SBDIO I4.0
Big Data
for Industry

O2.3

Prototipo finale cloud ibrido e service mesh federato di SBDIOI40

Code	O2.3
Data	15/06/2021
Type	Confidential
Participants	CIRI-ICT
Authors	Andrea Garbugli, Lorenzo Patera (UNIBO)
Corresponding authors	Luca Foschini

POR FESR EMILIA-ROMAGNA 2014/2020
FSC Fondo per lo Sviluppo e la Coesione
UNIONE EUROPEA Fondo europeo di sviluppo regionale
Regione Emilia-Romagna

Sommario

1 Cloud ibrido e federazione

In questo documento discuteremo il prototipo finale dell'infrastruttura cloud di SBDIOI40. Nelle prossime sezioni ripercorreremo i vari step evolutivi attraverso i quali è passato lo sviluppo di tale piattaforma.

Inizialmente la piattaforma è stata concepita come un cluster OpenStack. L'architettura del cluster è a singolo nodo, con la possibilità di aggiungere ulteriori macchine dinamicamente durante l'esecuzione. Il nodo ricopre i ruoli controller, compute e network.

Il modello del server fisico è un Acer AR585F1 con le seguenti caratteristiche hardware:

- 32 CPU,
- 32GB di memoria RAM,
- 15TB di memoria di massa,
- 4 interfacce di rete.

Per quanto riguarda le impostazioni di rete, la macchina utilizza 3 interfacce fisiche:

- Una prima interfaccia offre l'accesso alla rete pubblica, rendendo possibile la gestione remota dell'infrastruttura virtuale e di esporre servizi in esecuzione su Internet.
- Una seconda interfaccia collega il computer alla rete di gestione del cluster CIRI-ICT. Si tratta di una rete locale che viene utilizzata sia per accedere alle macchine per operazioni di manutenzione, sia come canale di comunicazione per il traffico di controllo fra i nodi di OpenStack. Tale interfaccia semplifica il processo di estensione del cluster fisico, fondamentale nella aggiunta di nuove risorse computazionali.
- La terza ed ultima interfaccia di rete fornisce l'accesso alla rete dedicata al traffico di controllo e applicativo della piattaforma Kubernetes, che verrà descritta nelle prossime sezioni.

L'infrastruttura appena descritta è stata integrata con un secondo cloud privato gestito da T3LAB per fornire l'astrazione di un singolo cluster distribuito geograficamente. Tale operazione ha richiesto un procedimento chiamato federazione degli accessi, descritto di seguito.

1.1 Federazione degli accessi

Con il termine cloud federato si intende la facilitazione dell'interconnessione di due o più cloud di elaborazione geograficamente separati [1]. Nel nostro caso, la federazione consiste in una autenticazione unica fra i due cloud OpenStack CIRI-ICT e T3LAB. L'approccio federato porta a molteplici vantaggi. È possibile delegare l'onere dell'autenticazione a una entità terza, scaricando i singoli cloud privati di questa responsabilità. Inoltre, diventa possibile sfruttare una sorgente di identità unica e condivisa: con le stesse credenziali è possibile accedere ed utilizzare i servizi di diverse piattaforme. Risulta possibile dunque connettere diversi cloud gestiti da entità separate in modo uniforme.

POR FESR EMILIA-ROMAGNA 2014/2020
FSC Fondo per lo Sviluppo e la Coesione
UNIONE EUROPEA Fondo europeo di sviluppo regionale
Regione Emilia-Romagna

Lo standard studiato ed utilizzato come protocollo di autenticazione tra due prototipi di piattaforma cloud all'interno di questo progetto è lo standard SAML (Security Assertion Markup Language). Quest'ultimo permette la descrizione e lo scambio di informazioni di sicurezza tra le più parti permettendo quindi di realizzare meccanismi di: single sign-on (SSO), autenticazione federata, ecc.

SAML propone un protocollo di comunicazione basato su XML per trasmettere dei "token" di sicurezza, chiamati assertion, tra un'autorità che li emette (l'Identity Provider) e un consumatore che li utilizza (il Service Provider). Queste assertion contengono informazioni fondamentali sull'utente che è in fase di autorizzazione. All'interno dello standard SAML, e pure in generale nel contesto della federazione, si distinguono tre attori fondamentali: il client, il Service Provider (SP) e l'Identity Provider (IdP).

Per un primo prototipo di federazione degli accessi tra le piattaforme OpenStack del CIRI-ICT e del T3LAB si è scelto di adottare lo schema con Identity Provider esterno e Keystone come semplice Service Provider. Per quanto riguarda l'IdP, la scelta è ricaduta su JumpCloud. JumpCloud è un servizio di directory che gli utenti possono utilizzare per effettuare procedure di autenticazione gestendo utenti e applicazioni multiple.

Una volta implementata tale modalità di autenticazione, la Dashboard di OpenStack Horizon è stata configurata affinché mostri, nella pagina di accesso, un menu addizionale che consenta di scegliere il metodo di autenticazione desiderato tra quello locale e quello federato.

1.2 Migrazione

La piattaforma federata è costituita da molteplici sistemi OpenStack, ciascuno sotto la supervisione di un partner del progetto. Diventa quindi importante disporre di meccanismi di migrazione delle applicazioni da un cloud all'altro dell'infrastruttura. L'obiettivo di questa sezione è quello di presentare il meccanismo di migrazione implementato.

Gli utilizzatori della piattaforma OpenStack hanno tre possibilità per accedere alle risorse:

- **Dashboard**, un pannello di controllo semplificato che è accessibile tramite un browser web,
- **Interfaccia da riga di comando (CLI)**, attraverso l'utilizzo di un terminale,
- **API REST**, dedicate allo sviluppo di applicazioni di controllo personalizzate.

Per quanto riguarda la piattaforma SBDIOI40, si è scelto di seguire due approcci paralleli.

