

Инструкция по созданию кластера

Оглавление

1. Введение.....	2
2. Установка Debian.....	4
3. Настройка кластера.....	6
4. Приложения.....	12

1. Введение

Инструменты:

1. Нароху – балансировщик нагрузки, отвечающий за распределение запросов между серверами
2. Pacemaker (Cluster Resource Manager) – управляет ресурсами (“Ресурс с точки зрения pacemaker - это все, что можно заскриптовать. Т.е. любой сервис или программа, которая может управляться через скрипты, начиная с простых сервисов типа apache, ip-адрес, и т.д.”). Подробнее можно прочитать [тут](#). Также Pacemaker чаще всего работает в связке с Corosync, который отвечает за “сетевое взаимодействие узлов, т.е. передачу сервисных команд (запуск/остановка ресурсов, узлов и т.д.), обмен информацией о полноте состава кластера(quorum)и т.д.”
3. Corosync – система группового общения для отказоустойчивого кластера, использующая протокол Totem.
4. Apache – веб-сервер, используемый для демонстрации распределения нагрузки
5. BIND9 – dns сервер

Примерная схема кластера:

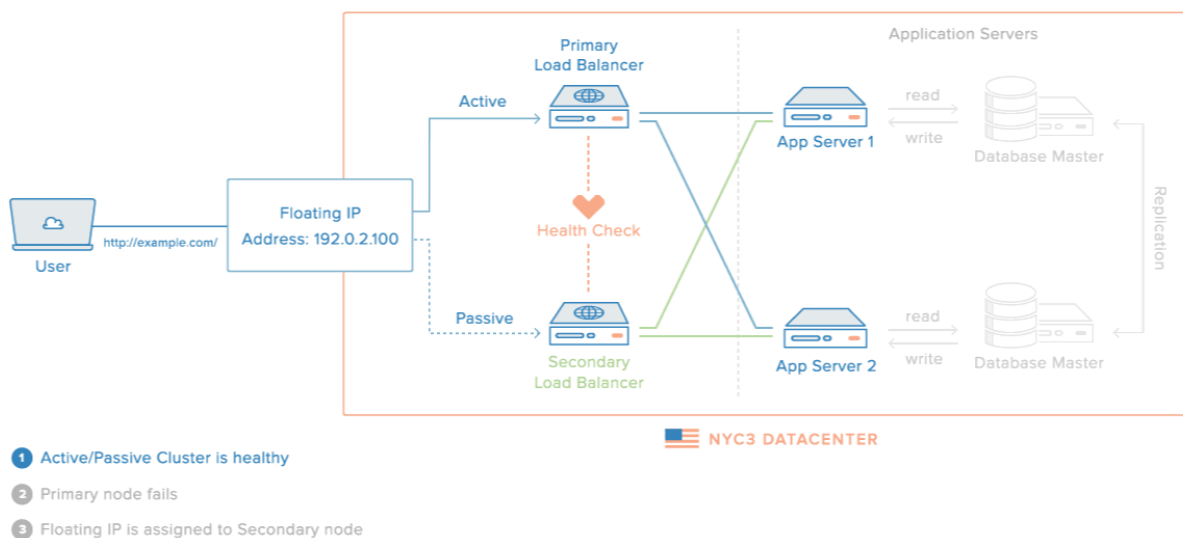


Рисунок 1. https://assets.digitalocean.com/articles/high_availability/ha-diagram-animated.gif

На рисунке изображено 2 сервера, на каждом из которых в роли load balancer'а выступает нароху, а в роли app server'а – apache.

В качестве гипервизора используем – Virtual box, а в качестве гостевой ОС – Debian.

