Virtualizzazione e tools vari

Otacon22
otacon22 - at - poul.org

Corsi GNU/Linux Avanzati 2014

Sezione 1

SSH Key Auth

Per fare login utilizzando una chiave ssh su un server remoto:

- Generiamo una coppia di chiavi ssh pubblica/privata sul nostro client

  ```
  ssh-keygen
  ```

  - scegliamo il percorso, eventualmente possiamo proteggere la chiave con una password necessaria a "sbloccarla"
- Sul server ssh su cui fare il login, nell'utente su cui vogliamo fare login:

  ```
  mkdir ~/.ssh #se non esiste gia'
  touch ~/.ssh/authorized_keys
  ```

SSH Key Auth

Per fare login utilizzando una chiave ssh su un server remoto:

- Generiamo una coppia di chiavi ssh pubblica/privata sul nostro client

  ```
  ssh-keygen
  ```

  - scegliamo il percorso, eventualmente possiamo proteggere la chiave con una password necessaria a "sbloccarla"

- Sul server ssh su cui fare il login, nell'utente su cui vogliamo fare login:

  ```
  mkdir ~/.ssh #se non esiste gia'
  touch ~/.ssh/authorized_keys
  ```

SSH Key Auth

- Copiamo la chiave ssh **pubblica** creata dal client (quella che finisce con .pub), nel file ~/.ssh/authorized_keys sul server
- Assicuriamoci poi sempre di aver impostato i **permessi giusti** nelle cartelle (se sono sbagliati non possiamo fare login con la chiave)

```
chmod -R 600 ~/.ssh
chown -R nomeutente:nomeutente ~/.ssh
```

- Sul client:

```
ssh nomeutente@hostname -i /path/to/ssh/private/key
```

SSH Key Auth

- Copiamo la chiave ssh **pubblica** creata dal client (quella che finisce con .pub), nel file ~/.ssh/authorized_keys sul server
- Assicuriamoci poi sempre di aver impostato i **permessi giusti** nelle cartelle (se sono sbagliati non possiamo fare login con la chiave)

```
chmod -R 600 ~/.ssh
chown -R nomeutente:nomeutente ~/.ssh
```

- Sul client:

```
ssh nomeutente@hostname -i /path/to/ssh/private/key
```

SSH Key Auth

- Copiamo la chiave ssh **pubblica** creata dal client (quella che finisce con .pub), nel file ~/.ssh/authorized_keys sul server
- Assicuriamoci poi sempre di aver impostato i **permessi giusti** nelle cartelle (se sono sbagliati non possiamo fare login con la chiave)

```
chmod -R 600 ~/.ssh
chown -R nomeutente:nomeutente ~/.ssh
```

- Sul client:

```
ssh nomeutente@hostname -i /path/to/ssh/private/key
```

SSH Key Auth

- Nel caso in cui vogliamo forzare il login con chiave pubblica

```
ssh nomeutente@hostname -i /path/to/ssh/private/key -o PubkeyAuthentication=yes
```

- Dualmente, per forzare il login con password:

```
ssh nomeutente@hostname -i /path/to/ssh/private/key -o PubkeyAuthentication=no
```

SSH Key Auth

- Nel caso in cui vogliamo forzare il login con chiave pubblica

```
ssh nomeutente@hostname -i /path/to/ssh/private/key -o PubkeyAuthentication=yes
```

- Dualmente, per forzare il login con password:

```
ssh nomeutente@hostname -i /path/to/ssh/private/key -o PubkeyAuthentication=no
```

SSH Security Tuning

Apriamo */etc/ssh/sshd_config*

- Forziamo il protocollo versione 2 (la versione 1 ha problemi di sicurezza)

```
Protocol 2
```

- Disabilitiamo il login per root (sapere uno username rende semplice un bruteforce, specialmente con auth con password)

```
PermitRootLogin no
```

- Forziamo l'autenticazione con chiave pubblica:

```
PubkeyAuthentication yes
PasswordAuthentication no
```

SSH Security Tuning

Apriamo */etc/ssh/sshd_config*

- Forziamo il protocollo versione 2 (la versione 1 ha problemi di sicurezza)

```
Protocol 2
```

- Disabilitiamo il login per root (sapere uno username rende semplice un bruteforce, specialmente con auth con password)

```
PermitRootLogin no
```

- Forziamo l'autenticazione con chiave pubblica:

```
PubkeyAuthentication yes
PasswordAuthentication no
```

SSH Security Tuning

Apriamo */etc/ssh/sshd_config*

- Forziamo il protocollo versione 2 (la versione 1 ha problemi di sicurezza)

```
Protocol 2
```

- Disabilitiamo il login per root (sapere uno username rende semplice un bruteforce, specialmente con auth con password)

```
PermitRootLogin no
```

- Forziamo l'autenticazione con chiave pubblica:

```
PubkeyAuthentication yes
PasswordAuthentication no
```

SSH/Sudo Tuning

Se utilizzate solo chiavi pubbliche ha senso togliere le password degli utenti e utilizzare solo sudo (senza password)