T3LAB ha sviluppato un insieme di script shell che eseguono la migrazione delle applicazioni in esecuzione sui nodi. Gli script sfruttano una connessione SSH per interagire con le macchine fisiche delle rispettive piattaforme cloud e poi impartiscono comandi tramite CLI di OpenStack. Trattandosi di una proof-of-concept, lo script presenta alcuni svantaggi: risulta dipendente da eventuali aggiornamenti dell'interfaccia CLI e richiede credenziali di accesso alla macchina, anziché autenticazione ai servizi di OpenStack tramite API (Application Programming Interface). La soluzione non è conforme ai requisiti, visto che le API OpenStack sono dedicate all'interazione da parte di

POR FESR EMILIA-ROMAGNA 2014/2020
FSC Fondo per lo Sviluppo e la Coesione
UNIONE EUROPEA Fondo europeo di sviluppo regionale
Regione Emilia-Romagna

software applicativi. In aggiunta, gli script sono ritagliati su una specifica applicazione (quella di Carpigiani) e non possono essere riutilizzati con applicazioni diverse rispetto a quella prestabilita.

Per i motivi sopra elencati, CIRI-ICT ha sviluppato un secondo applicativo software dedicato alla migrazione di servizi ed applicazioni. Per mantenere una elevata manutenibilità e visto il supporto riconosciuto alla piattaforma OpenStack [2], il prototipo software è stato scritto in linguaggio Go. Si tratta di un linguaggio moderno, che dispone di librerie standard ampie e strutturate, che semplificano lo sviluppo di software complesso. Un software scritto in Go risulta più robusto rispetto a script shell, grazie, ad esempio, alla peculiare centralità della gestione degli errori nel linguaggio e alla compilazione statica che rende ciascun eseguibile auto-contenuto e dunque indipendente dalle librerie installate sul sistema ospitante.

1.3 Cloud ibrido Kubernetes

Uno dei requisiti progettuali di SBDIO I4.0 è quello di supportare le future applicazioni industriali basate su microservizi. A tal fine, si è resa necessaria l'integrazione di un orchestratore di container nell'attuale sistema basato su OpenStack. La scelta è ricaduta su Kubernetes, un servizio open-source inizialmente sviluppato da Google e manutenuto da Linux Foundation [3].

Kubernetes è una piattaforma portatile ed estensibile per la gestione di carichi di lavoro e servizi containerizzati, che facilita sia la configurazione dichiarativa che l'automazione. Presenta un grande ecosistema in continua crescita.

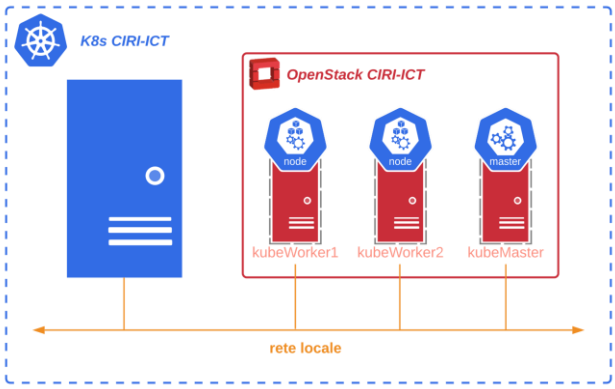


Figure 1. Architettura cloud ibrido CIRI-ICT.

In Figure 1 viene mostrato uno schematico del cluster ibrido risultato dell'attività di analisi di architetture innovative a microservizi del progetto SBDIOI40. Nel contesto di questa attività per cloud ibrido facciamo riferimento sia alla natura dei nodi che compongono il cluster Kubernetes, che all'integrazione di due diverse tecnologie: macchine virtuali gestite da OpenStack e container gestiti da Kubernetes. Essendo Kubernetes indipendente dall'infrastruttura sottostante, si è deciso di utilizzare OpenStack come piattaforma IaaS per dare la possibilità a Kubernetes di utilizzare nodi in forma di macchina virtuale e dei tradizionali server da utilizzare come nodi fisici.

Nella configurazione descritta, il cluster Kubernetes è composto da quattro nodi:

POR FESR EMILIA-ROMAGNA 2014/2020
FSC Fondo per lo Sviluppo e la Coesione
UNIONE EUROPEA Fondo europeo di sviluppo regionale
Regione Emilia-Romagna

- Il master, distribuito tramite macchina virtuale gestita da OpenStack,
- Due nodi worker, distribuiti anch'essi tramite macchine virtuali su OpenStack,
- Infine, un terzo worker bare-metal ad alte prestazioni.

L'approccio ibrido adottato garantisce diversi vantaggi. Innanzitutto, la possibilità di estendere il cluster in qualunque momento sia con nodi virtuali che con nodi fisici. Questo conferisce al sistema una notevole flessibilità. Inoltre, la presenza di un nodo fisico consente comunque di godere dei vantaggi prestazionali derivanti dall'uso di un hardware fisico piuttosto che esclusivamente virtualizzato.

2 Composizione di servizi

Negli ultimi anni, sempre più applicazioni enterprise stanno passando da un'architettura di tipo monolitico ad un basata su microservizi. Un'architettura a microservizi (rappresentata in

Figure 2) permette agli sviluppatori di creare applicazioni altamente flessibili, resilienti e scalabili. Inoltre, applicazioni che seguono questo tipo di architettura rendono possibile dei cicli di sviluppo software molto più rapidi.

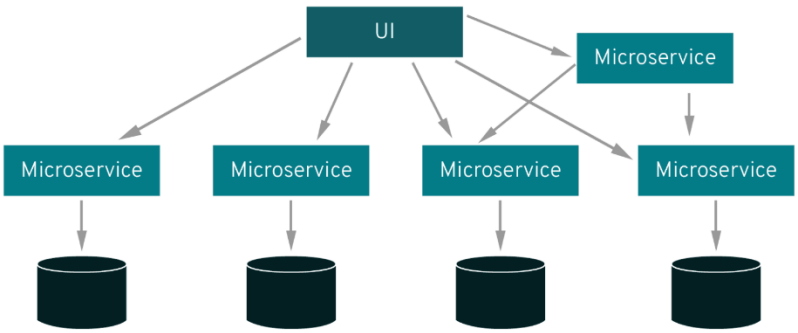


Figure 2. Schematico di una applicazione a microservizi.

