TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

Shibboleth IdP und keepalived
–
Eine einfache Lösung zur Hochverfügbarkeit

Thorsten Michels

RHRK

28. September 2016

RH Regionales Hochschul-Rechenzentrum Kaiserslautern RK

Ausgangssituation 1

IdP eine einzelne virtuelle Maschine
⇒ Vorteile für IdP-Admins:

- keine Sorgen wegen Ausfallsicherheit der Hardware
- keine Sorgen wegen Erreichbarkeit im Internet (Netzwerkinfrastruktur)
- Snapshots
- einfache Anpassung von Prozessorkernen, Festplattengröße, ...

Aber noch keine Absicherung gegen Ausfälle z. B. bei Updates oder Konfigurationsänderungen.

Ausgangssituation 2

Zwei virtuelle Maschinen + Service-IP-Adresse

- VMs laufen immer auf getrennten ESX-Hosts (Distributed Resource Scheduling per Affinitätsregel)
- Beide VMs haben identische Apache-, Tomcat- und IdP-Konfiguration auf Name bzw. IP der Service-IP-Adresse

$\Rightarrow$ Keine Unterschiede bei Wartung; Skripte und Konfigurationsdateien können ohne Änderung zwischen VMs kopiert werden.

Die VMs können sich also gegenseitig ersetzen.
Fehlt: Mechanismus, der das im Notfall automatisch macht.

keepalived

- Implementierung eines Hot-Standby-Protokolls
  (VRRP: Virtual Router Redundancy Protocol, RFC 5798)
- Healthcheck Framework:
  Überwachung von TCP-Verbindung, http/https-GET oder beliebiges Skript
- Loadbalancing Framework:
  benutzt „Linux Virtual Machine“ Kernelmodul, `ipvsadm`

Homepage: `http://www.keepalived.org`

keepalived

Dokumentation:

- https://www.keepalived.org/pdf/UserGuide.pdf
- `man keepalived.conf`
- https://github.com/acassen/keepalived/blob/master/doc/keepalived.conf.SYNOPSIS
- Google und hoffen

1. Szenario: Failover mit IdP V2

keepalived annonciert die Service-IP-Adresse,
falls es keine andere Instanz mit höherer Priorität macht.
Überprüfbar mit `ip addr`.

Auf beiden VMs läuft eine Instanz von keepalived:

- `apt-get install keepalived`
- `vim /etc/keepalived/keepalived.conf`
- `service keepalived start/stop`
- Logs in `/var/log/messages`

In unserer Konfiguration:

Es gibt keinen dezidierten Masterknoten.
Knoten 2 übernimmt die IP-Adresse, wenn Knoten 1 ausfällt,
z. B. bei Shutdown oder kontrolliertem Abschalten des Dienstes,
und behält sie dann, auch wenn Knoten 1 wieder hochfährt.

keepalived.conf

```
vrrp_instance VI_1 {
    state BACKUP
    nopreempt
    interface eth0
    virtual_router_id 53
    priority 100 # Kommentar
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass JOSHUA
    }
    virtual_ipaddress {
        93.184.216.34/28
        2606:2800:220:1:248:1893:25c8:1946/64
    }
}
```

1. Szenario: Failover mit IdP V2

Vorteile:

+ einfache Lösung, leicht umzusetzen
+ leichtes Fallback bei Updates und Konfigurationsänderungen
+ keine Probleme mit Verteilung einer Sessions auf verschiedene Knoten durch Loadbalancing

Nachteile:

– kein Loadbalancing (Ist das nötig?)
– Sessionverlust beim Schwenk
– funktioniert nicht bei scheintoten Maschinen

Weitere nötige Zusatzarbeiten:

- DB und LDAP müssen auch hochverfügbar sein.
- Ab V3: Cookie Encryption Key Rollover

Cookie Encryption Key Rollover

```
#!/bin/bash
IPS=‘ip addr‘
IP_IDP=‘dig +short idp.example.org | egrep ’^[0-9\.]*$’‘
MASTER=‘echo $IPS | grep $IP_IDP | wc -l‘

if [ $MASTER -ge 1 ]
then
  # Ich bin Master.
  java -cp "$IDP_HOME/webapp/WEB-INF/lib/*" \
    net.shibboleth.utilities.java.support.security.BasicKeystoreKeyStrategyTool \
    --storefile $IDP_HOME/credentials/sealer.jks \
    --versionfile $IDP_HOME/credentials/sealer.kver \
    --alias secret \
    --storepass \
      "$(grep ’^idp.sealer.storePassword’ $IDP_HOME/credentials/credentials.properties
       | cut -d ’ ’ -f 2)"

  cd $IDP_HOME/credentials
  scp sealer.jks sealer.kver root@knoten1:$IDP_HOME/credentials/
  scp sealer.jks sealer.kver root@knoten2:$IDP_HOME/credentials/
fi
```

2. Szenario: IdP V3 und Postgres

Master-/Slave-Replikation zwischen Postgres-Knoten

Mit neuer IdP-V3-Option

```
idp.session.StorageService = shibboleth.JPAStorageService
```

⇒ Kein Sessionverlust bei Schwenk (außer während eines Flows).

Dazu nötig:
Kommandos „an Postgres“ zum Wechsel zwischen Master- und Slave-Rolle.

Lösung: Notify-Skripte.

keepalived.conf

```
    ...
    }
    # notify_master "/pfad/zu/ichbinmaster.sh"
    # notify_slave "/pfad/zu/ichbinslave.sh"
    # notify_fault "/pfad/zu/ichbinfault.sh"
    notify "/pfad/zu/notify.sh"
    notify_stop "/pfad/zu/ichbinslave.sh"
}
```

`notify_master` und `notify_slave` werden nur bei Zustandsänderungen ausgeführt, nicht beim Start von keepalived.
Deshalb Verwendung von `notify`.

notify.sh

```
#!/bin/bash

TYPE=$1   # GROUP oder INSTANCE
NAME=$2   # Name der GROUP bzw. INSTANCE
STATE=$3  # Neuer Zustand

case $STATE in
  "MASTER") /pfad/zu/ichbinmaster.sh
            exit 0
            ;;
  "BACKUP") /pfad/zu/ichbinslave.sh
            exit 0
            ;;
  "FAULT") /pfad/zu/ichbinfault.sh
            exit 0
            ;;
  *) echo "Unknown State in keepalived-notify script"
     exit 1
     ;;
esac
```

3. Szenario: Besserer Statuscheck

Statt nur Existenz des keepalived von gegenüber zu überprüfen:
Laufen Apache, Tomcat und IdP?
→ Test auf Auslieferung der eigenen Status-Seite.

`statuscheck.sh`:

```
#!/bin/bash
wget -T 4 -q http://localhost/idp/status -O /dev/null
if [ $? -eq 0 ]
then
    exit 0 # erfolgreich
else
    exit 1 # irgendein Fehler
fi
```

keepalived.conf

```
vrrp_script statuscheck {
  script "/pfad/zu/statuscheck.sh"
  weight 50
  interval 5
  timeout 5
}
vrrp_instance VI_1 {
    state EQUAL
...
    priority 100
#    nopreempt
...
    track_script {
      statuscheck
    }
}
```

Ausblick

Noch genauere Überprüfung?

- ¿ Inhalt der Statusseite?
- ¿ Funktioniert ein Login am IdP?

→ Test durch Monitoring-System (OMD).

Vielen Dank

für die Aufmerksamkeit

und

an die Kollegen.