- Aggiungiamo una chiave pubblica per l'accesso via SSH nella home dell'utente "username", così come visto prima.
- Aggiungiamo l'utente al gruppo "sudo" e modifichiamo la riga relativa al gruppo "sudo" nel file /etc/sudoers come segue:

```
%sudo ALL=(ALL) NOPASSWD: ALL
```

- Disabilitiamo l'accesso con password all'account dell'utente

```
usermod -L username
```

In questo modo possiamo loggarci sul server via SSH con chiave e possiamo diventare root usando sudo, tutto senza utilizzare nessuna password. Un attaccante che vuole

SSH/Sudo Tuning

Se utilizzate solo chiavi pubbliche ha senso togliere le password degli utenti e utilizzare solo sudo (senza password)

- Aggiungiamo una chiave pubblica per l'accesso via SSH nella home dell'utente "username", così come visto prima.
- Aggiungiamo l'utente al gruppo "sudo" e modifichiamo la riga relativa al gruppo "sudo" nel file /etc/sudoers come segue:

```
%sudo ALL=(ALL) NOPASSWD: ALL
```

- Disabilitiamo l'accesso con password all'account dell'utente

```
usermod -L username
```

In questo modo possiamo loggarci sul server via SSH con chiave e possiamo diventare root usando sudo, tutto senza utilizzare nessuna password. Un attaccante che vuole

SSH/Sudo Tuning

Se utilizzate solo chiavi pubbliche ha senso togliere le password degli utenti e utilizzare solo sudo (senza password)

- Aggiungiamo una chiave pubblica per l'accesso via SSH nella home dell'utente "username", così come visto prima.
- Aggiungiamo l'utente al gruppo "sudo" e modifichiamo la riga relativa al gruppo "sudo" nel file /etc/sudoers come segue:

```
%sudo ALL=(ALL) NOPASSWD: ALL
```

- Disabilitiamo l'accesso con password all'account dell'utente

```
usermod -L username
```

In questo modo possiamo loggarci sul server via SSH con chiave e possiamo diventare root usando sudo, tutto senza utilizzare nessuna password. Un attaccante che vuole

SSH/Sudo Tuning

Se utilizzate solo chiavi pubbliche ha senso togliere le password degli utenti e utilizzare solo sudo (senza password)

- Aggiungiamo una chiave pubblica per l'accesso via SSH nella home dell'utente "username", così come visto prima.
- Aggiungiamo l'utente al gruppo "sudo" e modifichiamo la riga relativa al gruppo "sudo" nel file /etc/sudoers come segue:

```
%sudo ALL=(ALL) NOPASSWD: ALL
```

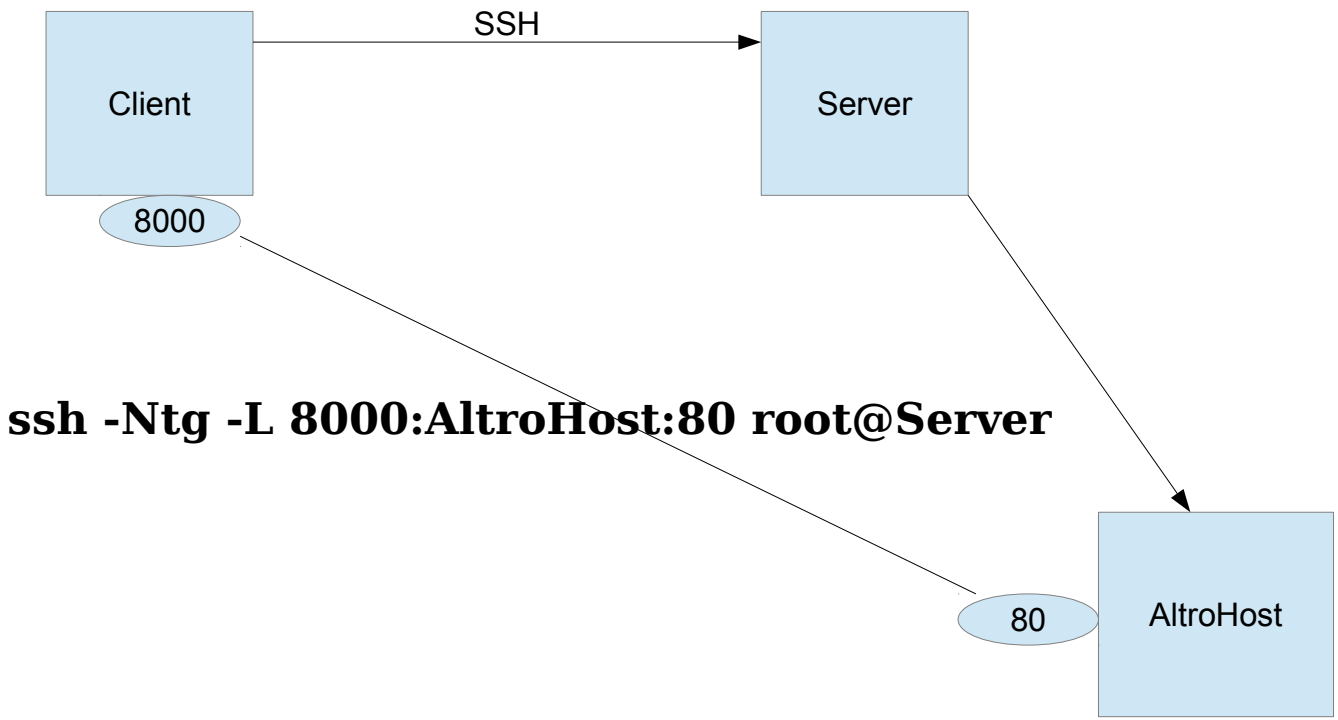
- Disabilitiamo l'accesso con password all'account dell'utente

```
usermod -L username
```

