DOCKER: POTENZIARE LE STRATEGIE DI SVILUPPO IN UN AMBIENTE DEVOPS

Gabriele Provinciali

La virtualizzazione, per il personale IT coinvolto nella preparazione degli ambienti di sviluppo, di collaudo e di esercizio, non è certo cosa nuova: l'evoluzione degli strumenti che consentono una replica completa di un sistema operativo – e delle componenti applicative necessarie – in una Virtual Machine, nel contesto di una più ampia gestione del ciclo di vita applicativo, ha indubbiamente reso più facile la gestione di un Data Center complesso. Allo stesso tempo, però, l'utilizzo intensivo di sistemi virtualizzati ha innescato una serie di preoccupazioni concernenti la possibile disseminazione non controllata di macchine virtuali (*VM sprawling*) che aprono la strada a rischi relativi alla sicurezza, ad una corretta gestione del *licensing*, ed alla efficienza operativa nella gestione di un elevato numero di VM, soprattutto nella fase di design e sviluppo del software.
La tecnologia dei *containers*, che consente la convivenza di multiple istanze *user-space* nel sistema operativo, ognuna delle quali gode di caratteristiche pregiate quali *isolamento* e *gestione delle risorse*, ha letteralmente rivoluzionato da qualche anno la comunità degli sviluppatori – specialmente nelle realtà DevOps, dove i meccanismi di continuità tra sviluppo e produzione corrono su binari ultraveloci coinvolgendo in maniera attiva entrambi i mondi – che ha adottato rapidamente la metodologia correlata e la possibilità di costruire un ambiente completo in pochi secondi.
Misurato con un cronografo, il tempo necessario per la creazione di un container, il download di un'immagine con l'ultima versione di Tomcat e la sua messa in esecuzione in un ambiente di sviluppo, può durare meno di 30 secondi: un setup valido consente di avere un ambiente isolato e pronto per il deployment di applicazioni Java senza grande dispendio di tempo e risorse.
I perché di questa diffusione e di questo successo, nel comparto di sviluppo, sono facilmente intuibili: l'*assenza* di vincoli relativi alla configurazione di un dispositivo intermediario (una VM), la *velocità* di esecuzione (le chiamate di sistema sono native e non sono dirette verso una virtualizzazione di tipo classico) e la *libertà* di essere indipendenti da piattaforme hardware normalmente supportate da Virtualizzatori di Sistema e HyperVisor rendono i containers appetibili da ogni team, a prescindere dalla loro filosofia di sviluppo del software.
Inoltre, le comunità Open Source e le aziende che producono software commerciale hanno messo a disposizione una enorme quantità di *container images* attraverso cataloghi e *registries* pubblici (ad esempio, DockerHub[1]) molto facili da consultare per creare – al volo – l'immagine containerizzata preferita.

1 http://hub.docker.com

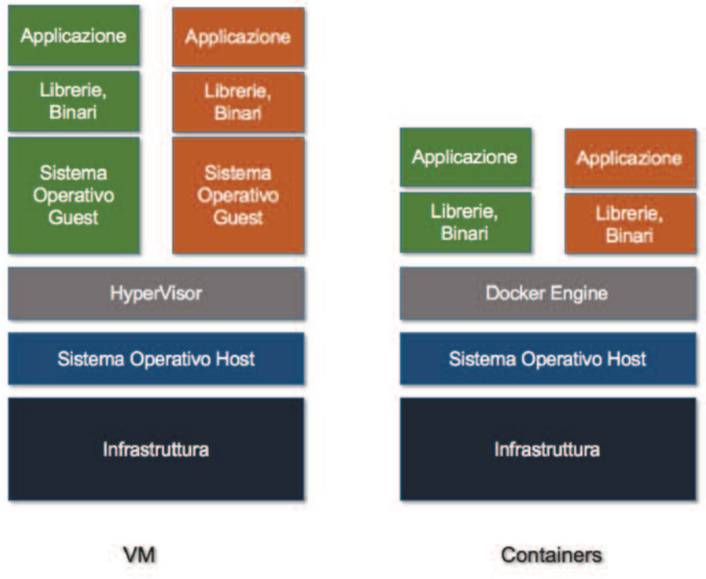
IL TEMPO NECESSARIO PER LA CREAZIONE DI UN CONTAINER, IL DOWNLOAD DI UN'IMMAGINE CON L'ULTIMA VERSIONE DI TOMCAT E LA SUA ESECUZIONE IN UN AMBIENTE DI SVILUPPO, PUO DURARE MENO DI 30 SECONDI

