Perchè scegliere una piattaforma di containerizzazione?
L'esperienza di CRIF Global Technologies

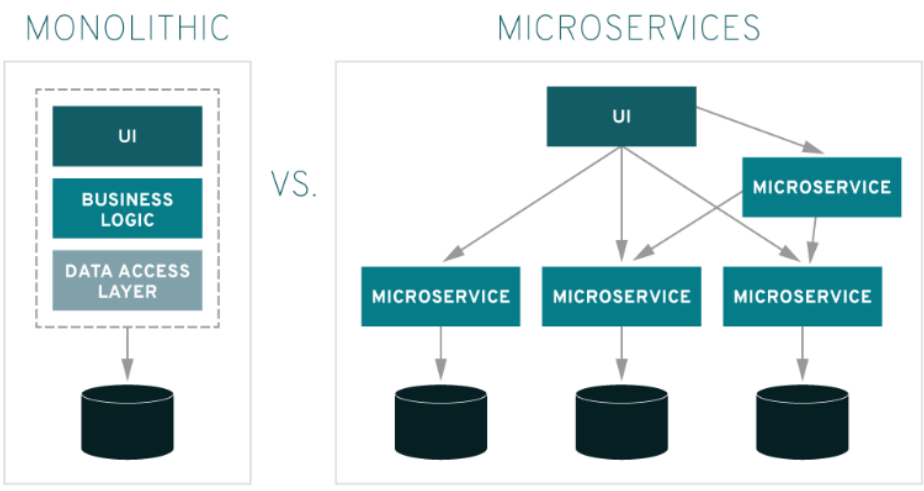
Il bisogno di flessibilità nell'operare in ambienti di produzione differenti, dall'on-premises al Cloud, ha determinato l'esigenza di astrarre le applicazioni disaccoppiandole sempre più dall'ambiente sottostante. Contributo fondamentale per la realizzazione di questo obiettivo è stato determinato dalla tecnologia a container. Attraverso tale astrazione, e con tecnologie di automazione sempre più performanti, si è ottenuto un deploy dell'applicazione più semplice con tempi di esecuzione e complessità ridotti. L'orchestrazione di tali oggetti è quindi divenuto punto centrale per l'innovazione dei processi di deploy. Per comprendere come si è arrivati all'adozione della piattaforma di orchestrazione OpenShift bisogna quindi partire da due aspetti chiave: l'architettura a microservizi e il suo rapporto con la containerizzazione.

1) Architetture a microservizi

In breve, "architettura a microservizi" è il termine che descrive la pratica di suddividere un'applicazione in un insieme di parti più piccole e specializzate, ciascuna delle quali comunica con le altre attraverso interfacce comuni come API e interfacce REST allo scopo di creare un'applicazione più grande. Con i microservizi le applicazioni sono quindi scomposte in elementi più piccoli comunicanti tra di loro pur rimanendo indipendenti da un punto di vista funzionale.

A differenza dell'approccio monolitico dove tutti i componenti sono creati e vivono all'interno di un unico elemento, i microservizi esprimono singole funzioni il cui comportamento non compromette il sistema d'interazione. Tale approccio di sviluppo, grazie alla particolare leggerezza di ogni componente, promuove la granularizzazione del codice e la condivisione dei processi simili tra più applicazioni fornendo le basi per un software di migliore qualità in tempi più rapidi rispetto alla programmazione convenzionale.

Data la natura della connessione tra i vari microservizi, in genere stateless, tale metodica permette di realizzare applicazioni con una maggiore tolleranza all'errore e preservando l'indipendenza del linguaggio di programmazione. Tuttavia, siccome la comunicazione tra i vari componenti avviene mediante interfacce, la parte di backend è completamente mascherata rispetto alla sua struttura e organizzazione. Pertanto, attraverso l'adozione dei microservizi si ottiene una scalabilità maggiore rispetto alle applicazioni monolitiche, una aumentata resilienza, più interoperabilità e apertura verso nuovi sviluppi e un veloce deployment delle applicazioni. Tutto questo si traduce in un time to market più tempestivo e una maggiore rispondenza alle esigenze di scalabilità del business.



