Dirks Art der Vernetzung virtueller Welten

Ein Programm zu Erstellung von Shellskripten

Dirk Geschke

Linux User Group Erding

23. Dezember 2020

Gliederung

Motivation

- testen virtueller Setups, z.B. Firewalls, VPNs, etc.
- sichere Vernetzung virtueller Systeme
- einfache Vernetzung virtueller Systeme
- flexible Vernetzung virtueller Systeme

Sicher

- saubere Trennung der VMs $\Longrightarrow$ keine Abkürzungen!
- keine Bridges auf dem Host
- Möglichkeit komplexe Netzwerke abzubilden
- mehrere Ethernetkarten pro VM möglich

Einfach

- **leicht** zu erstellende VMs
- **leicht** zu vernetzen
- **leicht** zu verwalten

Flexibel

- leicht änderbare Konfiguration der VM
- Vernetzung leicht anpassbar
- keine zufälligen MAC–Adressen
- feste MAC–Adressen pro VM

Shellskripte

- zum Verwalten werden Shellskripte verwendet
- Parameter über Shell–Variablen
- MAC–Addresse: Prefix:ID:Karte
- z.B.: 52:54:00:00:1f:01

  ID pro VM eindeutig: **2** Byte
  Prefix Herstellerprefix: **3** Byte
  Karte Netzwerkkarte der VM: **1** Byte
- $\Longrightarrow$ Eindeutige MAC-Adressen!
- Verwendung von `telnet`–Ports zur Wartung

Bis zu drei Ports

1. serieller Port: `localhost`:10000+ID
2. QEMU Monitorport: `localhost`:20000+ID
3. Optionaler VNC–Port: `localhost`:30000+ID

$\Longrightarrow$ auch ohne Netzwerk managebar!

Der VDE–Switch

VDE **V**irtual **D**istributed **E**thernet

- virtueller Switch
- kann direkt mit QEMU verlinkt werden
- einfach und flexibel
- managebar via **vdeterm**
- Vernetzung **ohne** Host möglich!
- Hostanbindung über `tap`–Interface möglich
- PXE-Boot (via Switch) möglich

Weitere Features

- VLANs, tagged und untagged
- (R)STP
- plugins möglich: Logging, sniffen per pcap, etc.

slirpvde virtueller DHCP-Router

vde_plug mit **dpipe** zur Verneztung von Switchen

wirefilter Testprogramm für z.B.:

  - packet loss
  - Bandbreiten- und Geschwindigkeitsbegrenzung
  - MTU-Beschränkung
  - Rauschen
  - Latenz

Host–Anbindung

- erfolgt über `tap`–Interface
- muss daher als **root** erfolgen
- ohne `tap`–Interface sind **root**–Rechte notwendig
- ermöglicht Zugang vom Host aber auch via Host:
  - Administrationszugang
  - Internetanbindung für Updates:
    - Webproxy
    - DNS-Server
    - oder NAT/RDR für Zugriffe
    - oder Routing
    - oder Bridge auf physikalisches Interface
- muss zuerst gestartet werden

Die Antwort auf die Idee: `kvmscript`

- erstellt passende Bash-Skripte für VMs
- setzt Ports für serielle Konsole & Co
- Konfiguration von
  1. RAM
  2. Netzwerkkarten
  3. CD-Image
  4. CPUs
  5. ...

Die Antwort auf die Idee: `kvmscript`

- Skripte leicht anpassbar via Variablen
- Voreinstellungen per `~/.kvmscript` einstellbar
- Netzwerk einfach per VDE–Shell–Variablen setzbar
- VDE–Switche werden bei Bedarf gestartet
- **Ausnahme** Admin-Switch, benötigt `root`–Rechte
- **Neu**: Live–Migration möglich

Zwischenfazit

1. Verwendung von QEMU
2. Vernetzung via VDE–Switche
3. Anbindung an den Host via `tap`–Interface möglich
4. Skripte zur einfachen Anpassbarkeit
5. Eindeutige ID pro VM und darüber eindeutige MAC–Adressen

Schmankerl. . .

- VDE–Switche sind per `vdeterm` verwaltbar
- die ID ermöglicht viele weitere nette Funktionen
- es sind auch ganz andere Szenarien realisierbar
- viele Hilfsskripte und -programme verfügbar
- Display per VNC realisierbar
- serielle Konsole via `telnet` erreichbar

Viele hilfreiche Skripte existieren:

`kvmlist`	Auflistung der laufenden VMs
`kvmrenew`	Erneuern von Skripten basierend auf alten Einstellungen
`konnect`	Zugriff per `telnet` auf seriellen Port oder QEMU–Monitorport, beendet bestehende `telnet`–Verbindung
`vdelist`	listet laufende VDE–Switche auf
`ethlist`	Welche Netzwerkkarte der VM hängt an welchem Switch?
`portlist`	An welchem Port hänge welche VM/Netzwerkkarte?
`vlan-vdecfg`	auslesen und setzen von VLANs pro Port
`name-vlan-vdecfg`	auslesen und setzen von VLANs pro VM
`net2nwdiag`	erstellt `nwdiag`–Skript für einen Netzplan

Das sind zwar keine Skripte, aber dennoch hilfreich:

`getnet` erstellt ein zufälliges RFC–1918 Netz der angegebenen Größe, sehr hilfreich!

`kvmstick` ermöglicht es einen virtuellen USB–Stick hinzuzufügen oder zu entfernen, nur bedingt tauglich

`tn-inst` automatische Installation einer Appliance bei Zugriff über seriellen Port, hat Lernmodus, auch anderweitig nutzbar!

Kann als VM–Serverdienst verwendet werden:

- Verwendung eines VDE–Switches mit `tap`–Interface
- Einhängen des `tap`–Interfaces in eine Bridge
- Hinzufügen des physikalischen Netzwerkinterfaces zur Bridge

$\Longrightarrow$ VM hängt nun direkt im LAN!

$\Longrightarrow$ VM kann ohne `root`–Rechte gestartet werden!

Weitere Option

Damit ist z.B. eine Live–Migration möglich:

- gemeinsames Dateisystem notwendig, z.B. `glusterfs`
- Live-Migration über Monitorport und dem Netzwerk
- halbautomatisch von `kvmscript` unterstützt
- hilfreich: Lock–Mechanismus
- keybasierter `ssh`-Zugang

Weitere Option

Auch eine HA–Funktionalität ist realisierbar, z.B.:

- Überwachung per Skript
- automatisches Starten auf anderem Knoten
- hilfreich: Lock–Mechanismus
- größtes Problem: VM darf nicht 2x laufen!

Fazit

- ein einfaches Projekt, dass einem das Leben sehr erleichtern kann
- viele hilfreiche Hilfsprogramme
- sehr flexibel
- nur bedingt `root`–Rechte notwendig

Praxis && Fertig :-)