Всего у нас будет 3 сервера:

1. Node1 (Pacemaker, Corosync, Haproxy, Apache)
2. Node2 (Pacemaker, Corosync, Haproxy, Apache)
3. DNS (BIND9)

2. Установка Debian

- 1) Установка Virtual Box - <https://www.virtualbox.org/wiki/Downloads>
- 2) Скачивание образа debian - <https://www.debian.org/CD/netinst/>
- 3) Создание виртуальной машины на основе образа

- a. Нужно выделить 1 Гб оперативной памяти, а также 3 Гб HDD для того, чтобы можно было использовать удобную псевдографическую установку, иначе система будет падать при выборе этого способа установки.
- b. В настройках сети виртуальной машины нужно выбрать тип подключения – сетевой мост, который позволяет виртуальной машине выступать в роли реального сетевого устройства с отдельным ip адресом (то есть это позволит вам заходить на ваш сайт с различных устройств, подключенных к одной и той же локальной сети).

c. Установка OS

- i. Язык – русский
- ii. Страна – РФ
- iii. Раскладка клавиатуры – русская
- iv. Способ переключения – Alt+shift
- v. Имя компьютера – node1
- vi. Имя домена – пустое
- vii. Пароль суперпользователя – 1234
- viii. Имя нового пользователя – user
- ix. Пароль для нового пользователя – 4321
- x. Часовой пояс – Красноярск (+4)
- xi. Метод разметки – авто – использовать весь диск
- xii. Диск для разметки – он единственный
- xiii. Схема разметки – все файлы в одном разделе
- xiv. Разметка дисков – закончить разметку и записать изменения на диск
- xv. Записать изменения на диск – да
- xvi. Просканировать другой CD – нет
- xvii. Страна, в которой расположено зеркало архива – РФ
- xviii. Зеркало архива – deb.debian.org
- xix. Информация о HTTP-прокси – пустое
- xx. Участвовать в опросе – нет
- xxi. Устанавливаемое ПО – ssh-сервер, стандартные системные утилиты.
- xxii. Установить системный загрузчик GRUB в главную загрузочную запись – да

- xxiii. Устройство для установки системного загрузчика -
/dev/sda/
- d. Клонировем виртуальную машину для node2:
 - i. Тип клонирования – полное клонирование
 - ii. Цель клонирования – все
 - iii. Политика mac-адреса – сгенерировать новые mac-адреса всех сетевых адаптеров
- e. Повторяем пункт d для DNS

3. Настройка кластера

NODE1:

1. Заходим в нашу виртуальную машину, используя user с паролем 4321
2. Переходим в режим администратора с помощью команды – **su** – с паролем 1234
3. Обновим пакеты на нашей машине:
 - a. **apt-get update**
 - b. **apt-get upgrade**
4. Устанавливаем aptitude – **apt-get install -y aptitude**
5. Устанавливаем Corosync, Pacemaker, Haproxy при помощи команды – **aptitude install corosync pacemaker haproxy**
6. Настраиваем Corosync:
 - a. Узнаем ip-адрес при помощи одной из следующих команд (обычно он будет иметь вид 192.168.*.* с названием адаптера: en*):
 - **ifconfig**
 - **ip a**
 - b. Желательно сделать ip-адрес статическим. Это можно сделать при помощи настроек роутера, либо изменив **/etc/network/interfaces**
 - c. Открываем файл – **nano /etc/corosync/corosync.conf**:
 - В директиве totem изменяем (либо если его нет, то создаем) interface на следующий: **interface {**
ringnumber: 0
bindnetaddr: 192.168.0.0
broadcast: yes
}
Bindnetaddr – адрес сети нашего кластера (зависит от ip-адреса нашей виртуальной машины)
 - В директиве nodelist добавляем следующее:
node {

name: node1
nodeid: 1
ring0_addr: 192.168.0.104

}
node {
name: node2
nodeid: 2
ring0_addr: 192.168.0.105

}
ring0_addr – это ip-адреса 1 и 2 узла (то есть NODE1 и NODE2, которая выдала команда из пункта [6.a](#))

name – это имена 1 и 2 узла (то есть NODE1 и NODE2)

- d. Сохраняем настройки (**Ctrl+S**) и выходим (**Ctrl+X**)
 - e. Переходим к [NODE2](#)
 - f. Генерируем ключ – **corosync-keygen**
 - g. С помощью команды **scp** отправляем сгенерированный ключ на NODE2 – **scp /etc/corosync/authkey user@192.168.0.105:.**
 - h. Переходим к **NODE2**([п. 4](#))
 - i. Добавляем в corosync автозапуск - **nano /etc/default/corosync**
И пишем **START=yes**
 - j. Перезапускаем виртуальную машину – **/sbin/reboot**
 - k. Прodelываем пункты i и j на другой машине
 - l. Можно проверить настройки corosync с помощью следующей команды – **/sbin/corosync-cmapctl | grep members**
7. Настраиваем Nаppoху:
- a. Открываем файл с конфигурациями - **nano /etc/haproxy/haproxy.cfg**