2) Nozioni di base sui container

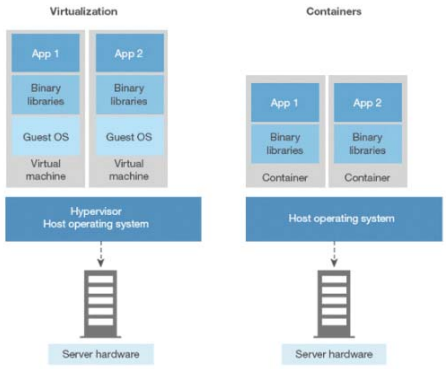
La tecnologia di base che di recente ha maggiormente contribuito al successo dei microservizi è quella dei container. Un container è simile a un'istanza di macchina virtuale, ma anziché includere un intero sistema operativo autonomo, può esser visto come uno "spazio autonomo" che si porta dietro solamente le librerie effettivamente necessarie al funzionamento dell'applicazione e tutte le loro dipendenze. Ciò significa che i container condividono il kernel del SO ma sono molto più leggeri di questo, si avviano più velocemente, utilizzano molta meno memoria rispetto a quanto ne richiederebbe l'avvio di un'intera VM, sono facili da distribuire rapidamente, localmente o in Cloud, e possono essere scalati facilmente orizzontalmente o verticalmente per soddisfare la domanda in relazione alle risorse disponibili.

Il disaccoppiamento tra container e sistema, su cui esso è in esecuzione, consente di eseguire facilmente e in modo coerente il deployment delle applicazioni basate su container, indipendentemente dal fatto che l'ambiente di destinazione sia un data center privato, Public Cloud o il computer portatile di uno sviluppatore. La containerizzazione consente una chiara separazione dei compiti, in quanto gli sviluppatori si concentrano sulla logica e le dipendenze dell'applicazione, mentre l'IT Operations può concentrarsi sul deploy e sulla gestione dell'applicazione, senza preoccuparsi di dettagli ad esso relativi, quali le versioni software e le configurazioni specifiche dell'applicazione.

Per quanto riguarda i microservizi questi prendono vantaggio dalla tecnologia a container per il fatto che ogni singolo microservizio può essere eseguito nel proprio contenitore, riducendo significativamente i costi di gestione dei servizi e favorendone la sua organizzazione.

Container e microservizi sono quindi partner perfetti, poiché rendono possibile, ad esempio, ridimensionare le singole applicazioni in modo indipendente, senza influire sugli altri servizi. Tuttavia, per ottenere questo risultato con una VM l'intero sistema dovrebbe essere ridimensionato. Inoltre, i container sono più piccoli, cosa che rende la distribuzione più veloce e c'è anche un enorme vantaggio in termini di portabilità. Per esempio, i microservizi "dockerizzati", possono essere eseguiti in container distribuiti su qualsiasi server con sw di containerizzazione già installato (per esempio Docker). Eseguire lo stesso come servizio su nuove macchine virtuali richiederebbe prima l'installazione del software con aggravio del tempo complessivo di deploy e una aumentata complessità dovuta alla preparazione in generale dell'ambiente ospite.

La maggior parte delle implementazioni di container dispone di strumenti di orchestrazione complementari che automatizzano l'implementazione, la gestione, il ridimensionamento, il networking e la disponibilità di applicazioni basate su container. È la combinazione di microservizi piccoli e facili da costruire e contenitori facili da distribuire che rende possibile la filosofia DevOps. Esistono diverse implementazioni del concetto di container, ma di gran lunga il più popolare è Docker, che è generalmente associato a Kubernetes come piattaforma di orchestrazione.



3) La gestione dei container nell'infrastruttura "classica". Problematiche e soluzioni

Ovviamente per l'esecuzione del container è necessaria una macchina su cui funzionare. Questa può essere con HW dedicato "bare metal" oppure, senza perdita di funzionalità, una VM dedicata. Dato che, come abbiamo detto in precedenza, un container impiega solo una frazione delle risorse che la stessa applicazione richiederebbe se fosse installata in maniera convenzionale, su una VM possono essere messe in esecuzione decine, a volte anche centinaia, di container.

In tale scenario, pur beneficiando dei vantaggi della containerizzazione, nel suo complesso non si evidenziano particolari miglioramenti in termini di efficienza e resilienza dell'applicazione. Questo perché utilizzando una macchina virtuale semplicemente come contenitore dove far eseguire i container non si risolvonoi problemi derivanti dalla gestione dell'applicazione stessa. Di seguito, alcuni esempi.

