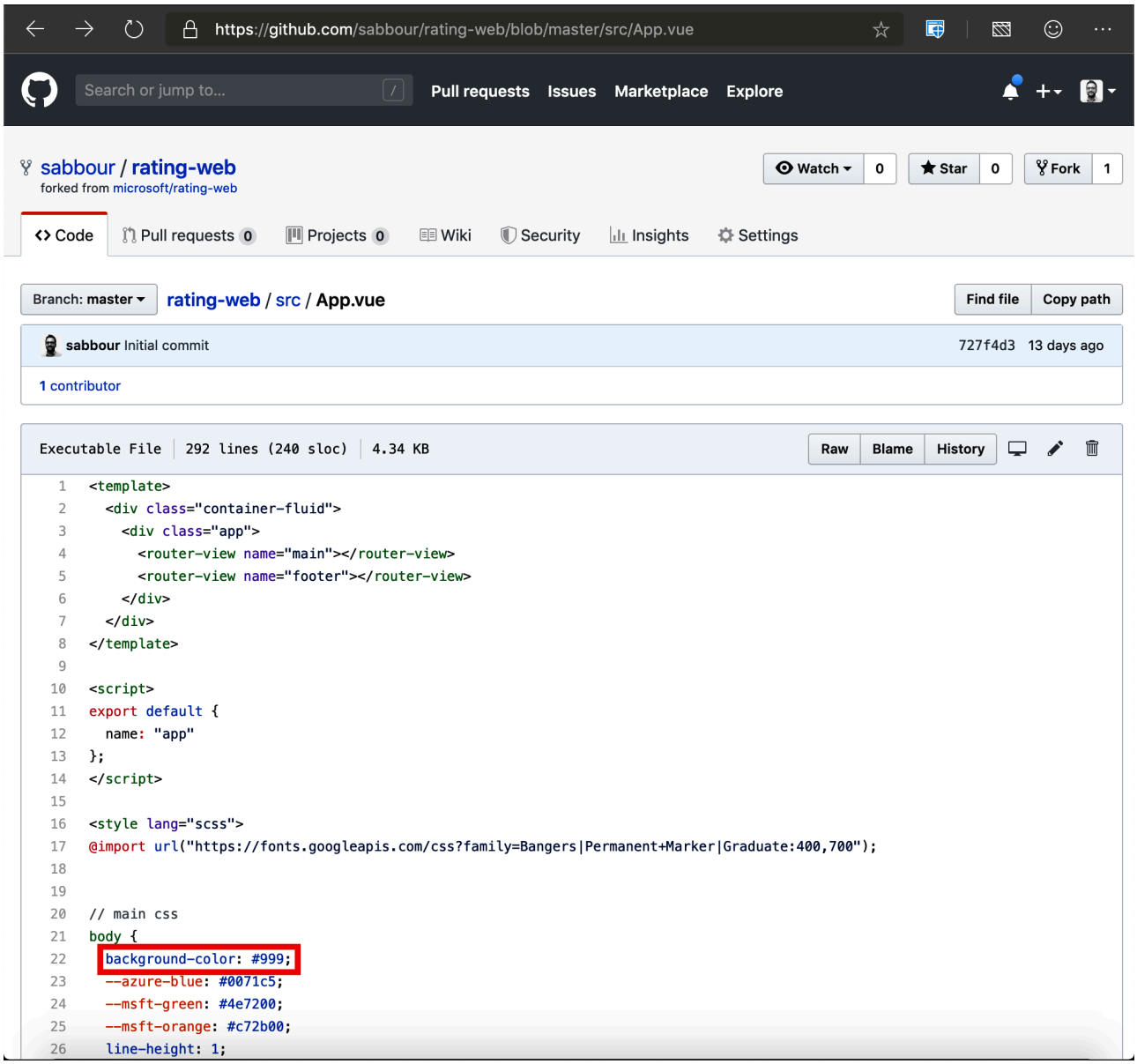


Figura 7.16: Commit delle modifiche al file nel branch master

Subito dopo passare alla scheda **Builds** nella web console di OpenShift. La nuova build, attivata dall'invio delle modifiche, appare nella coda. Completata la coda, viene attivato il deployment e sarà quindi possibile visualizzare il sito web con il colore aggiornato.

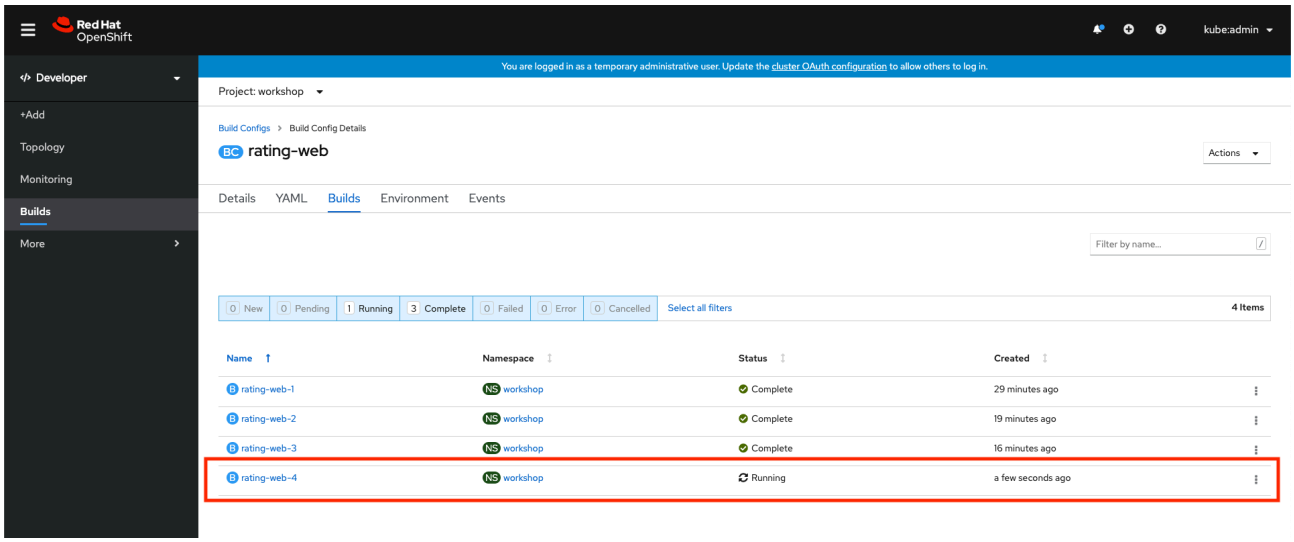


Figura 7.17: Scheda Builds con esecuzione della nuova build in corso

Tornare alla pagina `ratings-web`. Se le operazioni sono state completate correttamente, sarà visibile il nuovo colore di sfondo.

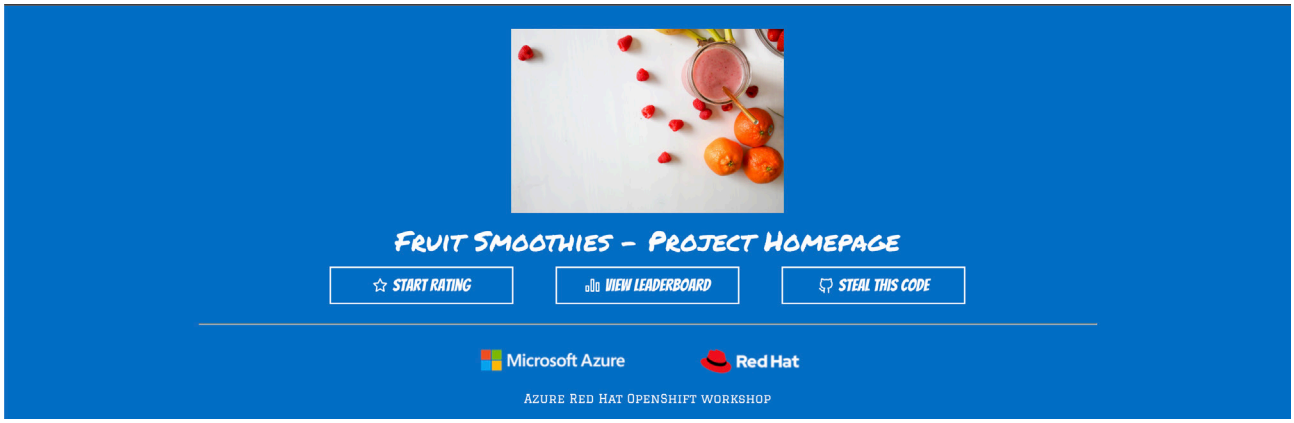


Figura 7.18: Homepage dell'app Fruit Smoothies con il nuovo colore di sfondo

Riepilogo

In questo capitolo abbiamo eseguito il deployment di una semplice applicazione formata da tre applicazioni basate su microservizi più piccole: un database di MongoDB, `ratings-api` e il codice di `ratings-web`. Benché non sia simile a un'applicazione di produzione, è un valido campionario dei concetti di cui tener conto durante la distribuzione di applicazioni in Azure Red Hat OpenShift. Le istruzioni qui delineate possono essere utilizzate come punto di partenza per la distribuzione delle tue applicazioni.

Red Hat OpenShift supporta anche il deployment di applicazioni a partire da OperatorHub, Helm Charts e sistemi di CI/CD esterni come Azure DevOps. La strategia di deployment più adatta all'organizzazione dipende dagli strumenti e dalle tecnologie che vengono utilizzate internamente.

Nel capitolo successivo verrà esplorato il valore aggiuntivo di Azure Red Hat OpenShift, che contraddistingue dalle altre questa piattaforma applicativa basata su Kubernetes. Approfondiremo le funzioni e le caratteristiche progettate per supportare le complesse esigenze delle applicazioni enterprise.

Capitolo 8

Esplorazione della piattaforma applicativa

Nei capitoli precedenti abbiamo appreso che Red Hat OpenShift fornisce numerosi servizi basati su Kubernetes. Dalla combinazione di questi servizi, che possono essere raggruppati in cinque categorie (servizi piattaforma, servizi applicativi, servizi dati, servizi per sviluppatori e servizi cluster Kubernetes) nasce un'autentica piattaforma applicativa.

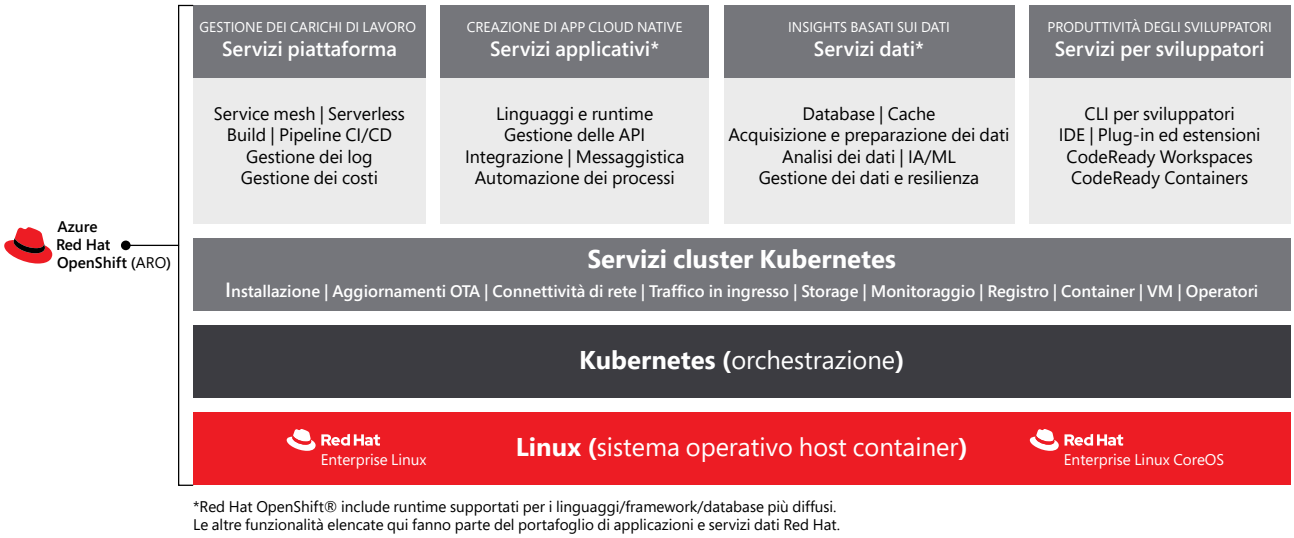


Figura 8.1: Servizi inclusi in Azure Red Hat OpenShift

I componenti raggruppati in **OpenShift Platform Plus (Multicluster Management**, **Cluster Security** e **Global Registry**) sono prodotti aggiuntivi che richiedono una sottoscrizione. Sono compatibili con Azure Red Hat OpenShift, ma non sono inclusi nell'offerta Azure Red Hat OpenShift.

Nelle seguenti sezioni, approfondiremo alcuni dei vantaggi principali offerti da questi servizi e indicheremo alcune risorse in cui reperire ulteriori informazioni.

Servizi cluster: registro dei container integrato

Red Hat OpenShift include un registro dei container integrato e disponibile immediatamente dopo la distribuzione del cluster. Tale registro viene utilizzato sia dai servizi interni del cluster, come gli operatori, sia dai container applicativi del cliente per impostazione predefinita. È un registro standard che non richiede ulteriore configurazione ed è gestito da un operatore di Infrastruttura.

Generalmente, tutti i clienti che distribuiscono un servizio di container devono distribuire un registro dei container personale e mantenere private le immagini dei container. Con OpenShift queste attività ordinarie di installazione e configurazione non sono necessarie, perché il registro dei container integrato è già disponibile con il cluster. È un esempio di funzionalità di OpenShift semplice ma in grado di far risparmiare tempo.

Panoramica del registro dei container integrato nella piattaforma OpenShift
Spesso un amministratore del cluster può scegliere di esporre questo registro dei container all'esterno del cluster stesso, in modo che anche utenti esterni a OpenShift possano inviare immagini dei container al registro. Azure Red Hat OpenShift supporta questo scenario; la relativa documentazione è reperibile nella **documentazione standard di OpenShift sull'esposizione del registro**.

Servizi piattaforma: OpenShift Pipelines

I clienti di Red Hat OpenShift hanno a disposizione diversi modi per creare le proprie applicazioni e molti strumenti diffusi di CI/CD, come Jenkins, CircleCI e GitHub Actions sono dotati di plugin che supportano OpenShift. La piattaforma OpenShift fornisce inoltre funzionalità per creare le applicazioni anche utilizzando pipeline di container cloud native, tramite l'operatore OpenShift Pipelines.

OpenShift Pipelines si basa su un progetto gestito dalla community e chiamato Tekton. Ogni fase della pipeline, come estrarre il codice da un repository Git, eseguire un compilatore Java o assemblare un pacchetto RPM viene effettuata in un container. Ciò significa che sviluppatori e operatori possono realizzare il software formando pipeline complesse e sofisticate che sfruttano tutti i vantaggi dei container.

È possibile installare OpenShift Pipelines tramite OperatorHub. Passare a **OperatorHub** e selezionare l'operatore per avviare l'installazione. Non è necessaria alcuna configurazione e l'installazione dell'operatore viene completata in genere in meno di un minuto.

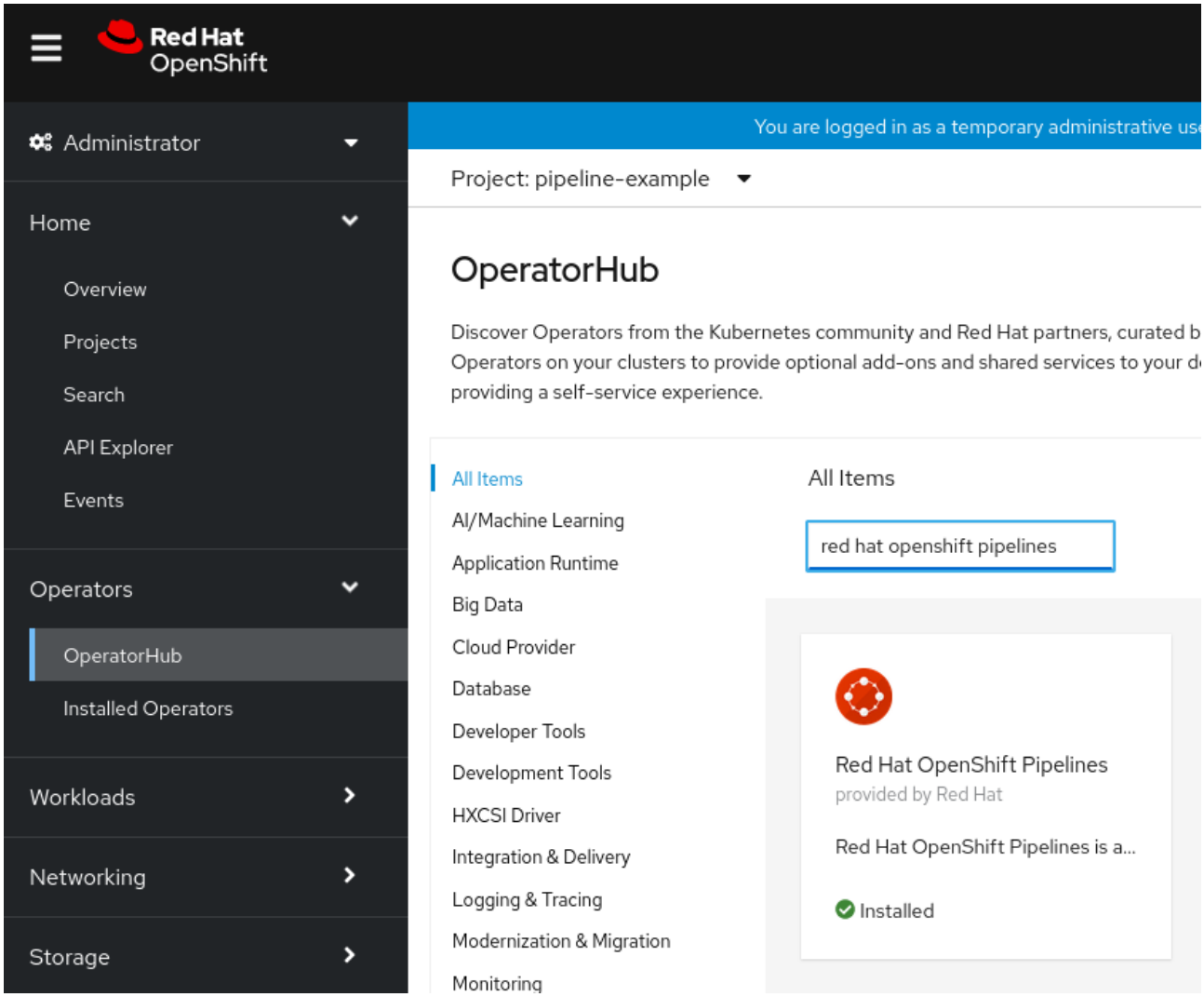


Figura 8.2: Installazione di OpenShift Pipelines tramite OperatorHub

Dopo aver installato l'operatore OpenShift Pipelines, nella barra laterale sarà visibile una nuova sezione **Pipelines**, insieme all'opzione che consente di aggiungere le pipeline agli elementi del catalogo, come durante la compilazione dell'esempio Node.js.

