Cinder: advanced features

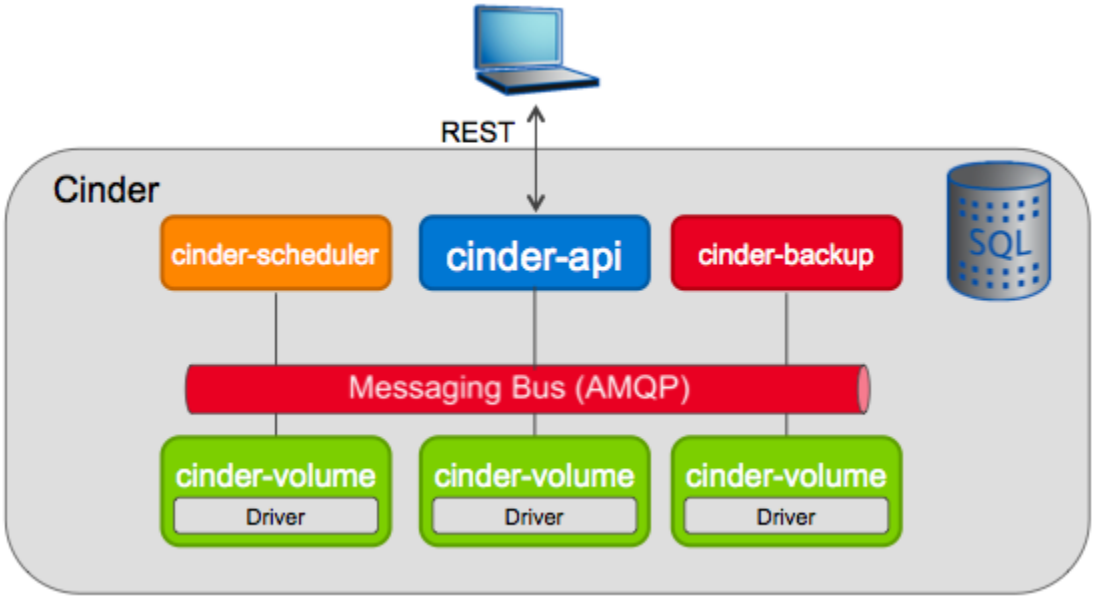
Marica Antonacci - INFN Bari

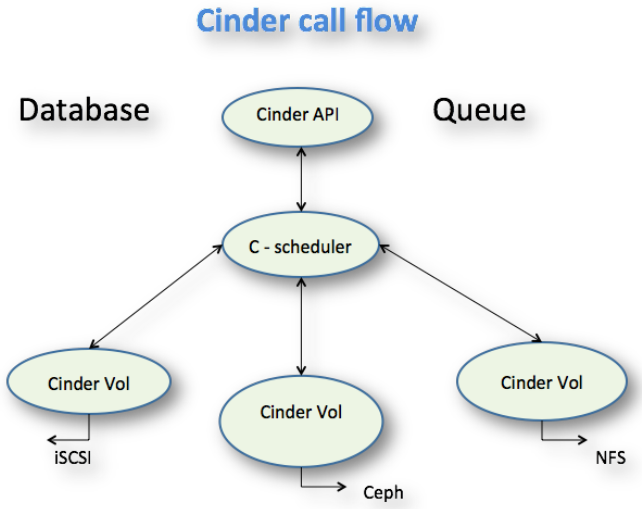
Scuola di Cloud Computing
Bari, 24-28 Novembre 2014

Outline

- Multi-backend
- QoS & Rate-limiting
- Encryption
- Backup & Disaster-Recovery

Cinder: Backend Multipli





Openstack Block Storage Drivers Support Matrix: https://wiki.openstack.org/wiki/CinderSupportMatrix

Configuration

- File /etc/cinder/cinder.conf
-
- **Debugging**

```
[DEFAULT]
verbose = True
debug = True
…
```