In questo modo possiamo loggarci sul server via SSH con chiave e possiamo diventare root usando sudo, tutto senza utilizzare nessuna password. Un attaccante che vuole

SSH Tunneling

Serve per dire al server ssh a cui ci colleghiamo di farci da TCP proxy verso un host che lui può raggiungere



SSH Tunneling

- Può tornare molto utile per collegarsi a qualche servizio che è raggiungibile solo dal server ssh remoto, senza stare a usare VPN o altro
- Ad esempio spesso è comodo se abbiamo un database MySQL (in ascolto su TCP 3306) e vogliamo accederci da remoto per amministrarlo via CLI, senza "esporlo" su internet.
- Con un tunnel ssh possiamo collegarci al server mysql usando come IP sorgente 127.0.0.1, dato che la connessione viene instaurata dal server che si collega a se stesso

```
ssh -Ntg -L 3308:127.0.0.1:3306 root@server
```

- Può tornare utile anche per fare connessioni ssh a catena

SSH Tunneling

- Può tornare molto utile per collegarsi a qualche servizio che è raggiungibile solo dal server ssh remoto, senza stare a usare VPN o altro
- Ad esempio spesso è comodo se abbiamo un database MySQL (in ascolto su TCP 3306) e vogliamo accederci da remoto per amministrarlo via CLI, senza "esporlo" su internet.
- Con un tunnel ssh possiamo collegarci al server mysql usando come IP sorgente 127.0.0.1, dato che la connessione viene instaurata dal server che si collega a se stesso

```
ssh -Ntg -L 3308:127.0.0.1:3306 root@server
```

- Può tornare utile anche per fare connessioni ssh a catena

SSH Tunneling

- Può tornare molto utile per collegarsi a qualche servizio che è raggiungibile solo dal server ssh remoto, senza stare a usare VPN o altro
- Ad esempio spesso è comodo se abbiamo un database MySQL (in ascolto su TCP 3306) e vogliamo accederci da remoto per amministrarlo via CLI, senza "esporlo" su internet.
- Con un tunnel ssh possiamo collegarci al server mysql usando come IP sorgente 127.0.0.1, dato che la connessione viene instaurata dal server che si collega a se stesso

```
ssh -Ntg -L 3308:127.0.0.1:3306 root@server
```

- Può tornare utile anche per fare connessioni ssh a catena

SSH Tunneling

- Può tornare molto utile per collegarsi a qualche servizio che è raggiungibile solo dal server ssh remoto, senza stare a usare VPN o altro
- Ad esempio spesso è comodo se abbiamo un database MySQL (in ascolto su TCP 3306) e vogliamo accederci da remoto per amministrarlo via CLI, senza "esporlo" su internet.
- Con un tunnel ssh possiamo collegarci al server mysql usando come IP sorgente 127.0.0.1, dato che la connessione viene instaurata dal server che si collega a se stesso

```
ssh -Ntg -L 3308:127.0.0.1:3306 root@server
```

- Può tornare utile anche per fare connessioni ssh a catena

SSH Reverse Tunneling

Diciamo al server: “se ti attiva una connessione TCP su questa porta A, rigiramela a me, poi io inoltro la connessione ad un altro host sulla porta B

- Ad esempio

```
ssh -Ntg -R *:8000:192.168.0.1:80 user@server_hostname
```

  - Dice al server “server_hostname” di restare in ascolto sulla porta TCP 8000 per connessioni da qualunque indirizzo
  - Quando arriva una connessione al server, lui (tramite ssh) ce la inoltra. Noi facciamo una connessione a 192.168.0.1 sulla porta 80 e gli inoltriamo i dati ricevuti dal server

SSH Reverse Tunneling

- Note:
  - Sul server dobbiamo aggiungere in /etc/ssh/sshd_config l'opzione

    ```
    GatewayPorts yes
    ```

  - Se il client è dietro NAT, il NAT potrebbe terminare la connessione ssh quando non stiamo ricevendo nessun dato. Per evitarlo dobbiamo aggiungere sul server, sempre in /etc/ssh/sshd_config:

    ```
    TCPKeepAlive yes
    ```

Sezione 2

1. SSH tuning
   - Security
   - Tunneling
2. Virtualizzazione
   - Concetti base
   - Tools

Perchè virtualizzare?

