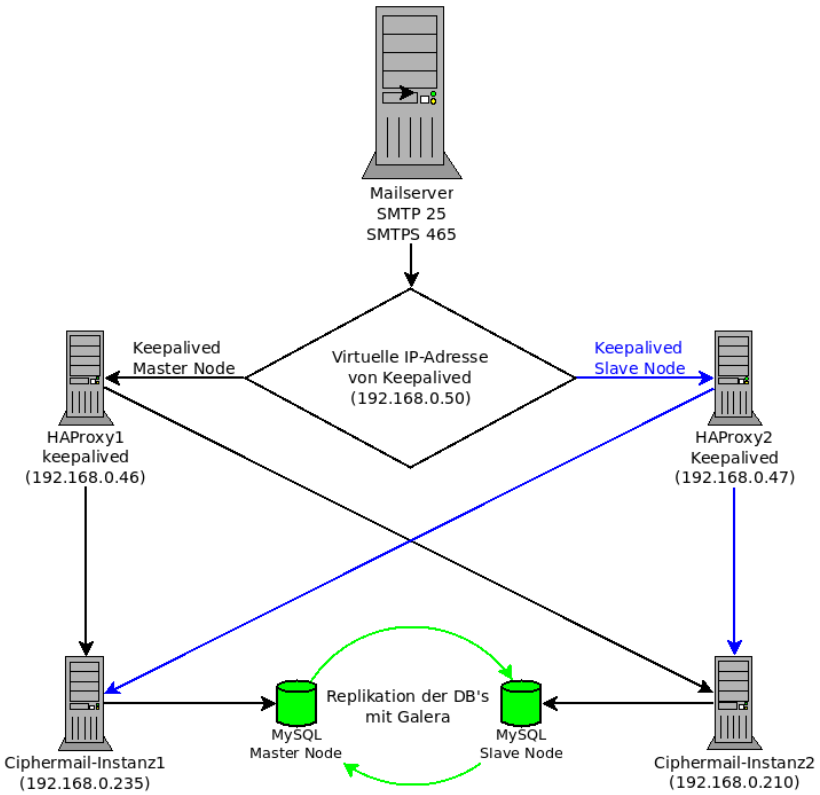
Ciphermail Galera Keepalived und HAProxy

Inhaltsverzeichnis

Zielsetzung und Diagramm

Ziel dieser Konfiguration ist es, Ciphermail ausfallsicher und performant zu betreiben.

- **HAProxy** verteilt die Anfragen vom Mailserver im Round-Robin-Verfahren abwechselnd an die Ciphermail-Instanzen (Loadbalancer).
- **Keepalived** stellt den redundanten Betrieb des Loadbalancers sicher.
- **Galera** ist für die Replikation der Datenbank zuständig.



Installation von Galera

Vorbereitungen (Ubuntu 14.04)

- Firewall

  ufw disable oder die entsprechenden IP's freigeben

- AppArmor

  ln -s /etc/apparmor.d/usr /etc/apparmor.d/disable/.sbin.mysqld

  service apparmor restart

Galera Repository einrichten

- Abhängigkeiten installieren

  apt install software-properties-common

- Schlüssel des Codership Repositories importieren

  apt-key adv --keyserver keyserver.ubuntu.com --recv BC19DDBA

- Codership Repository in die /etc/apt/sources.list eintragen:

  deb http://releases.galeracluster.com/mysql-wsrep-5.5/ubuntu trusty main

  deb http://releases.galeracluster.com/galera-3/ubuntu trusty main

- Vorzugsbehandlung des Codership Repository konfigurieren (/etc/apt/preferences.d/galera.pref)

  # Prefer Codership repository

  *Package: **

  Pin: origin releases.galeracluster.com

  Pin-Priority: 1001

Dies ist notwendig, um sicherzustellen, dass die gepatchten Versionen bevorzugt werden, z.B. wenn ein Drittanbieterprogramm *libmysqlclient20* benötigt und die OS-Version dieser Bibliothek aktueller ist.

- Aktualisieren des lokalen Caches

  apt update

Installation der Galera Pakete

apt install rsync galera-3 galera-arbitrator-3 mysql-wsrep-5.5

Sperren der Galera-Pakete gegen Updates

apt-mark hold galera-3 galera-arbitrator-3 mysql-wsrep-5.5

Anmerkungen

Galera Cluster for MySQL ist nun installiert. Diesen Vorgang muss man für jeden Knoten im Cluster wiederholen.