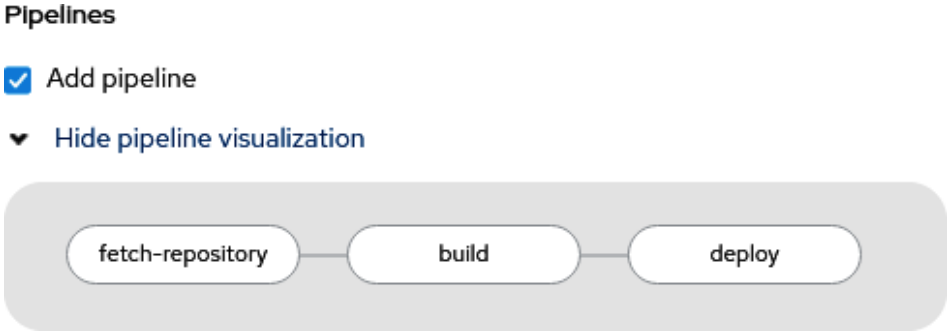


Figura 8.3: Aggiunta di pipeline

OpenShift Pipelines consente di utilizzare il builder grafico per impostare e creare anche pipeline complesse e con diramazioni. La seguente schermata è un esempio di pipeline complessa:

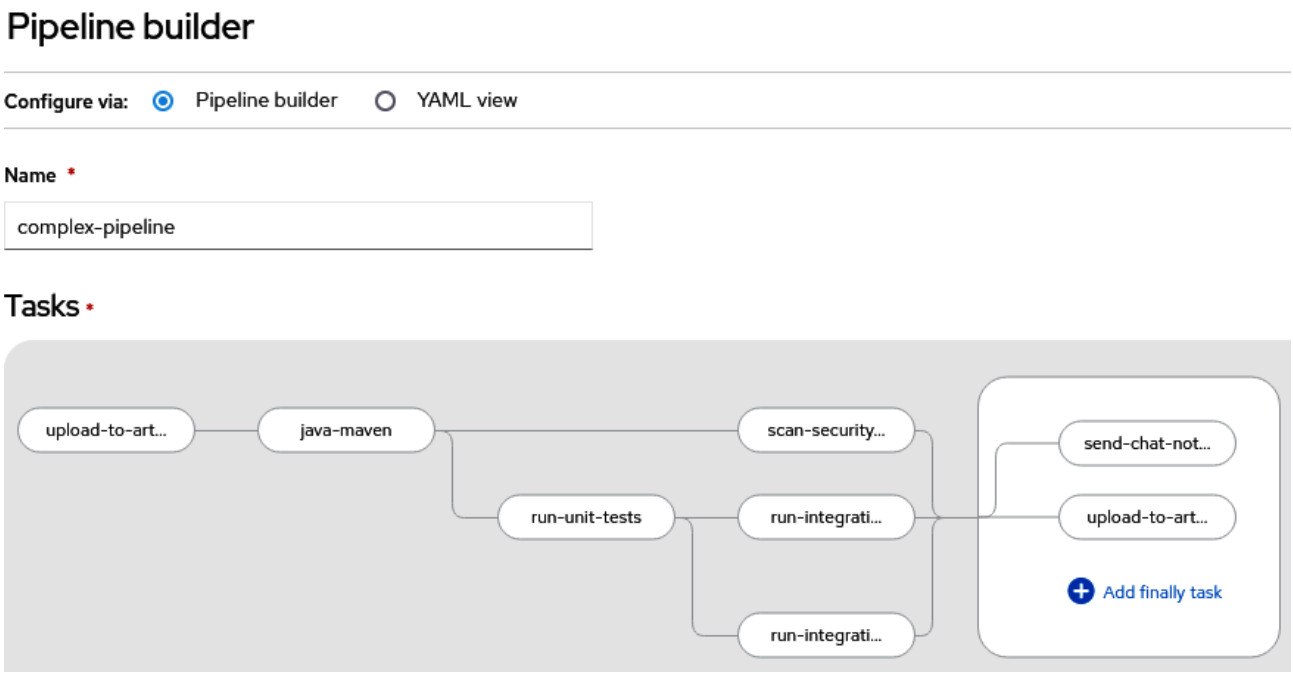


Figura 8.4: Pipeline complessa

Molte organizzazioni usano strumenti e tecnologie per realizzare il proprio software. OpenShift Pipelines semplifica l'integrazione della build direttamente in OpenShift, garantendo tutti i vantaggi derivanti dai container. È una soluzione CI/CD uniforme e di facile utilizzo, estremamente semplice da configurare qualunque sia l'infrastruttura sottostante impiegata.

Approfondimenti

- Funzionamento di OpenShift Pipelines in OpenShift
- Sito della community Tekton

Servizi piattaforma: OpenShift Serverless

È un preconcetto comune ritenere che i container non siano altro che una tecnologia utile per eseguire servizi a lunga durata. In realtà, molti processi brevi e funzioni serverless si avvalgono di container a breve durata. Offrendo inoltre vantaggi in termini di velocità di avvio, coerenza e facilità di arresto, si prestano bene anche ai carichi di lavoro serverless. Naturalmente tutti i servizi serverless necessitano di server sottostanti per eseguire il codice. Per questa ragione "serverless" spesso viene utilizzato come sinonimo di "function-as-a-service", ovvero funzione come servizio.

Red Hat OpenShift abilita i carichi di lavoro serverless, o function-as-a-service, tramite l'operatore OpenShift Serverless, che si basa sul noto progetto open source chiamato Knative.

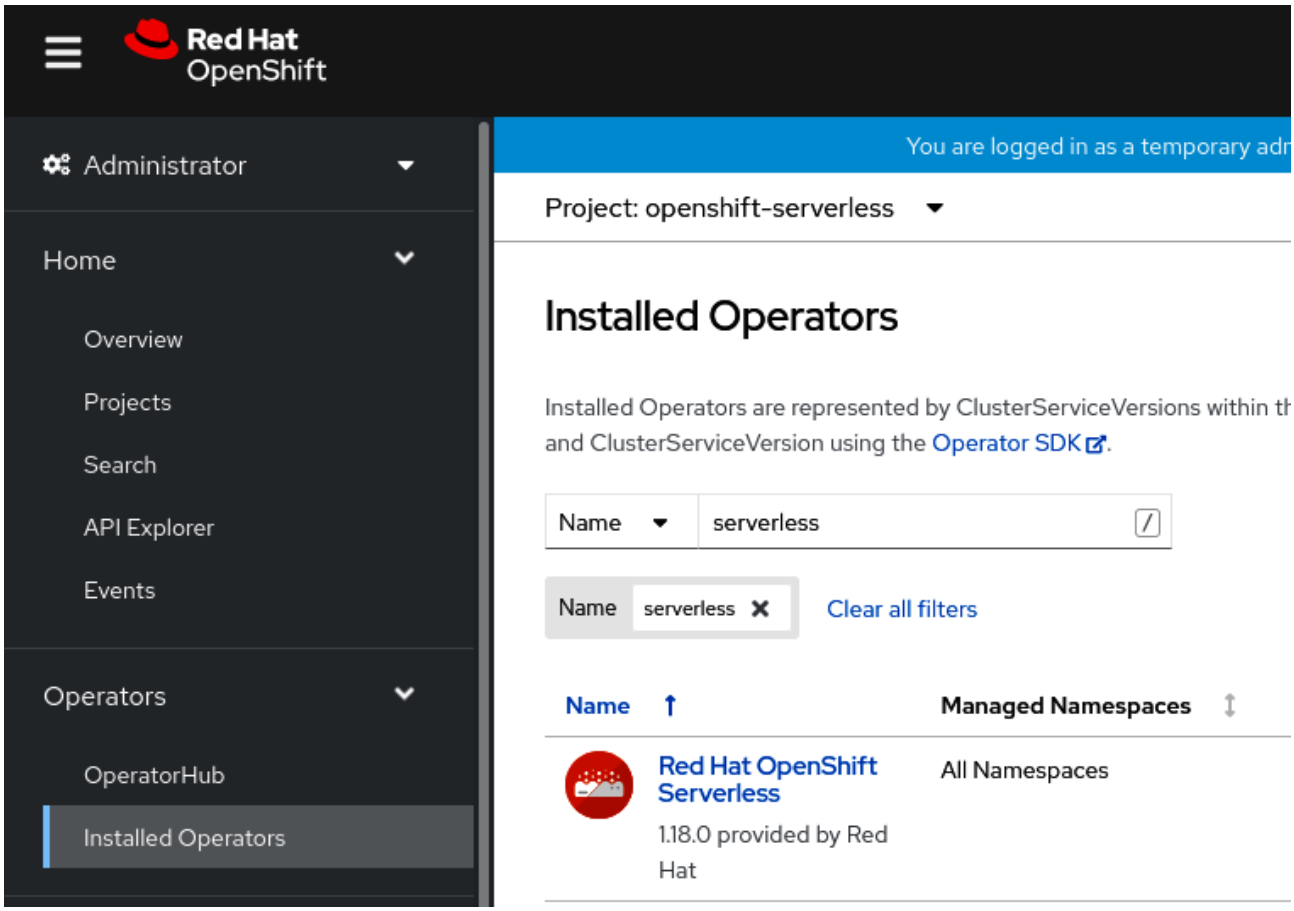


Figura 8.5: Vista del menu Installed Operators

Dopo il deployment nel cluster (attività che in genere richiede uno o due minuti), è necessario procedere con la configurazione dell'operatore. Occorre fare attenzione in particolare a due **definizioni di risorsa personalizzata (CRD)**: **Knative Serving** e **Knative Eventing**.

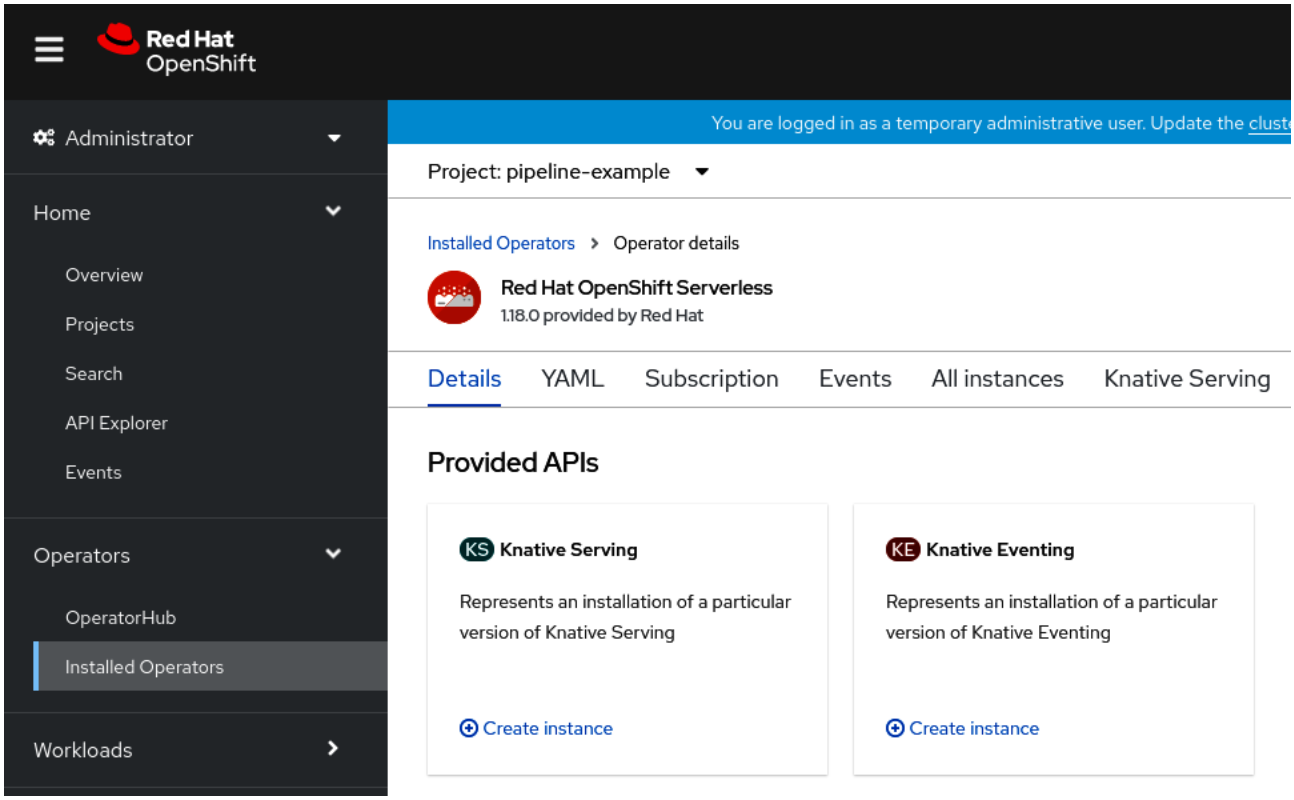


Figura 8.6: CDR Knative Serving e Knative Eventing nel menu Operators

- **Knative Serving** semplifica il deployment dell'applicazione, applica la scalabilità dinamica in base al traffico in ingresso e supporta strategie di rollout personalizzate con suddivisione del traffico, ad esempio una funzione che carica un'immagine in un bucket di storage e che registra tale caricamento in un database NoSQL.
- **Knative Eventing** consente l'associazione tardiva delle sorgenti degli eventi durante il runtime dell'applicazione e non durante la creazione della build, come nel caso di un'applicazione che risponde ai caricamenti di nuove immagini in un bucket di storage. Questa applicazione non deve conoscere il bucket di storage nel momento della compilazione o del deployment dell'evento, ma Knative Eventing semplifica l'associazione della sorgente dell'evento all'applicazione in fase di runtime.

Le due sezioni successive mostrano esempi di entrambi i tipi di applicazione serverless in OpenShift.

Servizi piattaforma: OpenShift Serverless, esempio di Knative Serving

Dimostreremo come Knative Serving rende possibile la scalabilità automatica di un'applicazione, in questo caso specifico la scalabilità a zero quando non ci sono richieste. Utilizzeremo un'applicazione molto semplice, una pagina web che fornisce immagini ASCII di "pet" ovvero cuccioli:

- php-ascii-pets GitHub repository

L'aggiunta di questo repository da Git è molto facile. Red Hat OpenShift rileva automaticamente un'immagine del builder compatibile di PHP.

OpenShift rileva automaticamente come creare questo progetto.

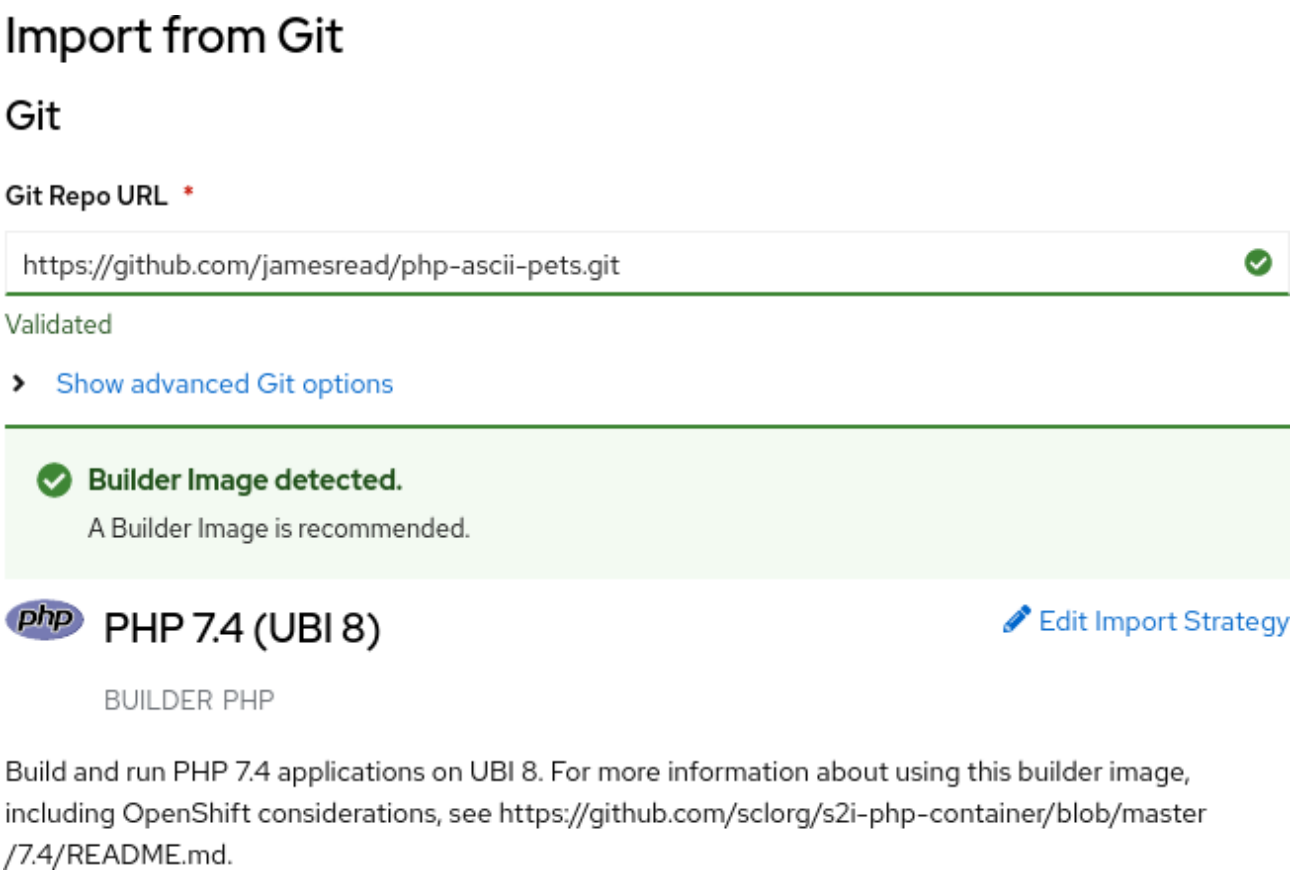


Figura 8.7: Rilevazione e suggerimento automatico dell'immagine del builder più adatta

La parte importante che rende questo deployment "serverless" è la selezione del tipo di deployment. Si tenga presente che nel momento in cui viene installato l'operatore OpenShift Serverless diventa disponibile un deployment "serverless".