- Дописываем в конец следующее:

```
frontend front
    bind 192.168.0.11:80
    default_backend back
backend back
    balance roundrobin
    server node1 192.168.0.104:81 check
    server node2 192.168.0.105:81 check
```

Раздел frontend определяет, каким образом перенаправлять запросы к бэкенду в зависимости от того, что за запрос поступил от клиента. (TCP, HTTP)

Секция backend содержит список серверов и отвечает за балансировку нагрузки между ними в зависимости от выбранного алгоритма.

80 – порт для haproxy, 81 – для apache2.

front – название frontend'а

bind 192.168.0.11:80 – ip адрес и порт виртуального интерфейса ([На рисунке](#) помечен как Floating IP)

default_backend back – название бэкенда, на который будут отправляться входящие запросы с frontend'а

back – название backend'а

balance roundrobin – выбираем алгоритм балансировки, а про другие алгоритмы балансировки(consistent-hashing) можно прочитать [тут](#) и [тут](#).

server node1 192.168.0.104:81 check – server — указывает имя и IP-адрес сервера, на который передается запрос;

check — указываем, что необходимо проверять состояние сервера, подробнее можно прочитать [здесь](#).

Также можно прочитать про sticky session [тут](#).

- b. Можно проверить настройки haproxy с помощью следующей команды – **haproxy -f /etc/haproxy/haproxy.cfg -c**
 - c. Передаем настройки на NODE2 – **scp /etc/haproxy/haproxy.cfg 192.168.0.105:/etc/haproxy/haproxy.cfg**
 - d. Добавляем в файл (на обоих узлах) **/etc/sysctl.conf** строку **net.ipv4.ip_nonlocal_bind=1** – разрешаем привязку нелокального ip-адреса (то есть несуществующему IP)
 - e. Полезные команды: **/sbin/service haproxy start/stop/reload/status**
8. Настраиваем Pacemaker ([хорошее описание теории команд Pacemaker'a](#)):
- a. Устанавливаем crmsh (Инструмент для упрощенной работы с XML файлами настройки Pacemaker'a, также распространяет настройки сразу на все узлы, чем облегчает жизнь) – **aptitude install crmsh**
 - b. Переходим в режим конфигурации – **crm configure**
 - c. Вводим следующие команды:
 - **property no-quorum-policy=ignore** – отключает необходимость набора 50% рабочих узлов кластер для жизнеспособности кластера.
 - **property stonith-enabled=false** – отключает Shoot-The-Other-Node-In-The-Head(подробнее по ссылке выше)
 - **primitive VIP ocf:heartbeat:IPaddr2 params ip=192.168.0.11 cidr_netmask=24 op monitor interval=1s** – создаём виртуальный ip-адрес(VIP, то есть создаем ресурс); с помощью **op monitor interval=1s** кластер будет следить за состоянием ресурса каждую секунду.
 - **primitive HAP lsb:haproxy op monitor interval=1s** – создаём ресурс haproxy(HAP);
 - **colocation CLC inf: VIP HAP** - это принуждение к размещению ресурсов на одном узле или наоборот на разных узлах
 - **order ORD inf: VIP HAP** – определяем порядок запуска ресурсов (сначала VIP, а потом HAP, зависящий от VIP)
 - Теперь сохраняем настройки – **commit**
 - Перезапускаем виртуальную машину – **/sbin/reboot**
 - d. Полезные команды в режиме конфигурации: **exit, status, show, delete**
 - e. Полезные утилиты:
 - **crm status** – состояние кластера
 - **crm configure show** – [конфигурация Pacemaker'a](#)

9. Настраиваем Apache:

a. Устанавливаем apache2 – **aptitude install apache2**

b. Открываем конфигурационный файл – **nano /etc/apache2/ports.conf**

И меняем

NameVirtualHost *:81

Listen 81

c. Создаем конфигурационный файл для сайта – **nano /etc/apache2/sites-available/mysite.com.conf**

И пишем следующее:

<VirtualHost 192.168.0.104:81>

ServerName fit.nsu

DocumentRoot /var/www/fit.nsu

</VirtualHost>

<VirtualHost 192.168.0.104:81>

ServerName fit2.nsu

DocumentRoot /var/www/fit2.nsu

</VirtualHost>

Создаем 2 виртуальных хоста (сайта) на нашей виртуальной машине **NODE1**.

