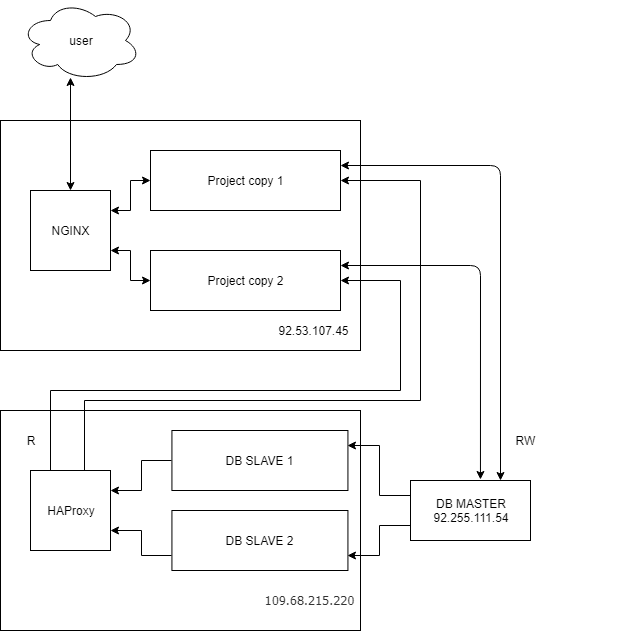
**Балансировка и отказоустойчивость.**

****

1. Разворачиваем Haproxy для балансировки доступа к слейвам Mysql. Для этого используем docker-compose.yml:

version: '3.7'

services:

mysql-master1:

image: uoles/mysql

container\_name: mysql-1

command: --default-authentication-plugin=mysql\_native\_password

env\_file: .env

ports:

- "6611:3306"

cap\_add:

- all

volumes:

- /home/uoles/project/mysql\_db\_haproxy\_1:/var/lib/mysql

- /home/uoles/project/mysql\_db\_haproxy\_1\_conf:/etc/mysql

environment:

- MYSQL\_ROOT\_PASSWORD:${MYSQL\_ROOT\_PASSWORD}

networks:

default:

aliases:

- mysql

mysql-master2:

image: uoles/mysql

container\_name: mysql-2

command: --default-authentication-plugin=mysql\_native\_password

env\_file: .env

ports:

- "6612:3306"

cap\_add:

- all

volumes:

- /home/uoles/project/mysql\_db\_haproxy\_2:/var/lib/mysql

- /home/uoles/project/mysql\_db\_haproxy\_2\_conf:/etc/mysql

environment:

- MYSQL\_ROOT\_PASSWORD:${MYSQL\_ROOT\_PASSWORD}

networks:

default:

aliases:

- mysql

haproxy:

image: haproxytech/haproxy-ubuntu:latest

container\_name: haproxy

ports:

- "2999:2999"

- "3306:3306"

cap\_add:

- all

volumes:

- ./haproxy.cfg:/usr/local/etc/haproxy/haproxy.cfg

networks:

default:

aliases:

- mysql

Для настройки Haproxy настраиваем конфиг haproxy.cfg:

#---------------------------------------------------------------------

# Example configuration for a possible web application. See the

# full configuration options online.

#

# https://www.haproxy.org/download/1.8/doc/configuration.txt

#

#---------------------------------------------------------------------

#---------------------------------------------------------------------

# Global settings

#---------------------------------------------------------------------

global

# to have these messages end up in /var/log/haproxy.log you will

# need to:

#

# 1) configure syslog to accept network log events. This is done

# by adding the '-r' option to the SYSLOGD\_OPTIONS in

# /etc/sysconfig/syslog

#

# 2) configure local2 events to go to the /var/log/haproxy.log

# file. A line like the following can be added to

# /etc/sysconfig/syslog

#

# local2.\* /var/log/haproxy.log

#

log 127.0.0.1 local2

chroot /var/lib/haproxy

pidfile /var/run/haproxy.pid

maxconn 4000

user haproxy

group haproxy

daemon

# turn on stats unix socket

stats socket /var/lib/haproxy/stats

# utilize system-wide crypto-policies

ssl-default-bind-ciphers PROFILE=SYSTEM

ssl-default-server-ciphers PROFILE=SYSTEM

#---------------------------------------------------------------------

# common defaults that all the 'listen' and 'backend' sections will

# use if not designated in their block

#---------------------------------------------------------------------

defaults

#mode http

log global

option tcplog

retries 2

timeout connect 4s

timeout client 30m #1440m

timeout server 30m #1440m

timeout check 10s

#maxconn 3000

listen stats

mode http

bind \*:2999

stats enable

stats uri /stats

stats realm Strictly\ Private

stats auth devops:Haproxymysql2021@

listen mysql-cluster

mode tcp

option tcpka

bind \*:3306

option mysql-check user haproxy\_user

balance roundrobin

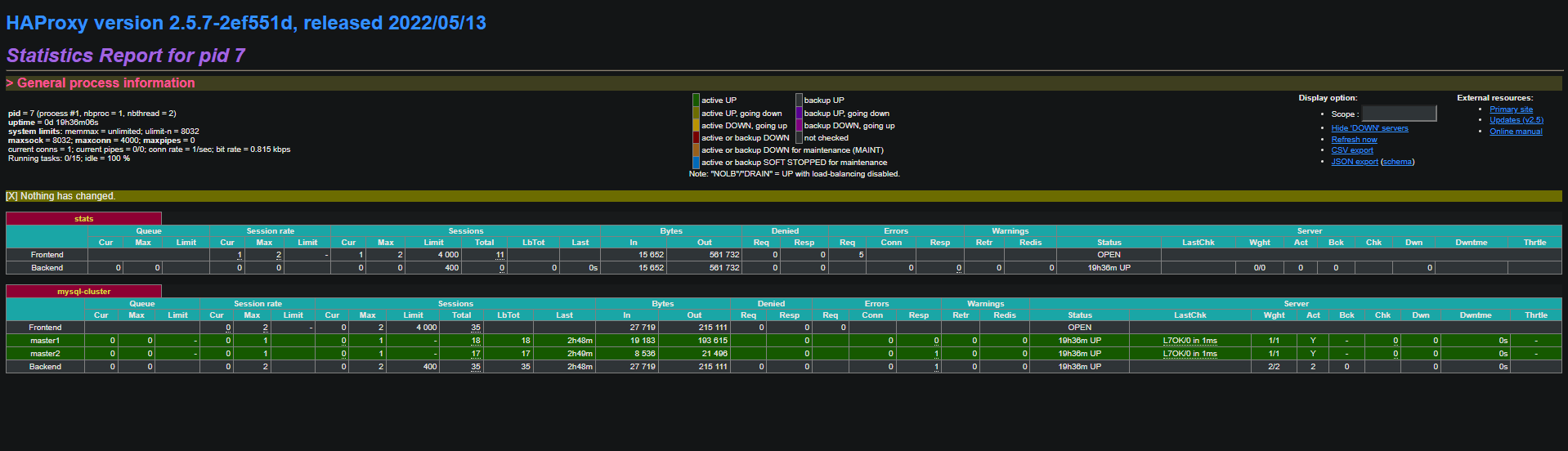
server master1 mysql-master1:3306 weight 1 check

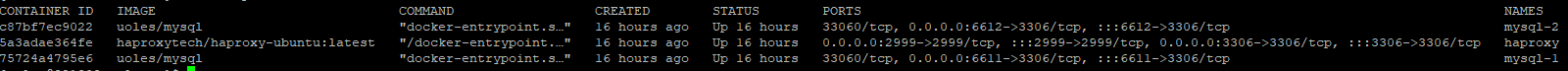
server master2 mysql-master2:3306 weight 2 check

Для работы балансировщика добавляем пользователя в слейвы:

CREATE USER 'haproxy\_user'@'%';

Проверяем работоспособность Haproxy:





1. Разворачиваем копии приложения и поднимаем nginx:

docker run --name social-project-1 \

-p 8081:8081 \

-d uoles/mooncake-social

docker run --name social-project-2 \

-p 8082:8081 \

-d uoles/mooncake-social

docker run --name social-nginx -p 80:80 \

-v /home/uoles/project/nginx/settings/nginx.conf:/etc/nginx/nginx.conf \

-v /home/uoles/project/nginx/log:/var/log/nginx \

-d uoles/nginx

Конфигурация nginx:

#user nobody;

worker\_processes 1;

events {

worker\_connections 1024;

}

http {

log\_format compression '[$time\_local] $remote\_addr - $remote\_user - $server\_name $host to: $upstream\_addr: '

'"$request" $status $body\_bytes\_sent '

'"$http\_referer" "$http\_user\_agent" "$gzip\_ratio"';

upstream nginx-backend {

ip\_hash;

server 92.53.107.45:8081 fail\_timeout=0 max\_fails=1;

server 92.53.107.45:8082 fail\_timeout=0 max\_fails=1;

}

server {

listen 80;

server\_name localhost;

gzip on;

access\_log /var/log/nginx/nginx.access.log compression;

location / {

proxy\_pass http://nginx-backend;

proxy\_set\_header Host $host;

proxy\_set\_header Cookie $http\_cookie;

proxy\_set\_header X-Forwarded-Host $http\_host;

proxy\_read\_timeout 1s;

proxy\_connect\_timeout 1s;

}

}

}

Dockerfile:

FROM centos:centos7

MAINTAINER Maksim Kulikov <max.uoles@rambler.ru>

# Configure my repo to use my custom Nginx with modsec

RUN yum -y install wget nano tar mc

RUN yum clean all

RUN yum update -y

RUN yum install yum-utils

RUN rpm -Uvh http://fedora-mirror01.rbc.ru/pub/epel/epel-release-latest-7.noarch.rpm