Sebbene l'architettura dei microservizi offra molti vantaggi, la sua adozione comporta alcune sfide riguardanti lo sviluppo e la gestione delle applicazioni. Poiché l'architettura è costituita da molti servizi individuali che lavorano insieme, è importante garantire una comunicazione senza interruzioni. Solitamente, i componenti di un'architettura a microservizi possono essere visti come delle entità indipendenti fra loro che espongono le proprie funzionalità attraverso delle API specifiche. Tuttavia, la gestione delle chiamate API, e del flusso di traffico che ne consegue, richiede molto tempo e impegno da parte del gruppo di sviluppo.

La soluzione a questo problema ci viene fornita da quelle chiamiamo composizione di servizi, o Service Mesh. La composizione di servizi permette infatti di facilitare la comunicazione service-to-service e permettere alle applicazioni distribuite di dialogare sia all'interno che all'esterno del cluster di cui fanno parte in modo trasparente e uniforme.

In questo capitolo mostreremo le principali caratteristiche di un'architettura basata su Service Mesh, per poi focalizzare la nostra attenzione su Istio, famosa piattaforma di Service Mesh open-source.

POR FESR EMILIA-ROMAGNA 2014/2020
FSC Fondo per lo Sviluppo e la Coesione
UNIONE EUROPEA Fondo europeo di sviluppo regionale
Regione Emilia-Romagna

2.1 Service mesh

Una Service Mesh rappresenta un livello di infrastruttura configurabile a bassa latenza progettato per gestire un volume elevato di comunicazione inter-processo. Una rete di servizi garantisce che la comunicazione tra servizi di infrastruttura applicativa containerizzati e spesso effimeri sia veloce, affidabile e sicura. Questo è reso possibile da funzionalità critiche, tra cui: rilevamento dei servizi, bilanciamento del carico, crittografia, osservabilità, tracciabilità, autenticazione e autorizzazione [4].

Uno dei modi con cui la rete mesh può essere implementata è attraverso l'utilizzo di un'istanza proxy, chiamata sidecar, associata ad ogni servizio. Le sidecar gestiscono le comunicazioni interservizio ed il monitoraggio. Inoltre, garantiscono una comunicazione sicura per tutte le interazioni tra i singoli servizi. In questo modo, i programmatori possono gestire lo sviluppo, il supporto e la manutenzione del codice applicativo ed i gruppi operativi possono manutenere la rete mesh e mantenere in esecuzione le applicazioni.

Istio [5], supportato da Google, IBM e Lyft, è attualmente l'architettura mesh di servizi più conosciuta. Kubernetes, originariamente progettato da Google, è attualmente l'unico framework di orchestrazione di container supportato da Istio. Nonostante tutto, Istio non rappresenta l'unica opzione e sono in fase di sviluppo anche altre implementazioni di reti mesh. Il modello proxy sidecar è il più popolare, come illustrato dai progetti di Buoyant, HashiCorp, Solo.io e altri. Esistono anche architetture alternative: la suite tecnologica di Netflix è uno degli approcci in cui la funzionalità di mesh di servizi è fornita da librerie di applicazioni (Ribbon, Hysterix, Eureka, Archaius). Piattaforme come Azure Service Fabric incorporano funzionalità simili a mesh di servizi nel framework dell'applicazione.

2.2 Istio

Fra i principali compiti di Istio troviamo la gestione della comunicazione e della condivisione dei dati tra i diversi microservizi. Inoltre, la piattaforma viene spesso aggiunta in modo da ridurre la complessità della gestione dei servizi di rete in applicazioni formate da molti microservizi.

Una volta installato, inietta dei proxy (sidecar) all'interno dei pod Kubernetes, accanto ai containers dell'applicazione. Ogni proxy è configurato per intercettare le richieste e instradare il traffico al servizio appropriato in base ai criteri adottati.

Istio consente il bilanciamento del carico, l'autenticazione da servizio a servizio e il monitoraggio, tipicamente senza modifica al codice di servizio. Il piano di controllo offre diverse funzionalità, tra cui:

- Comunicazione sicura da servizio a servizio in un cluster con crittografia TLS, autenticazione forte e autorizzazioni basate sull'identità,
- Bilanciamento automatico del carico per traffico HTTP, gRPC, WebSocket e TCP,
- Controllo granulare del comportamento del traffico con regole di routing avanzate, rinvio automatico, failover e iniezione di errori,
- Un livello di policy plug-and-play ed API di configurazione che supportano controlli di accesso, limiti di velocità e quote,

POR FESR EMILIA-ROMAGNA 2014/2020
FSC Fondo per lo Sviluppo e la Coesione
UNIONE EUROPEA Fondo europeo di sviluppo regionale
Regione Emilia-Romagna

- Metriche, log e tracce automatici per tutto il traffico all'interno di un cluster, inclusi il traffico in ingresso ed in uscita del cluster.

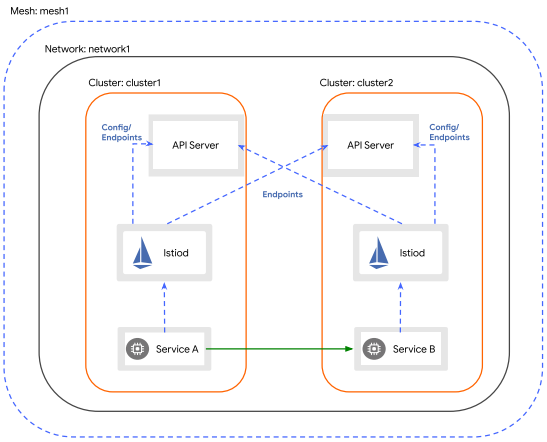
Istio è progettato per l'estensibilità e può gestire una vasta gamma di esigenze di distribuzione. Il piano di controllo di Istio viene eseguito su Kubernetes e supporta nuove applicazioni distribuite sia nel cluster principale che nella rete mesh. È possibile estendere la mesh ad altri cluster o connettere VM o altri endpoint in esecuzione al di fuori di Kubernetes.