- Per gestire meglio le **risorse** che abbiamo a disposizione
- Per poter amministrare in modo più semplice vari servizi
  - Se ho un sacco di servizi su una macchina sopra diventano un problema gli aggiornamenti, la migrazione (SPOF), sicurezza tra vari servizi
- Per **isolare** una applicazione poco sicura in un ambiente totalmente dedicato che possiamo snapshottare
- Per fornire a **tanti utenti** risorse completamente separate, su cui è possibile fare accounting

Perchè virtualizzare?

- Per gestire meglio le **risorse** che abbiamo a disposizione
- Per poter amministrare in modo più semplice vari servizi
  - Se ho un sacco di servizi su una macchina sopra diventano un problema gli aggiornamenti, la migrazione (SPOF), sicurezza tra vari servizi
- Per **isolare** una applicazione poco sicura in un ambiente totalmente dedicato che possiamo snapshottare
- Per fornire a **tanti utenti** risorse completamente separate, su cui è possibile fare accounting

Perchè virtualizzare?

- Per gestire meglio le **risorse** che abbiamo a disposizione
- Per poter amministrare in modo più semplice vari servizi
  - Se ho un sacco di servizi su una macchina sopra diventano un problema gli aggiornamenti, la migrazione (SPOF), sicurezza tra vari servizi
- Per **isolare** una applicazione poco sicura in un ambiente totalmente dedicato che possiamo snapshottare
- Per fornire a **tanti utenti** risorse completamente separate, su cui è possibile fare accounting

Perchè virtualizzare?

- Per gestire meglio le **risorse** che abbiamo a disposizione
- Per poter amministrare in modo più semplice vari servizi
  - Se ho un sacco di servizi su una macchina sopra diventano un problema gli aggiornamenti, la migrazione (SPOF), sicurezza tra vari servizi
- Per **isolare** una applicazione poco sicura in un ambiente totalmente dedicato che possiamo snapshottare
- Per fornire a **tanti utenti** risorse completamente separate, su cui è possibile fare accounting

Perchè virtualizzare?

- Per avere un **ambiente** di produzione **integrato** per un singolo servizio
  - Ad esempio un certo servizio deve girare con certe librerie che ci sono su una certa versione del sistema operativo, che però vanno in conflitto con altre cose
- Per avere "**scalabilità**" delle risorse
  - Avere la possibilità di aumentarle on-demand in base alle proprie necessità grazie al fatto di non essere legati ad una specifica macchina
  - Metto una VM su una macchina lenta, poi compro una macchina nuova e posso spostare la VM aumentanone le prestazioni, senza reinstallarla.

Perchè virtualizzare?

- Per avere un **ambiente** di produzione **integrato** per un singolo servizio
  - Ad esempio un certo servizio deve girare con certe librerie che ci sono su una certa versione del sistema operativo, che però vanno in conflitto con altre cose
- Per avere "**scalabilità**" delle risorse
  - Avere la possibilità di aumentarle on-demand in base alle proprie necessità grazie al fatto di non essere legati ad una specifica macchina
  - Metto una VM su una macchina lenta, poi compro una macchina nuova e posso spostare la VM aumentanone le prestazioni, senza reinstallarla.

Perchè virtualizzare?

- Per avere un **ambiente** di produzione **integrato** per un singolo servizio
  - Ad esempio un certo servizio deve girare con certe librerie che ci sono su una certa versione del sistema operativo, che però vanno in conflitto con altre cose
- Per avere "**scalabilità**" delle risorse
  - Avere la possibilità di aumentarle on-demand in base alle proprie necessità grazie al fatto di non essere legati ad una specifica macchina
  - Metto una VM su una macchina lenta, poi compro una macchina nuova e posso spostare la VM aumentanone le prestazioni, senza reinstallarla.

Metodi di virtualizzazione

- QEMU
- KVM
- LXC
- OpenVZ

Abbiamo poi tutta una serie di tools per sfruttare queste “tecnologie” in modo più o meno astratto

QEMU

. Permette di emulare una CPU e dei device collegati ad essa.

- Essendo un emulatore di CPU grezzo supporta varie piattaforme:
  - Ad esempio possiamo emulare una CPU ARM invece di x86 e installare una versione di Linux per ARM

- Le performance non sono ottimali perché viene emulato completamente tutto il comportamento del processore

QEMU

. Permette di emulare una CPU e dei device collegati ad essa.

- Essendo un emulatore di CPU grezzo supporta varie piattaforme:
  - Ad esempio possiamo emulare una CPU ARM invece di x86 e installare una versione di Linux per ARM

- Le performance non sono ottimali perché viene emulato completamente tutto il comportamento del processore

KVM

Kernel-based **V**irtual **M**achine

- Se il nostro processore supporta le estensioni per la **virtualizzazione hardware** (Intel-VTx o AMD-V) accelerata, possiamo utilizzare questa interfaccia del kernel che aumenta moltissimo le performance

Verifica estensioni

- Verifichiamo che la CPU del nostro server supporti la virtualizzazione hardware (attenzione: può essere che sia supportata ma disabilitata nel BIOS)

```
egrep -c '(vmx|svm)' /proc/cpuinfo
```

