23. Nov. 2024

Beschreibung und Gebrauchshinweise für den Internet Creator v0.8

Inhalt:

Die Version von Gerd’s Orginalskript, von der ich ausgegangen bin, ist von Mitte Februar 2024 und findet sich in der Datei `original_provider.sh`.
Der Ordner `streams` enthält ein Beispiel-Setup, an dem man ausprobieren kann, wie der *Internet Creator* arbeitet. Dabei handelt es sich um das denkbar einfachste Setup, nämlich ohne VRF, EVPN oder VXLAN’s. Es wird auch keine Verbindung zwischen den Providern hergestellt, sondern es geht vorläufig nur darum, die ISP’s aufzusetzen. Damit ist es auch für Neueinsteiger, Anfänger, Schüler und Auszubildende gut geeignet. Komplexere Setups sind natürlich für die Zukunft geplant.
Obwohl letztlich jeder einzelne Node (bzw. Provider) auf einem eigenen Rechner laufen soll und dann drei Rechner zu einem Cluster verbunden werden, findet hier zunächst alles auf nur auf einem einzigen Node statt, was den Vorteil hat, dass man auch vorläufig nur einen Rechner, einen DHCP-Server und eine pfSense braucht. Das Setup.pdf ist allerdings so geschrieben, dass es auch unmittelbar auf einen Aufbau mit drei einzelnen Rechnern übertragbar ist.

1. Funktionen

Der *Internet Creator* ist eine in Flask geschriebene Web App, die dazu dient, drei Netzwerke (ISP's) bestehend aus jeweils 9 Vyos-Routern automatisiert unter PVE aufzusetzen und mit Ansible zu konfigurieren.

Er ist darauf ausgelegt, sich bzgl. der Arbeitsgeschwindigkeit an die Gegebenheiten verschieden starker CPU's anzupassen. So gibt es einen Fast Modus für Rechner mit besonders starken CPU's, einen Normalmodus für schwächere CPU's und einen seriellen Modus für besonders schwache CPU's.

Man kann auswählen, ob die neueste Vyos Version vor dem Upgrade automatisch runtergeladen wird und es gibt einen (erweiterbaren) Filter für irrelevante Warnungen im Backend (siehe `inc.py`).

Es wird automatisch erkannt, ob es sich um ein Btrfs- oder ein ZFS-Dateisystem handelt und dementsprechende Skripte ausgeführt. (Für ext4 ist der Internet Creator nicht ausgelegt.)

Es gibt einen Schalter, mit dem man zwischen Darkmode und Lightmode wechseln kann.

Mit dem Ping Test kann man überprüfen, ob alle Router rauspingen können, sodass man sich dafür nicht mehr in jeden extra einloggen muss.

Auf der Seite Router Infos können Informationen über Konfiguration, Routing Tabelle, Routing Regeln, ARP-Tabelle, IPs, Interfaces und VRF einzelner Router abgerufen werden.

Backup, Restore und Upgrade einzelner Router, Serien von Routern bzw. kompletter Provider sind möglich. Außerdem sind Backup und Restore einzelner VMs anhand ihrer PVE-ID, wie insbesondere des/der DHCP-Server(s) und der pfSense(n), möglich.

Bei fehlerhaften Eingaben in die Formularfelder und bei erfolgreichen Skriptausführungen erscheinen entsprechende Systemrückmeldungen für 7 Sekunden, vorausgesetzt der Anwender hat während der Skriptausführung nicht die Seite verlassen, von der aus das Skript aufgerufen worden ist.

Seit Version v0.8 sind Eingaben für `Node` und `Limit` im Frontend überflüssig geworden: Jeder zu erstellende Router wird vorher zu zerstören versucht. Wenn er nicht existiert, wird dieser Schritt ausgelassen. Und der Ansible-Limit wird vom Programm automatisch gesetzt.