Alta affidabilità: se la VM dove i container sono in esecuzione va in crash, non c'è modo di garantire che i servizi che la utilizzano rimangano attivi, a meno di complicati artifici di fault-tolerance che implicano la ridondanza di container tra più host e l'utilizzo di bilanciatori di carico esterni. Ovviamente, l'applicazione deve tener conto di tale architettura.

Scalabilità: qualora le VM, che gestiscono il servizio, non siano più in grado di far fronte all'aumento dei volumi di carico, è necessario aggiungere altri host dove installare copie dei container da gestire con logiche di bilanciamento. Questa operazione richiede tempo per la configurazione dei nuovi ambienti aggiuntivi, introduce ulteriori costi per componenti di bilanciamento esterni e non dà garanzie sull'efficienza della scalabilità.

Durata del deploy: l'attività di deploy è eseguita macchina per macchina con importante impatto temporale e problematiche di gestione del servizio legate all'indisponibilità della macchina in questione. Eventuali meccanismi di versioning devono essere gestiti manualmente così come procedure di roll-back, qualora a seguito di un deploy dovessero insorgere problemi all'applicazione. Non è inoltre possibile, se non con complicati artifici, attuare strategie di test mirate per singole sorgenti di richiedenti (vedi "canary deploy", ad esempio).

Resilienza: se il container in esecuzione sulla VM va in crash è necessario implementare importanti meccanismi di controllo per averne evidenza, così come script per il loro riavvio automatico qualora possibile.

Sicurezza: a livello di singola VM non esiste la possibilità di effettuare la micro segmentazione dei container. Se questi insistono sulla medesima VLAN, ogni container può potenzialmente comunicare con tutti gli altri rendendo impossibile la segregazione a livello di rete a meno di complicati artifici a livello applicativo.

Monitoraggio: il monitoraggio delle risorse interne al container risulta essere difficoltoso qualora venga eseguito con metodiche convenzionali attraverso l'osservazione della VM. Può capitare che la memoria del container sia in uno stato di hang pur rilevando uno stato normale della memoria utilizzata dalla VM.

CI/CD: l'utilizzo di container in un ambiente VM è compatibile con l'automazione dei processi di CI/CD ma per averne il pieno controllo è necessaria una sovrastruttura spesso non facile da realizzare. Allo stesso modo, l'integrabilità con strumenti di autorizzazione e tracking (vedi Jira, ad esempio) si realizza a costo di enormi sforzi di configurazione e con limiti spesso non accettabili.

Questi sono solo alcuni degli aspetti che inducono a pensare che l'utilizzo della containerizzazione in ambiente VM sia una strada che non ottimizza i processi dell'IT Operations. Si rende quindi necessario pensare ad ambienti più evoluti che possano amplificare il vantaggio di questi oggetti, semplificando la vita alle IT Operations e fornendo un ambiente più robusto per le esigenze del business.

Pertanto, i pilastri per ottenere una gestione delle applicazioni semplice e flessibile sono il Cloud Computing e l'Orchestrazione.

Con Cloud computing intendiamo la distribuzione di risorse (computer, storage, ecc.) allo scopo di offrire risorse flessibili e scalabili, con struttura di costo prevalentemente operativo e capacità di astrazione dello strato infrastrutturale. In sintesi, si tratta di una struttura in grado di sottostare a logiche di automazione per il provisioning di risorse, scalabile sia orizzontalmente sia verticalmente, e che offra capacità di risposta rapida alle esigenze computazionali dei servizi collegati.

La disponibilità di tale servizio (IAAS – Infrastructure as-a-service), risolve essenzialmente il problema della scalabilità delle risorse su cui far funzionare i container ma non affronta la scalabilità dei container

stessi. Serve quindi un “ponte” tra lo strato infrastrutturale, abilitante alla gestione del servizio, e l’applicazione stessa. Si arriva quindi al PaaS (Platform as-a-service) che può essere visto come la connessione tra l’applicazione e la parte sottostante che ulteriormente rende “astratta” la parte HW e di Sistema Operativo permettendo all’utilizzatore di dedicarsi esclusivamente all’ambiente operativo di sviluppo o di produzione sfruttando le dinamicità e caratteristiche degli strati sottostanti senza doverli conoscere.