Konfiguration von Galera

Erster Knoten

Der Erste Knoten ist bei unserem Setup die erste Ciphermail-Instanz.

Bearbeiten der MySQL-Datenbank-Server-Konfigurationsdatei

cp -a /etc/mysql/my.cnf /etc/mysql/my.cnf.bak && vim /etc/mysql/my.cnf

folgende Zeilen sind bereits in der Konfigurationsdatei vorhanden und müssen wie nachfolgend zu sehen ist angepasst werden

bind-address = 127.0.0.1 - - > *bind-address = 0.0.0.0*
myisam-recover - - > *myisam-recover-options*

Folgende Zeilen müssen der Sektion **[mysql]** hinzugefügt werden

binlog_format=ROW
default_storage_engine=innodb
innodb_autoinc_lock_mode=2
innodb_flush_log_at_trx_commit=0
innodb_buffer_pool_size=122M
wsrep_provider=/usr/lib/galera/libgalera_smm.so
wsrep_provider_options="gcache.size=300M; gcache.page_size=300M"
wsrep_cluster_name="<muss auf allen Knoten identisch sein>"
wsrep_cluster_address="gcomm://<IP's von allen Knoten; inkl. Eigener; Kommasepariert>"
wsrep_sst_method=rsync
wsrep_node_name=<jeder Knoten hat einen individuellen Namen>
wsrep_node_address="<eigene IP Adresse>"

folgende Zeile müssen der Sektion **[mysql_safe]** hinzugefügt werden

log-error=/var/log/mysqld.log
pid-file=/var/run/mysqld/mysqld.pid

Starten des ersten Cluster-Knotens

MySQL kann nicht gleichzeitig zum in das Error Log und syslog schreiben

rm /etc/mysql/conf.d/mysqld_safe_syslog.cnf

service mysql bootstrap && tail -f /var/log/mysqld.log

Das Argument "bootstrap" darf nur beim Starten des ersten Cluster-Knotens verwendet werden.
Falls MySQL auf Grund der folgenden Meldung nicht starten kann,

mysqld_safe Directory '/var/run/mysqld' for UNIX socket file don't exists.

legt man das erforderliche Verzeichnis an:

mkdir -p /var/run/mysqld && chown mysql:mysql /var/run/mysqld/

Prüfen, ob erfolgreich gestartet wurde:

mysql -u root -p
SHOW STATUS LIKE 'wsrep_cluster_size';

der Wert von “wsrep_cluster_size“ muss **1** sein

Zweiter Knoten

Der zweite Knoten ist bei unserem Setup die zweite Ciphermail-Instanz. Hier ist die Konfiguration identisch mit der des ersten Knotens, mit der Ausnahme des Knoten-Namens und der (eigenen) Knoten-IP-Adresse

wsrep_node_name=<jeder Knoten hat einen individuellen Namen>
wsrep_node_address="<eigene IP Adresse>"

Starten des zweiten Cluster-Knotens

service mysql start && tail -f /var/log/mysqld.log

Auch hier ist zu prüfen, ob der Dienst erfolgreich gestartet wurde:

mysql -u root -p
SHOW STATUS LIKE 'wsrep_cluster_size';

der Wert von “wsrep_cluster_size“ sollte nun **2** sein

Testen des Galera Clusters

Kontrolle der Konfiguration und Installation:

```
mysql -p -u root
        SHOW STATUS LIKE 'wsrep%';
                | wsrep_local_state_comment   | Synced
                | wsrep_cluster_size          | 2
                | wsrep_ready                 | ON
```

- *wsrep_local_state_comment*: Der Wert *Synced* gibt an, dass der Knoten mit dem Cluster verbunden und betriebsbereit ist.
- *wsrep_cluster_size*: Der Wert gibt die Anzahl der Knoten im Cluster an.
- *wsrep_ready*: Der Wert ON zeigt an, dass dieser Knoten mit dem Cluster verbunden ist und Transaktionen verarbeiten kann.

