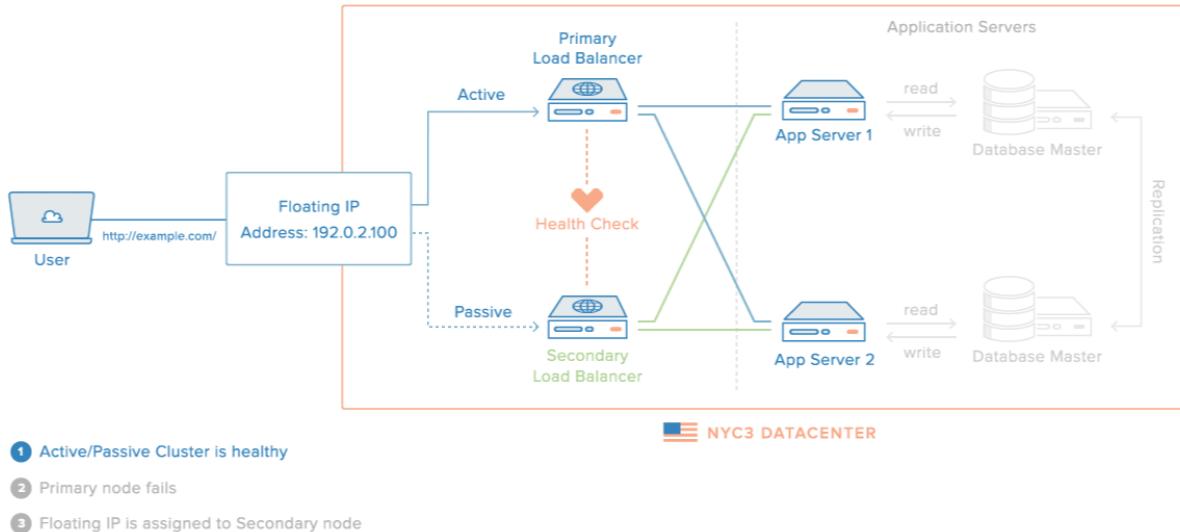




# Cluster

## Структура



1. Есть два **бэкенд-сервера**, на которых запущены веб-серверы. Эти веб-серверы могут хостить вообще разные, не связанные друг с другом сайты. Обозначим эти серверы **webserver-1** и **webserver-2**. На них запущены два сайта: [fit.nsu.ru](#) и [fit2.nsu.ru](#).

2. Есть два **фронтенд-сервера**, которые балансируют нагрузку между двумя веб-серверами (если они хостят одно и то же). Обозначим их **proxy-1** и **proxy-2**.
3. Фронтенд-сервера делят между собой **плавающий виртуальный IP адрес**. Если оба сервера онлайн, то запросы на плавающий IP поступают на proxy-1. Если proxy-1 оффлайн, то запросы поступают на proxy-2.
4. Есть **DNS-сервер**, который отвечает за зону **nsu.ru**.  
Он содержит записи для fit.nsu.ru и fit2.nsu.ru, в которых указан плавающий IP-адрес.

## Реализация

### Что понадобится:

- Около 35 Гб свободного места на диске
- VMware Workstation
- Debian 4.18

### Стек:

- **corosync** - отвечает за сетевое взаимодействие узлов (запуск/остановка ресурсов, узлов и т.д.)
- **pacemaker** - менеджер ресурсов. Он реагирует на события, происходящие в кластере: отказ или присоединение узлов, ресурсов, переход узлов в сервисный режим и другие административные действия.
- **pcs** - утилита для конфигурации pacemaker и corosync.
- **haproxy** - прокси высокой доступности. Обеспечивает балансировку нагрузки на бэкенд-сервера.
- **apache2** - веб-сервер.
- **bind9** - DNS-сервер.

## Приступаем

1. Сначала настроим виртуальную сеть в VMware для виртуальных машин.
  1. Вкладка Edit > Virtual Network Editor
  2. Выбираем сеть с типом NAT (если такой нет, то создайте). Убедитесь, что стоят галочки на 'Connect a host virtual adapter' и 'Use local DHCP...'.

Первая нужна, чтобы можно было из хост-машины стучаться на виртуальные, вторая - для управления распределением адресов внутри этой сети из VMware (чтобы IP виртуалок не менялись каждый раз при перезапуске хост-системы).
  3. OK
2. Создаём новую виртуальную машину с обычными настройками.