Parlare quindi di Paas Containerization significa fare riferimento a qualcosa che sia in grado di:

1) liberare gli utilizzatori dai compiti più meccanici e ripetitivi;
2) interagire con più gruppi di containers allo stesso tempo;
3) fornire servizi di rete, monitoraggio, telemetria, sicurezza e accounting;
4) pianificare e implementare un servizio di Registry distribuito geograficamente;
5) connettere in maniera trasparente più scenari per gestire un unico cloud virtuale geograficamente distribuito.

Pertanto, la scelta di una architettura SW organizzata in microservizi che poggia su un PaaS capace di offrire scalabilità, resilienza ed efficienza nell’infrastruttura permette di liberare l’utilizzatore da tutti gli oneri non direttamente connessi al servizio che sta sviluppando. **Tutto questo permette la realizzazione di una nuova metodica di erogazione dei servizi che sia quindi moderna, efficace, flessibile e time2market.**

Piattaforme in grado di realizzare quanto sopra descritto e offrire le caratteristiche necessarie a una corretta gestione dei container sono oggi molteplici, ognuna delle quali con specifiche caratteristiche che le differenziano dalle altre.

Dalle analisi condotte da CRIF Global Technologies, la divisione IT di CRIF, è emerso una maggior corrispondenza alle nostre caratteristiche, sia di sviluppo sia di IT Operations, per la piattaforma Kubernetes. Si tratta di una piattaforma Open Source di Google particolarmente adatta a cluster di medie e grandi dimensioni utilizzate per la gestione di applicazioni complesse. Kubernetes è particolarmente versatile ed è in grado di gestire deploy su larga scala.

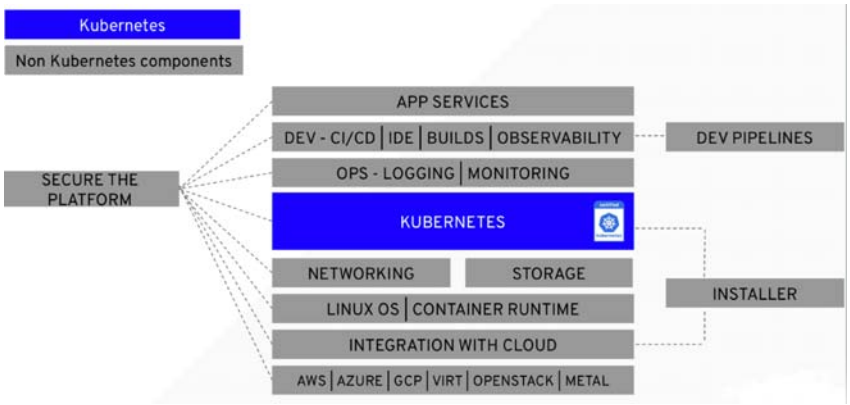
Di seguito i principali vantaggi:

1) orchestrazione di container tra più Host;
2) organizzazione in POD (gruppo di container);
3) ottimizzazione delle risorse;
4) controllo e automazione del deployment;
5) scalabilità in real-time delle applicazioni e risorse;
6) controllo e autocorrezione delle App.

Appurata l’aderenza di Kubernetes alle specificità aziendali, ci si è posti la domanda sulla opportunità di proseguire nella scelta di un prodotto totalmente open o se raccoglierne tutte le caratteristiche spostandosi verso un prodotto Enterprise. Dopo attenta analisi si è optato per la seconda strada scegliendo **OpenShift**, una piattaforma con tutte le caratteristiche di Kubernetes (condividendone la totale compatibilità) ma con il vantaggio di presentare una maggiore stabilità garantita dal supporto Red Hat e con l’aggiunta di alcune funzionalità non presenti nel prodotto open.

4) Prodotto commerciale vs open source: le ragioni della scelta

Come detto Kubernetes è un progetto “open source” o, meglio ancora, framework open. La sua caratteristica fondamentale è che la piattaforma nasce all’interno di una Community che lo mantiene e aggiorna con le funzionalità richieste dagli utilizzatori stessi, a scapito di una non facile configurazione. In particolare, i prodotti di livello Enterprise sono spesso degli adattamenti di versioni “open” ai quali sono aggiunte le funzionalità e adattamenti necessari per offrire una suite completa: adatta a un ambiente Corporate con una user experience semplice e una facile configurazione. Nel caso specifico di CRIF, la scelta è ricaduta sulla piattaforma OpenShift; sebbene al suo interno vi sia l’engine Kubernetes, su quale Red Hat ha costruito un sistema perfettamente integrato di componenti atti a facilitarne l’utilizzo e le funzionalità di deployment. In sintesi, **si potrebbe dire che OpenShift altro non è che Kubernetes in versione stabile con l’aggiunta di alcune funzionalità utili per semplificare il processo di deploy, per la gestione dei container e per la sicurezza.**