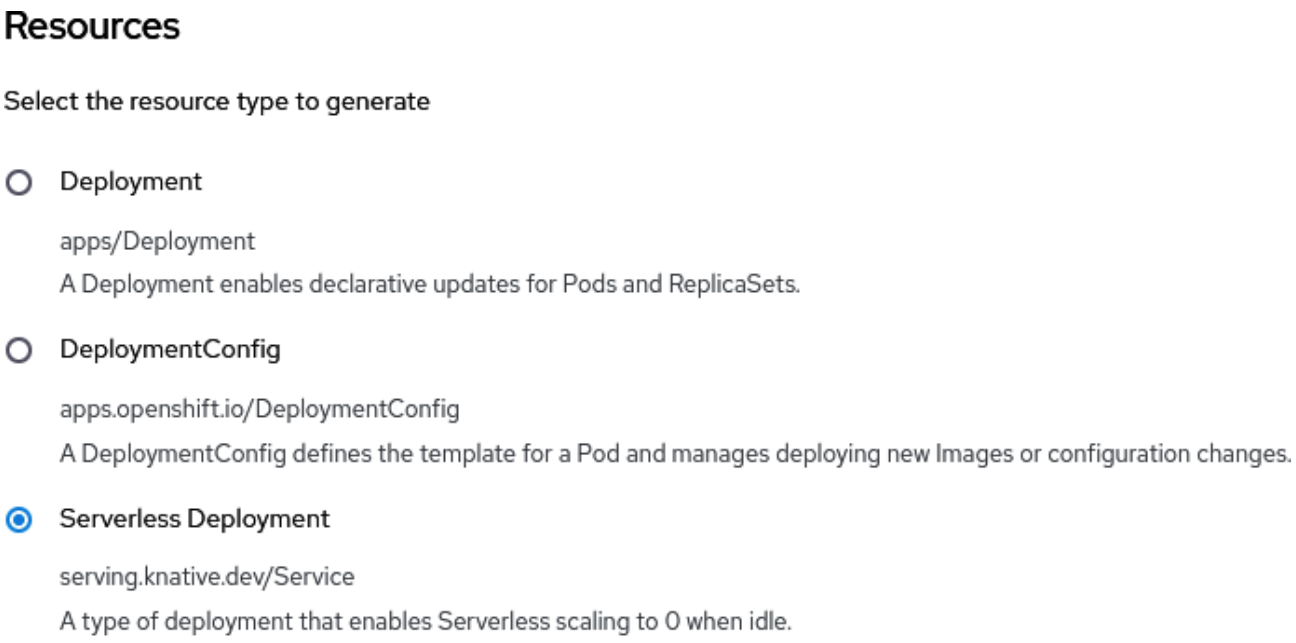


Figura 8.8: Tipo di deployment serverless

Il completamento della prima build richiede poco tempo. Una volta completata la build, viene distribuito un pod. La vista della topologia sarà simile alla seguente:

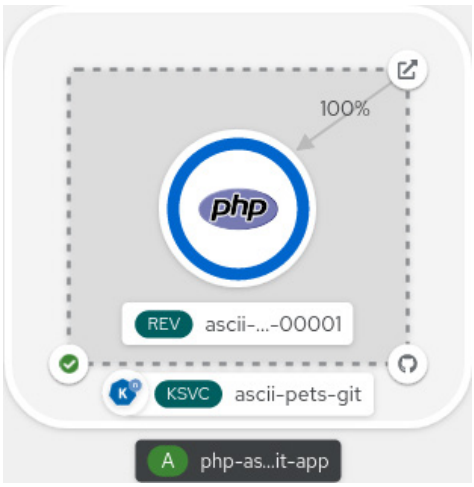


Figura 8.9: Un'app Knative Serving

La pagina dell'applicazione sarà simile a quella della *Figura 8.9*. Si tratta di un'applicazione molto semplice, ma il "pet", ovvero l'immagine ASCII del cucciolo, è vincolato al nome host del pod. In questa applicazione di esempio, quando aggiungiamo un ulteriore carico Knative Serving genererà pod aggiuntivi e sarà possibile vedere i numerosi "pet" presentati.

Se, tuttavia, l'applicazione rimane per un minuto senza richieste, Knative Serving ridimensiona l'applicazione a zero. La vista della topologia mostra che l'applicazione non è più in esecuzione.

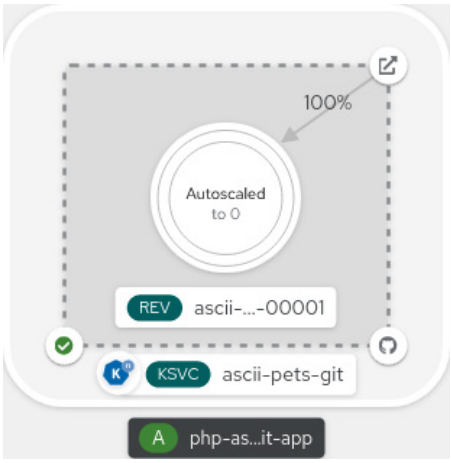


Figura 8.10: Deployment serverless eseguito con Knative Serving, con un'applicazione ridimensionata a zero

Infine, se un utente visita la pagina, l'applicazione verrà nuovamente ridimensionata a 1, 2 o 3 repliche, a seconda del numero di istanze ritenute necessarie da OpenShift per soddisfare la domanda.

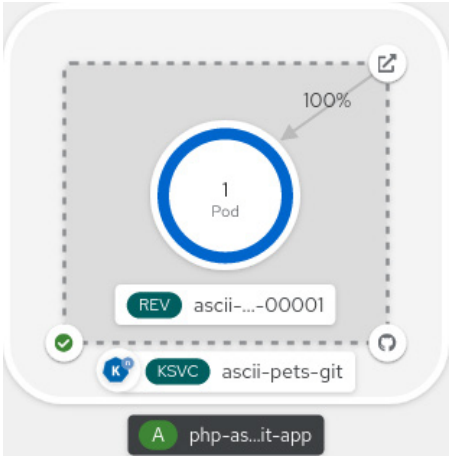


Figura 8.11: Scalabilità automatica in risposta al traffico

Red Hat OpenShift Serverless con Knative Serving consente alle organizzazioni di attivare la scalabilità orizzontale e verticale in modo dinamico, senza complesse configurazioni aggiuntive e senza modifiche alle applicazioni. L'operatore è molto utile per mantenere i deployment entro le dimensioni corrette ed evitare consumi e addebiti eccessivi delle risorse.

Approfondimenti

- Introduzione a OpenShift Serverless
- learn.openshift.com, che include un corso su OpenShift Serverless
- Sito della community Knative

Servizi piattaforma: OpenShift Serverless, esempio di Knative Eventing

A partire dall'applicazione di esempio e dai concetti di base del capitolo precedente, OpenShift Serverless può avviare la scalabilità di un'applicazione in base a metriche che non si limitano al solo traffico in ingresso. Soprattutto negli scenari con un'architettura basata sugli eventi, è utile poter disporre di una scalabilità verticale delle istanze di un'applicazione per elaborare eventi da una coda di messaggi o attivarli in base a un timer.

Utilizzando lo stesso deployment, la console OpenShift consente di configurare facilmente diverse sorgenti di eventi.

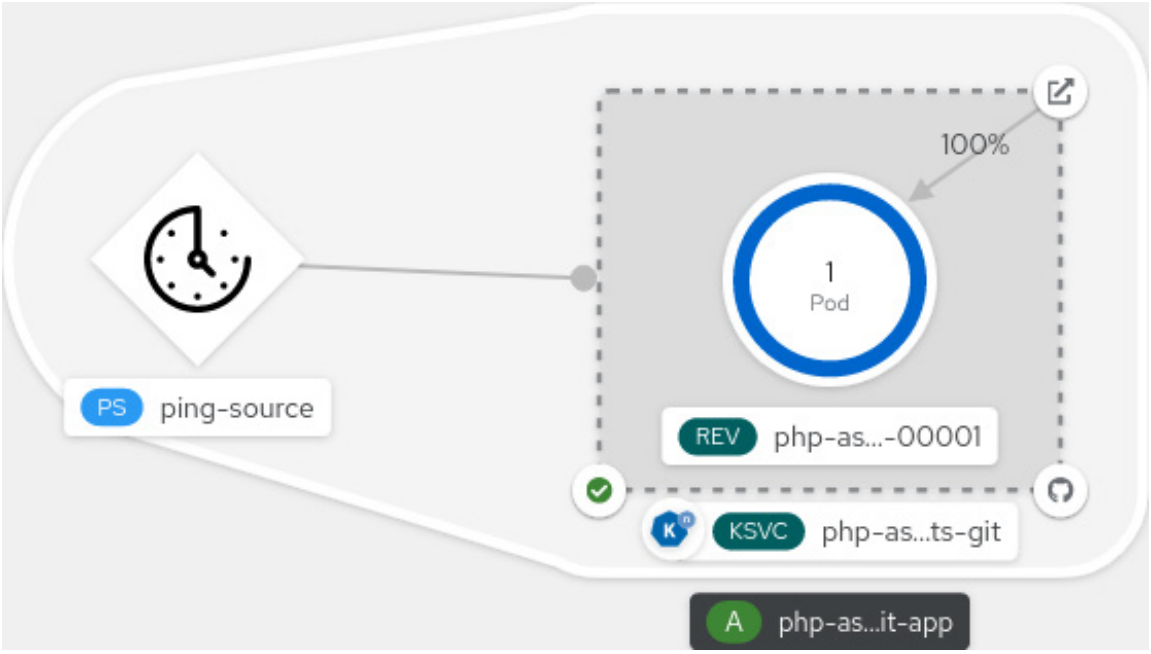


Figura 8.12: Sorgente di un evento "ping" che esegue il ping dell'applicazione in base a un timer

Uno scenario di utilizzo per una sorgente di evento ping è l'"attivazione" di un container di un processo di backup ogni notte a mezzanotte. Un altro esempio di utilizzo del timer del ping è il monitoraggio di un container che deve essere eseguito periodicamente.

Approfondimenti

- OpenShift Serverless Eventing explained in 5 minutes
- Introduzione a OpenShift Serverless

Servizi piattaforma: OpenShift Service Mesh

Grazie al progetto open source Istio, Red Hat OpenShift Service Mesh aggiunge alle applicazioni distribuite un livello trasparente, senza richiedere alcuna modifica al codice dei servizi. Per aggiungere il supporto Red Hat OpenShift Service Mesh ai servizi, è necessario distribuire uno speciale proxy sidecar nell'ambiente, che intercetterà tutte le comunicazioni di rete tra i microservizi. La service mesh viene configurata e gestita tramite le funzioni del piano di controllo.

Tra gli scenari di utilizzo che è possibile abilitare con OpenShift Service Mesh segnaliamo:

- Crittografia trasparente nella comunicazione tra servizi, con mTLS automatico.
- Fornitura di più versioni di un servizio, e abilitazione dei test A/B, ad esempio.
- Visibilità sulla modalità di comunicazione tra i servizi, con Kiali.
- Tracciabilità delle chiamate da servizio a servizio, con Jaeger.

Come tutte le altre funzioni per piattaforma di OpenShift, Service Mesh viene installata con un operatore Red Hat.

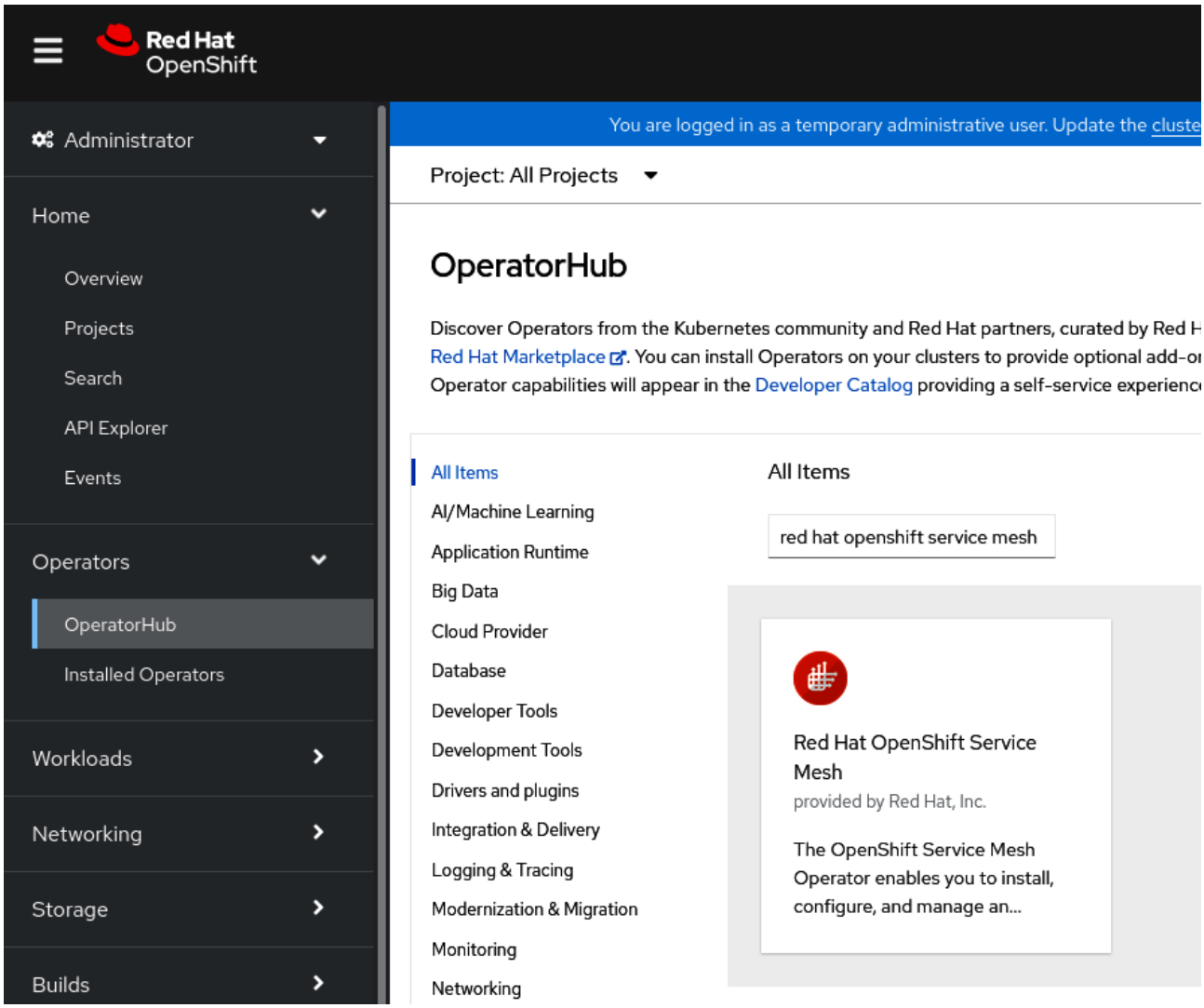


Figura 8.13: Installazione di Service Mesh tramite OperatorHub

Nella documentazione di OpenShift Service Mesh è inclusa un'eccellente applicazione di esempio chiamata BookInfo, un'app molto semplice che emula un negozio di libri ed è eseguita su OpenShift.