Un ampio ecosistema di contributori, partner, integrazioni e distributori estende e sfrutta Istio per un'ampia varietà di scenari. Istio può essere installato in modalità stand-alone, oppure tramite fornitori, con prodotti che integrano Istio e lo gestiscono in modo trasparente allo sviluppatore.

Il modello di service mesh di Istio si basa su due componenti: il piano dati e il piano di controllo. Il primo è dedicato alla comunicazione tra i servizi. Senza service mesh, il sistema non è in grado di instradare il traffico prodotto e non può quindi prendere alcuna decisione basata su di esso. Questo problema viene risolto mediante l'utilizzo di un proxy, consentendo un'ampia gamma di funzionalità ad-hoc alle applicazioni in base alle configurazioni scelte. Nel caso Istio, il proxy prende il nome di Envoy proxy e viene distribuito al fianco di ognuno dei servizi eseguiti nel cluster o in alternativa messo in esecuzione sulle VM. Il piano di controllo, basandosi sulla propria vista dei servizi, configura dinamicamente i server proxy, aggiornandoli al variare delle regole o dell'ambiente circostante.

Un ulteriore motivazione alla scelta di Istio è la possibilità di installazione in multicluster, in linea con il setting distribuito di SBDIOI40. Istio multicluster è configurabile in quattro differenti topologie su sistemi multi-cluster: Multi-Primary, Primary-Remote, Multi-Primary on different networks, Primary-Remote on different networks.

2.2.1 Multi-Primary



La configurazione in

Figure 3 rappresenta l'installazione Multi-primary su cluster con visibilità diretta (stessa rete) di Istio. Il control plane Istio è presente sia sul cluster1 che sul cluster2, rendendo ciascuno un cluster primario. Entrambi i cluster risiedono sulla rete network1, quindi esiste una connettività diretta tra tutti i pod.

POR FESR EMILIA-ROMAGNA 2014/2020
FSC Fondo per lo Sviluppo e la Coesione
UNIONE EUROPEA Fondo europeo di sviluppo regionale
Regione Emilia-Romagna

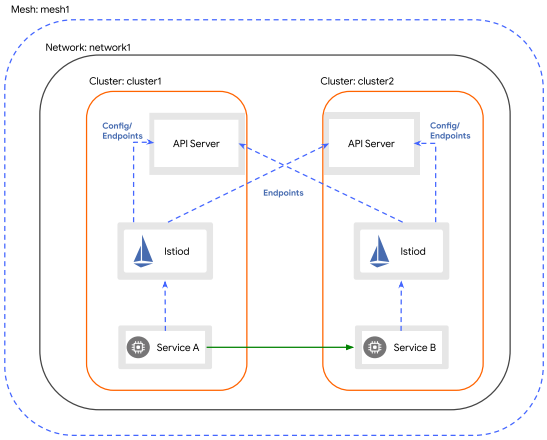


Figure 3. Configurazione Multi-Primary su stessa rete di Istio.

In questa configurazione, ogni piano di controllo dialoga con i server API in entrambi i cluster per fare discovery degli endpoint dei servizi. I carichi di lavoro comunicano direttamente (da pod a pod) attraverso i confini del cluster.

2.2.2 Primary-Remote

In questa configurazione (Figure 4), il cluster1 controlla entrambi i server API in entrambi i cluster per rendere disponibili gli endpoint dei servizi. In questo modo, il control plane sarà in grado di fornire il rilevamento dei servizi per i carichi di lavoro in entrambi i cluster. Il cluster 1, viene definito “primary cluster” ed il cluster 2 viene definito “remote cluster”.

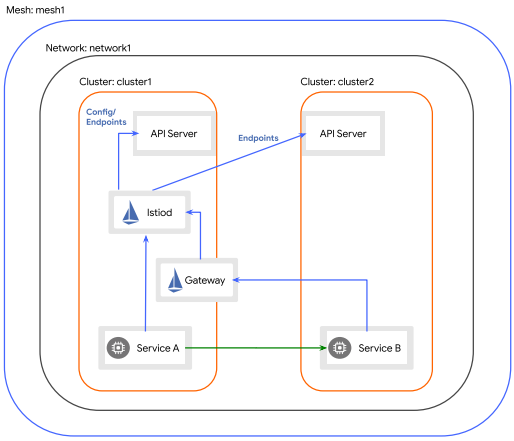


Figure 4. Configurazione Primary-Remote su stessa rete di Istio.

I carichi di lavoro del servizio comunicano direttamente (da pod a pod) attraverso i confini del cluster e sotto gli stessi endpoint di rete. I servizi nel cluster2 raggiungeranno il control plane nel cluster1 tramite un gateway dedicato per il traffico est-ovest.

2.2.3 Multi-Primary su reti separate

Nel caso di reti separate tra i cluster, è necessario applicare una differente configurazione, schematizzata in Figure 5, dove entrambi gli API server osservano i server API in ciascun cluster per gli endpoint.