Auf dem ersten Knoten wird nun zu Testzwecken eine Tabelle anlegt:

```
mysql -p -u root
        CREATE DATABASE galeratest;
        USE galeratest;
        CREATE TABLE test_table (
            id INT PRIMARY KEY AUTO_INCREMENT,
            msg TEXT ) ENGINE=InnoDB;
        INSERT INTO test_table (msg)
            VALUES ("Hello my dear cluster.");
        INSERT INTO test_table (msg)
            VALUES ("Hello, again, cluster dear.");
```

Auf dem zweiten Knoten wird dann überprüft, ob die Daten korrekt repliziert wurden

```
mysql -p -u root
        USE galeratest;
        SELECT * FROM test_table;
        | id | msg                          |
        +----+--------------------------------+
        |  1 | Hello my dear cluster.       |
        |  2 | Hello, again, cluster dear. |
```

weitere Tests siehe *http://galeracluster.com/documentation-webpages/testingcluster.html*

Galera starten und beenden (bootstrap)

Wenn der Cluster **geordnet** heruntergefahren wird, markiert Galera den Knoten, der zuletzt beendet und somit die aktuellsten Daten besitzt, mit dem Flag

safe-to-bootstrap: 1

in der Datei */var/lib/mysql/grastate.dat*

Diesen Knoten sollte man im Normalfall immer zuerst wieder starten. Alle anderen Knoten werden als "unsicher" (*safe_to_bootstrap: 0*) markiert. Falls man nun versuchen würde, einen als unsicher markierten Knoten als erstes zu starten, weigert sich Galera diesen zu starten. Falls man dennoch einen als unsicher markierten Knoten zuerst starten möchte, kann man in der Datei *grastate.dat* das Flag *safe-to-bootstrap* von 0 auf 1 setzen.

Falls der Cluster **nicht geordnet** (Crash etc.) heruntergefahren wird, haben alle Knoten das Flag *safe_to_bootstrap: 0*
Nun ist es wichtig festzustellen, welcher Knoten die letzte Transaktion im Cluster durchgeführt hat:

SHOW STATUS LIKE 'wsrep_last_committed'

In der Ausgabe nach "Recovered position" suchen und bei dem Node mit dem größten Wert das Flag *safe_to_bootstrap*: auf 1 setzen und diesen zuerst starten.

Ciphermail für MySQL vorbereiten

Alle Schritte, bis auf das Anlegen der DB, müssen auf beiden Nodes abgearbeitet werden.

Setzen der maximal erlaubten Paketgröße. Dieser Wert definiert die maximale Größe einer E-Mail und der Certificate Revocation List (CRL).

/etc/mysql/conf.d/ciphermail.cnf

```
[mysqld]
.....
max_allowed_packet = 128M
.....

service mysql restart
```

Anlegen einer Datenbank (es genügt die DB nur auf einem Node anzulegen):

```
mysql -u root -p
CREATE USER 'djigzo'@'localhost' IDENTIFIED BY 'djigzo';
CREATE DATABASE djigzo CHARACTER SET utf8 COLLATE utf8_general_ci;
GRANT DELETE,INSERT,SELECT,UPDATE,LOCK TABLES,DROP,CREATE,ALTER ON \
    djigzo.* TO 'djigzo'@'localhost';
```

Importieren der Ciphermail Tabellendefinitionen (nur auf dem Node, auf dem die DB angelegt wurde):

```
mysql -u root -p djigzo < /usr/share/djigzo/conf/database/sql/djigzo.mysql.sql
```

Konfiguration von Ciphermail an die Verwendung von MySQL anpassen. Dies geschieht in der Datei */usr/share/djigzo/wrapper/wrapper-additional-parameters.conf*

```
# this file can contain one or more parameters that will be passed as is to the JVM example:
-Dciphermail.hibernate.database.type=mysql
```

Ciphermail benötigt das Passwort der Datenbank für das Backup, dies ist in die Konfigurationsdatei */usr/share/djigzo/conf/database/mysql.cnf* einzutragen:

```
[client]
user=djigzo
password=djigzo

[mysqldump]
user=djigzo
password=djigzo
```

Setzen der Berechtigungen

```
chown djigzo:djigzo /usr/share/djigzo/conf/database/mysql.cnf && chmod 600
/usr/share/djigzo/conf/database/mysql.cnf
```

Neustart der Services

```
service djigzo restart && service tomcat7 restart
```

Entfernen der PostgreSQL Pakete

```
apt-get -y remove --purge djigzo-postgres postgresql postgresql-9.3 postgresql-client-9.3 \
        postgresql-client-common postgresql-common
```

HAProxy und Keepalived

Installation von HAProxy und Keepalived

apt install haproxy vim-haproxy keepalived

Konfiguration von Keepalived (beide Nodes)

