Casi d'uso di infrastrutture OpenStack
File system on-demand

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

9 Dicembre 2014

Sommario

Sommario

Introduzione

Obiettivi

- fornire ad istanze OpenStack filesystem condivisi POSIX-like
- i filesystem devono essere self-hosted su OpenStack
- l'intervento umano per rendere disponibile il filesystem deve essere ridotto al minimo

Bottomline

FSaaS (o qualcosa che gli assomigli...)

Intermezzo

Perché non usare direttamente OpenStack Manila?
In breve:

- non fa parte di nessuna release stabile
- ad oggi è ancora in incubation
- non sembrerebbe essere stato testato al di fuori di deployment con DevStack (!!)

Conviene aspettare *almeno* Kilo per capire come evolverà

Sommario

Cosa abbiamo

Block storage

- ottimo per fornire filesystem POSIX tradizionali
- provisioning on-demand già presente, sono solo necessari mkfs+mount nell'istanza
- ...ma anche usando filesystem cluster-aware i volumi **non** sono shareable

Object storage

Niente semantica POSIX, quindi non risponde al requisito fondamentale.

Cosa ci manca

Tutto?
Non proprio:

- Cinder può essere usato come backing store per filesystem distribuiti shared-nothing come Ceph o GlusterFS
- Nova può ospitare le istanze che forniscono il filesystem distribuito
- Neutron può fornire il trasporto di rete
- **Heat e cloud-init possono automatizzare la creazione di un filesystem distribuito**

Punti dolenti

Non è tutto oro ciò che luccica...

Problemi assortiti

- prestazioni pessime di Cinder in configurazione di default (iSCSI)
- effetto sensibile sulle quote di Nova per il tenant
- Heat non è molto user-friendly
- ...men che meno in release antecedenti a Juno! (Icehouse è una via di mezzo)

Punti dolenti

...e loro possibili soluzioni

- Cinder (e Nova) possono usare direttamente Fibre Channel al posto di iSCSI
- ...a patto di dotare una parte della flotta di compute node di HBA FC
- i template Heat vanno scritti una sola volta (più o meno)
- con un po' di fatica si può *quasi* ridurre la creazione di uno stack ad un paio di click su Horizon

Sommario

Candidati per il filesystem

Requisiti

- semantica POSIX
- accesso concorrente da più istanze
- **shared-nothing** (a causa di Cinder)
- buona scalabilità
- buona affidabilità (c'era bisogno di dirlo?)

Candidati per il filesystem

NFS

- semantica POSIX: ✓
- accesso concorrente da più istanze: $\pm$ (dipende da lockmgr et al.)
- shared-nothing: ✗
- buona scalabilità: ✗
- buona affidabilità: ✗(SPOF)

Candidati per il filesystem

AFS

- semantica POSIX: ✓
- accesso concorrente da più istanze: ✓
- shared-nothing: ±
- buona scalabilità: ✓
- buona affidabilità: ±
- (special guest) facilità di deployment ed amministrazione: ✗

Candidati per il filesystem

GlusterFS

- semantica POSIX: ✓
- accesso concorrente da più istanze: ✓
- shared-nothing: ✓
- buona scalabilità: ✓
- buona affidabilità: $\pm$ (a seconda della versione)

Candidati per il filesystem

CephFS

- semantica POSIX: ✓
- accesso concorrente da più istanze: ✓
- shared-nothing: ✓
- buona scalabilità: ✓
- buona affidabilità: ✓

Candidati per il filesystem

CephFS

- semantica POSIX: ✓
- accesso concorrente da più istanze: ✓
- shared-nothing: ✓
- buona scalabilità: ✓
- buona affidabilità: ✓

Abbiamo un "vincitore"

CephFS sembrerebbe essere il miglior candidato

Ceph su OpenStack

Prerequisiti

- disponibilità di Cinder nell'infrastruttura
- quote sufficienti per la creazione di almeno 3 volumi per cluster
- quote sufficienti per la creazione di almeno 3 OSD per cluster