POR FESR EMILIA-ROMAGNA 2014/2020
FSC Fondo per lo Sviluppo e la Coesione
UNIONE EUROPEA Fondo europeo di sviluppo regionale
RegioneEmilia-Romagna

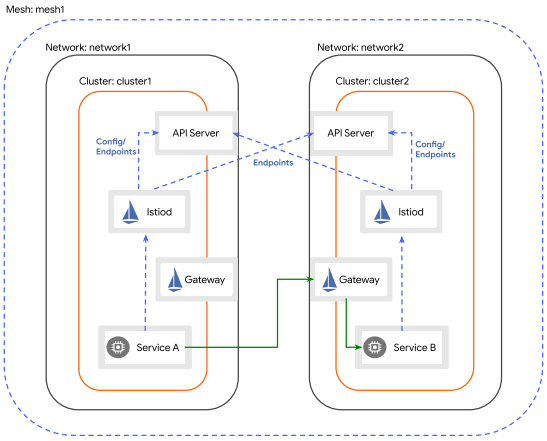


Figure 5. Configurazione Multi-Primary su reti separate di Istio.

I carichi di lavoro del servizio attraverso i confini del cluster comunicano indirettamente, tramite gateway dedicati per il traffico est-ovest. Il gateway in ogni cluster deve essere raggiungibile dall'altro cluster.

2.2.4 Primary-Remote su reti separate

In quest'ultima configurazione (Figure 6), il cluster cluster1 osserverà i server API in entrambi i cluster per gli endpoint. In questo modo, il piano di controllo sarà in grado di fornire il rilevamento dei servizi per i carichi di lavoro in entrambi i cluster.

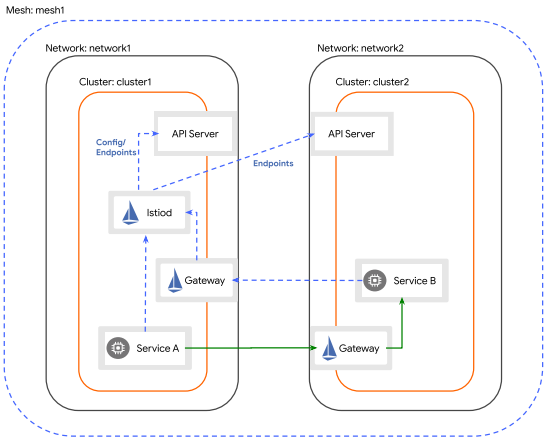


Figure 6. Configurazione Primary-Remote su reti separate di Istio.

I carichi di lavoro del servizio attraverso i confini del cluster comunicano indirettamente, tramite gateway dedicati per il traffico est-ovest. Il gateway in ogni cluster deve essere raggiungibile dall'altro cluster. I servizi nel cluster2 raggiungeranno il piano di controllo nel cluster1 tramite lo stesso gateway est-ovest.

POR FESR EMILIA-ROMAGNA 2014/2020
FSC Fondo per lo Sviluppo e la Coesione
UNIONE EUROPEA Fondo europeo di sviluppo regionale
Regione Emilia-Romagna

3 Orchestrazione di servizi

L'esistenza di migliaia di container in esecuzione in luoghi diversi e macchine fisiche (o virtuali) ha spinto gli sviluppatori a definire nuovi software necessari per coordinare e tenere sotto controllo le difficoltà. Queste applicazioni middleware vengono chiamati orchestratori.

Un orchestratore permette agli sviluppatori di avere una visione logica e di alto livello del sistema sottostante. In pratica, un orchestratore si occupa di: muovere i container di basso livello, replicare dei servizi e della tolleranza agli errori. Esistono diversi metodi per configurare un orchestratore. Ad esempio, esistono diverse applicazioni a riga di comando o applicazioni web che consentono di definire i parametri relativi ad ogni servizio: il giusto grado di replica, l'hostname desiderato e altre proprietà non funzionali. Tramite DNS di orchestrazione e piattaforme di discovery, il sistema può avere in ogni momento una vista aggiornata di tutti i container in esecuzione nelle diverse macchine e può fornire l'astrazione di avere un software unico su cui girano tutti i container ed i servizi.

L'orchestratore ad oggi più utilizzato è Kubernetes, il quale è già utilizzato come sistema di orchestrazione all'interno della piattaforma SBDIOI40. Per semplificare maggiormente la gestione dei servizi, il CIRI-ICT ha pensato di gestire i cluster della piattaforma tramite l'utilizzo di Rancher, strumento del quale parleremo nella prossima sezione.

3.1 Rancher

Rancher [6] non è propriamente un motore di orchestrazione, ma è definito dai suoi sviluppatori "Un software open source per fornire Kubernetes-as-a-Service".

È una piattaforma abilitante per la gestione del cloud computing multi-cluster. È stato originariamente sviluppato (a partire dalla versione 1.x) per funzionare con più orchestratori e incorpora un proprio orchestratore chiamato Cattle.

Grazie alla massiccia adozione di Kubernetes (tutti i principali provider di cloud prevedono una propria versione personalizzata), dalla versione 2.x, Rancher distribuisce e gestisce più cluster Kubernetes in esecuzione su qualsiasi provider. Una sola installazione del server Rancher può gestire centinaia di cluster Kubernetes dalla stessa interfaccia.

Rancher aggiunge anche funzionalità rilevanti a Kubernetes. Innanzitutto, permette di centralizzare la gestione di molteplici cluster dando la possibilità di controllarli in maniera uniforme. Consente quindi di integrare facilmente sistemi esterni di Continuous Integration (CI) e Continuous Deploy (CD) con semplici plug-in che, una volta configurati, automatizzano la consegna e la distribuzione dei carichi di lavoro. Con Rancher è inoltre possibile eseguire il monitoraggio e il controllo degli errori dei cluster e delle loro risorse. Infatti, consente di inviare tutte le informazioni di monitoraggio ad entità esterne (come Elasticsearch, Splunk e altri) in modo completamente trasparente e indipendente dall'applicazione. Pertanto, Rancher costruisce un intero ecosistema attorno a Kubernetes. Questo paragrafo mostrerà le principali caratteristiche e vantaggi che l'adozione di Rancher può portare agli sviluppatori.

POR FESR EMILIA-ROMAGNA 2014/2020
FSC Fondo per lo Sviluppo e la Coesione
UNIONE EUROPEA Fondo europeo di sviluppo regionale
Regione Emilia-Romagna

In Figure 7 è illustrato Rancher 2.0, composto da tre componenti logici principali: Application Workload Management, Unified Cluster Management e Rancher Kubernetes Engine (RKE).