Um Dienste an eine IP binden zu können, die auf dem System (noch) nicht existiert, ist es erforderlich, die Konfigurationsdatei */etc/sysctl.conf* zu ändern:

```
echo "net.ipv4.ip_nonlocal_bind = 1" >> /etc/sysctl.conf
sysctl -p
```

Konfiguration Keepalived Master Node

/etc/keepalived/keepalived.conf

```
global_defs {
        # Keepalived process identifier
        lvs_id haproxy_DH
        }
# Script used to check if HAProxy is running
vrrp_script check_haproxy {
        script "killall -0 haproxy"
        interval 2
        weight 2
        }
# Virtual interface
# The priority specifies the order in which the assigned interface to take over in a failover
vrrp_instance VI_01 {
        state MASTER
        interface eth0
        virtual_router_id 51
        priority 101
        # The virtual ip address shared between the two loadbalancers
```

```
        virtual_ipaddress {
        192.168.0.50
        }
track_script {
        check_haproxy
        }
}
```

Als virtuelle IP-Adresse dient in unserem Setup die 192.168.0.50 und diese muss auch in die Slave Node Konfiguration eingetragen werden.

Konfiguration Keepalived Slave Node

Die Konfiguration des Slaves entspricht der Master Konfiguration mit der Ausnahme, dass beim Slave Node der Zustand *SLAVE* ist und die Priorität niedriger ist als beim Master Node:

```
state SLAVE
priority 100
```

Abschließend den Dienst *keepalived* auf beiden Nodes starten

```
service keepalived start
```

Keepalived testen

Auf dem Master Node müssen 2 IP-Adressen konfiguriert sein: die eigene und die virtuelle

Master Node

ip addr |grep "scope global eth0"

inet 192.168.0.46/24 brd 192.168.0.255 scope global eth0

inet 192.168.0.50/32 scope global eth0

Slave Node

ip addr |grep "scope global eth0"

inet 192.168.0.47/24 brd 192.168.0.255 scope global eth0

Zieht man nun beispielsweise das Netzwerkkabel aus dem Master-Node, muss die virtuelle IP (192.168.0.50) automatisch dem Slave-Node zugeteilt werden.

HAProxy Konfiguration (beide Nodes)

Hier ist die Konfiguration der beiden Nodes identisch und sieht wie folgt aus:

HAProxy aktivieren:

```
echo "ENABLED=1" >> /etc/default/haproxy
```

Konfigurationsdatei bearbeiten:

```
cp -a /etc/haproxy/haproxy.cfg /etc/haproxy/haproxy.cfg.bak
```

/etc/haproxy/haproxy.cfg

```
listen smtp
        bind 0.0.0.0:25
        mode tcp
        option tcplog
        balance roundrobin

server ciphermail-instanz1 192.168.0.235:25 check
server ciphermail-instanz2 192.168.0.210:25 check

listen smtps
        bind 0.0.0.0:465
        mode tcp
        option tcplog
        balance roundrobin

server ciphermail-instanz1 192.168.0.235:465 check
server ciphermail-instanz2 192.168.0.210:465 check
```

Neustart des HAProxy

```
service haproxy restart
```

HAProxy testen

Mehrere E-Mails werden an die virtuelle IP von Keepalived verschickt, der HAProxy verteilt diese nun via Round-Robin abwechselnd an die beiden Ciphermail-Instanzen .

/var/log/haproxy.log

Feb 15 12:06:43 haproxy1 haproxy[11033]: 192.168.0.45:48602 [15/Feb/2018:12:06:43.666] smtp smtp/ciphermail-instanz1 1/1/295 288 -- 0/0/0/0/0 0/0

Feb 15 12:06:57 haproxy1 haproxy[11033]: 192.168.0.45:48604 [15/Feb/2018:12:06:57.714] smtp smtp/ciphermail-instanz2 1/1/240 288 -- 0/0/0/0/0 0/0

Feb 15 12:07:13 haproxy1 haproxy[11033]: 192.168.0.45:48608 [15/Feb/2018:12:07:13.523] smtp smtp/ciphermail-instanz1 1/1/228 288 -- 0/0/0/0/0 0/0

Feb 15 12:22:10 haproxy1 haproxy[11033]: 192.168.0.45:48736 [15/Feb/2018:12:22:10.173] smtp smtp/ciphermail-instanz2 1/1/220 288 -- 0/0/0/0/0 0/0