Практическая работа № 6. Построение отказоустойчивой архитектуры с использованием балансировщика HAProxy.

HAProxy (High Availability Proxy) - популярный прокси сервер для Linux, Solaris и FreeBSD с возможностью балансировки нагрузки TCP/HTTP с открытым программным кодом. Его основная задача - повышение производительности серверной среды путем распределения рабочей нагрузки среди нескольких серверов (web, приложения, базы данных). Им пользуются такие известные проекты как GitHub, Imgur, Instagram и Twitter.

Основные термины.

Список контроля доступа (ACL)

Выбор нужного правила по типу/постфиксу запроса и применение соответствующего действия (блокировка, перенаправление)

Использование ACL позволяет гибко управлять трафиком, перенаправляя его в зависимости от большего количества факторов, среди которых есть соответствие паттерну или ограничение количества соединений с бекэндом и др.

Пример ACL:

acl test_host hdr(host) -i mysite.loc
Правило test_host будет применено в случае, если пользователь обращается к домену `mysite.loc`.

Backend

Бекэндом называется набор серверов, получающих перенаправленные запросы пользователей. Они указываются в разделе настройки backend. В общем случае бекэнд определяется следующими параметрами: алгоритм распределения нагрузки , набор серверов и портов
Бекэнд может включать в себя как один, так и несколько серверов. Другими словами, добавляя больше серверов вы увеличиваете максимально возможную нагрузку, распределяя ее по всем серверам. Так же, таким образом, повышается отказоустойчивость в случае, если отказал один из серверов.
Пример настройки backend-а для нашего acl:

```
acl test_host hdr(host) -i mysite.loc
use_backend test if test_host
backend test
        balance roundrobin
        server srv1 172.17.0.2:80 check
        server srv2 172.17.0.4:80 check
```

Строка balance roundrobin указывает на алгоритм балансировки, также доступны (leastconn, source).

Параметр check означает, что эти серверы должны проверяться на работоспособность.

Как это работает - запрос, например mysite.loc, попадает на проксирующий-сервер, он ищет acl под которые попадает этот запрос (`acl test_host hdr(host) -i mysite.loc`), далее ищет подходящий backend (`use_backend test if test_host`), затем идет в секцию данного бэкенда видит:

```
backend test
        balance roundrobin
        server srv1 172.17.0.2:80 check
        server srv2 172.17.0.4:80 check
```

Значит используем балансировку round robin, если в прошлый раз такой запрос я отдал серверу srv1, значит теперь отдам его srv2, заодно проверю их состояние (рис. 1).

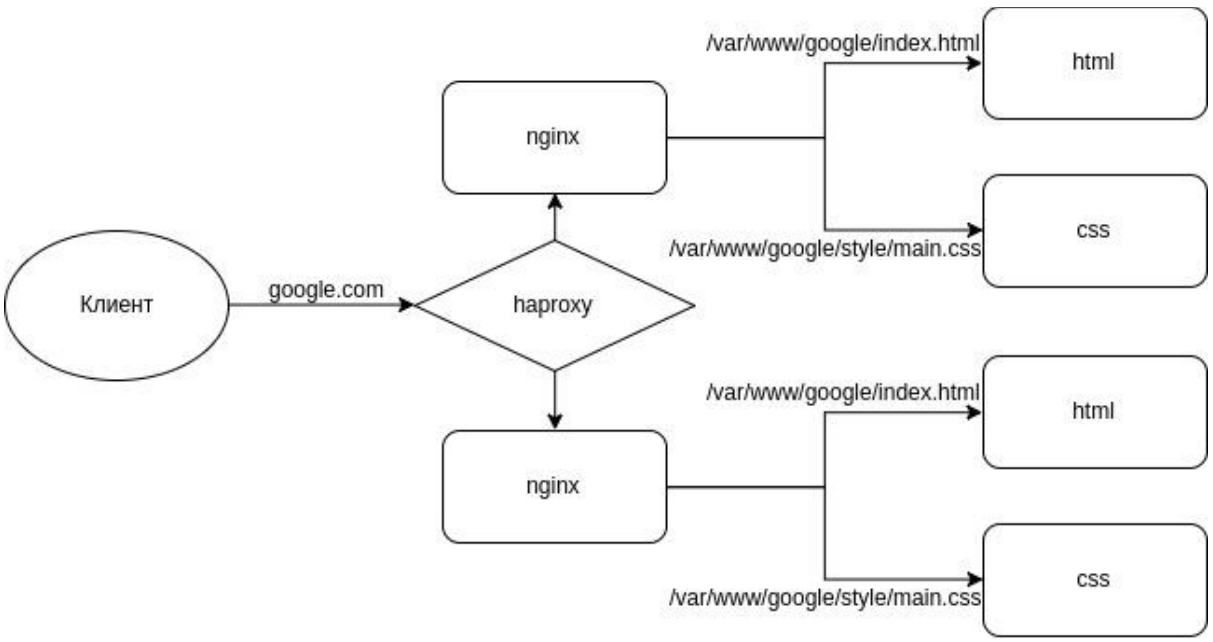

Рисунок 1. Общая схема функционирования HAProxy.

Установка HAProxy.

RHEL/CentOS: `yum install haproxy`
Ubuntu/Debian: `apt-get install haproxy`

Конфигурационный файл - `/etc/haproxy/haproxy.cfg`
Запуск/перезапуск службы - `service/systemctl haproxy start(restart)`

Пример конфигурационного файла:

```
global
        log /dev/log    local0 err
        log /dev/log    local1 err
        daemon

defaults
        log     global
        mode    http
        option  httplog
        option  dontlognull
        option  dontlog-normal
        option  forwardfor #except 127.0.0.1
        timeout client 900s
        timeout server 900s

frontend http
bind *:80
acl test_host hdr(host) -i mysite.loc   # ваш сайт
use_backend test if test_host

backend test
        balance roundrobin
        server srv1 172.17.0.2:80 check # ip первого сервера
        server srv2 172.17.0.4:80 check # ip второго сервера
```

Внимание
Конфигурации nginx на двух виртуалках должны быть идентичны.

Задание на лабораторную работу.

1. На трех виртуальных машинах развернуть - haproxy, nginx + nginx. Настроить так, чтобы при запросе на машине с haproxy по адресу вашего сайта из лабораторной 1, запрос проксировался на srv1/srv2 по round-robin. Убедится в том, что запрос ходит на сервера действительно по очереди, можно проверив `/var/log/nginx/access.log`. Там содержится тип запроса, URL, время.
2. Ограничить кол-во соединений на одной из машин в 1000 запросов в единицу времени. Протестировать через ab, убедится в том, что запросы, распределяются неравномерно.
3. Отключить одну виртуальную машину(остановить nginx), убедится в том, что запросы корректно обрабатываются другой виртуальной машиной.

Контрольные вопросы.

1. Перечислить алгоритмы балансировки, описать различия между ними.
2. Назначение балансировщика
3. От чего зависит надежность архитектуры? Какие существуют способы повышения надежности?