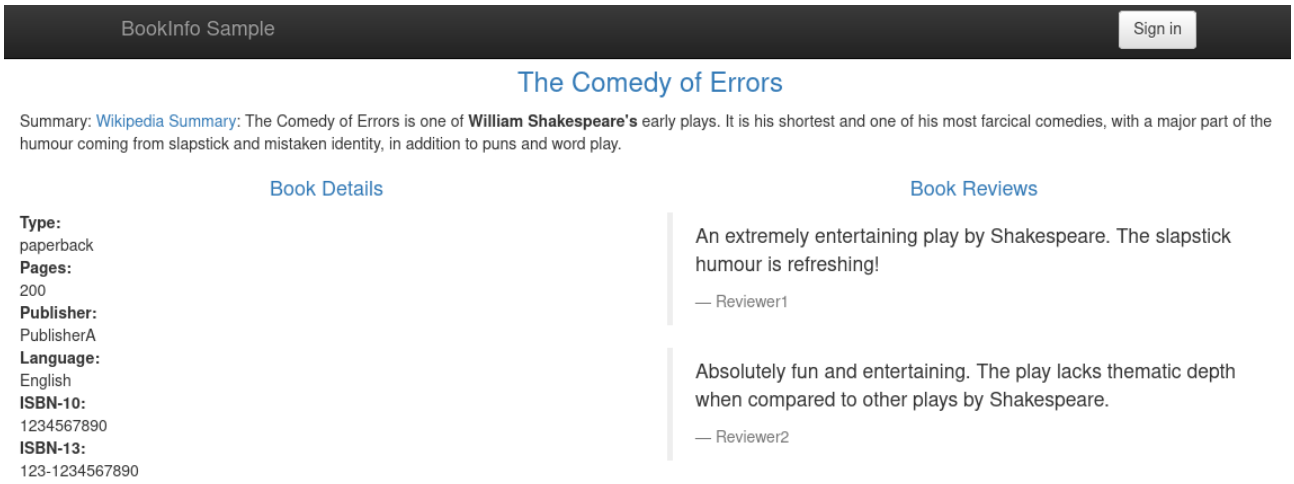


Figura 8.14: Applicazione BookInfo

Per fare in modo che l'applicazione BookInfo funzioni con la Service Mesh, è necessario creare un piano di controllo della Service Mesh. L'attività è facile da eseguire tramite l'interfaccia grafica di OpenShift, mostrata nella *Figura 8.15*:

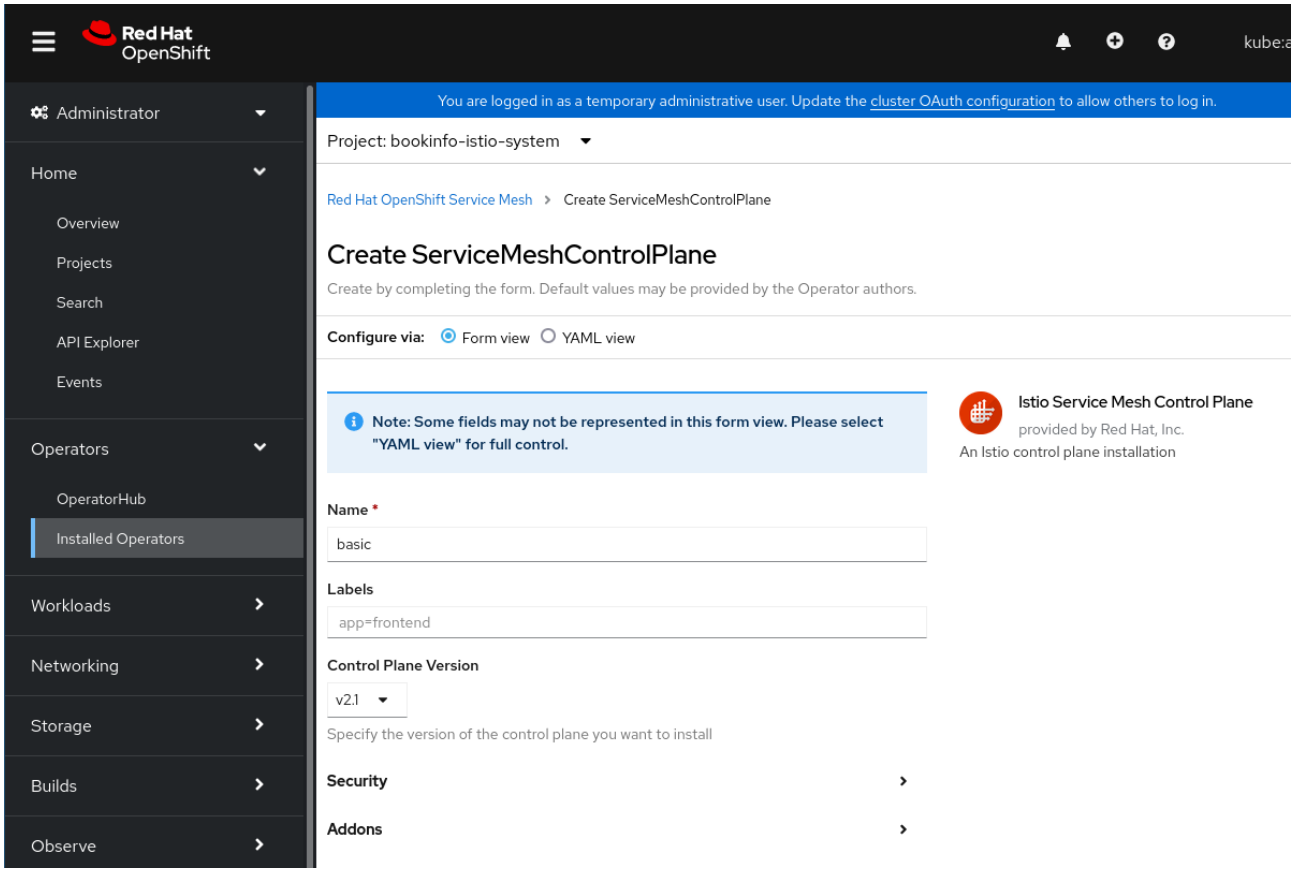


Figura 8.15: Configurazione del piano di controllo nel progetto bookinfo-istio-system di BookInfo

Il progetto BookInfo accetta il traffico in ingresso tramite il piano di controllo e, in questo caso, è stato creato un progetto distinto per ospitare tale piano, chiamato `bookinfo-istio-system`.

Non è necessario separare il piano di controllo di Service Mesh e il progetto, ed è anche possibile condividere un piano di controllo di Service Mesh con più progetti.

Nelle due sezioni seguenti, analizzeremo in dettaglio due scenari di utilizzo di Service Mesh, uno per l'osservabilità e uno per la tracciabilità distribuita.

Servizi piattaforma: OpenShift Service Mesh, osservabilità con Kiali

Se osserviamo i pod che costituiscono questa applicazione nella vista Topology di Red Hat OpenShift, possiamo vedere che è composta da sei microservizi.

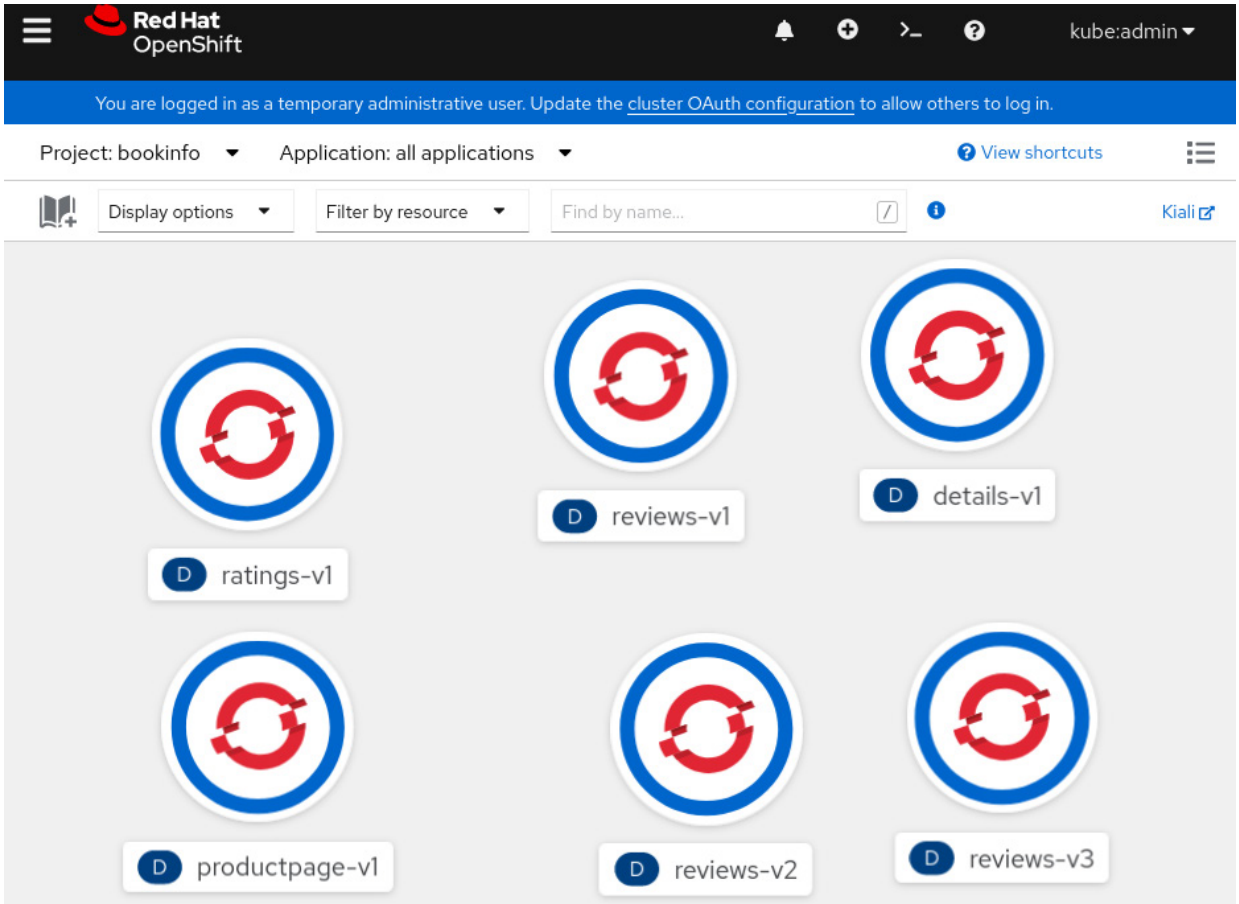


Figura 8.16: I sei servizi che compongono l'applicazione BookInfo

Sebbene questa vista sia utile, non spiega in modo esaustivo il modo in cui i servizi sono connessi. Iniziamo con il primo vantaggio fornito da Service Mesh, l'osservabilità. Se apriamo una console leggermente differente, chiamata Kiali e inclusa in Service Mesh, possiamo visualizzare una rappresentazione dell'architettura degli stessi sei servizi più accurata.

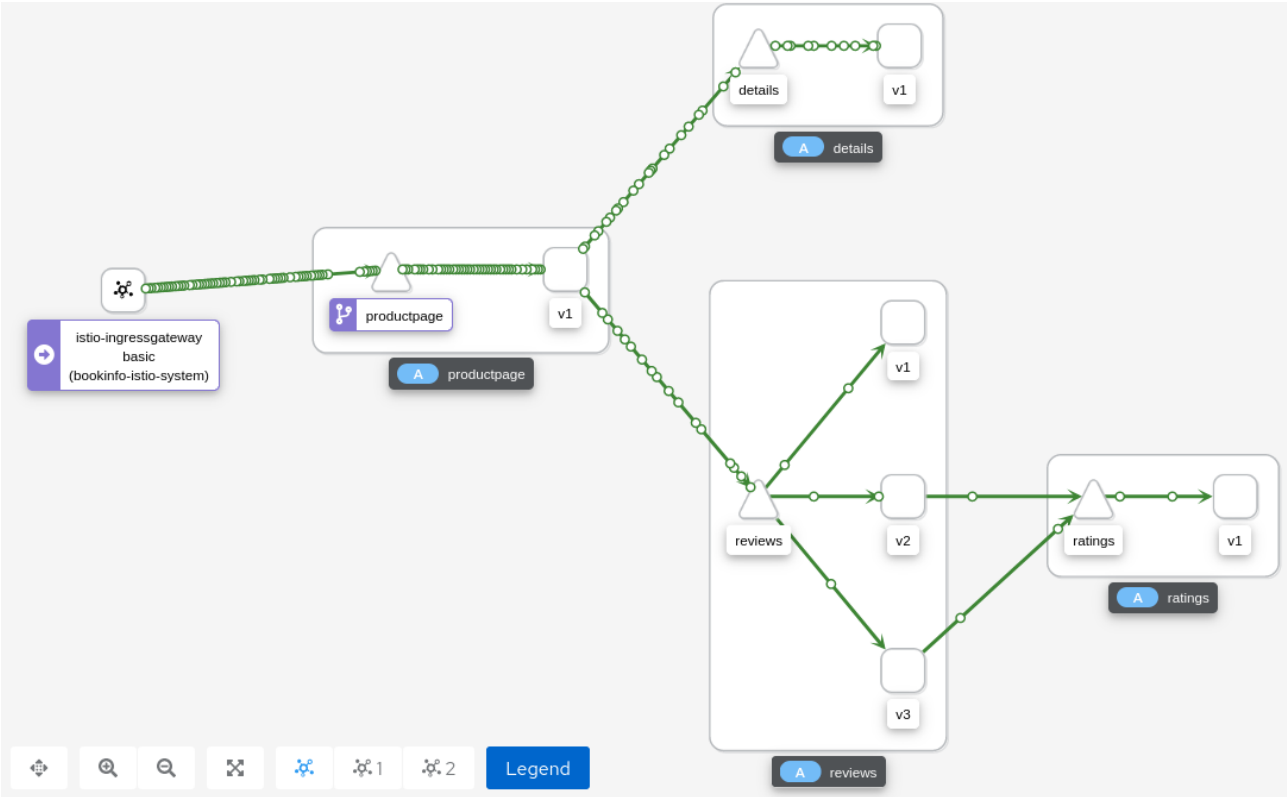


Figura 8.17: Grafico dell'architettura dell'applicazione generato automaticamente da Service Mesh

La vista mostra la reale connessione tra i servizi ma anche uno schema del traffico in tempo reale. In questo caso, all'applicazione arriva una quantità di traffico dimensionabile; ogni richiesta è rappresentata dall'animazione di un cerchio al momento della connessione.

Per sfruttare al massimo l'osservabilità, un amministratore può fare clic su una delle connessioni del servizio e visualizzare il profilo del traffico. Nel nostro caso, il traffico è principalmente nello stato **HTTP OK**.

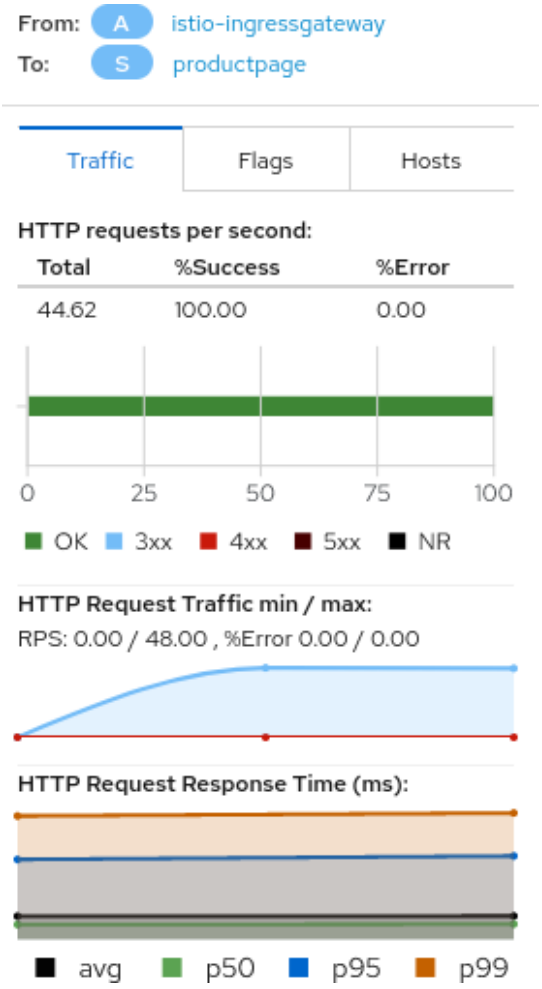


Figura 8.18: Vari stati HTTP

Proviamo a introdurre un errore artificiale nell'applicazione, arrestando il microservizio `details` e forzando la scalabilità a 0 repliche.

Figura 8.19: Zero repliche del servizio details, per simulare un errore

Tornando alla vista Kiali osserviamo l'aggiunta di alcuni componenti al diagramma, ma soprattutto che la connessione al servizio `details` è evidenziata in rosso per indicare l'errore.

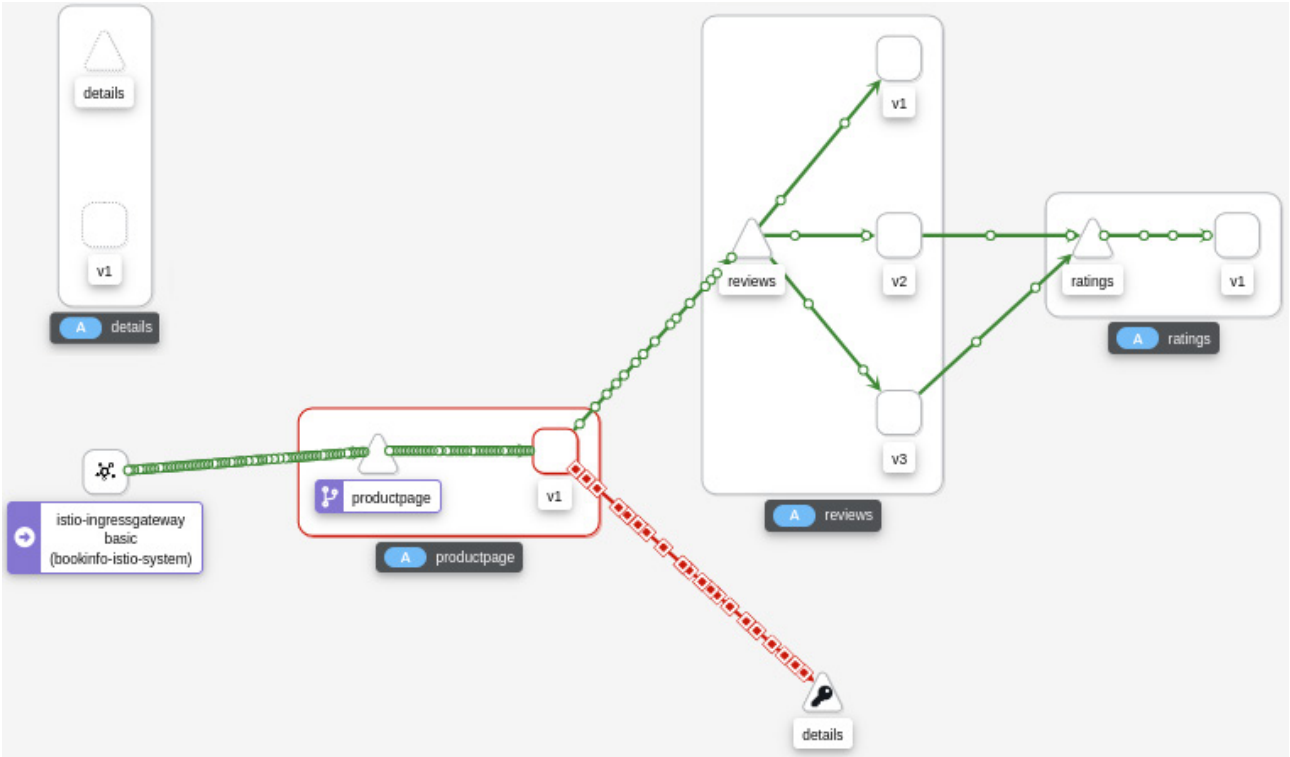


Figura 8.20: Connessione in rosso che indica un problema con il servizio

Abbiamo appreso quanto sia utile Kiali di Service Mesh per quanto riguarda l'osservabilità. In applicazioni di piccole dimensioni come BookInfo, Service Mesh si rivela utile, ma con applicazioni più grandi e complesse, che prevedono 20 o più microservizi e diversi tipi di interazione, strumenti come Service Mesh e Kiali diventano straordinariamente preziosi e anche imprescindibili.