Tali funzionalità sono ovviamente ottenibili anche con un prodotto opensource ma al costo di laboriose integrazioni di software e impegnative attività di customizzazione che spesso rendono non efficiente il processo di gestione degli ambienti di produzione in una logica Enterprise. Le funzionalità aggiuntive sono prevalentemente orientate alla gestibilità del container e alla sua sicurezza. Nella tabella sotto sono riportate alcune tra le principali differenze tra i due ambienti che hanno determinato la scelta finale.

	Openshift	Kubernetes
OpenShift product vs. Kubernetes project	You need to pay subscriptions but it provide supports and include several features that enhance it. Less release that Kubernetes	It is based on the community development and its model is free and self-supported. It is necessary to add external products for an enterprise usage.
Security	«Secure by default» approach. A lot of security features are enabled by default on OS (integration with AD, no root execution, logging, etc.)	There is a different approach and you need to configure the security «case by case». Big effort in working to secure the environment.
Publishing container	Complete integration with BIG-IP F5	Kubernetes uses a very good approach to do it but les mature the OS under the integration poin of view.
Approach to deployments	OpenShift DeploymentConfig has more options and support ImageStream	It works well but having less options often involve in more deeply configuration and effort for managing it
Management of container images	By using «imageStream» feature in OS, management of container is easier than Kubernetes and it requires les effort	The lack of integrated tools requares more effort.
Integration with Jenkins	additional feature of OpenShift makes it easy to deploy your apps with CI/CD pipelines. Native integration	Integration is possible but it needs more effort to be achieved and there is no support in case of isues.

In fase di analisi sono stati considerati i parametri tecnologici ma anche la diffusione e la quota di mercato del software in esame. Questo a garanzia di un più semplice reperimento delle competenze sia in ambito locale sia internazionale per altre realtà del Gruppo.

5) Architettura del prodotto di gestione container selezionato in CRIF

Una delle caratteristiche fondamentali del prodotto scelto per la gestione dei container è quella di essere una estensione della piattaforma Kubernetes e può essere installata su molteplici Cloud Provider o nel proprio Datacenter. A differenza di Kubernetes, che come detto è un framework opensource con funzionalità non immediatamente “pronte all’uso”, garantisce un approccio alla sicurezza cosiddetto “by design”, al costo però di una maggiore rigidità rispetto al software open che rimane senza dubbio la scelta più opportuna qualora si privilegi la flessibilità intrinseca.

Alla base del suo funzionamento c’è Red Hat Enterprise Linux CoreOS (RHCOS), un sistema operativo orientato ai container che combina le migliori funzionalità di CoreOS e Red Hat Atomic Host Operative System.