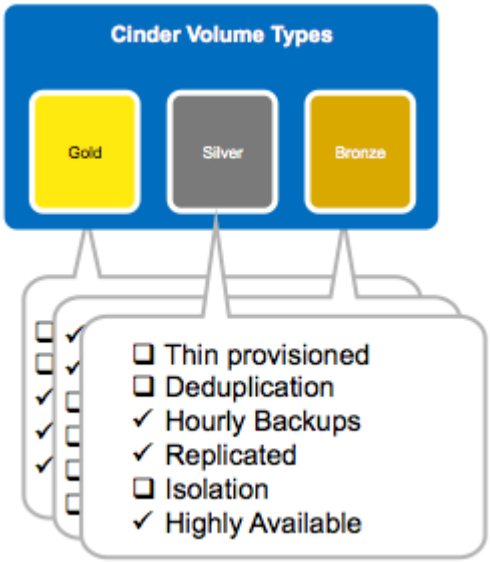
  - log in /var/log/cinder
  - può essere utile usare il client CLI python-cinderclient con l'opzione --debug

Advanced Storage Quality of Service

- Default: i volumi vengono distribuiti sui vari backend in modo da bilanciare lo spazio allocato.
- I **volume-type** possono essere utilizzati per controllare dove i volumi verranno allocati.
- Ogni volume-type contiene un set di coppie chiave-valore chiamati ***extra-specs***. Queste informazioni sono usate dal **Cinder-Scheduler** per prendere decisioni sul placement dei volumi in base alle capabilities dei backend disponibili.

QoS e volume type

- I **Volume type** possono essere usati per fornire agli utenti differenti livelli (tier) di storage:
  - performance diverse (p.e. HDD tier, mixed HDD-SDD tier, o SSD tier),
  - resilienza (selezionando differenti livelli di RAID, o replica)
  - specifiche features (p.e. compressione, data-deduplication, etc.).



QoS e volume type

- L’utente può specificare il tier in cui creare il volume.
- **Volume-Retype:** consente di cambiare il tipo di volume dopo la sua creazione. Questo è utile per esempio per modificare il livello di QoS dinamicamente (nel caso in cui un volume sia sottoposto ad utilizzo pesante nel tempo e si renda necessario il passaggio ad un tier che offra un servizio migliore).

Un esempio di configurazione

Per esempio, assumiamo di avere 2 pool su Ceph che utilizzano storage device differenti:

- il pool "*cinder-sata*" usa un rack **SATA**
- il pool "*cinder-ssd*" usa un rack **SSD**

```
# Multi backend options

# Define the names of the groups for multiple volume backends
enabled_backends=rbd-sata,rbd-ssd

# Define the groups as above
[rbd-sata]
volume_driver=cinder.volume.driver.RBDDriver
rbd_pool=cinder-sata
volume_backend_name=RBD_SATA
# if cephX is enable
#rbd_user=cinder
#rbd_secret_uuid=<None>
[rbd-ssd]
volume_driver=cinder.volume.driver.RBDDriver
rbd_pool=cinder-ssd
volume_backend_name=RBD_SSD
# if cephX is enable
#rbd_user=cinder
#rbd_secret_uuid=<None>
```

I backend vanno abilitati nel file di configurazione `/etc/cinder/cinder.conf` del nodo su cui gira cinder-volume

NOTA: in questo caso particolare, l'istanza di cinder-volume gestisce due backend diversi

Questo esempio mostra anche come poter sfruttare le capacità di tiering di Ceph

Rate-limiting

- Feature introdotta in **Havana**
- Implementa il supporto QoS in Nova e Cinder (sfruttando il rate limiting già supportato in KVM e QEMU attraverso libvirt) - utile nel caso in cui lo storage non espone questa funzionalità
- Il limiting può quindi essere realizzato dal “frontend” (hypervisor) o dal “backend” (storage subsystem) o entrambi
- **Backend**: campi specifici definiti dal vendor:
  - HP 3PAR (IOPS, tput: min, max; latency, priority)
  - Solidfire (IOPS: min, max, burst)
  - NetApp* (QoS Policy Group)
  - Huawei* (priority)