ServerName fit.nsu – доменное имя хоста

DocumentRoot /var/www/fit.nsu – директория, в которой находится наш сайт

d. Создаем папки для сайтов –

- **mkdir /var/www/fit.nsu**
- **mkdir /var/www/fit2.nsu**

e. Создаем стартовые страницы для сайтов –

- **nano /var/www/fit.nsu/index.html**
 - И пишем следующее:
<html>
<header><title>This is title </title></header>
<body>
Hello,I'm node1!(fit)
</body>
</html>
- **nano /var/www/fit2.nsu/index.html**
 - И пишем следующее:
<html>
<header><title>This is title </title></header>
<body>
Hello,I'm node1!(fit2)
</body>
</html>

- f. Запускаем сайт – **a2ensite mysite.com.conf**
- g. Запускаем apache2 – **service apache2 start**
 Повторяем пункты [9.a](#) – [9.g](#) для **NODE2**,
 заменив ip-адрес при создании виртуальных
 хостов (п. [9.c](#)) на ip-адрес **NODE2** и изменив
 текст приветствия в стартовых страницах на “
Hello,I'm node2!(fit)” и “ **Hello,I'm
 node2!(fit2)”**
- h. Можно проверить работоспособность кластера, перейдя в
 браузере по ip-адресу **192.168.0.11**, либо через терминал,
 используя команду – **curl 192.168.0.11**
- i. Переходим к [DNS](#)

NODE2:

1. Выполняем пункты 1 - 5.d из **NODE1**
2. Изменяем hostname на node2: **nano etc/hostname**
3. Оставляем запущенной машину и переходим к [NODE1\(п. 5.e\)](#)
4. Копируем authkey файл в нужную папку – **cp authkey /etc/corosync/**
5. Возвращаемся к [п. 6.i](#).

DNS:

1. Устанавливаем BIND9 – **aptitude install bind9**
2. Создаем новую зону – **nano /etc/bind/db.nsu**

а. И пишем следующее:

```
$TTL 604800
@      IN      SOA ns.nsu. root.nsu. (
                                6      ; Serial
                                10800 ; Refresh
                                3600  ; Retry
                                3600000 ; Expire
                                604800) ; Negative cache TTL
;
@      IN      NS       ns.nsu.
ns     IN      A        192.168.0.102
fit    IN      A        192.168.0.11
fit2   IN      A        192.168.0.11
```

192.168.0.102 – это ip-адрес виртуальной машины **DNS**

Про параметры выше можно прочитать [тут](#)

3. Настраиваем файл с зонами – **nano /etc/bind/named.conf.default-zones**
 - а. Добавляем туда:


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

};

4. Проверяем корректность файла зон – **named-checkconf -z**
5. Обновляем информацию о зонах – **rndc reload**
6. Настраиваем пути до DNS сервера – **nano /etc/network/interfaces**
 - а. Дописываем: **dns-nameservers 192.168.0.102**
7. Работоспособность нашего DNS сервера можно проверить с помощью команды – **nslookup <dns имя> <ip-адрес DNS сервера>**