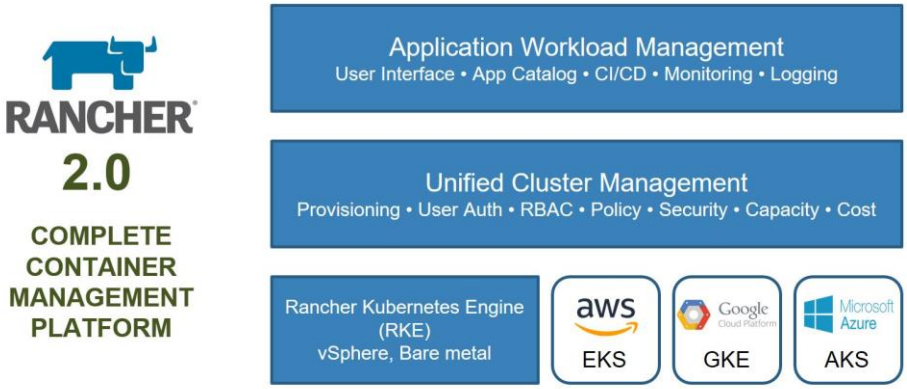


Figure 7. Componenti principali di Rancher.

3.1.1 Rancher Kubernetes Engine (RKE)

RKE è un semplice componente che permette di installare Kubernetes al volo in quei sistemi su cui non è preinstallato, come invece avviene in Amazon EKS, Google GKE e Microsoft AKS. È estremamente portabile e consente di installare Kubernetes su server bare metal, istanze VM in esecuzione su cluster OpenStack.

La distribuzione RKE consente di:

- Automatizzare il provisioning delle istanze container su molti cloud utilizzando Docker,
- Installare i master Kubernetes, i nodi etcd e i nodi di lavoro,
- Aggiungere o rimuovi nodi nei cluster Kubernetes esistenti,
- Aggiornare i cluster Kubernetes alle nuove versioni,
- Monitorare l'integrità dei cluster Kubernetes.

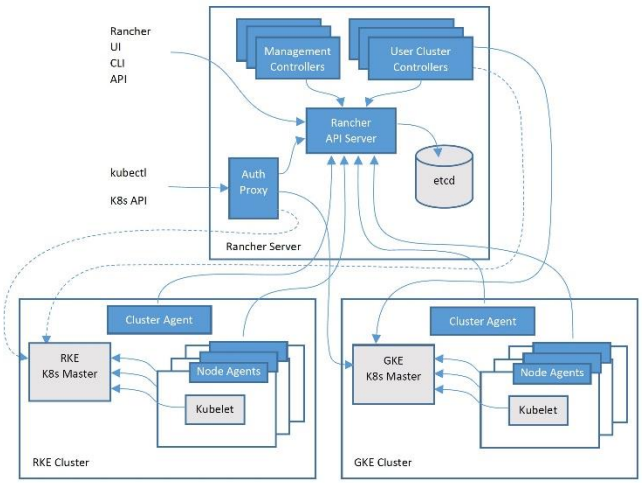


Figure 8. Schematico architetturale dei componenti di Rancher.

POR FESR EMILIA-ROMAGNA 2014/2020
FSC Fondo per lo Sviluppo e la Coesione
UNIONE EUROPEA Fondo europeo di sviluppo regionale
RegioneEmilia-Romagna

In Figure 8 viene mostrato la architettura di alto livello di Rancher e dei suoi componenti. Si può notare che i cluster in schematico sono di natura eterogenea, RKE messo in esecuzione da Rancher e GKE cluster di Google.

3.1.2 Gestione unificata dei cluster

Rancher consente agli sviluppatori di utilizzare i propri cluster Kubernetes o di utilizzare cluster gestiti da un provider cloud. La gestione unificata dei cluster garantisce un punto di autenticazione centralizzato univoco tra i cluster Kubernetes gestiti. Fornisce un proxy di autenticazione accessibile tramite API REST ai cluster.

È molto importante nelle grandi imprese disporre di un punto di accesso unico e uniforme a tutti i cluster gestiti. In effetti, tutti i principali fornitori di Kubernetes-as-a-service obbligano gli utenti ad avere un set di account (ad es. Account Google, account Amazon) per gestire i propri cluster. Con Rancher, uno sviluppatore aziendale può accedere alle piattaforme gestite da Kubernetes semplicemente utilizzando le credenziali di Active Directory gestite dall'azienda.

Un altro concetto che viene introdotto dalla Gestione Unificata dei Cluster è il Progetto. Quest'ultimo è rappresentato da uno o più spazi dei nomi Kubernetes associati alle loro politiche (come quota di risorse, proprietà e regole, ecc.). Gli sviluppatori possono creare progetti e assegnarli a singoli utenti o gruppi di utenti, suddividendo il sistema in un insieme di entità facilmente gestibili. In aggiunta a ciò, esiste la possibilità di personalizzare le operazioni CRUD che un utente o un gruppo di utenti può eseguire sul progetto, garantendo la massima sicurezza e flessibilità [7].

3.1.3 Gestione del carico di lavoro delle applicazioni

Tra le funzionalità più importanti fornite da Rancher, c'è quella di offrire un'interfaccia utente intuitiva per la gestione dei carichi di lavoro delle applicazioni. Rancher User Interface (UI) non tenta di nascondere i concetti di Kubernetes sottostanti, ma fornisce un'interfaccia utente di facile utilizzo per risorse Kubernetes native come pod e distribuzioni. Infatti, permette, con il semplice tocco di un pulsante, di replicare i contenitori in un pod.

In questo livello sono presenti anche applicazioni pronte per essere installate in un repository chiamato "Application Catalog". Questa funzione consente di avviare l'applicazione senza scaricare manualmente i container, distribuirli e scalarli sui cluster Kubernetes. Sono disponibili software per quasi tutti i sistemi CI/CD. Jenkins, Drone e Gitlab sono configurabili e pronti per essere utilizzati.