# install nginx

RUN yum install nginx -y

RUN systemctl enable nginx

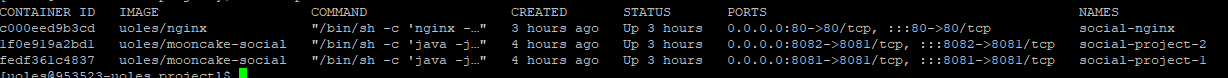
RUN mkdir /var/cache/nginx

# expose HTTP

EXPOSE 80

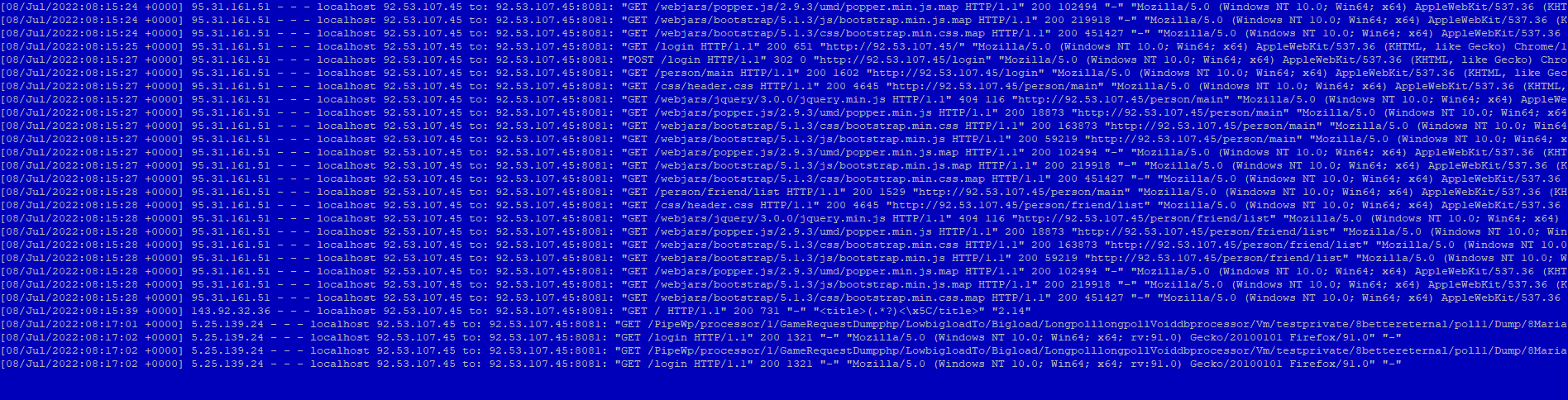
# Run

ENTRYPOINT nginx -g "daemon off;"