Gabriele Provinciali è Master Principal Solution Architect di Oracle Italia dal 2013. Svolge il ruolo di Technology Advisor con una particolare specializzazione sullo sviluppo software, sul Cloud Computing, sulle architetture orientate ai servizi, sulle piattaforme di Mobile Computing e sulle metodologie correlate a DevOps. Provinciali vanta una lunga esperienza nel settore delle telecomunicazioni (Ericsson, Marconi, British Telecom) e nelle aziende IT (Sun Microsystems, SilverStream, EMC, IONA Technologies, BEA Systems, CA Technologies). Lavorando in Italia e in UK, si è occupato di architetture software, di progetti mission critical in ambienti di elaborazione distribuita e di software design per i sistemi real-time.

CSA Italy è un'associazione no profit italiana nata nel 2011 come capitolo nazionale dell'associazione internazionale CSA (Cloud Security Alliance) a cui aderiscono le maggiori aziende del settore ICT ed Information Security che hanno scelto il Cloud Computing come parte rilevante del loro business. CSA coordina una community di professionisti che contribuiscono attivamente a sviluppare linee guida e buone pratiche per uno sviluppo ed utilizzo in sicurezza del Cloud.

COME FUNZIONA UN CONTAINER?

Anche se non gode di tutte le caratteristiche e dei privilegi di una Virtual Machine, un container è sicuramente più leggero e non richiede, almeno su Linux, l'utilizzo di grandi quantità di risorse. Come già accennato, la logica dei *lightweight containers* non richiede la replica di un intero sistema operativo ma utilizza le caratteristiche native del sistema operativo per istanziare una zona partizionata, all'interno della macchina, che incorpora caratteristiche interessanti (e necessarie) di isolamento rispetto al Sistema Operativo Host: l'idea di base è la pacchettizzazione di un'applicazione, disponibile su filesystem, che include tutto il necessario affinché possa essere posta in esecuzione correttamente, e che include l'ambiente di run-time, gli strumenti di sistema e le librerie ed i binari che normalmente si trovano su un server completo.
Questo approccio è differente rispetto al precedente, e garantisce un discreto livello di portabilità ed efficienza soprattutto nelle fasi di sviluppo.



L'APPROCCIO AI CONTAINER LEGGERI AGGIUNGE QUALCOSA IN PIU: UNA MAGGIORE POSSIBILITA DI SCAMBIO E COLLABORAZIONE "A BASSO IMPATTO" TRA SVILUPPATORI E SISTEMISTI (DEVOPS)

L'approccio ai container estende la promessa che la virtualizzazione ha già realizzato, come l'indipendenza dai sistemi fisici e la portabilità degli ambienti, ma aggiunge qualcosa in più: una maggiore possibilità di scambio e collaborazione "a basso impatto" tra sviluppatori e sistemisti (*DevOps*) e la possibilità di avere validi strumenti di implementazione di un'*architettura a microservizi*, definibili come un insieme di funzioni parcellizzate, di dimensioni contenute e non obbligatoriamente costrette a convivere in un monoblocco software di difficile manutenzione ed aggiornamento.

IMMAGINI E LAYERING

I team di sviluppo e di esercizio che stanno prendendo in considerazione l'adozione di **Docker**[2], considerata universalmente la tecnologia dominante dei container su Linux, potranno riscontrare diversi benefici grazie ad un approccio che consente la scomposizione funzionale di applicazioni complesse in mattoncini di base (*building blocks*) di dimensioni contenute al fine di produrre applicazioni portabili e composite, ed ottimizzarne la creazione di nuove che utilizzano gli stessi mattoncini.
Una immagine Docker è un *building block* definito da un Dockerfile (file di configurazione), che attraverso una serie di comandi genera degli strati (*Layers*) immutabili durante l'esecuzione del container.
La costruzione di una immagine Ubuntu Linux con una installazione di Apache Web Server e la semplice esecuzione di un comando Unix (creazione di un file e stampa su schermo) può essere condensata in un Dockerfile di poche righe, come questo:

```
FROM ubuntu
RUN apt-get update
RUN apt-get install –y apache2
RUN echo "Ciao CSA!" >> /tmp/CSA.txt |
cat /tmp/CSA.txt
```