*defined through extra specs

Rate-limiting

- **Frontend** QoS options:
  - throughput
    - *total_bytes_sec*: the total allowed bandwidth for the guest per second
    - *read_bytes_sec*: sequential read limitation
    - *write_bytes_sec*: sequential write limitation
  - IOPS
    - *total_iops_sec*: the total allowed IOPS for the guest per second
    - *read_iops_sec*: random read limitation
    - *write_iops_sec*: random write limitation
- Il file di definizione della VM a cui viene agganciato il volume con *qos-specs* conterrà un campo xml extra “<iotune>”. Es.

```
<iotune>
      <read_iops_sec>2000</read_iops_sec>
      <write_iops_sec>1000</write_iops_sec>
</iotune>
```

Solo da command-line (come admin)

create qos specs

```
cinder qos-create <name> <key=value> [<key=value>
...]
```

Associate qos specs with specific volume type

```
cinder qos-associate <qos_specs> <volume_type_id>
```

Esempi: extra-specs + qos-specs

Mettiamo insieme un po' tutto:

- volume-types,
- extra-specs,
- qos-specs

Volume Type	Extra Specs	QoS Specs
Gold	{netapp:disk_type=**SSD**, netapp_thick_provisioned=**True**}	{}
Silver	{}	{total_iops_sec=**500**}
Bronze	{volume_backend_name=**lvm**}	{total_iops_sec=**100**}

Workflow

- creare la specifica di QoS (cinder qos-create)
- creare il volume-type
- associare il volume-type al qos
- creare un volume con il volume-type definito
- agganciare il volume ad una VM
- controllare il definition file della VM (deve contenere il tag `<iotune>`)

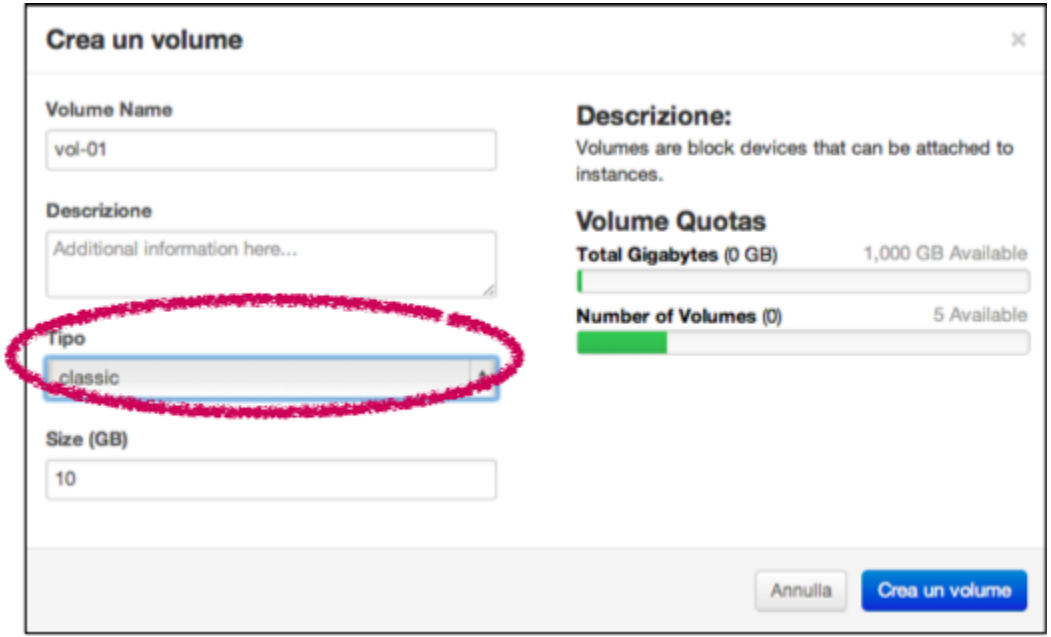
wiki per l’esercitazione:

https://github.com/infn-bari-school/Cinder/wiki

https://github.com/infn-bari-school/cinder-advanced/wiki/Multi-backend

https://github.com/infn-bari-school/cinder-advanced/wiki/Rate-limiting

Creazione volumi da Horizon



La dashboard consente la creazione dei volume-type, ma non permette al momento l’ associazione con i backend.