2. Systemvoraussetzungen: RAM

Ein aus 1 pfSense, 1 DHCP-Server und 9 Vyos-Routern bestehender Provider benötigt bei Verwendung eines ZFS-Dateisystems im fertig aufgesetztem Zustand je nach Provider 12 bis 13 GB RAM (bei Btrfs etwas weniger). Dabei habe ich die RAM-Größen ziemlich minimalistisch gewählt. Um ein Upgrade durchführen zu können, benötigt ein Vyos-Router allerdings erheblich mehr RAM als im Normalbetrieb, nämlich knapp 1,5 GB im Gegensatz zu knapp 600 MB. Da während der Installation auch immer ein Upgrade durchgeführt wird, muss bei einem Rechner mit nur 16 GB RAM eine Beschränkung auf die Erstellung (bzw. Upgrade bereits bestehender Router) von höchstens 5 Routern pro Durchlauf eingehalten werden. Diese Beschränkung kann auch durch Nutzung des *Serial Mode* nicht umgangen werden. Die restlichen Router können dann in einem weiteren Durchgang erstellt bzw. upgegradet werden.
Ab 20 GB RAM dürfte gerade so genug RAM vorhanden sein, um alle 9 Router in einem Durchgang aufzusetzen (von mir allerdings nicht getestet).
Den *Internet Creator* niemals mit Swap verwenden, weil das leicht zu Programmabstürzen führt.

Mit 32 GB RAM ist man auf jeden Fall auf der sicheren Seite, wenn man alle 9 Router in einem einzigen Durchlauf aufsetzen will.

3. Übersicht

Der *Provider Creator* und der *Fast Provider Creator* arbeiten beide insofern parallel, als die Upgrade- und die Konfigurations-Tasks unter Ansible immer alle aufzusetzenden Router in jeweils einem einzigen Durchgang abarbeiten. Der *Serial Mode* ist für besonders schwache CPUs gedacht, für die diese Art der parallelen Verarbeitung eine zu große Belastung darstellt. Im *Serial Mode*

werden die Upgrade- und die Konfigurations-Tasks immer nur für einen einzigen Router durchgeführt, d.h. der nächste Router kommt (innerhalb eines Playbooks) erst dann dran, wenn das Playbook für den vorigen Router abgeschlossen ist. Diese Vorgehensweise setzt die CPU-Auslastung sehr erheblich herab, was aber natürlich auch länger dauert.

4. Die Unterschiede zwischen dem Provider Creator und dem Fast Provider Creator

Es gibt drei Unterschiede zwischen dem normalen Modus des *Provider Creators* und dem Fast Modus des *Fast Provider Creators*, nämlich beim Upgrade, dem anschließenden Reboot und bzgl. des Start Delays, den es im Fast Modus nicht gibt:

Im provider_fast.sh gibt es länger anhaltende Lastspitzen insbesondere in der ersten Start-Phase, der Upgrade-Phase und den Reboot-Phasen.

Die langen Lastspitzen beim ersten Start und den Start-Teilen der Reboots ließen sich durch einen frei setzbaren Start Delay kontrollieren, der in einer wählbar langen sleep-Phase am Ende jedes Loop-Durchlaufs besteht. Die Delaywerte werden in der GUI in Sekunden angegeben.

Sehr einfach war auch die Verlangsamung des Shutdowns: Am Ende des `vyos_upgrade_fast.yml` wird mittels direktem CLI-Befehl in den Vyos Routern:

```
- name: reboot
  cli_command:
    command: "reboot now"
```

ein stark parallel stattfindender Bulk-Shutdown und -Restart ausgelöst, der eine erhebliche, anhaltende Lastspitze erzeugt. Das habe ich im `provider.sh` durch einen Shutdown mittels qm-Befehlen in einem kurzen Shutdown-Skript ersetzt (ks steht für *kurze Skripte*).

```
for i in $(seq $fr $lr); do
  sudo qm shutdown ${provider}0${provider}00$i
done
```

Der Loop durch die qm-Befehle läuft in einem für schwache Rechner genau richtigen, langsamen Tempo, sodass ein Delay-Parameter gar nicht mehr nötig ist.