Servizi piattaforma: Red Hat OpenShift Service Mesh, tracciabilità distribuita con Jaeger

Jaeger è un altro servizio molto valido fornito da Service Mesh, che abilita la tracciabilità distribuita per le applicazioni di microservizi complessi. Nella sezione precedente abbiamo osservato che quando il servizio `details` viene arrestato, le richieste generano un errore.

Nel nostro caso, il motivo dell'errore era semplice da identificare: il servizio `details` non era disponibile e generava l'errore. Se la causa fosse stata meno identificabile e avesse richiesto ulteriori indagini, la tracciabilità distribuita di Jaeger sarebbe stata di grande aiuto.

Jaeger traccia le richieste a partire dal primo servizio e nelle connessioni successive del sistema. Benché per abilitare Jaeger sia necessario del codice applicativo aggiuntivo, il lavoro degli sviluppatori è facilitato dalle librerie client e da lievi modifiche al codice.

Se osserviamo il servizio `productpage` (scritto in Python), possiamo scegliere il codice a noi necessario per abilitare la tracciabilità distribuita di Jaeger in tale servizio. Questo codice importa la libreria client di Jaeger nel servizio `productpage`:

```
from jaeger_client import Tracer, ConstSampler
from jaeger_client.reporter import NullReporter
from jaeger_client.codecs import B3Codec
```

Una volta importata la libreria client, sarà necessario configurare un nuovo tracer nella pagina del prodotto. Questo codice inizializza un `Tracer` Jaeger nel servizio `productpage`:

```
tracer = Tracer(
    one_span_per_rpc=True,
    service_name='productpage',
    reporter=NullReporter(),
    sampler=ConstSampler(decision=True),
    extra_codecs={Format.HTTP_HEADERS: B3Codec()}
)
```

Infine, sempre nel servizio `productpage`, indicheremo a Jaeger che intendiamo creare un nuovo span, ovvero una richiesta per più servizi:

```
span = tracer.start_span(
    operation_name='op', child_of=span_ctx, tags=rpc_tag
)
```

Jaeger traccerà quindi il tracer su numerosi servizi, i quali dovranno essere adattati per poter operare con Jaeger; in ogni caso, sono disponibili librerie client per i linguaggi di programmazione più diffusi.

L'intero codice sorgente per il servizio `productpage` è reperibile in **GitHub**.

Per rendere funzionale il codice sono possibili due opzioni: utilizzare le librerie client di Jaeger, che ora sono considerate deprecate o utilizzare le opzioni basate sullo standard open del progetto OpenTelemetry; quest'ultima scelta è quella preferibile:

- **Librerie di OpenTelemetry**

Al termine di questa attività l'applicazione sarà completa e le tracce saranno simili alle seguenti:

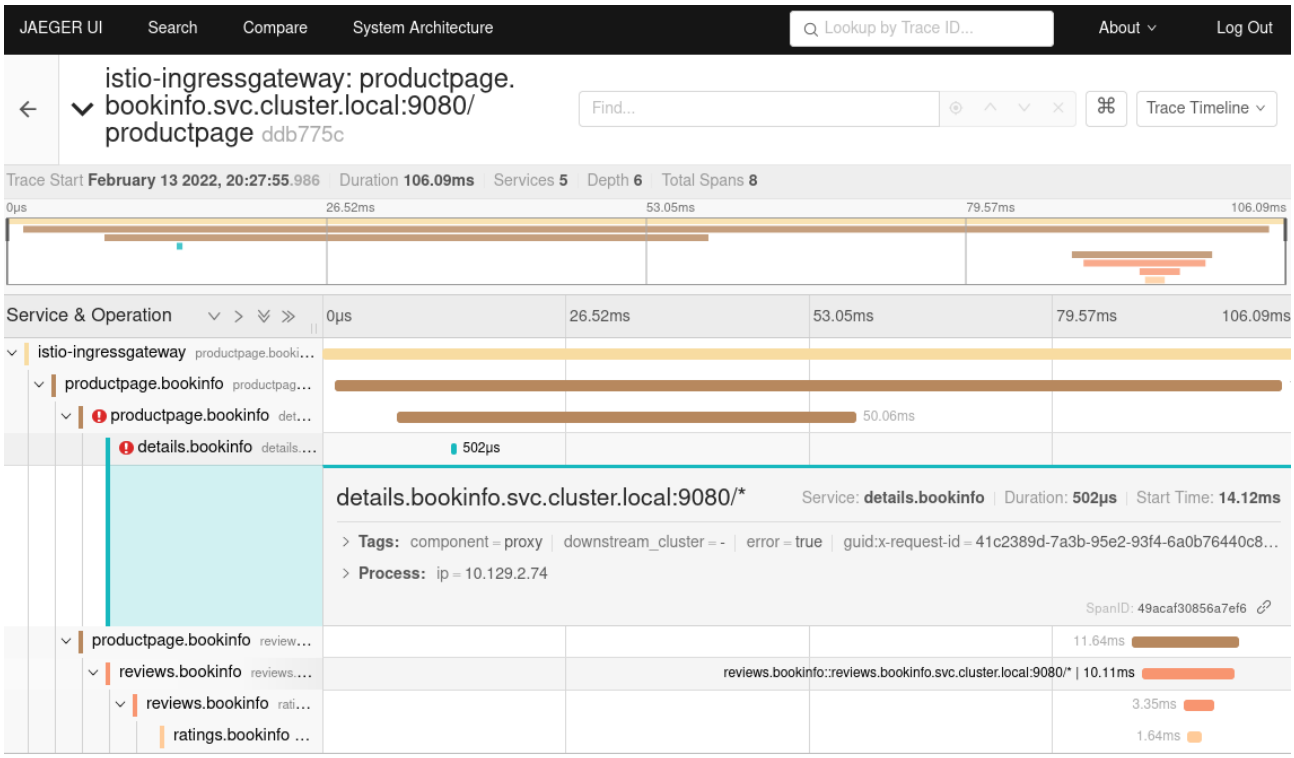


Figura 8.21: Una traccia di chiamata

La traccia di chiamata visualizzata nella vista è simile a quella visualizzata in Kiali, ma qui è mostrata con uno schema gerarchico. È possibile espandere ognuna delle righe visualizzate per ottenere informazioni sulla durata della richiesta, l'ora di avvio, l'indirizzo IP dell'origine e diversi altri dettagli. Questo livello di informazione diventa fondamentale in presenza di applicazioni basate su microservizi complessi.

Se torniamo all'errore che abbiamo tentato di diagnosticare, la vista della traccia evidenzia chiaramente che si stanno verificando problemi sul microservizio `details`. L'errore in questo caso è semplice, ma espandendo la vista emerge che il servizio sta emettendo i codici di errore `HTTP 503`.

Figura 8.22: Dettagli dell'errore

HTTP 503 è il codice di errore standard che indica un "servizio non disponibile". In questo caso, il motivo è che il servizio è stato scalato a zero repliche. Riattivando la scalabilità del servizio, ne verrà ripristinato il funzionamento.

Insieme, Jaeger e Kiali sono strumenti potenti e utili quando le applicazioni basate su microservizi acquisiscono più complessità. Vengono forniti come parte del servizio Azure Red Hat OpenShift e non implicano ulteriori spese o sottoscrizioni.

Riepilogo

In questo capitolo sono stati illustrati alcuni tra i principali servizi a valore aggiunto offerti dalla piattaforma applicativa Red Hat OpenShift, che contrastano con lo scarno ambiente Kubernetes. Sebbene sia possibile replicare un livello di funzionalità simile a quello di OpenShift installando tutti i rispettivi progetti upstream open source su OpenShift, l'integrazione, il ciclo di vita e il supporto costituiscono la complessità erogata con Azure Red Hat OpenShift.

In definitiva, i vari servizi piattaforma, applicativi, dati, per sviluppatori e cluster rendono gli sviluppatori e gli operatori più produttivi quando lavorano con Kubernetes e quando distribuiscono più applicazioni enterprise complesse.

Capitolo 9

Integrazione con altri servizi

È normale che un'applicazione non possa esistere esclusivamente in Red Hat OpenShift, in quanto nella maggior parte dei casi dipende da database esterni o servizi di Azure di qualche tipo.

Azure Red Hat OpenShift non interrompe alcun tipo di connettività. Se, ad esempio, un'applicazione si basa su Azure CosmosDB, sarà comunque in grado di connettersi da Azure Red Hat OpenShift ad Azure CosmosDB senza che sia necessario modificarla. In base all'applicazione e all'organizzazione, è possibile distribuire alcune di queste dipendenze esterne separatamente da Azure Red Hat OpenShift o nello stesso momento.

Se si ritiene vantaggioso distribuire queste dipendenze esterne con l'applicazione, Azure Service Operator potrà semplificare il processo. Invece di dover usare la CLI di Azure, Azure Cloud Shell o i modelli ARM, avvalendosi di Azure Service Operator sarà possibile distribuire queste risorse come se fossero native in Kubernetes.

Azure Service Operator

Azure Service Operator è un altro operatore in grado di semplificare la vita di sviluppatori o amministratori che distribuiscono applicazioni in Azure Red Hat OpenShift. Il primo vantaggio è la capacità di eseguire il provisioning dei servizi Azure senza dover uscire dalla console di OpenShift, il che facilita e accelera il deployment delle applicazioni con potenziali dipendenze dai servizi di Azure, ad esempio Azure CosmosDB.

Il secondo vantaggio, forse più importante, è che Azure Service Operator consente l'archiviazione di tali dipendenze dai servizi di Azure insieme alla definizione dell'applicazione OpenShift, applicando un approccio di tipo Infrastructure-as-Code che utilizza file Kubernetes YAML standard.

Il funzionamento è molto semplice: Azure Service Operator cerca nuove **CRD**, definite in YAML, che corrispondono a una definizione di risorsa in Azure. Azure Service Operator tradurrà quindi il file YAML nelle chiamate alle API di Azure necessarie per creare quanto richiesto in Azure. Una volta installato l'operatore, gli utenti possono sfogliare il catalogo per gli operatori di OpenShift e visualizzare la schermata di creazione di numerose risorse Azure comuni, come gli indirizzi IP pubblici di Azure, i database Azure SQL o Azure Firewall.

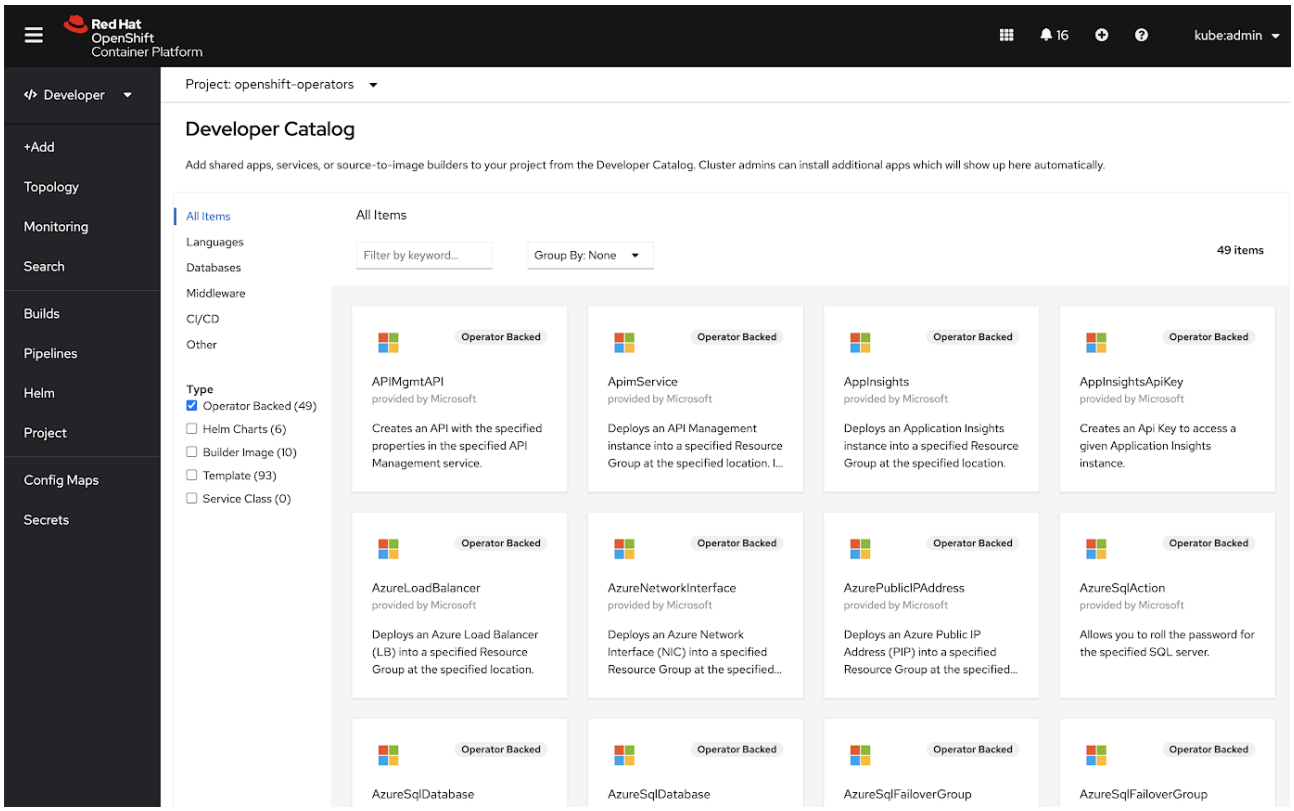


Figura 9.1: Alcuni dei servizi di Azure esposti da Azure Service Operator

Azure Service Operator può essere installato tramite OperatorHub e reso così disponibile a tutti gli utenti in OpenShift.

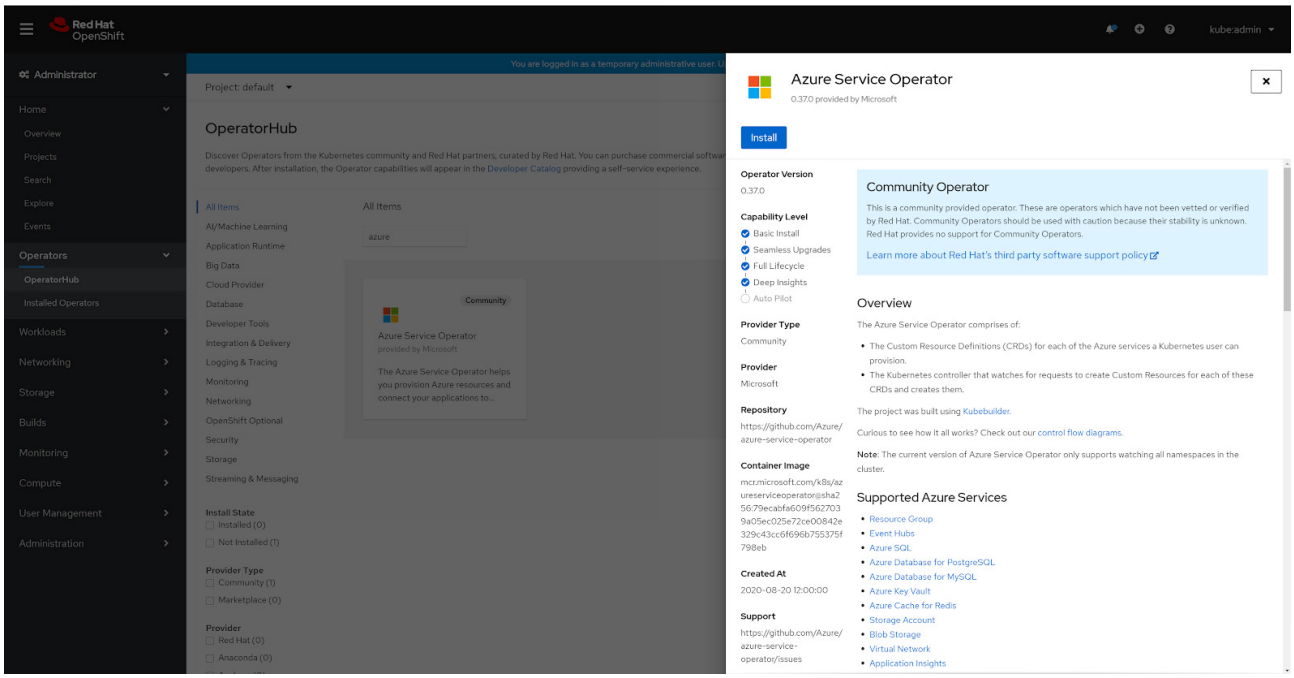


Figura 9.2: La schermata di installazione di Azure Service Operator, con la galleria OperatorHub nello sfondo

Non è obbligatorio utilizzare Azure Service Operator per i servizi in esecuzione su Azure, ed è importante anche comprendere che i servizi Azure sottostanti vengono distribuiti in modo nativo in Azure e non in OpenShift. Azure Service Operator semplifica e accelera questi processi per le applicazioni che hanno dipendenze dai servizi Azure.