Infine, questo livello consente di monitorare i log e gli errori di ciascun cluster, includendo un insieme di applicazioni per l'esportazione e l'analisi dei log.

4 Deployment multi-cluster

Come step finale per lo sviluppo dell'infrastruttura cloud federata di SBDIOI40 è stato scelto di estendere il concetto di federazione anche alla comunicazione e composizione di servizi appartenenti a cloud separati fra loro. In particolare, come vedremo nelle prossime sezioni, è stato integrato un ulteriore cluster Kubernetes chiamato "K8s Imolab", il quale ci ha permesso di creare un service mesh distribuito sue due cluster con reti diverse gestito uniformemente attraverso Rancher.

POR FESR EMILIA-ROMAGNA 2014/2020 FSC Fondo per lo Sviluppo e la Coesione UNIONE EUROPEA Fondo europeo di sviluppo regionale Regione Emilia-Romagna

4.1 Testbed

In Figure 9 mostriamo la configurazione con la quale abbiamo testato la federazione dei due cluster Kubernetes. Il primo cluster è distribuito in modalità ibrida, ovvero:

- Tre nodi Kubernetes sono rappresentati da macchine virtuali create grazie all'infrastruttura cloud OpenStack del CIRI-ICT. Due dei nodi costituiscono dei worker Kubernets, mentre il terzo nodo svolge la funzione di master del cluster,
- Una macchina fisica è connessa al cluster con la funzione di worker Kubernetes.

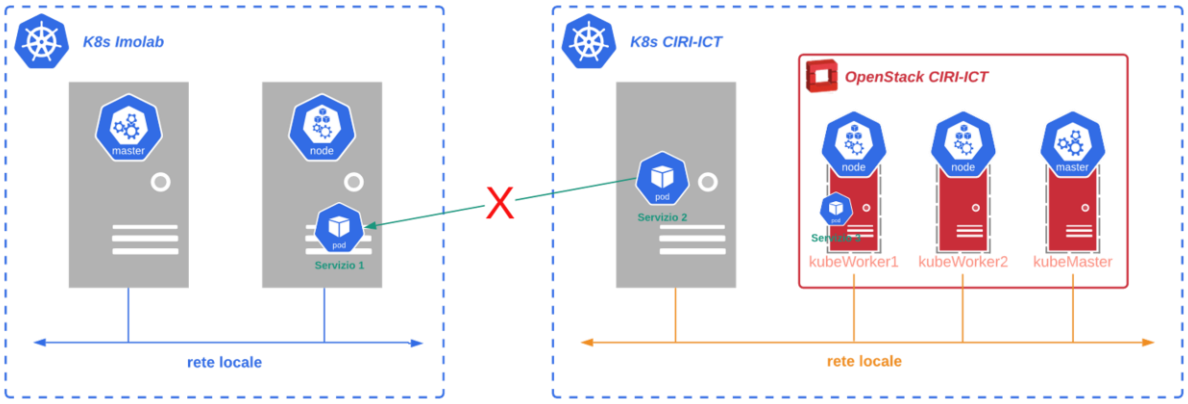


Figure 9. Architettura multi-cluster SBDIOI40 senza service mesh.

Il secondo cluster invece, è creato all'interno dell'infrastruttura cloud Imolab (gestita dall'azienda partner Imola Informatica). In questo caso il cluster è composto solamente da macchine virtuali.

Nella configurazione appena esposta, un servizio depositato sul cluster "K8s CIRI-ICT" non ha modo di comunicare con un servizio presente nel cluster "K8s Imolab". Questa limitazione è presente in quanto i due cluster risiedono su due cloud separati fra loro e di conseguenza su due reti che non sono direttamente connesse.

Per risolvere questo problema ed ampliare quindi la nostra infrastruttura, è stato necessario procedere con la federazione dei cluster Kubernetes grazie all'utilizzo di Istio in configurazione Multi-Primary su reti separate (sezione 2.2.3). Nella prossima sezione vengono esposti i risultati ottenuti.

4.2 Service mesh federato e risultati

Per quanto riguarda l'installazione di Istio, abbiamo proceduto con una configurazione manuale anziché utilizzare quella messa a disposizione da Rancher. Tale procedura si è resa necessaria in quanto i due cluster non condividono la stessa rete. Come si può vedere in Figure 10, grazie alla federazione, i servizi su K8s Imolab vengono resi accessibili a K8s CIRI-ICT tramite un reindirizzamento trasparente da parte del servizio Istio.

Di seguito sono descritti i passi necessari alla configurazione di Istio. Quanto descritto si riferisce ad un singolo cluster, ma è necessario ripetere tutti i passi su ogni cluster partecipante alla federazione.

POR FESR EMILIA-ROMAGNA 2014/2020
FSC Fondo per lo Sviluppo e la Coesione
UNIONE EUROPEA Fondo europeo di sviluppo regionale
Regione Emilia-Romagna

Per prima cosa una distribuzione mesh di servizi multicluster richiede di stabilire l'attendibilità tra tutti i cluster nella mesh. A seconda dei requisiti del sistema, potrebbero essere disponibili più opzioni per stabilire l'attendibilità. Per impostazione predefinita, la Certification Authority (CA) Istio genera un certificato radice e una chiave auto-firmati e li utilizza per firmare i certificati del carico di lavoro. Una CA Istio può firmare certificati del carico di lavoro utilizzando il certificato e la chiave specificati dall'amministratore e distribuire un certificato radice specificato dall'amministratore ai carichi di lavoro come radice di attendibilità.

Dopo aver messo in sicurezza Istio, si può procedere con l'installazione in versione Multi-Primary su reti differenti. Il tool utilizzato per l'installazione è `istioctl` il quale è stato scaricato e installato tramite repository ufficiale. Tramite i comandi forniti da `istioctl` è possibile impostare come "primary" entrambi i cluster e successivamente abilitare i due endpoint per il discovery reciproco.