Was den Kopiervorgang beim Upgrade betrifft, wurde das ansible-spezifische `net_put`, das im vyos_upgrade_fast.yml verwendet wird, im vyos_upgrade.yml durch ein scp ersetzt, wodurch es möglich wurde, in jeder Iteration eine kleine Pause einzubauen:

```
hosts:
    - router

  tasks:
    - name: "Copying {{vyos_file}} to system"
      net_put:
        src: "{{ vyos_dir }}{{ vyos_file }}"
        dest: "{{ vyos_file }}"
```

gegenüber

```
 tasks:
```

```
   - name: "Copying {{ vyos_file }} to system via scp"
     shell: "scp {{ vyos_dir }}{{ vyos_file }} {{ ansible_user }}@{{ ansible_host
}}:/home/{{ ansible_user }}/{{ vyos_file }} ; sleep 2"
     delegate_to: localhost   # scp wird auf dem Steuerungsrechner ausgeführt
     become: no   # Keine erhöhten Berechtigungen
     vars:
       ansible_ssh_pass: "{{ ansible_password }}"
```

Das Ansible-Pausenmodul hatte sich zu diesem Zweck nicht als hilfreich erwiesen.

5. Gebrauchshinweise

(1) Die Datei `create-vm-vyos.sh` habe ich dahingehend geändert, dass bei einem Neustart des PVE kein automatischer Start aller Router erfolgt, sondern die Router können über die Start-Funktion unter *General* mit einstellbarer Verzögerung gestart werden, wodurch zu große Lastspitzen bei schwachen Rechnern vermieden werden können.

(2) Der *Internet Creator* wird im Terminal durch den Befehl `python inc.py` aufgerufen. Es empfiehlt sich, in der `.bashrc` folgenden Alias zu setzen:

```
alias inc='python inc.py'
```

Auf diese Weise muss man allerdings vorher noch zwei weitere Befehle eingeben, was den Startvorgang etwas umständlich macht, nämlich:

```
source .venv/bin/activate
cd streams
```

Dies lässt sich folgendermaßen vermeiden: Führe `./go.sh` unter `/home/user/` aus - noch einfacher mit **`alias go='./go.sh'`** in der `.bashrc`

Abbrechen kann man den *Internet Creator* durch `Strg+C`.

(3) *Vermutlich* ist es bei der rein lokalen Nutzung im Heimnetz gar nicht nötig, die Datei `generate_secret_key.py` in VSCode auszuführen und den Secret Key in Zeile 10 von `inc.py` einzufügen:

```
10  app.secret_key = 'y0ur$ecr3t_k3y_w1th_r@ndomCh@racters'
```

Wenn man das aber aus irgendwelchen Gründen machen wollen würde, wird man den Secret Key nicht im Klartext ins Skript schreiben, sondern muss ihn in einer Umgebungsvariable ablegen und diese persistent machen und zwar so:

Zeile 10 im Skript ändern zu

```
10  app.secret_key = os.getenv('SECRET_KEY', 'default-secret-key')
```

Folgende Zeile in die `~/.bashrc` oder `~/.bash_profile` eintragen:

```
export SECRET_KEY='y0ur$ecr3t_k3y_w1th_r@ndomCh@racters'
```

(4) Es empfiehlt sich aus zwei Gründen, auf jeden Fall – auch wenn man auf dem PVE schon einen anderen User hat - das Skript `useradd.sh` (als root) laufen zu lassen, denn:

1) Es wird automatisch eine `.venv` angelegt und alle nötigen Python Pakete installiert, was man sonst händisch machen muss:

```
python3 -m venv .venv
source .venv/bin/activate
pip3 install -U setuptools wheel scp ansible paramiko flask flask-socketio
```