Выбираем ISO-образ дистрибутива Debian. Даём машине имя. Пусть это будет **proxy-1**. Выбираем расположение файлов на хост-машине.

Дисковое пространство - **не менее 7 Гб** (у меня система с кластером занимает 5.3 Гб). ОЗУ достаточно 1 Гб. Число процессоров - 1, число ядер ставим равным числу ядер процессора (с учетом потоков). Принтер и аудио можно удалить.
3. Устанавливаем Debian, обязательно выбираем в конце зеркало репозитория apt. Имя пользователя и машины - произвольные, но желательно дать осмысленные имена.
4. Запускаем Debian. Графический интерфейс не понадобится, можно отключить его автозапуск:

```
sudo systemctl set-default multi-user.target
```

Перезагружаемся.
5. Устанавливаем необходимый стек:

```
sudo apt-get update && sudo apt-get upgrade
```

```
sudo apt-get install corosync pacemaker pcs apache2 haproxy
```
6. Клонируем машину (**полностью**) в VMware 4 раза. Запускаем каждый клон.

Необходимо поменять имена хостов на каждой машине, т.к. сейчас они все одинаковые:

```
sudo nano /etc/hostname
```

На первой строчке пишем осмысленное имя хоста (proxy-1, proxy-2, webserver-1, webserver-2, dns-server соответственно).

В каждом клоне выполняем

```
ip a
```

Запоминаем inet адрес интерфейса ens33 для каждого хоста:

```
proxy-1@proxy-1:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 00:0c:29:c0:a3:08 brd ff:ff:ff:ff:ff:ff
    inet 192.168.26.128/24 brd 192.168.26.255 scope global dynamic noprefixroute ens33
        valid_lft 1225sec preferred_lft 1225sec
    inet6 fe80::20c:29ff:fec0:a308/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
proxy-1:~$
```

Для удобства обращения к другим хостам в каждом хосте добавим новые записи в /etc/hosts:

```
sudo nano /etc/hosts
```

Для каждого хоста дописываем после 127.0.0.1 строку вида 'IP имя':

```
127.0.0.1      localhost
192.168.26.128 proxy-1
192.168.26.129 proxy-2
```

## Создание кластера

1. Переходим в режим суперпользователя (**далее все команды выполнять в этом режиме**):

```
sudo su -
```

**Далее команды необходимо выполнять на каждом узле-балансировщике.**

2. Для управления кластером будем пользоваться утилитой pcs. При установке pacemaker автоматически был создан пользователь hacluster. Для использования pcs нужно задать пароль пользователю hacluster:

```
passwd hacluster
```

### 3. Создаём кластер:

```
pcs host auth proxy-1 proxy-2
```

Вместо proxy-1 и proxy-2 пишем имена из /etc/hostname каждого узла.

Вводим имя пользователя **hacluser**, пароль задавали ранее.

```
pcs cluster setup proxy-1 addr=192.168.26.128 proxy-2 addr=192.168.26.129
```

**Далее команды можно выполнять на любом из двух узлов-балансировщиках.**

### 4. Запускаем кластер на всех узлах:

```
pcs cluster start --all
```

Автоматический запуск при загрузке:

```
pcs cluster enable --all
```

Посмотреть статус кластера:

```
pcs status
```

## Настройка кластера

1. Отключаем **STONITH** (этот механизм позволяет выключать/включать/перезагружать узлы кластера. Обычно этот механизм реализуется с помощью специальных сетевых устройств с удаленным управлением или через специальные платы управления сервером):

```
pcs property set stonith-enabled=false
```

**Кворум** определяет минимальное число работающих узлов в кластере, при котором кластер считается работоспособным. По умолчанию, кворум считается неработоспособным, если число работающих узлов меньше половины от общего числа узлов. Так как узла у нас всего два, то кворума у нас не будет, поэтому отключаем эту политику:

```
pcs property set no-quorum-policy=ignore
```

Посмотреть какие у нас в итоге получились настройки можно командой:

```
pcs property
```

```
root@proxy-1:~# pcs property
Cluster Properties:
  cluster-infrastructure: corosync
  cluster-name: cluster
  dc-version: 2.0.1-9e909a5bdd
  have-watchdog: false
  no-quorum-policy: ignore
  stonith-enabled: false
```

2. Проверим правильность настроек:

```
crm_verify -V -L
```

Если всё хорошо, то ничего не должно быть напечатано в терминале.