In Figure 10 è schematizzato il flusso di comunicazione tra i due servizi sui cluster.

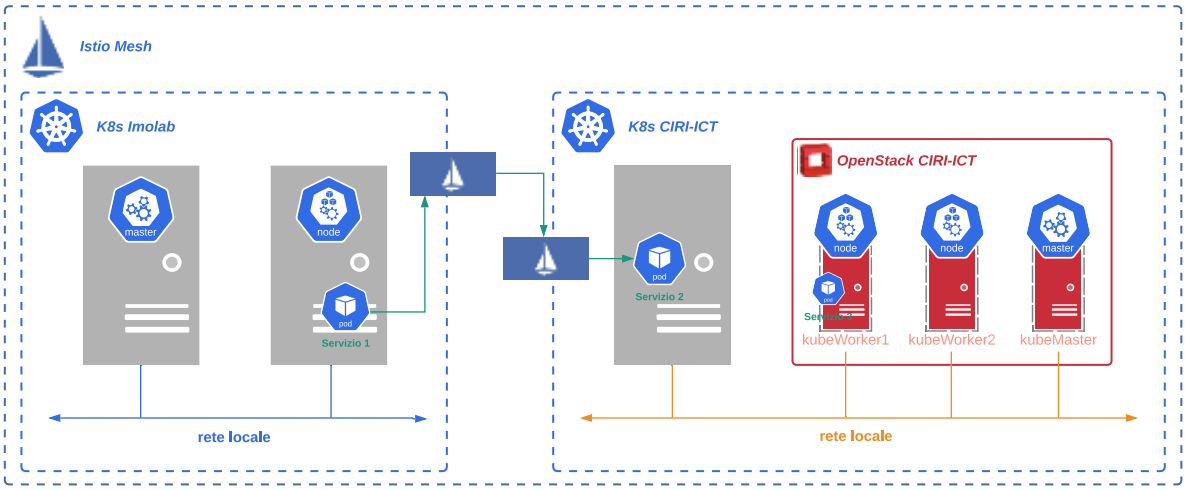


Figure 10. Architettura multi-cluster SBDIOI40 con service mesh.

Infine, mostriamo la dashboard per la gestione dei namespace presenti sui due cluster federati. In particolare, in Figure 11 è possibile visualizzare il progetto "SBDIO Istio Service Mesh" con auto-injection attiva per le sidecar di Istio. Grazie alle sidecar, tutti i servizi sviluppati e messi in esecuzione su tale progetto disporranno in maniera trasparente delle caratteristiche elencate nella sezione 2.1.

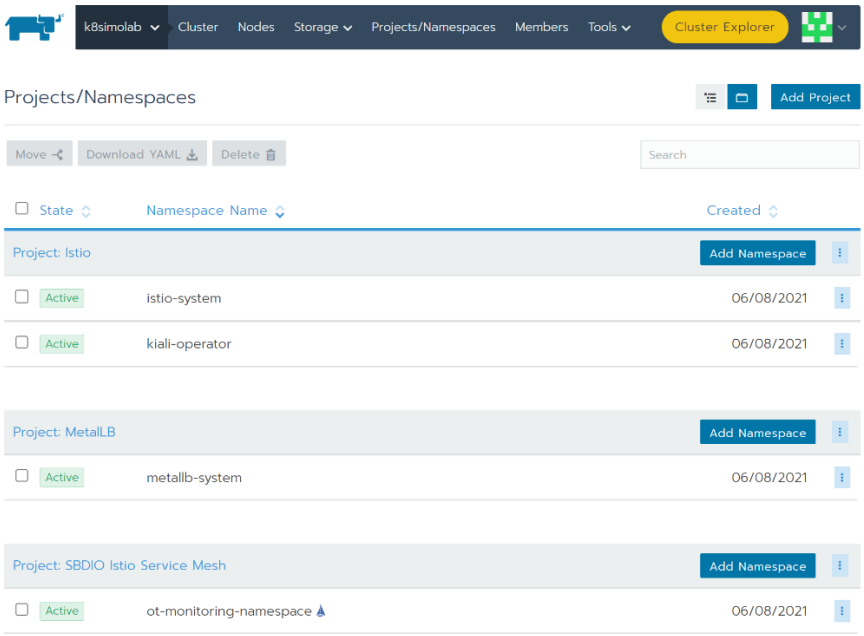


Figure 11. Interfaccia web integrata di Rancher per la gestione dei cluster.

5 Bibliografia

[1] DZone, «Hybrid and Federated Cloud Computing,» [Online]. Available: https://dzone.com/articles/common-hybrid-cloud-federation-models. [Consultato il giorno 1 Luglio 2021].

[2] OpenStack, «Software Development Kits,» [Online]. Available: https://wiki.openstack.org/wiki/SDKs. [Consultato il giorno 1 Luglio 2021].

[3] The Linux Foundation, «What is Kubernetes?,» [Online]. Available: https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/. [Consultato il giorno 1 Luglio 2021].

[4] NGINX, «What Is a Service Mesh?,» [Online]. Available: https://www.nginx.com/blog/what-is-a-service-mesh/. [Consultato il giorno 1 Luglio 2021].

[5] Istio , «Istio Service Mesh,» [Online]. Available: https://istio.io/. [Consultato il giorno 1 Luglio 2021].

[6] Rancher Labs, «Enterprise Kubernetes Management | Rancher,» [Online]. Available: https://rancher.com/. [Consultato il giorno 1 Luglio 2021].

[7] Rancher Labs, *Rancher: Technical Architecture - Version 2.1,* September 2018.

POR FESR EMILIA-ROMAGNA 2014/2020
FSC Fondo per lo Sviluppo e la Coesione
UNIONE EUROPEA Fondo europeo di sviluppo regionale
Regione Emilia-Romagna