4. Приложения

(Некоторые параметры в Corosync закомментированы – это необязательно)

```
corosync.conf.txt
# Please read the corosync.conf.5 manual page
totem {
    version: 2

    # Corosync itself works without a cluster name, but DLM needs one.
    # The cluster name is also written into the VG metadata of newly
    # created shared LVM volume groups, if lvmlockd uses DLM locking.
    cluster_name: debian
    transport: udpu
    interface {
        ringnumber: 0
        bindnetaddr: 192.168.0.0
        broadcast: yes
    }
    # crypto_cipher and crypto_hash: Used for mutual node authentication.
    # If you choose to enable this, then do remember to create a shared
    # secret with "corosync-keygen".
    # enabling crypto_cipher, requires also enabling of crypto_hash.
    # crypto works only with knot transport
    crypto_cipher: none
    crypto_hash: none
}

logging {
    # Log the source file and line where messages are being
    # generated. When in doubt, leave off. Potentially useful for
    # debugging.
    fileline: off
    # Log to standard error. When in doubt, set to yes. Useful when
    # running in the foreground (when invoking "corosync -f")
    to_stderr: yes
    # Log to a log file. When set to "no", the "logfile" option
    # must not be set.
    to_logfile: yes
    logfile: /var/log/corosync/corosync.log
    # Log to the system log daemon. When in doubt, set to yes.
    to_syslog: yes
    # Log debug messages (very verbose). When in doubt, leave off.
    debug: off
    # Log messages with time stamps. When in doubt, set to hires (or on)
    #timestamp: hires
    logger_subsys {
        subsys: QUORUM
        debug: off
    }
}

quorum {
    # Enable and configure quorum subsystem (default: off)
    # see also corosync.conf.5 and votequorum.5
    provider: cprosync_votequorum
    #
    two_node: 1
}
```

```
corosync.conf.txt

logging {
    # Log the source file and line where messages are being
    # generated. When in doubt, leave off. Potentially useful for
    # debugging.
    fileline: off
    # Log to standard error. When in doubt, set to yes. Useful when
    # running in the foreground (when invoking "corosync -f")
    to_stderr: yes
    # Log to a log file. When set to "no", the "logfile" option
    # must not be set.
    to_logfile: yes
    logfile: /var/log/corosync/corosync.log
    # Log to the system log daemon. When in doubt, set to yes.
    to_syslog: yes
    # Log debug messages (very verbose). When in doubt, leave off.
    debug: off
    # Log messages with time stamps. When in doubt, set to hires (or on)
    #timestamp: hires
    logger_subsys {
        subsys: QUORUM
        debug: off
    }
}

quorum {
    # Enable and configure quorum subsystem (default: off)
    # see also corosync.conf.5 and votequorum.5
    provider: cprosync_votequorum
    #
    two_node: 1
}

nodelist {
    # Change/uncomment/add node sections to match cluster configuration

    node {
        # Hostname of the node
        name: node1
        # Cluster membership node identifier
        nodeid: 1
        # Address of first link
        ring0_addr: 192.168.0.104
        # When knet transport is used it's possible to define up to 8 links
        #ring1_addr: 192.168.1.1
    }
    node {
        name: node2
        nodeid: 2
        ring0_addr: 192.168.0.105
    }
    # ...
}
```

```
haproxy.cfg.txt
global
    log /dev/log      local0
    log /dev/log      local1 notice
    chroot /var/lib/haproxy
    stats socket /run/haproxy/admin.sock mode 660 level admin expose-fd listeners
    stats timeout 30s
    user haproxy
    group haproxy
    daemon

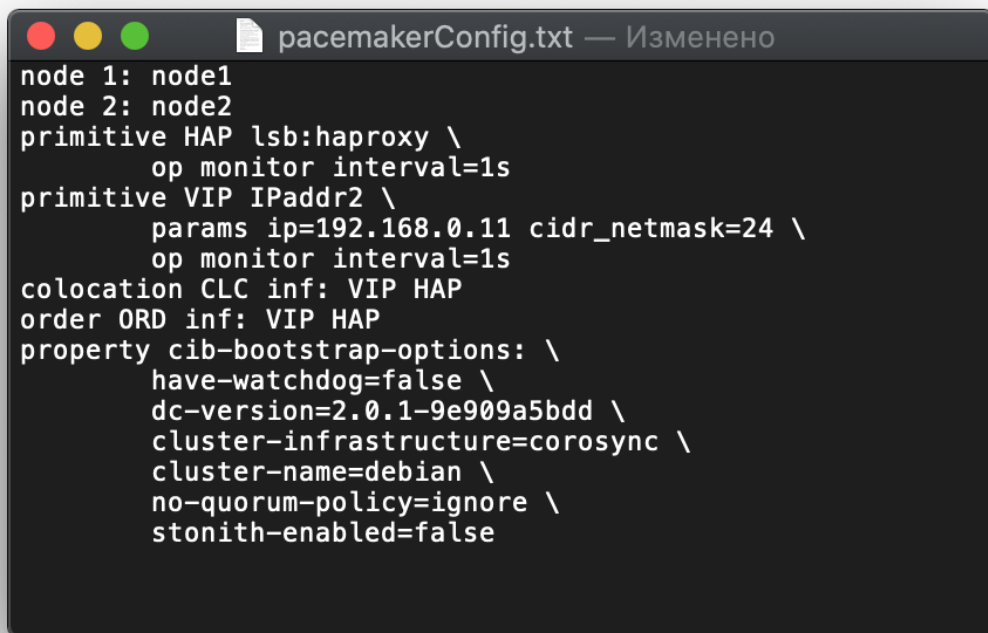
    # Default SSL material locations
    ca-base /etc/ssl/certs
    crt-base /etc/ssl/private

    # Default ciphers to use on SSL-enabled listening sockets.
    # For more information, see ciphers(1SSL). This list is from:
    # https://hynek.me/articles/hardening-your-web-servers-ssl-ciphers/
    # An alternative list with additional directives can be obtained from
    # https://mozilla.github.io/server-side-tls/ssl-config-generator/?server=haproxy
    ssl-default-bind-ciphers
ECDH+AESGCM:DH+AESGCM:ECDH+AES256:DH+AES256:ECDH+AES128:DH+AES:RSA+AESGCM:RSA+AES:!aNULL:!
MD5:!DSS
    ssl-default-bind-options no-sslsv3

defaults
    log          global
    mode          http
    option        httplog
    option        dontlognull
    timeout connect 5000
    timeout client  50000
    timeout server  50000
    errorfile 400 /etc/haproxy/errors/400.http
    errorfile 403 /etc/haproxy/errors/403.http
    errorfile 408 /etc/haproxy/errors/408.http
    errorfile 500 /etc/haproxy/errors/500.http
    errorfile 502 /etc/haproxy/errors/502.http
    errorfile 503 /etc/haproxy/errors/503.http
    errorfile 504 /etc/haproxy/errors/504.http

frontend front
    bind 192.168.0.11:80
    default_backend back

backend back
    balance roundrobin
    server node1 192.168.0.104:81 check
    server node2 192.168.0.105:81 check
```