Non sono neanche implementate le funzioni relative alla gestione degli encryption-type.

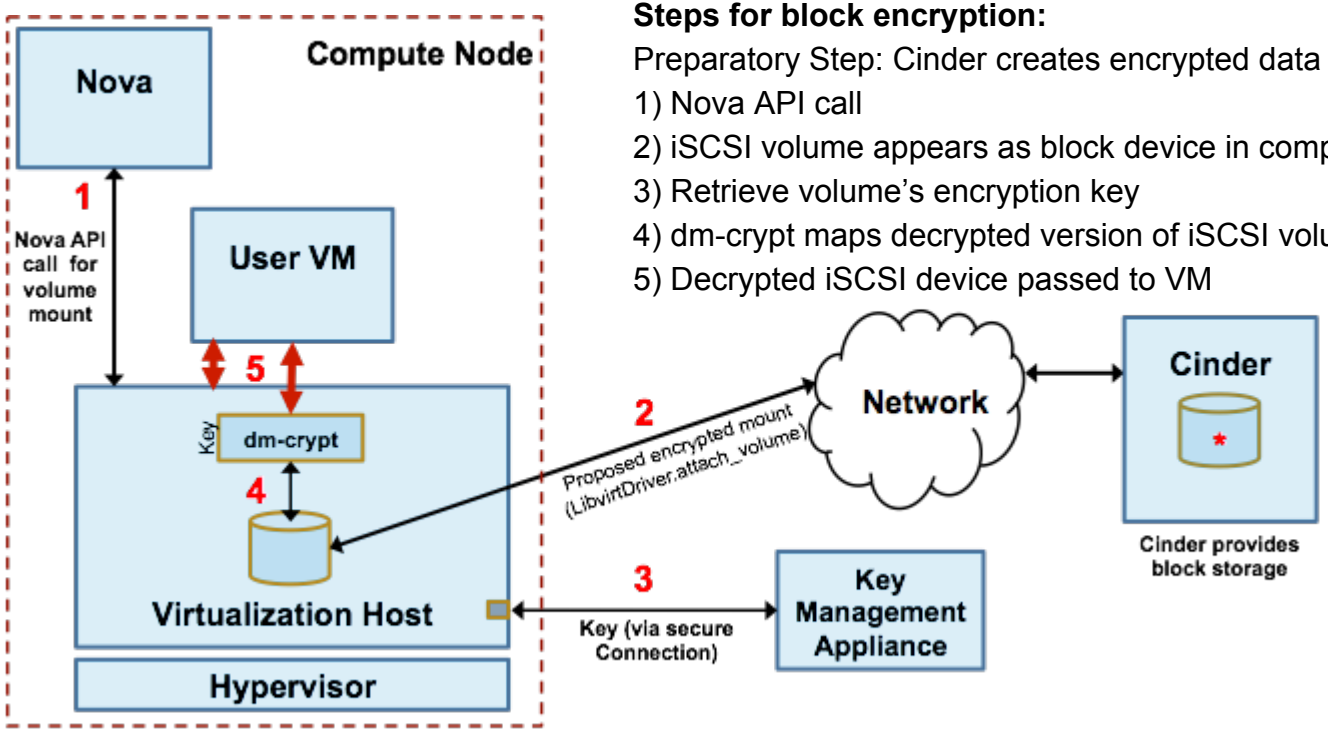
Data Encryption

- Semplice da configurare, trasparente per l'utente finale (creare un volume cifrato richiede le stesse operazioni di un volume non cifrato)
- Il transito dei dati è sicuro
  - p.e. non è necessario usare IPsec per proteggere il traffico iSCSI
- Supporta:
  - le funzionalità esistenti in cinder (p.e. snapshot)
  - boot da volumi criptati
  - possibilità di scegliere il key-manager da usare per gestire le chiavi