- Per aumentare le performance è meglio utilizzare un kernel a 64 bit.
- Per verificare se la nostra CPU è a 64 bit:

```
egrep -c ' lm ' /proc/cpuinfo
```

- Per verificare se il nostro kernel installato è a 64 bit:

```
uname -m
```

Verifica estensioni

- Verifichiamo che la CPU del nostro server supporti la virtualizzazione hardware (attenzione: può essere che sia supportata ma disabilitata nel BIOS)

  ```
  egrep -c '(vmx|svm)' /proc/cpuinfo
  ```

- Per aumentare le performance è meglio utilizzare un kernel a 64 bit.
- Per verificare se la nostra CPU è a 64 bit:

  ```
  egrep -c ' lm ' /proc/cpuinfo
  ```

- Per verificare se il nostro kernel installato è a 64 bit:

  ```
  uname -m
  ```

Verifica estensioni

- Verifichiamo che la CPU del nostro server supporti la virtualizzazione hardware (attenzione: può essere che sia supportata ma disabilitata nel BIOS)

  ```
  egrep -c '(vmx|svm)' /proc/cpuinfo
  ```

- Per aumentare le performance è meglio utilizzare un kernel a 64 bit.

- Per verificare se la nostra CPU è a 64 bit:

  ```
  egrep -c ' lm ' /proc/cpuinfo
  ```

- Per verificare se il nostro kernel installato è a 64 bit:

  ```
  uname -m
  ```

Verifica estensioni

- Verifichiamo che la CPU del nostro server supporti la virtualizzazione hardware (attenzione: può essere che sia supportata ma disabilitata nel BIOS)

  ```
  egrep -c '(vmx|svm)' /proc/cpuinfo
  ```

- Per aumentare le performance è meglio utilizzare un kernel a 64 bit.

- Per verificare se la nostra CPU è a 64 bit:

  ```
  egrep -c ' lm ' /proc/cpuinfo
  ```

- Per verificare se il nostro kernel installato è a 64 bit:

  ```
  uname -m
  ```

Verifica estensioni

- Verifichiamo che la CPU del nostro server supporti la virtualizzazione hardware (attenzione: può essere che sia supportata ma disabilitata nel BIOS)

  ```
  egrep -c '(vmx|svm)' /proc/cpuinfo
  ```

- Per aumentare le performance è meglio utilizzare un kernel a 64 bit.

- Per verificare se la nostra CPU è a 64 bit:

  ```
  egrep -c ' lm ' /proc/cpuinfo
  ```

- Per verificare se il nostro kernel installato è a 64 bit:

  ```
  uname -m
  ```

QEMU-KVM

```
sudo apt-get install qemu-kvm
```

- Facciamo il boot di una livecd su una macchina con 2 core, 128MB di ram e vediamo l'output video su VNC

```
kvm -m 128 -smp 2 -runas nobody -vnc 127.0.0.1:0 -cdrom /path/to/livecd.iso
```

- Aggiungendo l'opzione "-net tap,ifname=vmint0" aggiungiamo una interfaccia di rete virtuale, che sulla macchina host avrà nome vmint0 (mentre sulla macchina virtuale viene vista normalmente come eth0).

- Possiamo poi decidere se fare NAT, agganciare l'interfaccia ad un bridge o utilizzare la rete solo punto-punto

QEMU-KVM

```
sudo apt-get install qemu-kvm
```

- Facciamo il boot di una livecd su una macchina con 2 core, 128MB di ram e vediamo l'output video su VNC

```
kvm -m 128 -smp 2 -runas nobody -vnc 127.0.0.1:0 -cdrom /path/to/livecd.iso
```

- Aggiungendo l'opzione "-net tap,ifname=vmint0" aggiungiamo una interfaccia di rete virtuale, che sulla macchina host avrà nome vmint0 (mentre sulla macchina virtuale viene vista normalmente come eth0).

- Possiamo poi decidere se fare NAT, agganciare l'interfaccia ad un bridge o utilizzare la rete solo punto-punto

QEMU-KVM

```
sudo apt-get install qemu-kvm
```

- Facciamo il boot di una livecd su una macchina con 2 core, 128MB di ram e vediamo l'output video su VNC

```
kvm -m 128 -smp 2 -runas nobody -vnc 127.0.0.1:0 -cdrom /path/to/livecd.iso
```

- Aggiungendo l'opzione "-net tap,ifname=vmint0" aggiungiamo una interfaccia di rete virtuale, che sulla macchina host avrà nome vmint0 (mentre sulla macchina virtuale viene vista normalmente come eth0).
- Possiamo poi decidere se fare NAT, agganciare l'interfaccia ad un bridge o utilizzare la rete solo punto-punto

QEMU-KVM

```
sudo apt-get install qemu-kvm
```

- Facciamo il boot di una livecd su una macchina con 2 core, 128MB di ram e vediamo l'output video su VNC

```
kvm -m 128 -smp 2 -runas nobody -vnc 127.0.0.1:0 -cdrom /path/to/livecd.iso
```

- Aggiungendo l'opzione "-net tap,ifname=vmint0" aggiungiamo una interfaccia di rete virtuale, che sulla macchina host avrà nome vmint0 (mentre sulla macchina virtuale viene vista normalmente come eth0).