Uno scenario di utilizzo comune per Azure Service Operator è il deployment di database dipendenti insieme all'applicazione, come nel caso di un'applicazione web che utilizza un'istanza di Azure CosmosDB, in cui il deployment di Azure CosmosDB con Azure Service Operator è parte del deployment dell'applicazione in OpenShift. Azure Service Operator supporta Azure SQL Database, Azure Database per MySQL, Azure PostgreSQL e altri.

Presupponendo che Azure Service Operator sia installato, il seguente manifesto Kubernetes potrebbe essere archiviato con l'applicazione OpenShift e utilizzato per eseguire il provisioning di un database MySQL:

Fonte: esempio proveniente dal repository di esempi di Azure Service Operator

```
apiVersion: azure.microsoft.com/v1alpha1
kind: MySQLServer
metadata:
  name: aso-wpdemo-mysqlserver
spec:
  location: eastus2
  resourceGroup: aso-wpdemo-rg
  serverVersion: "8.0"
  sslEnforcement: Disabled
  minimalTLSVersion: TLS10 # Possible values include: 'TLS10', 'TLS11', 'TLS12', 'Disabled'
  infrastructureEncryption: Enabled # Possible values include: Enabled, Disabled
  createMode: Default # Possible values include: Default, Replica, PointInTimeRestore (not
implemented), GeoRestore (not implemented)
  sku:
    name: GP_Gen5_4 # tier + family + cores eg. - B_Gen4_1, GP_Gen5_4
    tier: GeneralPurpose # possible values - 'Basic', 'GeneralPurpose', 'MemoryOptimized'
    family: Gen5
    size: "51200"
    capacity: 4
```

Sarebbe insolito utilizzare Azure Service Operator per servizi che coinvolgono molti servizi Azure con dipendenze complesse; in questi casi, strumenti come i modelli Azure ARM o Bicep risultano più idonei.

Per approfondire Azure Service Operator, leggi i seguenti articoli del blog:

- Articoli del blog – Settembre 2020
- Azure Service Operator su OperatorHub.io
- Azure Service Operator su GitHub

Integrazione con Azure DevOps

Oltre che con OpenShift Pipelines, molti utenti integrano Azure Red Hat OpenShift con Azure DevOps. Queste soluzioni possono coesistere senza difficoltà; alcuni progetti ne usano una, altri utilizzano l'altra, altri ancora possono utilizzarle entrambe. Decidere quando utilizzare una o l'altra soluzione dipende da alcuni fattori.

In generale, Azure DevOps offre un livello di integrazione maggiore con altri strumenti di Azure, ma può essere facilmente distribuito in OpenShift come accade con altre risorse di elaborazione.

D'altro canto, OpenShift Pipelines è un'offerta ben integrata abbinata a OpenShift, e offre un'esperienza multicloud omogenea.

Azure DevOps considera OpenShift alla stregua di qualsiasi altro cluster Kubernetes. Pertanto, tutte le interfacce e le API Kubernetes standard funzioneranno come previsto.

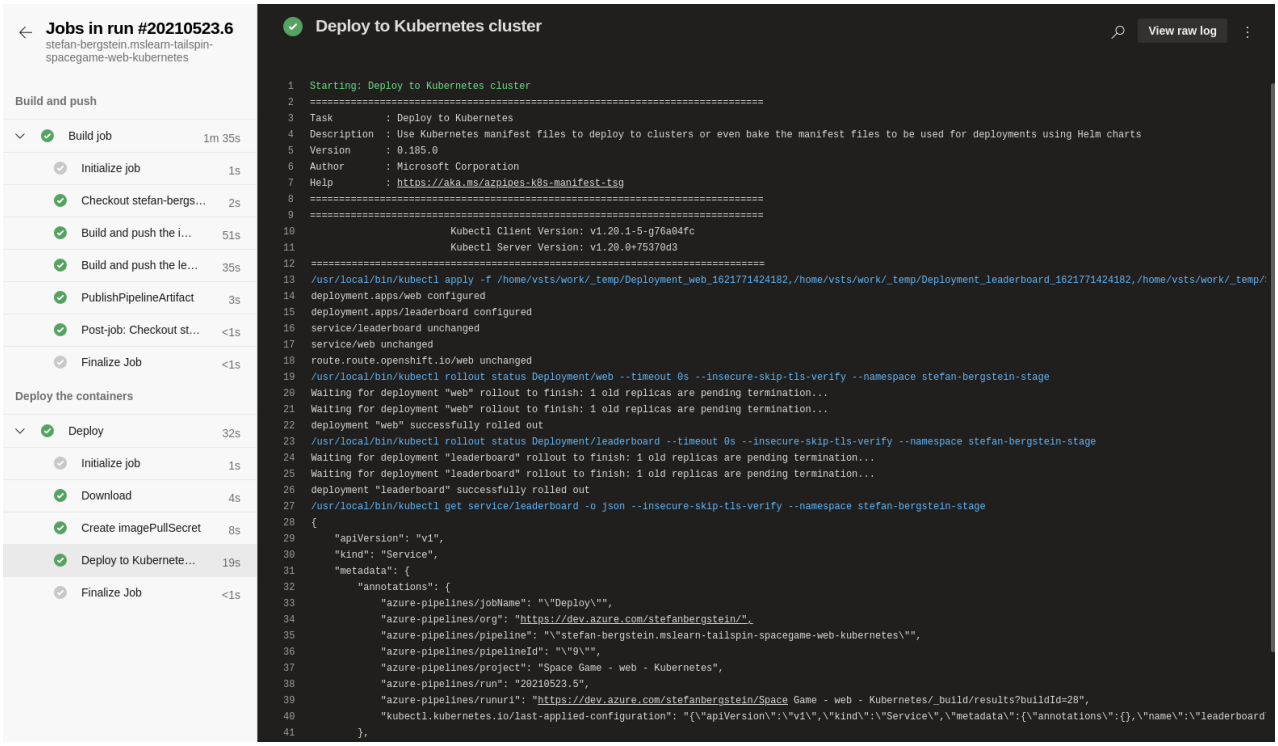


Figura 9.3: Pipeline di Azure DevOps che invia contenuto a OpenShift

Approfondimenti

L'articolo del blog della community indicato di seguito offre un valido tutorial per iniziare a utilizzare Azure Red Hat OpenShift e Azure DevOps:

- Articolo del blog – Maggio 2021

Integrazione con Visual Studio Code

Per valorizzare l'integrazione per sviluppatori OpenShift, esistono plugin per molti degli IDE più comuni, incluso Visual Studio Code. Ciò consente a sviluppatori e amministratori di accedere alle risorse Kubernetes e OpenShift in modo rapido e semplice, rimanendo nel proprio IDE.

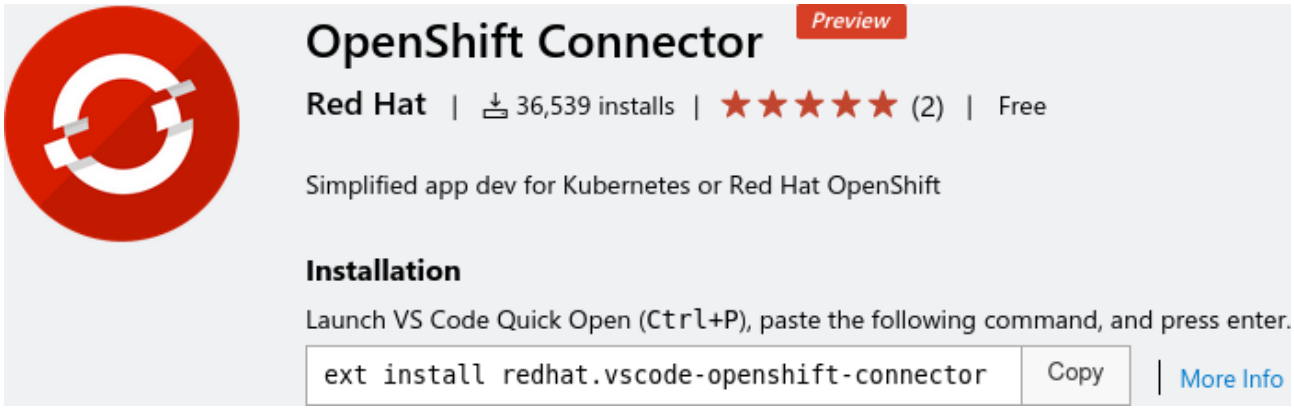


Figura 9.4: Installazione di OpenShift Connector

Una volta scaricato e installato il connettore in Visual Studio Code, il sistema chiederà di accedere al cluster OpenShift. L'esempio seguente, nello stile dei progetti "hello world", include una connessione ad Azure Red Hat OpenShift:

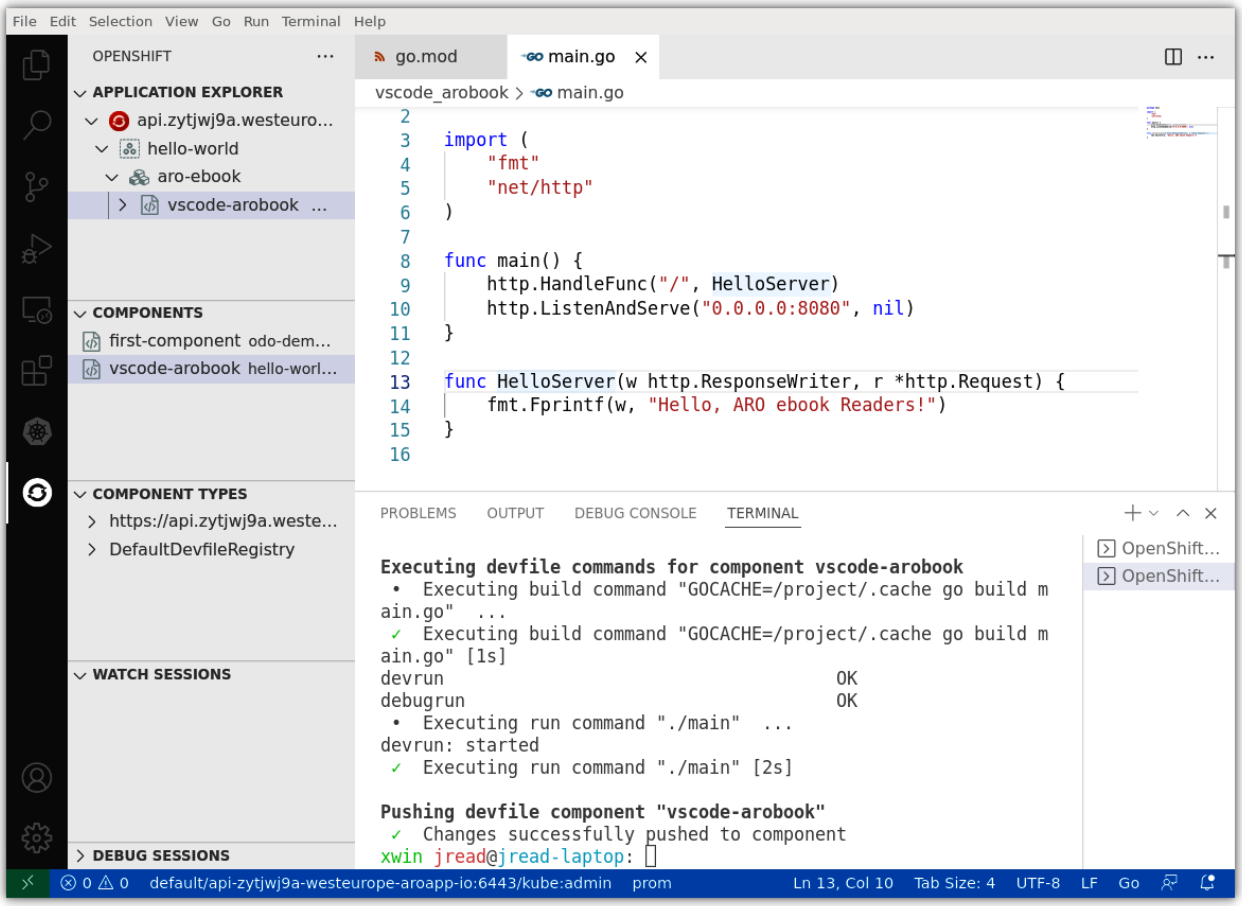


Figura 9.5: Progetto Golang appena distribuito con il connettore OpenShift Visual Studio Code

Utilizzare il connettore OpenShift con Visual Studio Code consente di avere a disposizione l'interfaccia grafica del nostro strumento OpenShift "OpenShift Do", chiamato semplicemente "odo", che consente agli sviluppatori di operare con concetti di livello più elevato rispetto ai semplici componenti Kubernetes, senza doversi occupare in dettaglio di servizi come Deployments, ReplicaSets e così via. Come mostra la schermata precedente, si tratta di un progetto semplice con un servizio HTTP esposto.

Il connettore è dotato anche di funzionalità per l'invio diretto delle modifiche al codice a OpenShift, evitando l'utilizzo del controllo della sorgente. Questo approccio accelera lo sviluppo, non richiedendo ogni volta di eseguire il controllo della sorgente, compilare un nuovo container e attenderne il deployment.

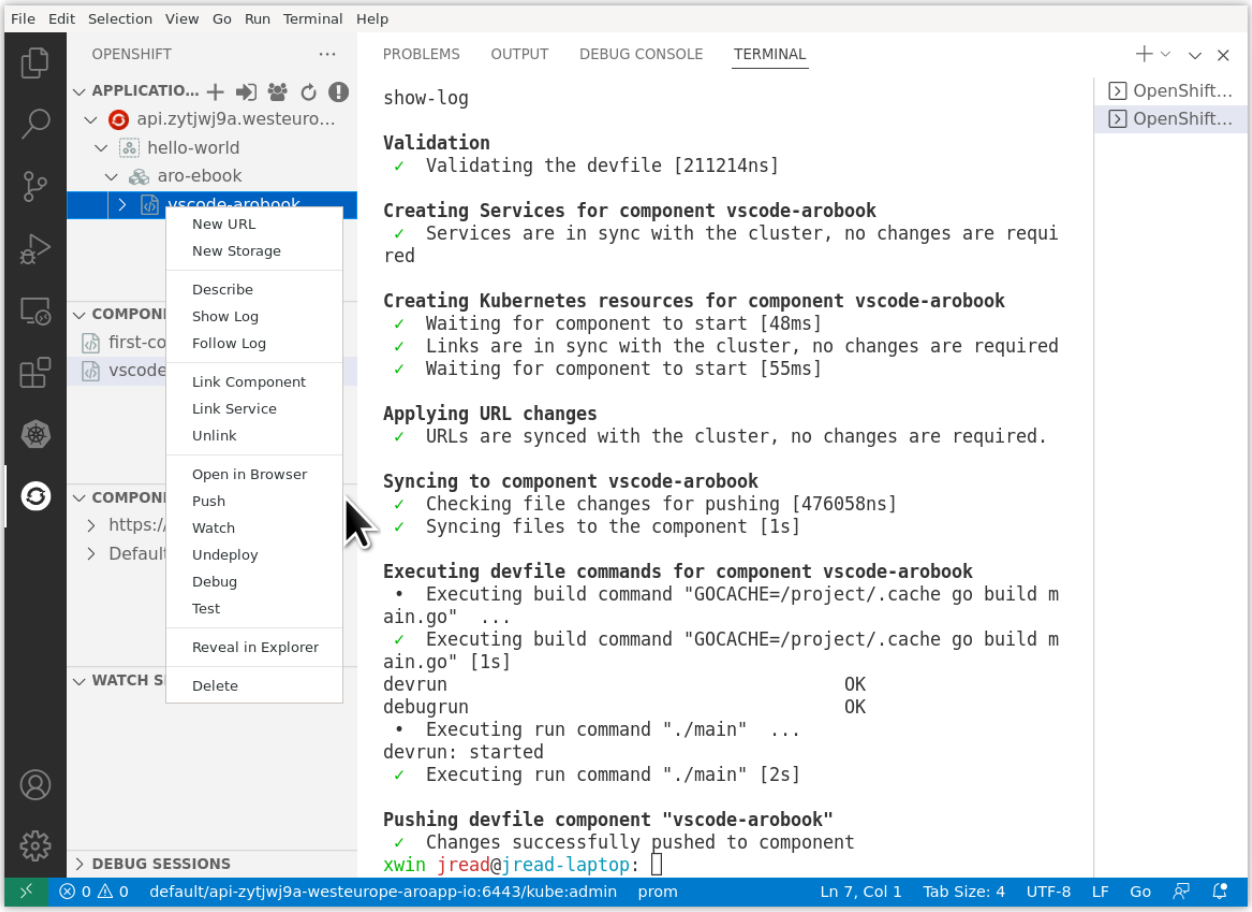


Figura 9.6: Il menu "push" in un progetto, con un push recente effettuato nella finestra del terminale

Questo connettore accelera i cicli di sviluppo e test per gli sviluppatori che preferiscono utilizzare Visual Studio Code.

Figura 9.7: Applicazione di base Golang in esecuzione su OpenShift

Gli sviluppatori non sono ovviamente vincolati a utilizzare solo Visual Studio Code; molti infatti lavorano con Vim, Eclipse e altri editor, ma Visual Studio Code è apprezzato da coloro che preferiscono un'esperienza in stile "IDE leggero".

Approfondimenti

- Video dimostrativo sull'utilizzo di Visual Studio Code con OpenShift
- Connettore OpenShift per Visual Studio Code

Integrazione con GitHub Actions

Ora è possibile utilizzare GitHub Actions per distribuire in qualsiasi ambiente Red Hat OpenShift, incluso Azure Red Hat OpenShift. Da qualsiasi repository di GitHub passare ad **Actions** → **New Workflow** e selezionare **OpenShift** dall'elenco delle azioni disponibili.