Key Manager

- Il key manager di default è “configuration-based”
  - supporta singola chiave statica usata per tutti i volumi
    - da NON usare in produzione. La sicurezza dei dati dipende dalla segretezza della chiave
  - consigliato utilizzare un key-manager esterno (e.g. Barbican)
- Il key-manager espone un’interfaccia astratta che consente di integrare qualunque key-manager (incluso sistemi commerciali, p.e. Safenet, IBM, HP, etc.)

Componenti



Steps for block encryption:

Preparatory Step: Cinder creates encrypted data volume

1) Nova API call
2) iSCSI volume appears as block device in compute host
3) Retrieve volume's encryption key
4) dm-crypt maps decrypted version of iSCSI volume
5) Decrypted iSCSI device passed to VM

https://wiki.openstack.org/wiki/VolumeEncryption

Encrypted Volume types

- estensione dell'astrazione del **volume-type**
- utilizzo di metadata predefiniti
  - ***cipher***: modalità di cifratura
    - ***e.g. aes-cbc-essiv:sha256 o aes-xts-plain64***
  - ***key-length***: dimensione della chiave in bits
    - ***e.g. 128 o 256***
  - ***Provider***: classe responsabile dell'attachment/detachment del volume criptato
    - ***nova.volume.encryptors.cryptsetup.CryptsetupEncryptor: uses "raw" cryptsetup***
    - ***nova.volume.encryptors.luks.LuksEncryptor: uses LUKS extensions to cryptsetup***
  - ***Control location***: servizio che esegue l'encryption
    - ***'front-end' → Nova; 'back-end' → Cinder***
      - **'back-end' (i.e., encryption by Cinder) not yet implemented**

Encryption & Ceph

- Ceph supporta dm-crypt

```
# ceph-deploy osd --dmcrypt [--dmcrypt-key-
  dir KEYDIR] create|prepare HOST:DISK
```

- creare pool su OSD criptati
- configurare in cinder.conf un nuovo backend associandolo al pool encrypted
- creare un volume-type specifico

Workflow

- configurare il key-manager
- creare il volume-type
- creare l'encryption-type associandolo al volume-type
- creare il volume del volume-type definito
- agganciare il volume ad una VM attiva
- montare il volume e scrivere sul volume
- verificare che il volume è stato criptato

wiki per l’esercitazione:

https://github.com/infn-bari-school/cinder-advanced/wiki/Encryption

Cinder Backup

- Un backup è una copia del volume archiviata nell'Object Store
- Gestito da un servizio a parte: **cinder-backup** (non attivo di default)
- Driver configurabili:
  - Ceph
  - Swift
  - IBM Tivoli Storage Manager

Backup su Swift - wiki

- Installare il pacchetto **cinder-backup** sui nodi su cui gira cinder-volume
- Configurare il backend in /etc/cinder/cinder.conf

- **wiki**: https://github.com/infn-bari-school/cinder-advanced/wiki/Backup-su-Swift

Verso il Disaster Recovery

- estensione delle API di cinder backup:
  - **import/export dei metadata**
- In Juno le API di cinder verranno ulteriormente arricchite per supportare la **replica** dei volumi

Altre funzionalità di Cinder...

- Migrare i volumi tra backend differenti (admin API)
  - `cinder migrate <volume_id> <target host>`
- Estendere la dimensione di un volume
  - `cinder extend <vol-id> <newsize>`
- Trasferire un volume da un tenant ad un altro

```
cinder transfer-create <volume_id> #TenantA

cinder transfer-accept <transfer_id> <auth_key>
#TenantB
```