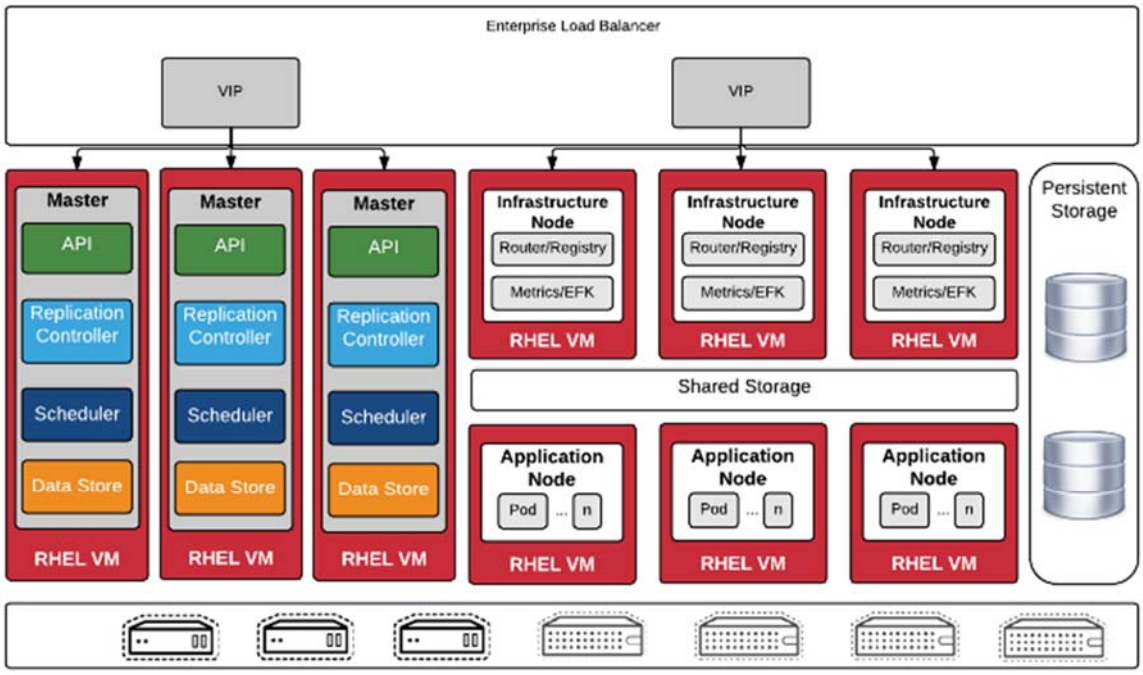
A partire dalla versione 4.x, attraverso uno specifico account, è possibile l’installazione della piattaforma attraverso un singolo comando all’interno dei principali cloud pubblici o direttamente collegandosi alla consolle VMWare. Questa caratteristica, non presente in Kubernetes, permette il deploy della piattaforma in modalità automatica e, in esercizio, permette di disporre di tutte le VM di cui necessita in logica di auto-provisioning attraverso l’infrastruttura di macchine virtuali.

Nella figura sotto riportata è possibile vedere una schematizzazione della sua architettura, nella quale si identificano 3 ruoli: *Master Node*, *Infrastructure Node* e *Application Node* (anche chiamati worker).

Kubernetes e la piattaforma di controllo risiedono sui nodi Master che, per questioni di affidabilità, devono rispettare la regola del 2n+1. Abbiamo quindi che il numero minimo di nodi Master contenuti in un cluster è pari a 3.

Sui nodi cosidetti “workers” risiedono i microservizi in esecuzione. Questi rappresentano la capacità elaborativa vera e propria del cluster. Essendo l’esecuzione delle applicazioni totalmente gestita da questi nodi, è evidente che alla loro scalabilità orizzontale è affidata la capacità dell’intera infrastruttura di soddisfare il livello di performance richiesto. Da qui emerge una delle più importanti caratteristiche della piattaforma. In caso di overload, se la configurazione è stata opportunamente eseguita, OpenShift attraverso il controll plane dell’infrastruttura (cloud, vmware, ecc.) scala orizzontalmente aggiungendo nodi affinché il livello di performance richiesto sia sempre soddisfatto. Per quanto riguarda subscription, i worker sono gli unici nodi che devono disporre di licenze d’uso.

Inoltre, esistono altri componenti all’interno della piattaforma necessaria al suo funzionamento (routers, registry, storage, ecc.) per le quali non scendiamo nei dettagli tecnici, ma ne diamo una visione d’insieme nella figura sotto.



6) La piattaforma di containerizzazione come elemento d’innovazione e opportunità per il business

È ormai opinione diffusa e consolidata che stiamo vivendo una esperienza senza precedenti di cambiamento nelle modalità in cui il business opera, guidato prevalentemente dalla necessità d’inseguire il comportamento dei clienti a una velocità a volte anche maggiore rispetto a quella con cui gli utenti finali manifestano le loro esigenze.

Negli ultimi anni, e il futuro non lascia presagire cambiamenti in tal senso, la velocità con la quale si riesce a soddisfare i mercati e la capacità di disegnare i propri prodotti e servizi in maniera conforme alle singole esigenze dei clienti sono elementi specifici della capacità di competizione nel proprio business. Al fine di permettere questo nuovo modo di fare mercato, è necessario che l’IT si doti di strumenti flessibili, capaci di automatizzare le operazioni necessarie alla messa in produzione dei nuovi prodotti, scalabili e veloci nella realizzazione delle integrazioni.

Contesti di deployment ibridi e multicloud sono quindi divenuti i nuovi standard e sono sempre meno diffuse le organizzazioni che in una qualche forma non stiano pensando, o già lo fanno, di adottare strategie cloud su cui installare le proprie applicazioni costruite su logiche a microservizi facendo uso di container.