- Possiamo poi decidere se fare NAT, agganciare l'interfaccia ad un bridge o utilizzare la rete solo punto-punto

Creare un immagine disco

Come disco per la macchina virtuale possiamo utilizzare diverse cose, a seconda delle esigenze/comodità.

- Quanto grande ci serve il disco?
  - Una immagine minimale di Linux occupa intorno ai 2GB, mentre una completa intorno agli 8GB
  - Possiamo quasi sempre ridimensionare il disco della macchina virtuale, ma è sempre un problema e richiede tempo.

Creare un immagine disco

Come disco per la macchina virtuale possiamo utilizzare diverse cose, a seconda delle esigenze/comodità.

- Quanto grande ci serve il disco?
  - Una immagine minimale di Linux occupa intorno ai 2GB, mentre una completa intorno agli 8GB
  - Possiamo quasi sempre ridimensionare il disco della macchina virtuale, ma è sempre un problema e richiede tempo.

Creare un immagine disco

- File, logical volume o SAN
  - Utilizzare un file come disco per la VM è sempre l'opzione più semplice, anche se può avere qualche problema di performance come vedremo tra poco
  - Se il sistema host che ospita la VM ha un disco che utilizza LVM, possiamo decidere di allocare un volume logico LVM per la macchina virtuale. Abbiamo prestazioni migliori.
  - SAN: Se abbiamo uno storage iSCSI sulla rete locale (vedi *openiscsi* e simili).
  - Utilizzare un disco intero della macchina host è solitamente sconsigliato per motivi di sicurezza.

Creare un immagine disco

- File, logical volume o SAN
  - Utilizzare un file come disco per la VM è sempre l'opzione più semplice, anche se può avere qualche problema di performance come vedremo tra poco
  - Se il sistema host che ospita la VM ha un disco che utilizza LVM, possiamo decidere di allocare un volume logico LVM per la macchina virtuale. Abbiamo prestazioni migliori.
  - SAN: Se abbiamo uno storage iSCSI sulla rete locale (vedi *openiscsi* e simili).
  - Utilizzare un disco intero della macchina host è solitamente sconsigliato per motivi di sicurezza.

Creare un immagine disco

- File, logical volume o SAN
  - Utilizzare un file come disco per la VM è sempre l'opzione più semplice, anche se può avere qualche problema di performance come vedremo tra poco
  - Se il sistema host che ospita la VM ha un disco che utilizza LVM, possiamo decidere di allocare un volume logico LVM per la macchina virtuale. Abbiamo prestazioni migliori.
  - SAN: Se abbiamo uno storage iSCSI sulla rete locale (vedi *openiscsi* e simili).
  - Utilizzare un disco intero della macchina host è solitamente sconsigliato per motivi di sicurezza.

Creare un immagine disco

- File, logical volume o SAN
  - Utilizzare un file come disco per la VM è sempre l'opzione più semplice, anche se può avere qualche problema di performance come vedremo tra poco
  - Se il sistema host che ospita la VM ha un disco che utilizza LVM, possiamo decidere di allocare un volume logico LVM per la macchina virtuale. Abbiamo prestazioni migliori.
  - SAN: Se abbiamo uno storage iSCSI sulla rete locale (vedi *openiscsi* e simili).
  - Utilizzare un disco intero della macchina host è solitamente sconsigliato per motivi di sicurezza.

Creare un immagine disco

L'opzione più comune è l'utilizzo di files come dischi.

- Files a dimensione **variabile**:
  - Possiamo utilizzare files **sparsi**: La VM vede un disco di certe dimensioni, ma lo spazio nel file sul sistema host viene allocato man mano che la VM lo utilizza.
  - Files **qcow2**: Simili ai files sparsi, è un formato utilizzato da QEMU per i dischi che cresce man mano che viene allocato dalla VM. qcow2 supporta gli snapshot del disco. È il formato più lento.
  - Qcow2 e i files sparsi sono lenti perché il sistema host deve allocare un nuovo settore ogni volta che la VM va a scrivere nuovi dati. Inoltre se la macchina host finisce lo spazio a disposizione, il disco della VM può corrompersi.
- Files a dimensione **fissa**:
  - Possiamo allocare un semplice file di dimensioni identiche a quelle del disco visto dalla VM. In questo modo risolviamo il problema del tempo di allocazione dei blocchi

Creare un immagine disco

L'opzione più comune è l'utilizzo di files come dischi.

- Files a dimensione **variabile**:
  - Possiamo utilizzare files **sparsi**: La VM vede un disco di certe dimensioni, ma lo spazio nel file sul sistema host viene allocato man mano che la VM lo utilizza.
  - Files **qcow2**: Simili ai files sparsi, è un formato utilizzato da QEMU per i dischi che cresce man mano che viene allocato dalla VM. qcow2 supporta gli snapshot del disco. È il formato più lento.
  - Qcow2 e i files sparsi sono lenti perché il sistema host deve allocare un nuovo settore ogni volta che la VM va a scrivere nuovi dati. Inoltre se la macchina host finisce lo spazio a disposizione, il disco della VM può corrompersi.
- Files a dimensione **fissa**:
  - Possiamo allocare un semplice file di dimensioni identiche a quelle del disco visto dalla VM. In questo modo risolviamo il problema del tempo di allocazione dei blocchi

Creare un immagine disco

L'opzione più comune è l'utilizzo di files come dischi.