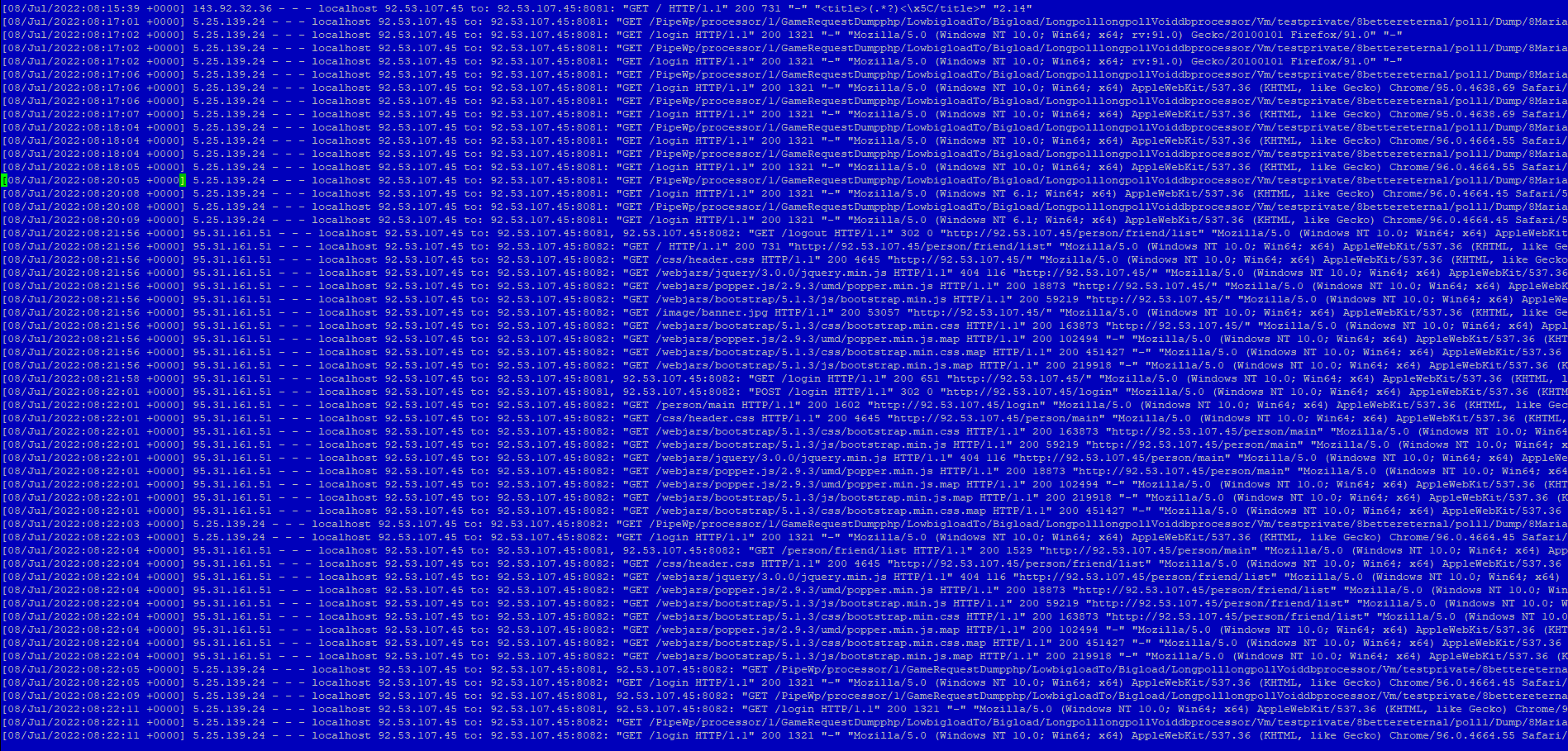


1. Проверка работы системы при отключении узлов приложения или БД:

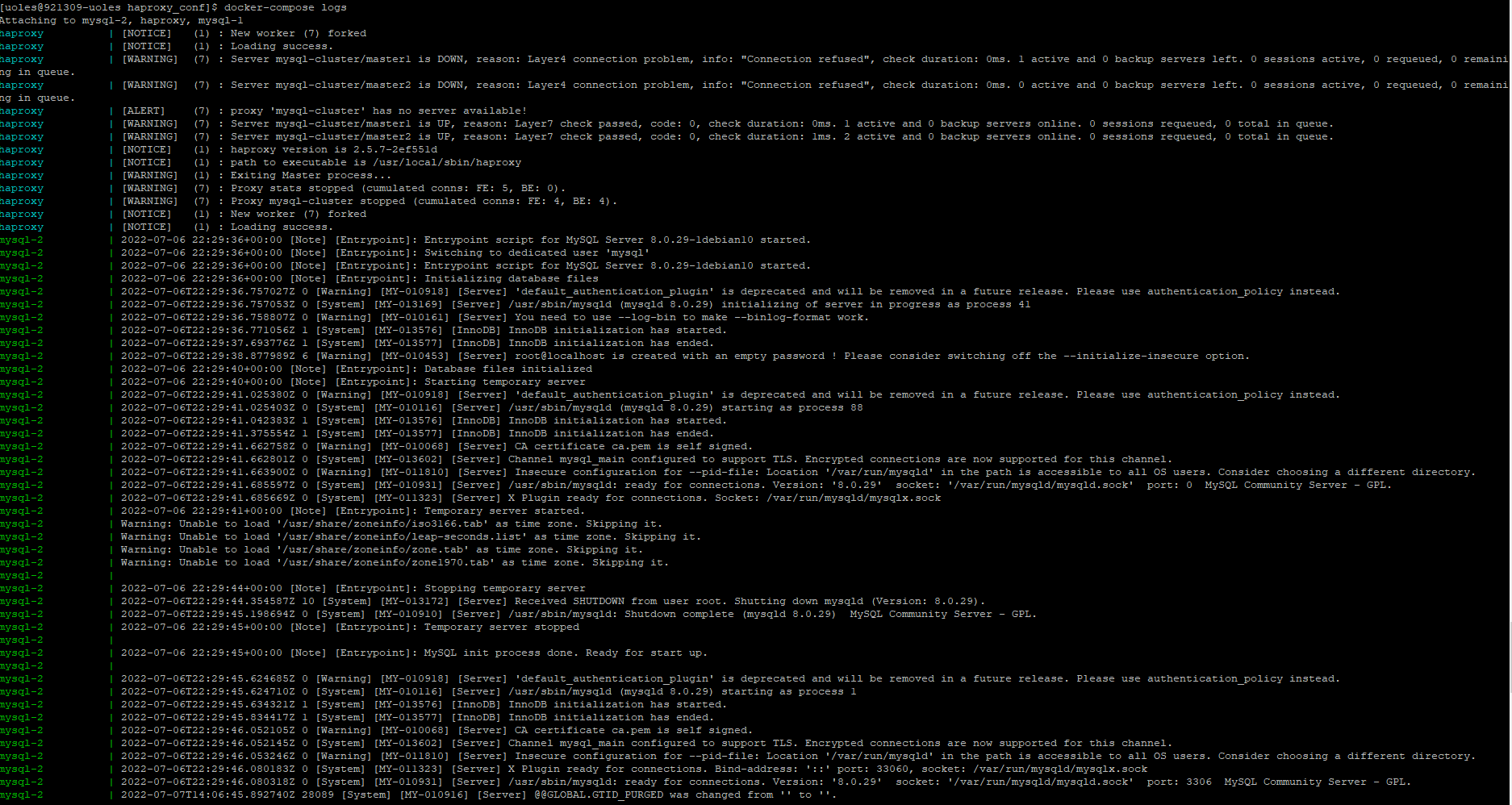
Запросы до отключения приложения на порту 8081:

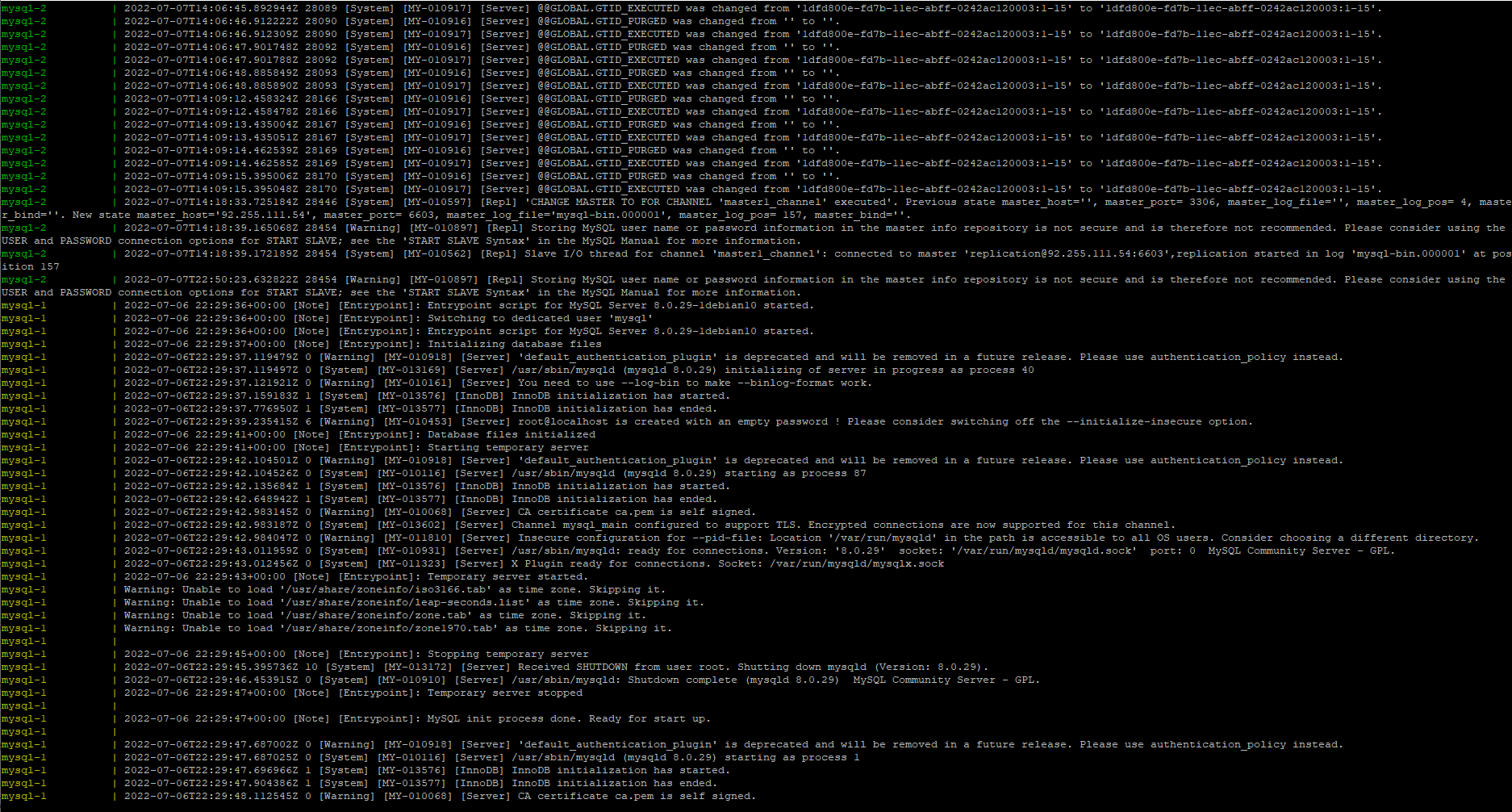


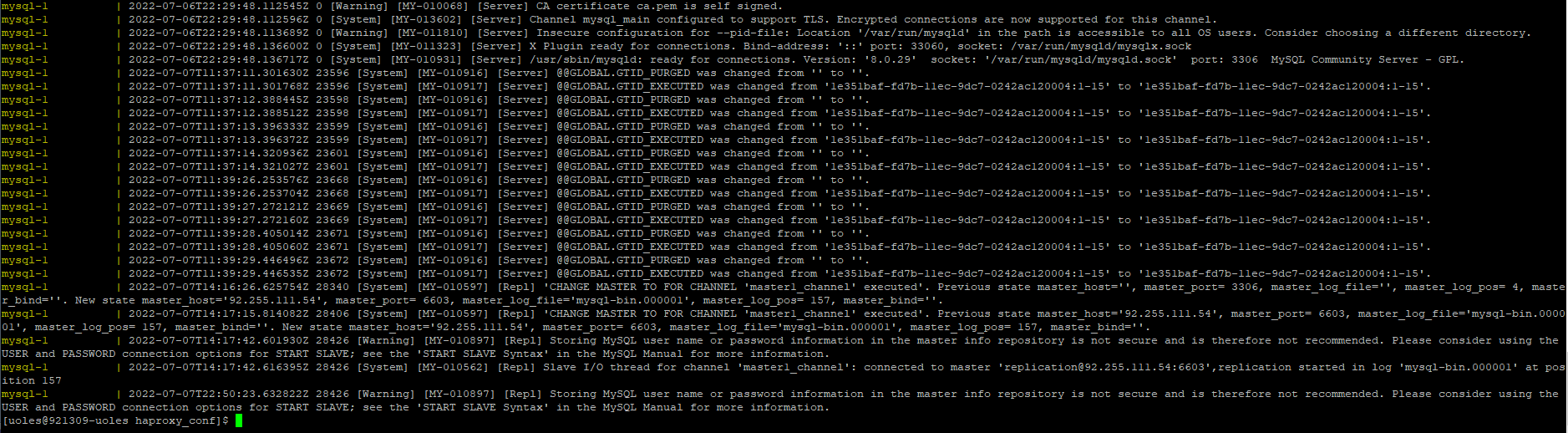
После отключения приложения на 8081 порту запросы стали ходить через приложение