La costruzione dell'immagine definita nel DockerFile, poi, viene invocata attraverso il comando Docker *build*:

```
Docker build –t myRepo/CSAimage
```

che produrrà un certo numero di Layers: uno per ogni direttiva contenuta nel Dockerfile. L'operazione di build scaricherà contestualmente i Layer da un Registry (pubblico o privato) richiesti dalle istruzioni contenute nel DockerFile, e l'immagine prodotta (*CSAimage*) conterrà un Web Server Apache attivo ed isolato rispetto alla macchina Host oltre all'esecuzione del comando Unix specificato. I benefici relativi al *Layering* sono facilmente intuibili se le esigenze di sviluppo o di esercizio ci obbligano a produrre un'immagine Docker che condivide parte dei Layer della precedente e ne estende le funzionalità. In questo caso potremmo modificare il DockerFile già presente modificando il comando Unix come segue:

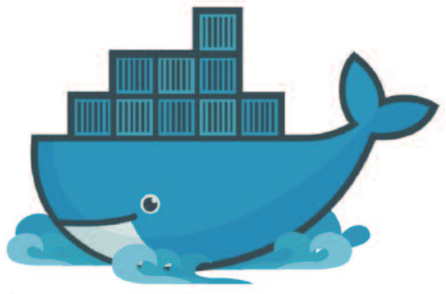
```
FROM ubuntu
RUN apt-get update
RUN apt-get install –y apache2
RUN ps –ef | grep httpd
```

La costruzione di un'altra immagine con il comando Docker *build*, specificando un altro nome, è la stessa:

```
Docker build –t myRepo/CSAimage2
```

e sarà questa volta molto più veloce, in quanto la creazione dell'immagine CSAimage rende disponibile i Layer prodotti ad altri potenziali container: in questo caso, l'immagine CSAimage2 riuserà i primi tre Layer (FROM e i due RUN iniziali) e l'unico Layer creato da zero sarà effettivamente il comando Unix che stampa su schermo la presenza dei processi Apache in esecuzione. Il riuso dei Layer è efficace soprattutto in ambienti DevOps che utilizzano logiche di *Continuous Integration* e *Continuous Delivery*, al fine di produrre cicli di costruzione del software ottimizzati per installazioni in esercizio giornaliere.

2 http://docker.io



Fonte: http://www.docker.com

IL LAYERING DI DOCKER E EFFICACE IN AMBIENTI DEVOPS CHE UTILIZZANO LOGICHE DI CONTINUOUS INTEGRATION E CONTINUOUS DELIVERY, PRODUCENDO APPLICAZIONI OTTIMIZZATE PER INSTALLAZIONI GIORNALIERE IN ESERCIZIO

TRA DEV E OPS

Il passaggio epocale dal software "monoblocco" al software "parcellizzato" porta con sé – ovviamente – tematiche complesse, perlopiù afferenti alla sfera organizzativa e culturale dell'azienda possono consentire cicli di manutenzione più agili. Non tutte le applicazioni aziendali possono essere migrate facilmente ad una architettura di nuova generazione che include i Containers e i microservizi, generalmente per la presenza di vincoli di sicurezza, normativi ed afferenti alla sfera funzionale della protezione dei dati (la presenza di una funzione immutabile in cui l'accesso al dato è essenzialmente replicato in un proprio Database applicativo visibile in maniera esclusiva dal microservizio, lascia dubbi in molti dei team che gestiscono ambienti di esercizio complessi). Molte applicazioni di Front-End, comunque, possono sfruttare in pieno i vantaggi di questo approccio per la replicazione di immagini identiche senza particolari sforzi, soprattutto per la gestione di picchi di traffico provenienti dal Web nel caso di Click-day o nel lancio di nuovi servizi per dispositivi mobili. E, in un mondo iperconnesso dove il traffico proveniente dai dispositivi ha già superato quello proveniente dai personal computer, le opzioni offerte dai Container rappresentano certamente una valida alternativa alle architetture IT classiche. ■