- Files a dimensione **variabile**:
  - Possiamo utilizzare files **sparsi**: La VM vede un disco di certe dimensioni, ma lo spazio nel file sul sistema host viene allocato man mano che la VM lo utilizza.
  - Files **qcow2**: Simili ai files sparsi, è un formato utilizzato da QEMU per i dischi che cresce man mano che viene allocato dalla VM. qcow2 supporta gli snapshot del disco. È il formato più lento.
  - Qcow2 e i files sparsi sono lenti perché il sistema host deve allocare un nuovo settore ogni volta che la VM va a scrivere nuovi dati. Inoltre se la macchina host finisce lo spazio a disposizione, il disco della VM può corrompersi.
- Files a dimensione **fissa**:
  - Possiamo allocare un semplice file di dimensioni identiche a quelle del disco visto dalla VM. In questo modo risolviamo il problema del tempo di allocazione dei blocchi

Creare un immagine disco

L'opzione più comune è l'utilizzo di files come dischi.

- Files a dimensione **variabile**:
  - Possiamo utilizzare files **sparsi**: La VM vede un disco di certe dimensioni, ma lo spazio nel file sul sistema host viene allocato man mano che la VM lo utilizza.
  - Files **qcow2**: Simili ai files sparsi, è un formato utilizzato da QEMU per i dischi che cresce man mano che viene allocato dalla VM. qcow2 supporta gli snapshot del disco. È il formato più lento.
  - Qcow2 e i files sparsi sono lenti perché il sistema host deve allocare un nuovo settore ogni volta che la VM va a scrivere nuovi dati. Inoltre se la macchina host finisce lo spazio a disposizione, il disco della VM può corrompersi.
- Files a dimensione **fissa**:
  - Possiamo allocare un semplice file di dimensioni identiche a quelle del disco visto dalla VM. In questo modo risolviamo il problema del tempo di allocazione dei blocchi

Creare un immagine disco

- Dimensione **variabile**

```
qemu-img create -f qcow2 guest.qcow2 8192M
```

  - Crea un file qcow2 per un disco da 8192MB
- Dimensione **fissa**

```
dd if=/dev/zero of=guest.img bs=1M count=8192
```

  - Alloca un file da 8192MB chiamato guest.img

Creare un immagine disco

- Dimensione **variabile**

```
qemu-img create -f qcow2 guest.qcow2 8192M
```

  - Crea un file qcow2 per un disco da 8192MB
- Dimensione **fissa**

```
dd if=/dev/zero of=guest.img bs=1M count=8192
```

  - Alloca un file da 8192MB chiamato guest.img

Configurare la rete

Le scelte sono solitamente 3:

- Scheda "**solo host**": Sulla macchina host lasciamo una interfaccia di rete che funge da interfaccia punto-punto verso quella della VM
- Scheda in **bridge**: Mettiamo l'interfaccia della VM in bridge con una interfaccia fisica della macchina host.
  - Un bridge tra due interfacce è un po' come se fossero fisicamente "switchate" insieme. Tutti i dati che arrivano su una vanno nell'altra

```
# brctl show
bridge name bridge id     STP enabled   interfaces
brvm            8000.001ec9daa349    no            eth0
                                                   tapVM
```

Configurare la rete

Le scelte sono solitamente 3:

- Scheda "**solo host**": Sulla macchina host lasciamo una interfaccia di rete che funge da interfaccia punto-punto verso quella della VM
- Scheda in **bridge**: Mettiamo l'interfaccia della VM in bridge con una interfaccia fisica della macchina host.
  - Un bridge tra due interfacce è un po' come se fossero fisicamente "switchate" insieme. Tutti i dati che arrivano su una vanno nell'altra

```
# brctl show
bridge name bridge id     STP enabled   interfaces
brvm           8000.001ec9daa349    no            eth0
                                                  tapVM
```

Configurare la rete

- Scheda in **NAT**: Possiamo mettere le varie interfacce delle VM del sistema host in un bridge. Quindi le vm è come se fossero tutte collegate ad uno switch comune. Poi possiamo cambiare le regole di firewalling dell'host per far comunicare le VM tramite NAT

```
iptables -t nat -A POSTROUTING -s 192.168.100.0/24 -j MASQUERADE
iptables -A FORWARD -i brvm -o eth0 -j ACCEPT
iptables -A FORWARD -i eth0 -o brvm -j ACCEPT
iptables -t nat -A PREROUTING -p tcp --dport 80 -j DNAT 192.168.100.2:80
```

libvirt/virsh

- libvirtd è un demone per gestire macchine virtuali basate su QEMU/KVM/LXC

  ```
  sudo apt-get install libvirt-bin
  ```