2) Die spezielle Kombination von .ssh/config- und .bashrc-Einträgen auf dem Steuerungsrechner und dem PVE erlaubt es am Ende, sehr komfortabel per SSH auf die Router zugreifen zu können, indem man z.B. nur **`u1`** auf dem Steuerungsrechner eingibt (womit man in den PVE-User einloggt) und dann **`p1r1v`** auf dem PVE, womit man dann sofort auf dem Router ist:

Steuerungsrechner:

`.ssh/config:`
```
Host u1
  HostName 192.168.10.11
  User user
```

`.bashrc:`
```
alias u1='ssh u1'
```

PVE-User:

`.ssh/config:`
```
Host p1r1v
  HostName 10.20.30.11

Host p*r*v
  User vyos
```

`.bashrc:`
```
alias p1r1v='ssh p1r1v'
```

(5) Für den automatischen Download der neuesten Vyos Version braucht man das Programm `jq` . Es muss aber nicht unbedingt auf dem PVE installiert werden, sondern kann in einem Container laufen gelassen werden. In diesem Fall muss man in den Skripten `provider.sh, provider_serial.sh, provider_Fast.sh` und `single_router.sh` den Befehl:

```
download_url=$(curl -s https://api.github.com/repos/vyos/vyos-nightly-
build/releases/latest | jq -r ".assets[0].browser_download_url")
```

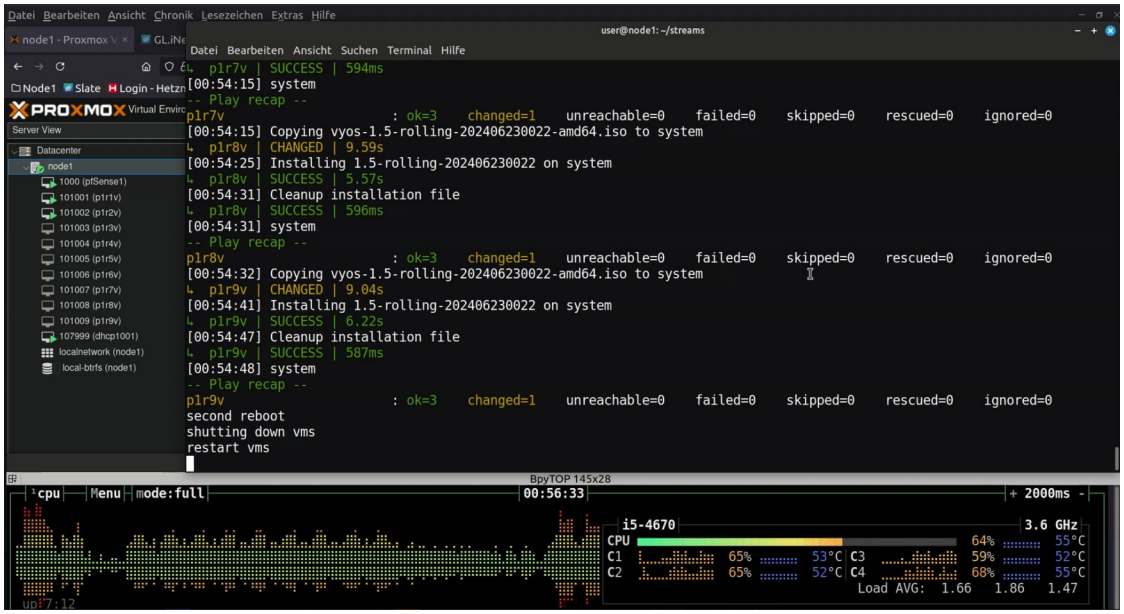
ersetzen durch:

```
download_url=$(sudo lxc-attach -n <Container ID> -- /bin/sh -c 'curl -s
https://api.github.com/repos/vyos/vyos-nightly-build/releases/latest | jq -r
".assets[0].browser_download_url"')
```

(6) Wer einen einzelnen Router erstellen, backupen, restoren oder upgraden möchte, gibt für `First Router` und `Last Router` die gleiche Zahl ein.