A terminal window with a dark background and a title bar. The title bar contains three colored circles (red, yellow, green) on the left and the text 'pacemakerConfig.txt — Изменено' on the right. The terminal displays a series of configuration commands for Pacemaker, including node definitions, primitive configurations for HAP and VIP, colocation and order constraints, and a set of properties for the cluster bootstrap options.

```
node 1: node1
node 2: node2
primitive HAP lsb:haproxy \
    op monitor interval=1s
primitive VIP IPAddr2 \
    params ip=192.168.0.11 cidr_netmask=24 \
    op monitor interval=1s
colocation CLC inf: VIP HAP
order ORD inf: VIP HAP
property cib-bootstrap-options: \
    have-watchdog=false \
    dc-version=2.0.1-9e909a5bdd \
    cluster-infrastructure=corosync \
    cluster-name=debian \
    no-quorum-policy=ignore \
    stonith-enabled=false
```

```
mysite.com.conf.txt
<VirtualHost 192.168.0.104:81>
    ServerName fit.nsu
    DocumentRoot /var/www/fit.nsu
</VirtualHost>

<VirtualHost 192.168.0.104:81>
    ServerName fit2.nsu
    DocumentRoot /var/www/fit2.nsu
</VirtualHost>
```

```
db.nsu.txt
$TTL 604800
@      IN      SOA  ns.nsu. root.nsu. (
        6      ; Serial
        10800   ; Refresh
        3600    ; Retry
        3600000 ; Expire
        604800) ; Negative cache TTL
;
@      IN      NS   ns.nsu.
ns     IN      A    192.168.0.102
fit    IN      A    192.168.0.11
fit2   IN      A    192.168.0.11
```



```
named.conf.default-zones.txt
// prime the server with knowledge of the root servers
zone "." {
    type hint;
    file "/usr/share/dns/root.hints";
};

// be authoritative for the localhost forward and reverse zones, and for
// broadcast zones as per RFC 1912

zone "localhost" {
    type master;
    file "/etc/bind/db.local";
};

zone "127.in-addr.arpa" {
    type master;
    file "/etc/bind/db.127";
};

zone "0.in-addr.arpa" {
    type master;
    file "/etc/bind/db.0";
};

zone "255.in-addr.arpa" {
    type master;
    file "/etc/bind/db.255";
};

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