- Ci fornisce una interfaccia per definire e amministrare macchine virtuali
- Definiamo le macchine virtuali tramite dei files di definizione in formato XML
- Un tool per interfacciarsi alle API di libvirt è **virsh**

libvirt/virsh

- libvirtd è un demone per gestire macchine virtuali basate su QEMU/KVM/LXC

  ```
  sudo apt-get install libvirt-bin
  ```

- Ci fornisce una interfaccia per definire e amministrare macchine virtuali
- Definiamo le macchine virtuali tramite dei files di definizione in formato XML
- Un tool per interfacciarsi alle API di libvirt è **virsh**

libvirt/virsh

- libvirtd è un demone per gestire macchine virtuali basate su QEMU/KVM/LXC

  ```
  sudo apt-get install libvirt-bin
  ```

- Ci fornisce una interfaccia per definire e amministrare macchine virtuali

- Definiamo le macchine virtuali tramite dei files di definizione in formato XML

- Un tool per interfacciarsi alle API di libvirt è **virsh**

libvirt/virsh

- libvirtd è un demone per gestire macchine virtuali basate su QEMU/KVM/LXC

  ```
  sudo apt-get install libvirt-bin
  ```

- Ci fornisce una interfaccia per definire e amministrare macchine virtuali

- Definiamo le macchine virtuali tramite dei files di definizione in formato XML

- Un tool per interfacciarsi alle API di libvirt è **virsh**

libvirt/virsh

- Creo una VM a partire dal file xml

```
$ virsh define /tmp/foo_new.xml
```

- Guardiamo la lista delle VM disponibili

```
$ virsh list --all
 Id Name                         State
----------------------------------------
  1 foo                          running
```

- Operazioni con la VM:

```
$ virsh start foo
$ virsh reboot foo
$ virsh shutdown foo
$ virsh suspend foo
$ virsh resume foo
```

libvirt/virsh

- Creo una VM a partire dal file xml

```
$ virsh define /tmp/foo_new.xml
```

- Guardiamo la lista delle VM disponibili

```
$ virsh list --all
 Id Name                         State
----------------------------------------
  1 foo                          running
```

- Operazioni con la VM:

```
$ virsh start foo
$ virsh reboot foo
$ virsh shutdown foo
$ virsh suspend foo
$ virsh resume foo
```

libvirt/virsh

- Creo una VM a partire dal file xml

```
$ virsh define /tmp/foo_new.xml
```

- Guardiamo la lista delle VM disponibili

```
$ virsh list --all
 Id Name                         State
----------------------------------------
  1 foo                          running
```

- Operazioni con la VM:

```
$ virsh start foo
$ virsh reboot foo
$ virsh shutdown foo
$ virsh suspend foo
$ virsh resume foo
```

libvirt/virsh

- Per cancellare una VM:

```
$ virsh destroy foo_new
$ virsh undefine foo_new
```

- Oltre ad utilizzare vnc, possiamo collegarci ad una console seriale

```
$ virsh console foo_new
```

  - Teniamo conto che la macchina virtuale deve avere nella kernel line l'opzione per utilizzare una console seriale, altrimenti non funzionerà.

libvirt/virsh

- Per cancellare una VM:

```
$ virsh destroy foo_new
$ virsh undefine foo_new
```

- Oltre ad utilizzare vnc, possiamo collegarci ad una console seriale

```
$ virsh console foo_new
```

  - Teniamo conto che la macchina virtuale deve avere nella kernel line l'opzione per utilizzare una console seriale, altrimenti non funzionerà.

virt-install

virt-install è un frontend per generare la configurazione di macchine virtuali

```
virt-install -r 512 --accelerate -n GuestVM2\
     --cdrom debian-7.4.0-amd64-CD-1.iso\
     --vcpus=2 --description="descrizione vm"
     --disk /var/lib/libvirt/images/guest.qcow2,format=qcow2,bus=virtio\
     --network=bridge=testbridge\
     --graphics vnc,password=foobar,port=5904 --virt-type kvm
```

vmbuilder

Esiste un altro frontend, *vmbuilder*, che permette di creare e installare automaticamente VM basate su ubuntu:

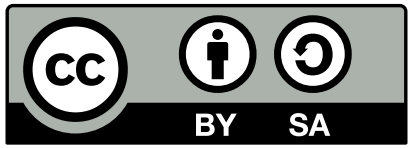
```
sudo vmbuilder kvm ubuntu --suite precise\
     --flavour server --cpus=2 --arch amd64\
     -o --libvirt qemu:///system\
     --ip 192.168.0.100 --mask=255.255.255.0\
     --net=192.168.0.0 --bcast=192.168.0.255\
     --gw 192.168.0.1 --hostname vmcorsilinux\
     --swapsize=1024 --rootsize=10240\
     --user poul --pass antani\
     --name poul --addpkg=openssh-server
```

Fonti

- Documentazione Ubuntu su KVM, libvirt e virsh: `https://help.ubuntu.com/community/KVM`

Ultima slide

Grazie per l'attenzione!



Queste slides sono licenziate Creative Commons Attribution-ShareAlike 4.0

`http://www.poul.org`