The hard way: installazione manuale

Pro

- non richiede una configurazione speciale di OpenStack
- non ci sono race condition dovute all'ordine di boot delle istanze
- possibilità di customizzare in maniera profonda il cluster

Contro

- complesso
- relativamente time-consuming
- decisamente l'opposto della definizione di "on-demand"

The hard way: installazione manuale

Passi necessari

- creare le istanze (3+1)
- installare i repository Ceph su tutte le istanze
- installare e configurare ceph-deploy su un'istanza
- effettuare il setup con ceph-deploy

OpenStack Heat: setup (semi-)automatico

Pro

- l'utente finale deve solo effettuare il deploy di un template
- setup (quasi) completamente automatizzato
- margine di errore ridotto

Contro

- è **necessario** OpenStack Icehouse (Heat in Havana è incompleto)
- sono necessarie immagini specializzate con heat-cfntools
- elevata complessità dei template
- (ceph-deploy non è consigliato per deployment di produzione)

OpenStack Heat: setup (semi-)automatico

Passi necessari

- eseguire l'upload del template da Horizon (o via CLI)
- specificare i parametri per il deployment
- attendere che Heat faccia il lavoro sporco :-)

Questioni aperte

Cosa non va (come vorremmo)

- i template Heat sono eccessivamente complessi ed usano ceph-deploy
- le prestazioni di Cinder non sono entusiasmanti
- estrarre i dati di autenticazione per Ceph è ancora un processo manuale
- i pacchetti Ceph vengono scaricati dai repository ogni volta che viene istanziato uno stack

Questioni aperte

Possibili soluzioni

- utilizzare Heat solo per il bootstrap e far dirigere il setup da Puppet
- utilizzare Fibre Channel al posto di iSCSI per fornire storage agli OSD
- disabilitare l'autenticazione?
- specializzare le immagini (una per ogni ruolo)

Sommario

Come non detto...

- la demo **avrebbe dovuto** girare localmente sul mio portatile...
- ...nested dentro VMware Workstation...
- ...e con poca RAM a disposizione ("chi vuoi che abbia bisogno di più di 8GB di RAM su un portatile?")

Purtroppo...

La CPU non prende tanto bene l'idea di avviare 4 guest nested e va in overheating :-(

Sommario

Heat Orchestration Template

Sulla carta

- permette di descrivere facilmente infrastrutture complesse
- fornisce primitive per istanziare semplicemente gruppi di istanze con la stessa configurazione
- sincronizza il lifecycle delle risorse all'interno di uno stack
- può anche gestire il lifecycle delle applicazioni nelle istanze e riconfigurarle dinamicamente

Heat Orchestration Template

La dura realtà

- la complessità del template è (nella migliore delle ipotesi) $O(n^2)$ rispetto a quella dell'infrastruttura
- le primitive di gruppo (AWS::AutoScaling::LaunchConfiguration) sono incomplete in Icehouse
- alcune sono addirittura inesistenti anche se presenti nella documentazione (OS::Heat::WaitCondition)
- la sincronizzazione è *estremamente* fragile e richiede estrema attenzione
- la gestione del lifecycle delle applicazioni richiede script speciali (tanto vale usare Puppet)

Considerazioni amministrative

L'amministrazione di Heat è straordinariamente controintuitiva e complessa:

- sono necessarie immagini speciali con heat-cfntools installato
  - possibilmente una versione **recente** di heat-cfntools
  - (per RHEL e derivate questo significa heat-cfntools da RDO/Icehouse, non da EPEL!)
- l'uso di AWS::CloudFormation::WaitCondition su Havana **richiede** `role:admin`!
- da Icehouse in poi, gli utenti autorizzati ad interagire con Heat devono avere anche `role:heat_stack_owner` nel tenant
- ...ma è documentato solo nelle release notes di Juno

Domande?