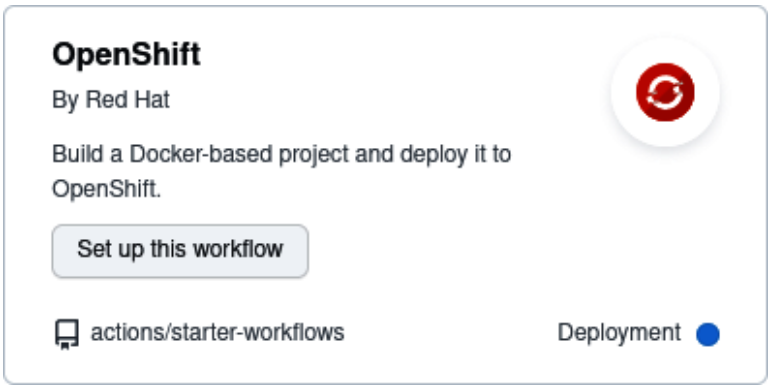


Figura 9.8: Distribuzione in OpenShift da GitHub

Il modello predefinito include un ottimo set di azioni al quale è sufficiente aggiungere qualche variabile di ambiente per eseguire la connessione a un cluster OpenShift:

```
name: OpenShift

env:
  # 🖉 EDIT your repository secrets to log into your OpenShift cluster and set up the context.
  # See https://github.com/redhat-actions/oc-login#readme for how to retrieve these values.
  # To get a permanent token, refer to https://github.com/redhat-actions/oc-login/wiki/Using-a-
Service-Account-for-GitHub-Actions
  OPENSHIFT_SERVER: ${{ secrets.OPENSHIFT_SERVER }}
  OPENSHIFT_TOKEN: ${{ secrets.OPENSHIFT_TOKEN }}
  # 🖉 EDIT to set the kube context's namespace after login. Leave blank to use your user's
default namespace.
  OPENSHIFT_NAMESPACE: ""
  ...
```

Di seguito il collegamento a un eccellente articolo del blog che illustra come fare, insieme a un video dimostrativo:

- Articolo del blog
- Video dimostrativo sull'utilizzo di GitHub Actions con OpenShift

Riepilogo

In questo capitolo sono state esaminate molte delle integrazioni più comuni utilizzate dai clienti con Azure Red Hat OpenShift. Come si è detto nell'introduzione del capitolo, le API OpenShift standard consentono l'integrazione con molti altri servizi non presentati in questo documento. Molti altri possono essere facilmente integrati tramite gli operatori elencati in OperatorHub.

Il capitolo successivo illustra alcune procedure ottimali e raccomandazioni sull'onboarding dei team che si occupano dell'applicazione e su come incentivare l'adozione del servizio nell'organizzazione.

Capitolo 10

Onboarding di carichi di lavoro e team

Dopo aver completato i passaggi del *pre-provisioning*, il provisioning di Azure Red Hat OpenShift e i necessari passaggi di post-provisioning, si passa alla fase di supporto e onboarding degli sviluppatori e delle applicazioni nel cluster. Il modo in cui questa fase viene affrontata dipende sostanzialmente dalla cultura dell'organizzazione: alcuni team preferiscono immergersi nell'attività e iniziare, mentre altri preferiscono ricevere più supporto.

Questo capitolo intende fornire una serie di checklist da utilizzare come base di partenza per accertarsi che i team di sviluppo e delle applicazioni siano preparati e in grado di utilizzare rapidamente Azure Red Hat OpenShift.

Le checklist possono essere estese e adattate alla struttura e alla cultura di ogni organizzazione.

Checklist: fase di pre-onboarding

Prima di eseguire l'onboarding di un nuovo carico di lavoro, è importante poter comunicare al team dell'applicazione o ai proprietari di tale carico di lavoro che Azure Red Hat OpenShift è stato distribuito e progettato secondo modalità già adeguate alle procedure ottimali in uso nell'organizzazione:

- Azure Red Hat OpenShift è stato distribuito in una zona di destinazione di Azure o in un altro ambiente Azure approvato per le applicazioni dell'organizzazione.
- Il cluster è già dotato di tutte le configurazioni necessarie per le attività ordinarie, come le classi di storage e l'autenticazione, come descritto nel *Capitolo 6, Post-provisioning - Operazioni ordinarie*.
- Il cluster è già completamente configurato e supportato dal team della piattaforma e sostenuto dal supporto integrato offerto da Red Hat e Microsoft.
- Il cluster è aggiornato alla versione stabile più recente e sono state applicate tutte le patch dovute. L'applicazione delle patch continuerà anche in futuro.

- Il cluster Azure Red Hat OpenShift è connesso alle risorse essenziali richieste:
  - Connessione all'ambiente on premise tramite Azure ExpressRoute o VPN
  - Connessione a un repository di artefatti aziendali (ad esempio il repository Java Maven)
  - Connessione alla rete Internet pubblica per il download delle dipendenze durante la creazione dell'applicazione

Onboarding dei carichi di lavoro pilota

Un modello comune per il rollout di Azure Red Hat OpenShift in un'organizzazione prevede l'onboarding di almeno due carichi di lavoro pilota:

- **Il carico di lavoro significativo minimo**. Teoricamente, come team SRE, il primo carico di lavoro pilota è un'applicazione conosciuta e di piccole dimensioni, con requisiti tecnici davvero semplici. È un'applicazione poco più avanzata della semplice "Hello World", in quanto dovrebbe avere un reale proprietario nell'organizzazione. Con il primo carico di lavoro, tuttavia, l'enfasi va posta sul verificare che il cluster Azure Red Hat OpenShift possa soddisfare le esigenze aziendali specifiche del cluster in termini di connessione ad Azure, accesso ad Azure Active Directory e identificazione delle problematiche più immediate e comuni presenti in quasi qualsiasi applicazione.
- **Un carico di lavoro di riferimento significativo.** Una volta completato il deployment di un carico di lavoro minimo e semplice, l'adozione sarà incentivata se il secondo carico di lavoro è sufficientemente complesso e significativo, cioè importante per l'azienda o perfino business-critical. Questo carico di lavoro aiuterà a identificare e risolvere problemi quali la connessione ai database strategici, il test delle prestazioni, l'accesso e le metriche su larga scala. Inoltre, tale carico di lavoro potrà essere utilizzato come **riferimento interno** dell'organizzazione e aiuterà i futuri team di sviluppo e dell'applicazione ad acquisire fiducia nella piattaforma Azure Red Hat OpenShift.

Esistono anche altri schemi di onboarding dei primi carichi di lavoro che potrebbero essere adatti alla tua organizzazione. In alcuni casi è necessario puntare ad applicazioni in un data center che verrà dismesso a breve o ad applicazioni in esecuzione su un server applicativo Java ormai obsoleto.

Checklist: incontro di onboarding con team aggiuntivi

Quando si esegue l'onboarding di un nuovo team di sviluppo o dell'applicazione nel cluster, è buona prassi ospitare un primo incontro di presentazione:

- Creare uno spazio dei nomi o una serie di spazi dei nomi che potranno essere utilizzati dal team.
- Presentare una breve demo di Red Hat OpenShift al team, evidenziando le funzionalità principali come Deploy da GitHub o altre equivalenti.
- Verificare che il cluster disponga di capacità sufficiente per distribuire il carico di lavoro di destinazione (CPU, RAM, storage e così via).
- Verificare che i membri del team possano accedere al cluster; per agevolare questa operazione la connessione ad Azure Active Directory è ottimale.
- Fornire i dettagli di contatto del team SRE, o simile, che si occupa della gestione del cluster.
- Fornire un elenco dei collegamenti che offrono una panoramica di OpenShift:
    - http://learn.openshift.com
    - https://github.com/openshift-labs/starter-guides
    - http://docs.openshift.com

Checklist: prevedere chiamate regolari per il controllo della situazione

Dopo che il team di sviluppo o applicativo ha avviato il deployment del cluster, è bene organizzare incontri regolari per verificare l'andamento delle attività. Si può pensare a chiamate che rientrino in una ruotine quotidiana, ma anche una frequenza settimanale di incontri da 30 minuti potrebbe essere sufficiente. Ecco un elenco degli aspetti da controllare:

- Andamento generale del team: è stato eseguito il deployment di qualche applicazione?
- Sono stati riscontrati problemi recenti con l'utilizzo del cluster (problemi di connettività o di conoscenza di Red Hat OpenShift)?
- Ci sono state interruzioni in Azure o con il cluster che hanno avuto un impatto negativo sull'esperienza?
- Promemoria sulla disponibilità del team e informazioni di contatto per rispondere alle domande.

Si possono considerare altri elementi già utilizzati in chiamate per progetti simili, e aggiungere alla checklist altri aspetti ritenuti utili.

Schemi da evitare: creare un ambiente playground/sandbox per gli sviluppatori

Uno schema comune da evitare quando si intendono incoraggiare gli sviluppatori all'adozione è la proposta di un ambiente di tipo sandbox, comunemente chiamato "parco giochi per sviluppatori" e aspettarsi che sviluppatori e applicazioni inizino ad adottare Azure Red Hat OpenShift. Senza supporto o risorse aggiuntive di follow up, Azure Red Hat OpenShift può apparire un ambiente ostile, con complessità difficili da metabolizzare. Nella maggior parte dei casi, l'adozione di un modello con "sandbox" può causare un'inutile spesa in cloud di elaborazione e cluster vuoti.

Tuttavia, nelle organizzazioni i cui team di sviluppo hanno utilizzato in precedenza gli ambienti "sandbox" è possibile seguire alcuni suggerimenti per fare in modo che l'esperienza sia comunque positiva:

- Fornire almeno una breve demo che mostri quanto Azure Red Hat OpenShift è in grado di fare.
- Fornire la documentazione e i collegamenti introduttivi indicati nelle sezioni precedenti.
- Organizzare un evento "hackathon", sfidando gli sviluppatori a creare qualcosa con OpenShift nell'ambito di una piccola competizione.
- Fornire un'offerta che estenda il supporto, presentare il team SRE e un'esperienza di onboarding più personalizzata come ulteriore opzione.

Se si segue una checklist con uno schema di adozione guidata, come indicato in precedenza, le possibilità di adozione della piattaforma aumentano.

Azure Red Hat OpenShift Developer Workshop

Azure Red Hat OpenShift Developer Workshop (https://aroworkshop.io) è un utile workshop pronto all'uso che è possibile proporre all'organizzazione per un'esperienza pratica e guidata su Azure Red Hat OpenShift. Il workshop è suddiviso in due esercitazioni laboratoriali, entrambe pensate per offrire un tutorial guidato a sviluppatori o operatori che non hanno mai usato OpenShift. Entrambi i laboratori evidenziano le principali funzionalità di OpenShift e mostrano come utilizzarle. Il primo è un'introduzione alla progettazione di microservizi moderni e il secondo entra nel merito del servizio.

Un'utile attività per invogliare l'organizzazione ad adottare Azure Red Hat OpenShift è l'uso dei contenuti di aroworkshop.io nel cluster interno di Azure Red Hat OpenShift. Durante la presentazione del workshop è bene illustrare come Azure Red Hat OpenShift sia stato distribuito nell'ambito della sottoscrizione di Azure Red Hat OpenShift già esistente. L'uso del servizio nel workshop può offrire un'esperienza simile all'uso di Azure Red Hat OpenShift per l'hosting delle applicazioni di produzione.

Riepilogo

In questo capitolo sono state fornite alcune checklist e linee guida utili per un onboarding efficace dei team e delle applicazioni in Azure Red Hat OpenShift. È stato descritto uno schema comune da evitare, ovvero creare cluster "sandbox" senza offrire supporto. Nessuna guida è in grado di offrire un elenco completo di risorse e di voci di checklist valide per qualsiasi organizzazione; è perciò importante adattare i contenuti di questo capitolo alle esigenze specifiche.

Il cloud offre un'interessante gamma di servizi, molti dei quali tuttavia vengono sottovalutati o adottati in modo inadeguato, perché le organizzazioni si concentrano di più sulla tecnologia e sulle sue modalità di deployment. Comprendere queste tematiche è senz'altro importante, ma rappresenta solo una parte dell'equazione nel momento in cui il servizio deve entrare in produzione ed essere ampiamente adottato. In questo capitolo abbiamo cercato di sottolineare come la formazione, i controlli regolari e attività simili sono indispensabili per garantire il successo dell'adozione di OpenShift.

Capitolo 11

Conclusioni

I team DevOps della tua organizzazione possono dedicare la maggior parte delle ore di lavoro alle attività di provisioning, configurazione, gestione e supervisione dei cluster e della pipeline CI/CD. Se è così, non rimarrà loro molto tempo da dedicare a ciò che sanno fare meglio: mantenere le applicazioni sempre al passo con l'innovazione.

Come illustrato in questa guida, Azure Red Hat OpenShift consente di distribuire cluster Red Hat OpenShift completamente gestiti senza doversi occupare di creare e gestire l'infrastruttura su cui eseguirli. Abbiamo visto come l'esecuzione del solo Kubernetes ha delle conseguenze, principalmente legate alle attività che potrebbero essere automatizzate con Azure Red Hat OpenShift.

Nella scelta della strategia di gestione del cluster più adatta, occorre considerare i pro e i contro che derivano dall'adozione di una piattaforma di tipo Kubernetes rispetto ad Azure Red Hat OpenShift, che si basa sul framework Kubernetes ma offre numerosi vantaggi out-of-the-box aggiuntivi.

Per approfondire Azure Red Hat OpenShift, visita la pagina del prodotto o leggi la relativa sezione nella documentazione. È inoltre possibile partecipare a workshop pratici e registrarsi per guardare un webinar in totale libertà. infine, Microsoft e Red Hat ti offrono il loro supporto nella valutazione di Azure Red Hat OpenShift.

Autori e versioni

James Read <james@redhat.com>
Principal Solution Architect presso Red Hat,
si occupa di Microsoft; aggiornamento e revisione per AROv4

Ahmed Sabbour <asabbour@microsoft.com>
Senior Product Marketing Manager presso Microsoft,
si occupa di Azure Red Hat OpenShift; ha redatto la versione iniziale per AROv3

Autori delle versioni precedenti

Oren Kashi <okashi@redhat.com>
Senior Principal Technical Product Marketing Manager presso Red Hat

Grazie a Brooke Jackson, Nermina Miller, Jose Moreno, Ahmed Sabbour, Aditya Datar, Vince Power, Alex Patterson e a tutti coloro che hanno gentilmente dedicato il loro tempo e offerto il loro feedback nella revisione di questa guida.

Capitolo 12

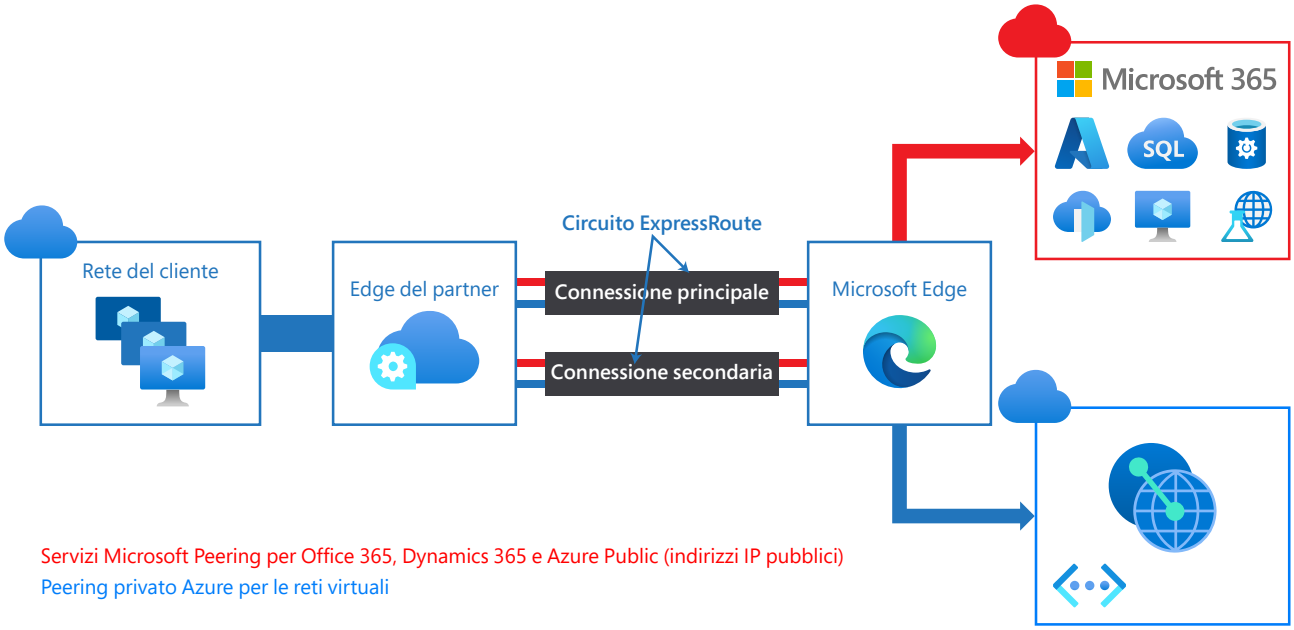
Glossario

Questo glossario intende essere un punto di riferimento rapido per alcuni dei termini utilizzati in questa guida e nell'ecosistema Azure Red Hat OpenShift. I termini sono in ordine alfabetico.

Azure ExpressRoute

ExpressRoute consente di estendere le reti on premise al cloud Microsoft mediante una connessione privata e l'aiuto di un provider di connettività. Grazie a ExpressRoute è possibile stabilire le connessioni a servizi cloud Microsoft, come Microsoft Azure e Microsoft 365.

Di seguito un'immagine di una rete ExpressRoute, proveniente dalla documentazione di Microsoft Azure:



Per leggere di più su ExpressRoute, vedere la pagina Che cos'è Azure ExpressRoute? nella documentazione di Microsoft Azure.

Per comprendere il funzionamento di ExpressRoute in Azure, leggere il *Capitolo 4, Domande sul pre-provisioning dell'architettura enterprise*.

Build e flussi di immagini

Con build, o compilazione, si intende il processo di trasformazione dei parametri di input nell'oggetto risultante. Spesso il processo è utilizzato per trasformare i parametri di input o il codice sorgente in un'immagine eseguibile. Un oggetto BuildConfig è la definizione dell'intero processo di compilazione.

Azure Red Hat OpenShift utilizza Kubernetes creando container formattati per Docker da immagini generate e trasferendole in un registro delle immagini dei container.