на порту 8082. Работоспособность системы сохранилась. Логи:



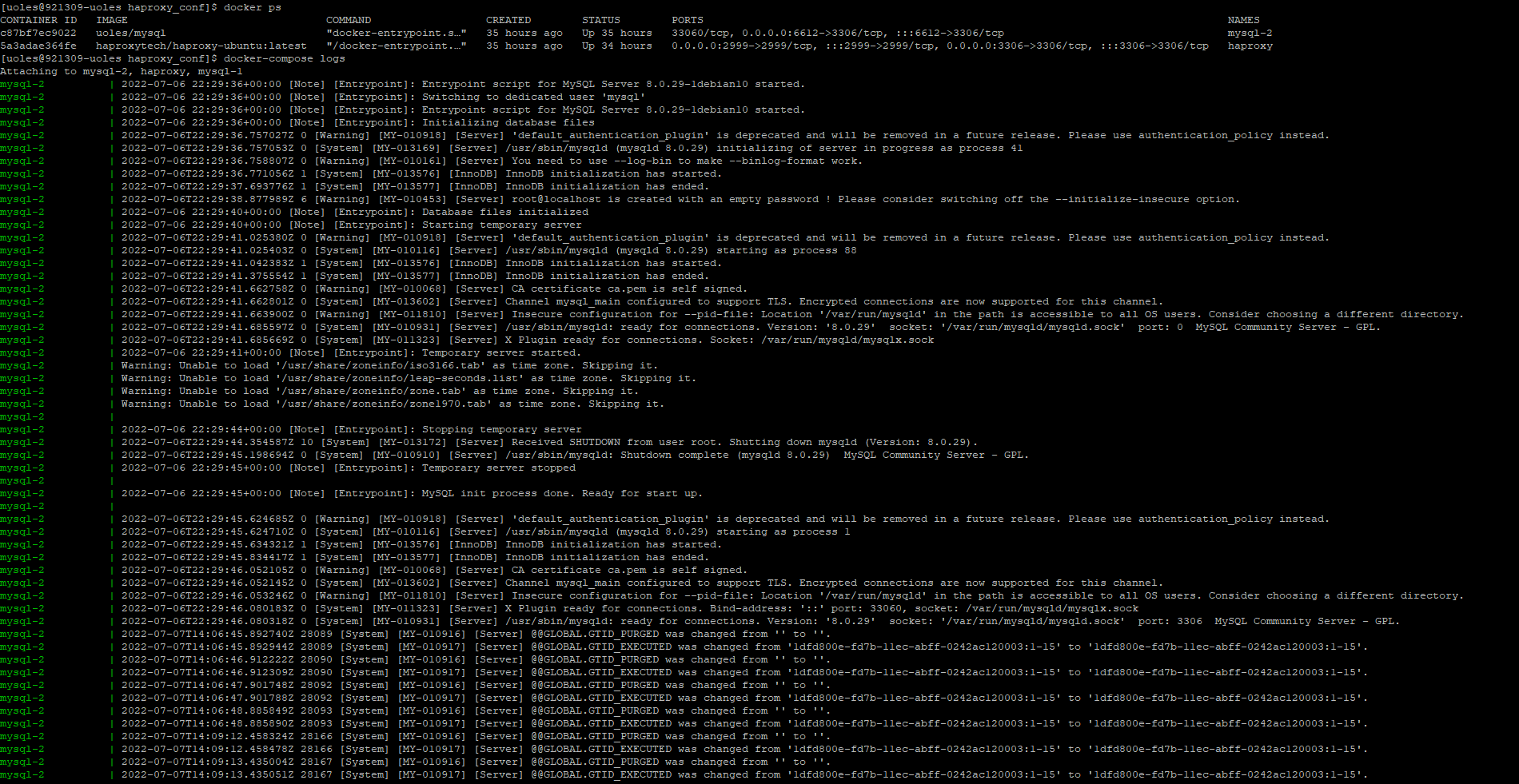
Логи работы Haproxy и Mysql до отключения одного слейва:

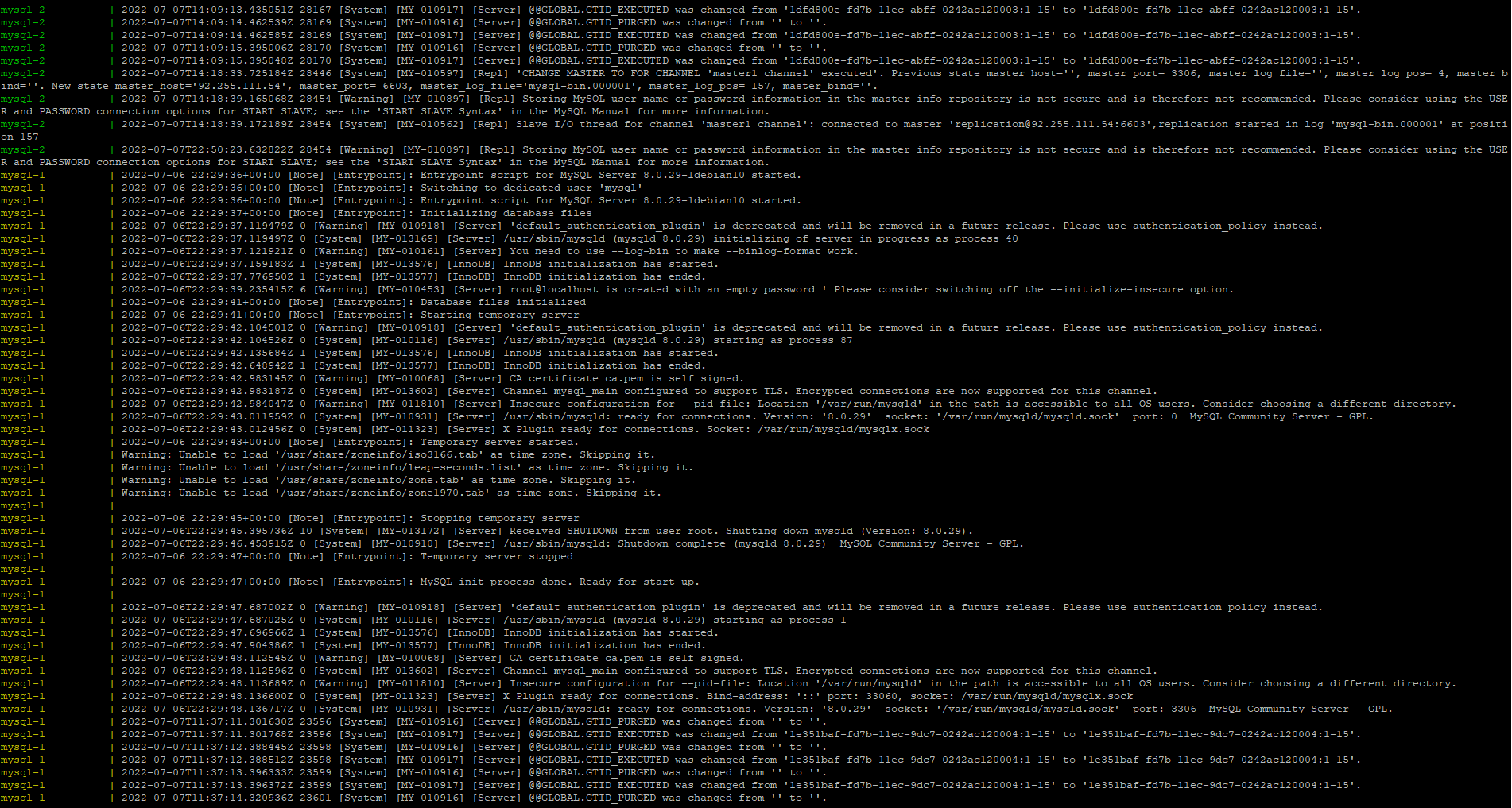


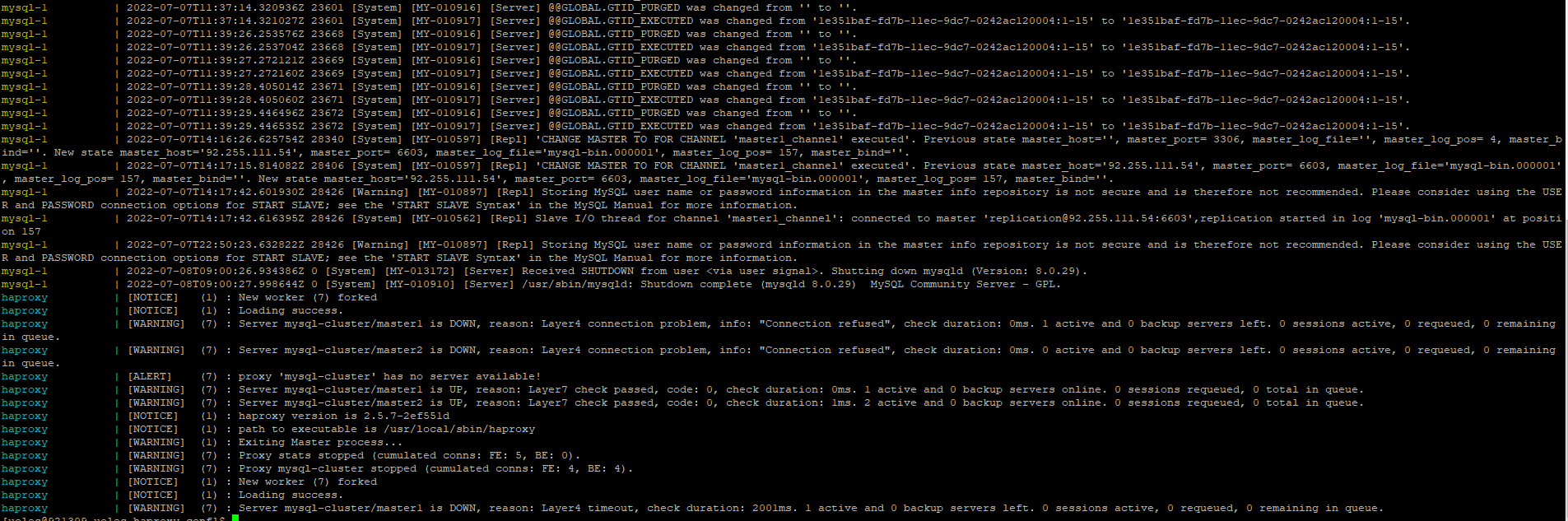


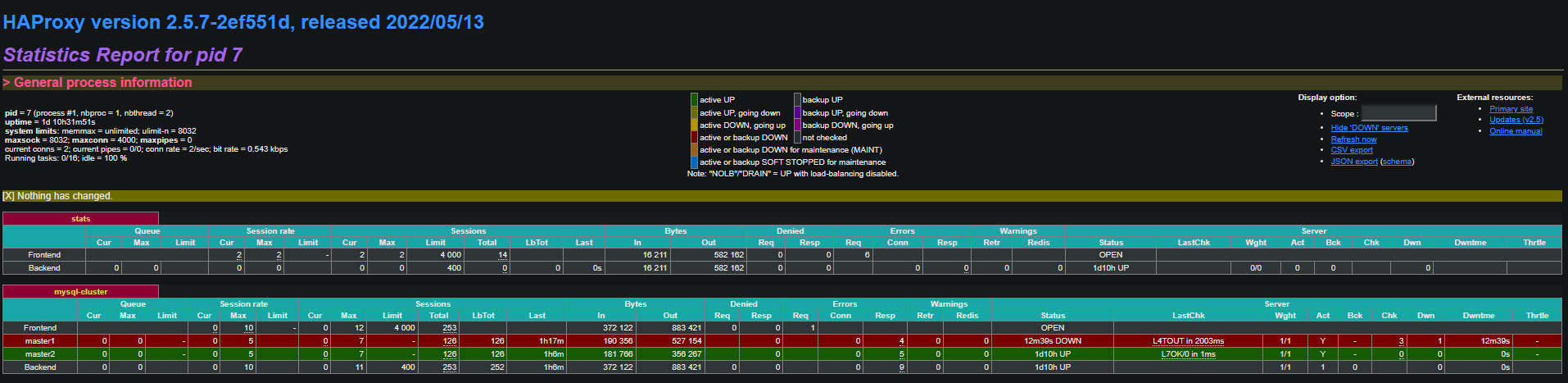


Отключаем слейв mysql. Как видно из логов mysql-1, контейнер завершил работу. В логах Haproxy появилась строка, что master1 завершил работу. Логи:

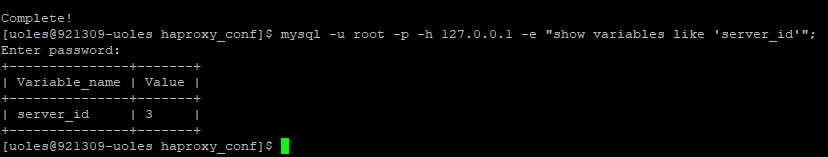








Проверка работоспособности Mysql. Подключаюсь локально к Haproxy, приходит ответ от инстанса mysql-2. Работоспособность всей системы сохранилась:



mysql-1 master1 server\_id=2

mysql-2 master2 server\_id=3