Red Hat OpenShift è stato disegnato proprio per rispondere a queste esigenze e fornire una piattaforma in grado di operare con un approccio di "automation-first" fornendo l'esperienza di cloud per il mondo ibrido.

In tale contesto, la piattaforma si pone come obiettivo quello di fornire i seguenti elementi:

1) Capacità di autogestione mediante utilizzo di procedure automatiche per l'update e la manutenzione della sua infrastruttura, offrendo trasparenza alle IT Operations che non necessitano di verticalità in tale contesto.
2) Facile integrazione con la maggior parte dei cloud esistenti (AWS, Google Cloud, Azure, ecc.) e con framework tecnologici per private cloud come, ad esempio, Open Stack o virtualizzazione.
3) Deployment semplificato e gestione del ciclo di vita dell'applicazione attraverso operatori Kubernetes.

Con tali caratteristiche è necessario scegliere la piattaforma di gestione dei container che meglio supporta l'accelerazione digitale delle aziende attraverso una piattaforma in grado di fornire agli sviluppatori e personale di IT Operations caratteristiche quali:

1) Possibilità di disporre di logiche di self-provisioning per i servizi necessari alle loro applicazioni fornendo strumenti totalmente automatizzati per build e deploy. Questo permette agli sviluppatori di creare e pianificare il deploy delle applicazioni presenti nel catalogo mediante processi automatizzati, assicurandone tuttavia il pieno controllo da parte del team delle IT Operations.
2) Gestire le logiche di sicurezza nella comunicazione tra microservizi mediante la funzionalità di Service Mesh liberando sviluppatori e gestori di rete dalle incombenze necessarie per la sua realizzazione in modalità classica.
3) Disporre di una piattaforma ideale per la gestione di servizi installati su logiche "server-less" o "function as a service" (FaaS).
4) Possibilità di operare in ambiente Multi Cluster attraverso più Datacenter e/o Cloud aumentando la disponibilità delle applicazioni senza rinunciare alla velocità e semplicità dei deploy.
5) Possibilità di muovere le applicazioni attraverso più Cloud provider e possibilità di utilizzare questi ultimi solamente per porzioni di applicazioni, pur rimanendo il controllo centralizzato.
6) Gestire una sola release di prodotto a livello di sviluppo e portarlo in UAT e successivamente in produzione con un solo click. Questo approccio abbatte drasticamente i tempi di deploy in quanto non è più necessario, una volta che l'applicazione è stata testata sia sotto il profilo della qualità sia della sicurezza, richiedere l'intervento delle IT Operations per portare un'applicazione in produzione.
7) POD Autoscaling per fornire la possibilità di far crescere (o descrescere) le risorse a disposizione dell'applicazione qualora ci sia una variazione in termini di richieste cliente senza l'intervento del sistemista.
8) Elevata portabilità dei container tra un ambiente e l'altro in maniera trasparente all'applicazione permettendo di ridistribuirli su altre piattaforme, di testare e "deployare" l'applicazione su qualsiasi tipologia di ambiente, cloud e on-premises e su diverse tecnologie che supportino i container Docker-formatted evitando il vendor lock-in e consentendo facilmente la migrazione dell'ambiente di sviluppo.
9) Possibilità di eseguire "canary deployment" o "deployment blu-green" affinché sia possibile il rilascio delle applicazioni prevedendo il trasferimento "graduale" del traffico utente da una versione a un'altra, per testare nuove funzionalità solo per gruppo ristretto di utenti senza necessità d'installare nuovi sistemi o laboriose configurazioni di rete.
10) Integrabilità completa con tool di automazione e tracking quali Jenkins e Jira per l'esecuzione completa di deploy a partire dalle richieste di business.
11) Ripartenza con versione pregressa in caso di deploy non positivo o in caso di problemi sulla qualità del nuovo codice.

Queste caratteristiche, assieme a tante altre funzionalità utili per semplificare e velocizzare la "fabbrica IT", rendono evidente quanto impattante e innovativo sia l'utilizzo di tale piattaforma all'interno di un contesto che richieda un time2market sempre più ridotto.