Gli oggetti build condividono caratteristiche comuni: input della compilazione, necessità di completare un processo di compilazione, registrazione del processo di compilazione, pubblicazione delle compilazioni riuscite, pubblicazione dello stato finale della compilazione. Gli oggetti build si avvalgono di restrizioni delle risorse e specificano i limiti per risorse come il consumo di CPU e di memoria, il tempo di esecuzione della build o del pod.

Il sistema di compilazione di Azure Red Hat OpenShift fornisce supporto esteso alle strategie di compilazione basate su tipi selezionabili specificati nell'API di compilazione. Sono disponibili tre tipologie principali di strategie di build:

- **Build Docker**: utilizza un Dockerfile, che può essere caricato o inviato da un repository sorgente
- **Build Source-to-Image (S2I)**: utilizza un repository sorgente (come Git) e determina come compilare l'applicazione da file di build in linguaggi noti (ad esempio file Maven `.pom` per progetti Java)
- **Build personalizzata**

Per impostazione predefinita, sono supportate le build Docker e S2I.

L'oggetto risultante di una build dipende dal builder con cui è stata creata. Gli oggetti risultanti delle build Docker e S2I sono immagini eseguibili. Gli oggetti risultanti delle build personalizzate sono qualsiasi elemento specificato dall'autore dell'immagine del builder.

Container

Le unità di base che costituiscono le applicazioni di Azure Red Hat OpenShift sono chiamate container. Le tecnologie dei container Linux sono meccanismi ottimizzati, che isolano i processi in esecuzione in modo che interagiscano esclusivamente con le risorse a loro assegnate.

Nei container di un singolo host possono essere eseguite molte istanze di applicazioni, senza che abbiano visibilità sui processi, i file, la rete delle altre. Sebbene i container possano essere utilizzati anche per carichi di lavoro arbitrari, in genere ogni container fornisce un solo servizio, spesso definito "microservizio", ad esempio un web server o un database.

Deployment e configurazioni di deployment

A partire dai controller di replica, con il concetto di deployment Azure Red Hat OpenShift aggiunge ulteriore supporto per lo sviluppo software e il ciclo di vita del deployment. Nel caso più semplice, un deployment crea un nuovo ReplicationController (controller di replica) e fa sì che questo avvii i pod. I deployment di Azure Red Hat OpenShift, tuttavia, consentono anche di passare da un deployment di un'immagine esistente a uno nuovo, oltre a definire gli hook da eseguire prima o dopo avere creato il controller di replica.

L'oggetto DeploymentConfig di Azure Red Hat OpenShift definisce i seguenti dettagli di un deployment:

1. Gli elementi di una definizione di ReplicationController.
2. I trigger per la creazione automatica di un nuovo deployment.
3. La strategia di transizione tra i deployment.
4. Gli hook del ciclo di vita.

Ogni volta che un deployment viene attivato, che sia in modo manuale o automatico, un pod di deployment gestisce il deployment (inclusa la riduzione del controller di replica obsoleto, l'aumento di quello nuovo e l'esecuzione degli hook). Il pod di deployment rimane per un periodo di tempo indefinito dopo aver completato il deployment, così da conservare i registri dell'operazione. Quando un deployment viene sostituito da un altro, il controller di replica precedente viene mantenuto per facilitare un eventuale rollback.

Per istruzioni dettagliate su come creare e interagire con i deployment, vedere l'argomento Deployment e DeploymentConfigs.

Immagini dei container

I container in Azure Red Hat OpenShift si basano su immagini di container formattati per Docker. Un'immagine è un file binario che contiene tutto il necessario per eseguire un singolo container e i metadata che ne descrivono esigenze e capacità.

Possiamo immaginarla come una tecnologia per la creazione di pacchetti. I container possono accedere soltanto alle risorse definite nell'immagine, a meno che in fase di creazione non vengano concessi al container accessi aggiuntivi. Distribuendo la stessa immagine in più container su più host e bilanciando il carico tra di loro, Azure Red Hat OpenShift garantisce ridondanza e scalabilità orizzontale del servizio contenuto in un'immagine.

Modelli

Un modello descrive un set di oggetti che possono essere elaborati e i cui parametri possono essere impostati per produrre un elenco di oggetti che verranno creati da Azure Red Hat OpenShift. È possibile elaborare un modello affinché crei qualsiasi elemento consentito dalle autorizzazioni disponibili allo sviluppatore, ad esempio servizi, configurazioni di build e configurazioni di deployment. In un modello è inoltre possibile definire un set di etichette da applicare a ogni oggetto in esso definito.

L'elenco di oggetti di un modello può essere creato tramite la CLI o, se il modello è stato caricato nel progetto o nella libreria dei modelli globale, tramite la web console. Per un set selezionato di modelli, consultare la libreria dei flussi di immagine e dei modelli di OpenShift.

Pod e servizi

Azure Red Hat OpenShift utilizza il concetto Kubernetes di pod, che è costituito da uno o più container distribuiti insieme su un host; si tratta della più piccola unità di calcolo che può essere definita, distribuita e gestita.

I pod equivalgono all'incirca all'istanza di una macchina (fisica o virtuale) per un container. A ciascun pod è assegnato un indirizzo IP interno, pertanto esso possiede tutto lo spazio della propria porta e i container all'interno dei pod possono condividere lo storage e la rete locali.

I pod hanno un ciclo di vita: vengono definiti, quindi assegnati per l'esecuzione su un nodo, poi eseguiti fino alla chiusura del container o rimossi per altri motivi. In funzione dei criteri e del codice di uscita i pod possono essere rimossi dopo l'uscita o conservati per consentire l'accesso ai registri dei rispettivi container.

In Azure Red Hat OpenShift i pod sono tendenzialmente immutabili: non è possibile apportare modifiche alla definizione di un pod in esecuzione. Per apportare le modifiche, la piattaforma chiude un pod esistente e lo ricrea modificando la configurazione o l'immagine di base, o entrambe. I componenti di runtime di un pod sono considerati espandibili e vengono ricreati dall'immagine del container definita. Pertanto i pod dovrebbero essere gestiti da controller di livello superiore e non direttamente dagli utenti.

Processi

Un processo è simile a un controller di replica in quanto la sua finalità è quella di creare pod per motivi specifici. La differenza è che i controller di replica sono progettati per pod in continua esecuzione, mentre i processi sono progettati per i pod temporanei. Un processo tiene traccia di qualsiasi completamento riuscito. Quando viene raggiunto il numero specificato di completamenti, anche il processo viene completato.

Vedere l'argomento *Jobs* per ulteriori informazioni su come utilizzare i processi.

Progetti e utenti

Un progetto è uno spazio dei nomi Kubernetes con annotazioni aggiuntive e costituisce il vettore centrale tramite il quale viene gestito l'accesso alle risorse degli utenti standard. Un progetto consente a una community di utenti di organizzare e gestire i propri contenuti in modo isolato dalle altre community. L'accesso degli utenti ai progetti viene concesso dagli amministratori; se gli utenti sono autorizzati a creare progetti, potranno accedere automaticamente ad essi. I parametri name, `displayName` e description di un progetto possono essere differenti.

name è obbligatorio e rappresenta l'identificatore univoco del progetto, visibile soprattutto quando si utilizza l'interfaccia CLI o l'API. La lunghezza massima di questo parametro è di 63 caratteri. `displayName` è facoltativo e indica come verrà visualizzato il progetto nella web console (il valore predefinito è `name`). description è facoltativo e può rappresentare una descrizione più dettagliata del progetto. Anch'esso è visibile nella web console.

Sviluppatori e amministratori possono interagire con i progetti tramite CLI o web console.

Registro dei container

Azure Red Hat OpenShift fornisce un registro delle immagini dei container integrato, denominato **OpenShift Container Registry (OCR)**, che consente di eseguire il provisioning automatico di nuovi repository di immagini on demand. In questo modo gli utenti dispongono di una posizione integrata a cui le build delle applicazioni possono inviare le immagini risultanti.

Ogni volta che una nuova immagine viene inviata a OCR, il registro invia una notifica ad Azure Red Hat OpenShift insieme a informazioni sull'immagine quali lo spazio dei nomi, il nome e i metadati. I diversi componenti di Azure Red Hat OpenShift vengono attivati dalla notifiche di nuove immagini, creando nuove build e deployment.

Inoltre, Azure Red Hat OpenShift può utilizzare qualsiasi server sul quale è presente l'API del registro delle immagini dei contenitori come sorgente di immagini, incluso Docker Hub e il Registro Azure Container.

ReplicaSets

Simile a un controller di replica, un ReplicaSet (set di repliche) garantisce che in un determinato momento venga eseguito il numero specificato di repliche del pod. La differenza tra un set di repliche e un controller di replica è che il primo supporta requisiti del selettore basati sul set mentre il secondo supporta requisiti del selettore basati solo sull'uguaglianza.

```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
    name: frontend-1
    labels:
        tier: frontend
spec:
    replicas: 3
    selector:
        matchLabels:
            tier: frontend
        matchExpressions:
        - {key: tier, operator: In, values: [frontend]}
    template:
        metadata:
            labels:
                tier: frontend
        spec:
            containers:
                - image: openshift/hello-openshift
                  name : helloworld
                  ports:
                  - containerPort: 8080
                    protocol: TCP
            restartPolicy: Always
```

Nel codice precedente troviamo:

- Una query etichetta su un set di risorse. Ciò che risulta da `matchLabels` e `matchExpressions` è congiunto logicamente.
- Selettore basato sull'uguaglianza per specificare risorse con etichette che corrispondono al selettore.
- Selettore basato su set per filtrare le chiavi. Vengono selezionate tutte le risorse con `key` uguale a `tier` e `value` uguale a `frontend`.

ReplicationController

Un controller di replica garantisce l'esecuzione costante del numero specificato di repliche di un pod. Se i pod terminano o vengono eliminati, il controller di replica agisce per creare le istanze necessarie a raggiungere il numero specificato. Allo stesso modo, se sono presenti più pod di quelli necessari, il controller elimina quelli in più, fino a ottenere la quantità specificata.

La configurazione di un controller di replica implica:

- Il numero di repliche desiderate (può essere rettificato in fase di runtime).
- Una definizione di pod da utilizzare quando si crea un pod replicato.
- Un selettore per identificare i pod gestiti.
- Un selettore è un insieme di etichette assegnato ai pod gestiti dal controller di replica. Le etichette sono incluse nella definizione del pod le cui istanze vengono create dal controller di replica. Il controller di replica usa il selettore per determinare quante istanze del pod sono già in esecuzione, così da adeguare il numero come necessario.

Questo controller non dimensiona automaticamente le istanze in base al carico o al traffico, perché non tiene traccia di questi elementi. Ciò richiederebbe che il conteggio della replica sia regolato da uno strumento esterno per la scalabilità automatica.

Il controller di replica è un oggetto Kubernetes primario chiamato ReplicationController. Di seguito un esempio di definizione di ReplicationController:

```
apiVersion: v1
kind: ReplicationController
metadata:
    name: frontend-1
spec:
    replicas: 1
    selector:
        name: frontend
    template:
        metadata:
            labels:
                name: frontend
        spec:
            containers:
                - image: openshift/hello-openshift
                  name: helloworld
                  ports:
                - containerPort: 8080
                  protocol: TCP
            restartPolicy: Always
```

Nel codice precedente troviamo:

- Il numero di copie del pod da eseguire.
- Il selettore di etichette del pod da eseguire.
- Un modello per il pod creato dal controller.
- Le etichette nel pod devono includere quelle derivanti dal selettore di etichette.
- La lunghezza massima del nome dopo l'espansione di tutti i parametri è di 63 caratteri.

Route e Ingress

Azure Red Hat OpenShift supporta le route e le risorse Ingress. Entrambe sono utilizzate per esporre un servizio tramite un nome DNS, come www.example.com, in modo che i client esterni possano accedervi.

Le route sono state ideate originariamente in Red Hat OpenShift versione 3. Dal momento del rilascio, Red Hat ha collaborato con la community Kubernetes per standardizzare questa funzionalità nella risorsa che oggi è nota come **Ingress**.

In OpenShift Container Platform, la risorsa Ingress di Kubernetes implementa il controller Ingress con un servizio di router condiviso che viene eseguito come un pod all'interno del cluster. Il controller Ingress rappresenta il modo più comune per gestire il traffico in ingresso. Come qualsiasi altro pod anche questo è scalabile e replicabile. Il servizio router si basa su HAProxy, una soluzione open source di bilanciamento del carico.

La route OpenShift Container Platform fornisce il traffico in ingresso ai servizi nel cluster. Le route forniscono funzionalità avanzate che potrebbero non essere supportate dai controller Ingress Kubernetes standard quali ri-crittografia TLS, passthrough TLS e il traffico suddiviso per i deployment blue green.

Il traffico in ingresso accede ai servizi nel cluster tramite una route. Route e Ingress sono le principali risorse per la gestione del traffico in ingresso. Una risorsa Ingress fornisce funzioni simili a quelle di una route, ad esempio accetta richieste esterne e le concede in delega in base alla route. Una risorsa Ingress invece consente solo alcuni tipi di connessione: HTTP/2, HTTPS, **Server Name Identification (SNI)** e TLS con certificato. In OpenShift Container Platform le route vengono generate per soddisfare le condizioni specificate dalla risorsa Ingress.

Source-to-Image (S2I)

S2I è un toolkit e un workflow per la creazione di immagini dei container riproducibili a partire dal codice sorgente. S2I genera immagini pronte all'esecuzione tramite l'inserimento del codice sorgente in un container che lo predispone all'esecuzione. Creando immagini del builder auto-assemblanti è possibile controllare l'ambiente della build nello stesso modo in cui si utilizzano le immagini dei container per controllare la versione degli ambienti di runtime.

In un linguaggio dinamico come Ruby, gli ambienti di build-time e di run-time in genere corrispondono. A partire da un'immagine del builder che descrive l'ambiente (con Ruby, Bundler, Rake, Apache, GCC e gli altri pacchetti necessari a impostare ed eseguire un'applicazione Ruby installata) S2I effettua i passaggi indicati di seguito:

1. Avvia un container dall'immagine del builder con il codice sorgente dell'applicazione inserito in una directory nota.
2. Il processo del container trasforma il codice sorgente nella configurazione eseguibile appropriata; in questo caso, installa le dipendenze con Bundler e sposta il codice sorgente in una directory in cui Apache è stato preconfigurato per cercare il file Ruby `config.ru`.
3. Esegue il commit del nuovo container e imposta l'entrypoint dell'immagine come uno script (fornito dall'immagine del builder) che avvia Apache per l'hosting dell'applicazione Ruby.

Per i linguaggi compilati come C, Go o Java, le dipendenze necessarie per la compilazione potrebbero superare di molto la dimensione dei componenti di runtime effettivi. Per non appesantire le immagini di runtime, S2I consente processi di compilazione in più fasi, in cui un componente binario, come un file eseguibile o un file WAR di Java, viene creato nella prima immagine del builder, estratto e inserito in una seconda immagine di runtime che colloca l'eseguibile nella posizione corretta per l'esecuzione.

Ad esempio, per creare una pipeline di compilazione riproducibile per Tomcat (web server Java molto diffuso) e Maven, procede come indicato di seguito:

1. Creare un'immagine del builder contenente OpenJDK e Tomcat, che prevede l'inserimento di un file WAR.
2. Creare una seconda immagine che si pone a un livello superiore alla prima immagine Maven e a qualsiasi altra dipendenza standard, e che preveda l'inserimento di un progetto Maven.
3. Invocare S2I usando il codice sorgente dell'applicazione Java e l'immagine Maven per creare l'applicazione WAR desiderata.
4. Invocare S2I una seconda volta usando il file WAR del passaggio precedente e l'immagine Tomcat iniziale per creare l'immagine di runtime.

Posizionando la logica di compilazione all'interno delle immagini e combinando le immagini in più passaggi, possiamo tenere l'ambiente di runtime vicino all'ambiente di compilazione (lo stesso JDK, gli stessi file JAR di Tomcat) senza dover distribuire in produzione gli strumenti di compilazione.

Le finalità e i vantaggi dell'uso di Source-To-Image (S2I) come strategia di compilazione sono i seguenti:

- **Riproducibilità:** consente di correlare gli ambienti di compilazione alle versioni, incapsulandoli in un'immagine del container e definendo una semplice interfaccia (codice sorgente inserito) per i chiamanti. Le build riproducibili sono fondamentali per consentire gli aggiornamenti di sicurezza e l'integrazione continua nell'infrastruttura containerizzata; le immagini del builder garantiscono la ripetibilità e la capacità di commutare i runtime.
- **Flessibilità**: qualsiasi sistema di compilazione esistente eseguibile su Linux può essere eseguito anche all'interno di un container; ogni singolo builder può far parte di una pipeline più grande. Inoltre, gli script che elaborano il codice sorgente dell'applicazione possono essere inseriti nell'immagine del builder, consentendo così agli autori di adattare le immagini esistenti per abilitare la gestione dei codici sorgente.
- **Velocità**: invece di compilare più livelli in un unico file Docker, S2I incoraggia gli autori a rappresentare l'applicazione in un unico livello immagine. In questo modo si risparmia tempo nelle fasi di creazione e deployment e si ottiene un maggior controllo sull'output dell'immagine finale.
- **Sicurezza**: le build che utilizzano i file Docker vengono eseguite con un numero ridotto di controlli operativi dei container; di solito vengono eseguiti come root e con accesso alla rete dei container. S2I può controllare le autorizzazioni e i privilegi disponibili all'immagine del builder, poiché la compilazione viene avviata in un solo container. In armonia con piattaforme come OpenShift, S2I consente agli amministratori un controllo più rigido dei privilegi degli sviluppatori in fase di build-time.

Zone di destinazione di Azure

Le zone di destinazione di Azure costituiscono un diffuso schema di deployment utilizzato dalle organizzazioni che pianificano distribuzioni su larga scala mediante Azure, tenendo conto delle caratteristiche di scalabilità, governance della sicurezza, rete e identità.

[Introduzione alle zone di destinazione di Azure]

In fase di sviluppo al momento della stesura di questo documento, esiste un progetto della community GitHub che offre alcuni consigli su come distribuire Azure Red Hat OpenShift in un'architettura con zone di destinazione di Azure:
https://github.com/Azure/Enterprise-Scale/tree/main/workloads/ARO