Casi d'uso di infrastrutture OpenStack
Database e container

Matteo Panella – matteo.panella@lngs.infn.it

9 Dicembre 2014

Sommario

Sommario

Introduzione

DataBase as a Service

- nascondere la complessità del setup di un database server
- utilizzare le risorse solo quando necessario
- fornire diversi DBMS (relazionali, KV, document-oriented...)

Introduzione

Container

- “chroot-on-steroids”
- overhead ridotto rispetto a paravirtualizzazione e HVM
- istanziati da immagini che già contengono l’applicazione
- creare un’immagine per un’applicazione (generalmente) non richiede tool speciali

Un momento: ma non c'è già OpenStack Trove?

Sulla carta, Trove implementa DBaaS.
Ma...

- la prima release "stabile" (Icehouse) è *letteralmente* inutilizzabile
- attualmente supporta solo DBMS relazionali (e tra questi solo MySQL)
- richiede immagini e flavor altamente specializzati
- ogni istanza di un DB consuma quote su Nova e Neutron

Sommario

Anatomia di Docker

Linux namespace + AUFS3/devicemapper + MAC + Go == Docker

Punti di forza di Docker

- build riproducibili
- bassissimo consumo di risorse
- ambienti self-contained e facilmente migrabili
- a volte è sufficiente un comando per avviare un'applicazione complessa

Un esempio: MongoDB

Creare ed avviare un'istanza di MongoDB

```
sudo docker run --name mongo-db1 -p 27017:27017 -d mongo:2.6
```

Terminare l'istanza

```
sudo docker stop mongo-db1
```

Distruggere l'istanza

```
sudo docker rm -v mongo-db1
```

Limiti di Docker

Limiti strutturali

- non possono esistere più di 42 (o 127, a seconda del driver) overlay per immagine/container
- il modello di networking è limitato (un'unica rete piatta e NAT per esporre i servizi)
- la gestione dei "volumi" (directory condivise tra container e container o container e host) è macchinosa
- seri problemi di resource management (a meno che non si usino kernel estremamente recenti)

Limiti di Docker

Creazione delle immagini

- rimanere sotto i 42 (o 127) overlay è complesso
- alcuni comportamenti nei Dockerfile sono controintuitivi
- i layer sono solo **additivi**, operazioni di cleanup eseguite in step successivi non riducono la dimensione finale

Limiti di Docker

Sicurezza?

- non tutti i sottosistemi del kernel sono sottoposti a namespacing
- root nel container va evitato **ad ogni costo**
- SELinux o AppArmor non sono un optional
- usare un kernel patchato con grsecurity è *caldamente* consigliato

Docker Horror Story

Il security track di Docker non è esattamente esemplare:

- CVE-2014-3499: Host local privilege escalation
- Container breakout in Docker 0.11 (no CVE assigned)
- CVE-2014-6407: Archive extraction allowing host privilege escalation
- CVE-2014-6408: Security options applied to image could lead to container escalation

CVE-2014-6407

Un'immagine malevola poteva ottenere root sull'host in fase di download!!

Disclaimer

(uso Docker quasi quotidianamente, non fatevi spaventare (troppo) dalle slide precedenti)

Sommario

Perché?

Docker al momento è grossomodo paragonabile a libvirt/kvm:

- necessita di tooling esterno per gestire scenari complessi
- lifecycle dei container gestito manualmente
- accesso al control socket di Docker == root sull'host

OpenStack già svolge operazioni simili per libvirt, estendere il supporto a Docker è quasi "naturale".

Stato dell'arte

Le buone notizie

- esiste un driver Nova per Docker
- glance supporta l'hosting di container Docker
- istanziare un container è grossomodo identico alla creazione di un'istanza KVM

Stato dell'arte

...e quelle cattive

- il driver Nova è "out-of-tree" e non supportato
- non è possibile far convivere KVM e Docker nello stesso deployment Nova
- la documentazione mette molta enfasi sulla necessità che solo gli utenti con role:admin possano fare l'upload di immagini

Speranze?

Pur essendo immaturo per la produzione, il driver Docker per Nova ha notevoli potenzialità.
È previsto il reintegro “in-tree” durante il ciclo di sviluppo di Kilo.

Nota Bene

Rimangono da affrontare i problemi strutturali di Docker – specialmente sul fronte della sicurezza

Sommario

Recap

- OpenStack-Docker è immaturo
- OpenStack Trove è inutilizzabile
- ...ma abbiamo comunque bisogno di avviare dei database on-demand

Come se ne esce?

Déjà vu/1: KVM

- gestione delle risorse affidabile
- garanzia di contenimento del DBMS (a meno di sgradevoli sorprese)
- è supportato adesso
- non è necessario un repurposing di Nova

Déjà vu/2: Heat

- permette di scalare un cluster in maniera (semi) automatica
- (...a patto che ci sia anche Ceilometer)
- l'istanza del DB potrebbe far parte di un intero stack
- sarebbe comunque necessario per orchestrare OpenStack-Docker :-)

Tutto qui?

Purtroppo no:

- l'overhead computazionale rispetto a Docker è inevitabile
- sono necessarie immagini speciali
- in alternativa va orchestrato il deployment del DBMS (Heat+puppet?)
- per database long-lived bisogna usare Cinder (che non è necessariamente un male)

Sommario

Conclusioni

- OpenStack Trove sarebbe un'ottima soluzione
- ...peccato che non funzioni
- Docker richiede un'attenzione alla sicurezza eccessiva rispetto a KVM
- OpenStack-Docker non è production-ready ed è incompatibile con un deployment Nova esistente
- alcuni componenti di OpenStack che si "sovrappongono" a Docker (ad es. Heat) sarebbero comunque necessari
- non è chiaro come OpenStack-Docker interagisca col resto di OpenStack (uno su tutti: Neutron)

Possibile scenario per un deployment

- immagini “master” con preinstallati i pacchetti per heat-cfntools e il DBMS scelto
- template Heat che gestisca il provisioning del DBMS e configuri OpenStack per l’autoscaling (assistito da Puppet?)
- Cinder come backing store per i DB che necessitano di data-at-rest

“If all you have is a hammer, everything looks like a nail”

Domande?