3. На **всех** балансировщиках настроим автозагрузку кластера:

```
systemctl enable pcsd
```

```
systemctl enable corosync
```

```
systemctl enable pacemaker
```

```
pcs cluster enable --all
```

## Создание виртуального плавающего IP-адреса

1. Выбираем незанятый IP-адрес в подсети, созданной VMware. Возьмём следующий за самым последним IP из уже занятых нашими хостами. У меня это 192.168.26.132.

2. Создаем ресурс (можно на 1 узле):

```
pcs resource create virtual_ip ocf:heartbeat:IPAddr2 ip= выбраный_ip cidr_netmask=32
op monitor interval=30s
```

3. Проверка:

```
pcs status resources
```

```
root@proxy-1:~# pcs status resources
Resource Group: HAProxyGroup
  virtual_ip (ocf::heartbeat:IPAddr2):           Started proxy-2
```

(у вас может запуститься на proxy-1)

На узле, на который привязался IP, проверяем вывод команды `ip a`:

```
proxy-2@proxy-2:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
        inet 127.0.0.1/8 scope host lo
            valid_lft forever preferred_lft forever
        inet6 ::1/128 scope host
            valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 00:0c:29:3a:cb:ff brd ff:ff:ff:ff:ff:ff
        inet 192.168.26.129/24 brd 192.168.26.255 scope global dynamic noprefixroute ens33
            valid_lft 1428sec preferred_lft 1428sec
        inet 192.168.26.132/32 brd 192.168.26.255 scope global ens33
            valid_lft forever preferred_lft forever
        inet6 fe80::20c:29ff:fe3a:cacb/64 scope link noprefixroute
            valid_lft forever preferred_lft forever
```

У интерфейса ens33 появился еще один IP-адрес.

## Настройка haproxy

**Команды выполнять на всех балансировщиках.**

1. `nano /etc/haproxy/haproxy.cfg`

В конец файла дописываем следующие строчки:

```
frontend front
    bind 192.168.26.132:80
    default_backend back

backend back
    balance roundrobin
    server webserver-1 192.168.26.130:81
    server webserver-2 192.168.26.131:81
```

front и back - имена, можно придумать свои.

В поле bind указываем **плавающий IP-адрес**.

В полях server пишем имена серверов и IP соответствующих узлов.

2. Установим ресурс кластера haproxy (т.к. по умолчанию он не идет с pacemaker):

```
cd /usr/lib/ocf/resource.d/heartbeat
curl -O https://raw.githubusercontent.com/thisismitch/cluster-agents/master/haproxy
chmod +x haproxy
```

3. Добавляем новый ресурс:

```
pcs resource create haproxy ocf:heartbeat:haproxy binpath=/usr/sbin/haproxy
conffile=/etc/haproxy/haproxy.cfg op monitor interval=10s
```

Объединим haproxy и virtual\_ip в одну группу, чтобы они работали одновременно только на одном узле:

```
pcs resource group add HAProxyGroup virtual_ip haproxy
```

Чтобы сначала запускался virtual\_ip, затем haproxy:

```
pcs constraint order virtual_ip then haproxy
```

#### 4. Проверяем:

```
pcs status
```

```
root@proxy-1:~# pcs status
Cluster name: cluster
Stack: corosync
Current DC: proxy-2 (version 2.0.1-9e909a5bdd) – partition with quorum
Last updated: Fri May 14 06:36:51 2021
Last change: Fri May 14 04:44:24 2021 by root via cibadmin on proxy-1

2 nodes configured
2 resources configured

Online: [ proxy-1 proxy-2 ]

Full list of resources:

Resource Group: HAProxyGroup
    virtual_ip (ocf::heartbeat:IPAddr2):           Started proxy-2
    haproxy   (ocf::heartbeat:haproxy):           Started proxy-2

Daemon Status:
    corosync: active/enabled
    pacemaker: active/enabled
    pcsd: active/enabled
root@proxy-1:~#
```

## Промежуточный итог

Теперь можно попробовать послать HTTP-запрос на плавающий IP:

```
curl 192.168.26.132
```

Однако сейчас мы увидим только сообщение об ошибке. Но на самом деле эта ошибка - с backend-сервера. haproxy перенаправил наш запрос на один из таких серверов на порт 81, но там у нас пока ничего не запущено.