7) Sicurezza

Se da una parte la piattaforma scelta è nata per semplificare e automatizzare le operazioni di sviluppatori e sistemisti fornendo strumenti agili per la gestione dei container, dall'altra è necessario che abbia provata robustezza sotto il profilo della sicurezza. Pertanto è fondamentale fare una scelta di piattaforma dove sia fondamentale il concetto di "security by design" rendendo disponibile una moltitudine di funzionalità a garanzia di un elevato livello di sicurezza dei container. In una logica applicativa basata su microservizi, la comunicazione tra i singoli componenti è quindi imprescindibile. Se in scenari formati da pochi elementi la modalità di comunicazione può essere facilmente gestita, in contesti complessipuò diventare difficile una gestione manuale. È quindi importante gestire le policy passando da un livello applicativo a quello infrastrutturale. **Nell'architettura scelta da CRIF si è data particolare attenzione alla scelta di caratteristiche in grado di combinare il controllo e la sicurezza dei container con una efficace gestibilità e tracciabilità del deploy dei microservizi.** Gli sviluppatori possono quindi focalizzarsi sulla realizzazione dei singoli componenti logici lasciando al sistema l'onere di gestirne la comunicazione attraverso policy estremamente puntuali e fornendo una visualizzazione delle varie comunicazioni e interdipendenze. Molto utile in fase di analisi delle problematiche anche il tracking delle performance, in stile APM, dei vari container basandosi su baseline opportunamente predisposte dal sistema stesso. Queste funzionalità e altre focalizzate al controllo e visibilità di cosa sta accadendo nella comunicazione tra microservizi sono gestibili attraverso API messe a disposizione degli sviluppatori per una migliore esperienza generale del sistema.

Attraverso la configurazione di regole di accesso e opportune politiche di routing è possibile quindi controllare il traffico tra i vari microservizi e le chiamate API gestendo l'autenticazione, l'autorizzazione e l'encryption di ogni canale di comunicazione e segregando l'applicazione o parte di essa a utenti o altri servizi non autorizzati. Tutto questo senza far ricorso a infrastrutture di sicurezza esterne (Firewall, proxy, ecc.).

In un tipico scenario multi tenancy, dove più clienti condividano la stessa piattaforma, la possibilità di gestirli in contesti completamente separati dal punto di vista logico fino ad arrivare al singolo microservizio diventa quindi punto di forza nell'utilizzo di questa applicazione.

Attraverso strumenti aggiuntivi, quali ad esempio Clair, è possibile condurre analisi statiche di vulnerabilità dei container intercettando eventuali minacce presenti nelle immagini e riducendo la probabilità di portare vulnerabilità in produzione. Questo aspetto è coadiuvato da altre funzionalità facilmente configurabili sulla piattaforma che hanno lo scopo di rendere maggiormente sicure le immagini riducendo eventuali disservizi una volta in produzione. È il caso, ad esempio, del Container Signing, funzionalità attraverso la quale solo le immagini "certificate" possono essere utilizzate per produrre applicazioni o particolari configurazioni delle autorizzazioni affinché non sia mai possibile scalare su utenze di root attraverso container compromessi.

Le possibilità d'intervento per raggiungere un buon livello di sicurezza dei container sono quindi molteplici. A seconda delle particolari esigenze è possibile utilizzare quelle direttamente messe a disposizione dalla piattaforma o rivolgersi a tool facilmente integrabili.

8) Conclusioni

Mediante l'utilizzo di una corretta piattaforma per la gestione dei container, i team Devops sono in grado di sviluppare e rendere disponibili applicazioni e nuove funzionalità occupandosi sempre meno degli aspetti relativi al deploy e alla gestione dell'infrastruttura su cui questi sono eseguiti. Il rilascio del codice può essere demandato a processi di CI/CD opportunamente orchestrati così da ridurre notevolmente il tempo necessario nella fase finale della messa in produzione aumentando nel contempo la qualità totale e il livello di performance per l'azienda.

Infine, osservando la parte economica, bisogna dire che potendo operare su una piattaforma condivisa attraverso l'utilizzo di container, microservizi e supporto nativo per multitenancy, il numero totale di

server necessari per singola applicazione si riduce drasticamente permettendo importanti risparmi a vantaggio di una rinnovata marginalità per il business. Tale soluzione è già nativa all'interno di **CRIF Digital Next**, la piattaforma aperta e collaborativa che ti consente di accelerare la trasformazione digitale dei clienti, di far evolvere la user experience degli stessi e di generare un processo di innovazione veloce e continuo.

Per maggiori informazioni: infoglobaltechnologies@crif.com