6. Was ist eine optimale CPU-Auslastung?

Das entscheidet natürlich letztlich jeder selber. Mein Ansatz sieht dabei so aus:
Das Grundprinzip ist, dass *idealerweise* eine Auslastung von 100% möglichst selten bis gar nicht vorkommen sollte und wenn dann nur ganz kurz, ansonsten aber eine Auslatung von über 90% sooft und so lange wie möglich angestrebt wird, damit die Erzeugung eines ISP's auch nicht länger dauert als nötig. Wenn man sich daran hält, gibt es immer noch genügend Phasen, in denen die Auslastung auf 50%, 30% oder noch weniger abfällt. Zur Überwachung der CPU-Auslastung und -Temperatur hat sich *bpytop* als gut geeignet erwiesen:



6.1 Erfahrungswerte

Hier einige Erfahrungswerte aus meinen Testläufen, um einen ungefähren Eindruck von Größenordnungen zu bekommen:

J4115 Es konnte nicht mal ein einziger Router erstellt werden. => Programm stürzt ab.

i5-4570 (Kein Serial Mode)
9 Router: Start Delay mindestens **12**
6 Router: Start Delay mindestens **6**
Bei Upgrade, Konfiguration und (Re-)Start teilweise ziemlich lang anhaltende CPU-Auslastung von bis zu 100%, allerdings bei geringer Erwärmung der CPU, sodass eine Erstellung von 9 Routern durchaus noch möglich ist. Bei schwächeren CPU's sollte m.E. für 9 Router in einem einzigen Durchgang der *Serial Mode* und ein höheres Start Delay gewählt werden.

i7-4770 (Kein Serial Mode)
9 Router: Start Delay mindestens **7**

6 Router: Start Delay mindestens **3**
Unproblematische CPU-Auslastung und niedrige Temperaturen meist zwischen 50 und 60 Grad.

i7-8700 (Kein Serial Mode)
9 Router: Start Delay mindestens **5**
Geringe CPU-Auslastung: Erreicht kaum 80%, meistens zwischen 50% und 60%, aber sehr starke Erwärmung bei Upgrade und (Re-)Start auf bis zu 87 Grad wegen häufigem Fast Boost der Taktrate auf 4,3 Mhz.
Im ***Fast Mode*** gibt es eine noch stärkere Erwärmung. Wer das vermeiden möchte kann den Turbo Boost im BIOS deaktivien, sodass eine Taktung von (hier) maximal 3,2 MHz eingehalten wird und dementsprechend auch die Temperatur wesentlich geringer ausfällt.

Fazit: Man könnte grundsätzlich auch sehr schwache Rechner wie z.B. den i5-4570 im *Fast Mode* laufen lassen. Dann kommt es über lange Strecken zu 100% CPU-Auslastung und einer starken Erwärmung. Sollte die CPU zu heiß werden, reduziert sie automatisch ihre Taktrate (außer bei extrem alten CPU's aus dem letzten Jahrhundert) und arbeitet dementsprechend langsamer. Demnach kann es vorkommen, dass der Normalmodus - nämlich ohne Drosselung der Taktrate - schneller durchläuft, als der Fast Mode, wenn er weitgehend mit einer stark reduzierten Taktung läuft.

7. To-Do-Liste für mögliche Weiterentwicklungen

(a) Einbindung von Mikrotiks.

(b) Automatisches Aufsetzen der pfSense und des DHCP-Servers. Die Skripte im Ordner Skripte sind ein erster Schritt in diese Richtung und soll zunächst durch Restore von Backups realisiert werden.

(c) Automatisches Erstellen der vyos.qcow2 und der seed.iso .