Займёмся этим.

## Настройка веб-серверов

Мы будем создавать два сайта - fit.nsu.ru и fit2.nsu.ru. Они могут как содержать одинаковый контент, т.е. по сути быть зеркалами друг друга, так и быть абсолютно не связанными.

Для наглядности пусть запросы на fit.nsu.ru на первый backend-сервер возвращают одну строку:

*this is first webserver for fit.nsu.ru*

Запросы на fit.nsu.ru на второй backend-сервер возвращают

*this is second webserver for fit.nsu.ru*

И то же самое для fit2.nsu.ru:

*this is first webserver for fit2.nsu.ru*

*this is second webserver for fit2.nsu.ru*

## Настройка Apache

**Настройку проделать на обоих веб-серверах.**

- Поменяем порт для Apache на 81:

```
sudo nano /etc/apache2/ports.conf
```

В начале изменить строчку (или добавить, если ее нет):

*Listen 81*

- Создадим папки /var/www/fit.nsu.ru, /var/www/fit2.nsu.ru:

```
sudo mkdir /var/www/fit.nsu.ru /var/www/fit2.nsu.ru
```

- В каждой из этих папок поместим файл index.html с простым содержанием:

*this is first/second webserver for fit.nsu.ru/fit2.nsu.ru*

- Создаём конфигурационные файлы для сайтов:

```
sudo nano /etc/apache2/sites-available/fit.nsu.ru.conf
```

```
sudo nano /etc/apache2/sites-available/fit2.nsu.ru.conf
```

Они должны содержать следующее:

```
<VirtualHost 192.168.26.131:81>
```

```
    ServerName fit2.nsu.ru
```

```
DocumentRoot /var/www/fit2.nsu.ru
</VirtualHost>
```

Вместо 192.168.26.131 пишем IP-адрес backend-сервера, на котором сейчас редактируем данный файл.

ServerName и DocumentRoot меняем в `fit.nsu.ru.conf` и `fit2.nsu.ru.conf` на соответствующие.

#### 5. Добавляем сайты в Apache:

```
sudo a2ensite fit.nsu.ru.conf
sudo a2ensite fit2.nsu.ru.conf
```

#### 6. Перезапускаем Apache:

```
sudo service apache2 restart
```

#### 7. Включаем автозагрузку для Apache:

```
sudo service apache2 enable
```

## Почти готово

Теперь если попробовать сделать HTTP-запрос на плавающий IP, то получим какую-то из страничек. Если сделать еще один, то получим страничку от второго backend-сервера. И так по кругу. Однако приходят страницы с `fit.nsu.ru`. Чтобы была возможность выбирать, на какой хост мы хотим сделать запрос, настроим собственный локальный DNS-сервер.

## Настройка DNS-сервера

#### 1. На 5-й машине устанавливаем bind9:

```
sudo apt install bind9
```

#### 2. В конец файла `/etc/bind/named.conf.local` дописываем следующие строки:

```
zone "nsu.ru" {
    type master;
    file "/etc/bind/db.nsu.ru";
};
```

#### 3. `sudo touch /etc/bind/db.nsu.ru`

В этом файле должно быть следующее содержание:

```
root@test:~# cat /etc/bind/db.nsu.ru
$TTL    604800
@       IN      SOA     ns.nsu.ru. lak.localhost. (
                          2021051401      ; Serial
                          604800          ; Refresh
                          86400           ; Retry
                          2419200         ; Expire
                          604800 )        ; Negative Cache TTL
;
@       IN      NS      ns.nsu.ru.
ns      IN      A       192.168.26.137
www    IN      A       192.168.26.132
nsu.ru. IN      A       192.168.26.132
fit.nsu.ru. IN      A       192.168.26.132
fit2.nsu.ru. IN      A       192.168.26.132
```

Вместо 192.168.26.132 указываем плавающий IP. Вместо 192.168.26.137 указываем IP машины, на которой сейчас настраиваем DNS.

4. `sudo service bind9 restart`
5. `sudo service bind9 enable`
6. На остальных хостах редактируем файл /etc/resolv.conf.

Вместо указанного там IP после nameserver пишем IP машины с DNS.  
Теперь можно пробовать `ping fit.nsu.ru` и `ping fit2.nsu.ru`.