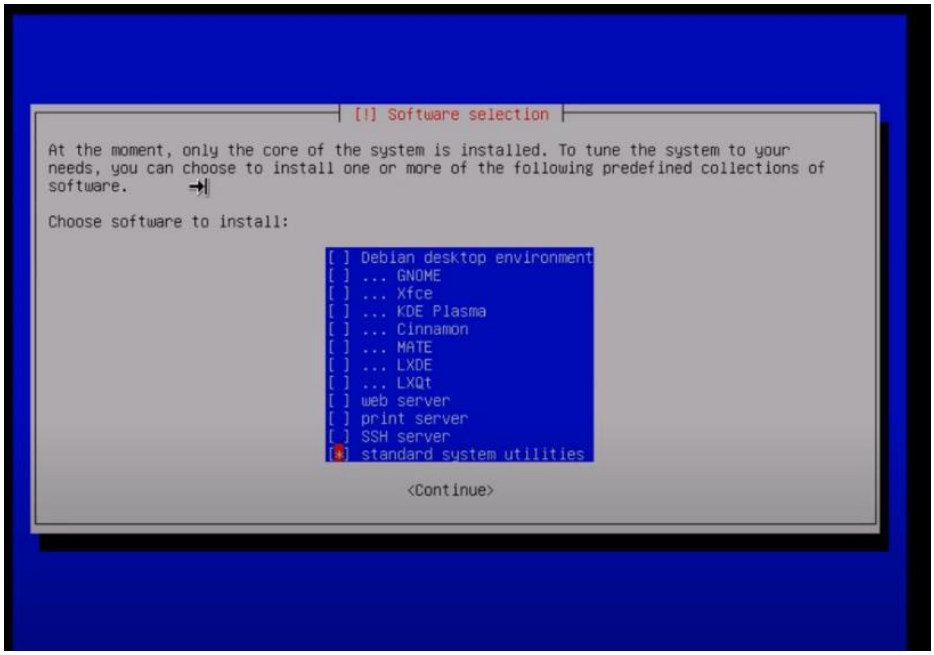
Создание HA-кластера с балансировкой нагрузки

Данный документ является переработанной памяткой HA_help.pdf (источник https://vk.com/topic-64649419_36089444) для выполнения лабораторной работы по распределительным системам

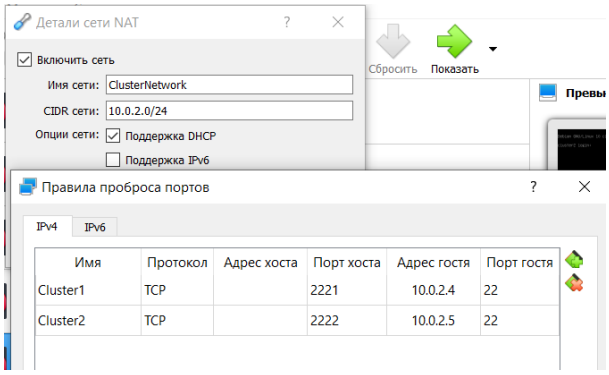
Предисловие: я потратил 3 дня на лабу. В том числе, чтобы прогуглить все узкие моменты, которые были в этом помощнике ha_help.pdf. Также я находил ошибки в командах в первоначальном файле.
Переделан весь файл. Удачи.

1) Подготовка. Первичная настройка оборудования и сети интернет.

   a) Установить VirtualBox, используя установщик с сайта VirtualBox.
   b) Поставить виртуальную машину без графической оболочки в VirtualBox https://www.debian.org/CD/netinst/.
      Я скачивал именно по этой ссылке
      https://cdimage.debian.org/debian-cd/current/amd64/bt-cd/debian-10.9.0-amd64-netinst.iso.torrent
      (2021 год)
      Для виртуальной машины достаточно 1Гб RAM 3Гб HDD.
      https://www.youtube.com/watch?v=EkX3Q4AArPw — видео (смотреть до конца, в конце чел отключает графический интерфейс в таком же окна, как на скрине чуть ниже в доке).
      При установке выбрать web server, ssh server, standard system utilities; язык - английский; страна - Россия, зеркало - mirror.yandex.ru (нет такого зеркала в настройках при установке, ну или я не понял)
   c) Чтобы установить версию без графического интерфейса (один терминал) стоит при выборе параметров отключить всяческие GNOME и т.п.
      Нужны только web server, ssh server, standard system utilities



   d) Клонировать машину 2 раза (с генерацией новых MAC). (эта услуга функция есть в VBox по умолчанию, можно заменить все MAC-address)
   e) Настройки сети в VBox: https://litl--admin-ru.turbopages.org/litl-admin.ru/s/virtualizaciya/kak-nastroit-set-s-virtualnymi-mashinami-virtualbox.html
      Делаем свою сеть NAT в настройках VirtualBox (не в настройках машины в VirtualBox, а в параметрах, скрин экрана ниже).
   f) Опционально можно подключиться к машине по ssh, чтобы удобно вводить вставкой ctrl+shift+v в терминале на хост машине (на самом ПК).
      Для этого нужно настроить проброс портов в настройках вашей сети NAT.



Надо узнать ip двух машин через **ifconfig**, сначала её нужно установить **apt-get install net-tools**.

Так же можно использовать **ip addr** для определения ip машины (установки не требует)

```
root@cluster1:~# ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.4  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::a00:27ff:fecd:ec5e  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:cd:ec:5e  txqueuelen 1000  (Ethernet)
        RX packets 442  bytes 286223 (279.5 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 385  bytes 38389 (37.4 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

g) Мои ip: 10.0.2.4 cluster1; 10.0.2.5 cluster2.
https://comp-security.net/подключиться-к-виртуальной-машине-по-ssh/

h) Скачиваем Windows Terminal (прикольное оформление) Можно просто открыть PowerShell😊.

i) Настраиваем машины на доступ руту по ssh.
https://www.dmosk.ru/miniinstruktions.php?mini=ubuntu-ssh-root
По умолчанию недоступная опция. Лично я далее делал всё из-под root-пользователя, иначе вводить sudo муторно (sudo, кстати, нет по умолчанию, нужно ставить, настраивая права пользователей в config-файлах) Тут я затестил на новой машине создание файла в etc (без прав суперпользователя это невозможно сделать, sudo отсутствует, как мы видим)

```
user@cluster2:/etc$ touch test.txt
touch: cannot touch 'test.txt': Permission denied
user@cluster2:/etc$ sudo touch test.txt
-bash: sudo: command not found
user@cluster2:/etc$ _
```

Далее все мои команды предназначены для root-пользователя.

j) По ssh подключаемся к машинам из windows terminal (он же power shell).
shh root@localhost -p 2221(-p 2222) — это порты локального хоста из таблицы проброса портов.
Иначе набирать самому команды. Двунаправленный буфер обмена не очень хочет работать.

k) Да! Вероятно, вы подумали, зачем мне твой ssh, Кирилл. Да всё просто, терминал без графического интерфейса – это боль была лично для меня. Процедура вводы = много времени, редактирование в mcedit – неудобно совсем.
Ставьте terminal себе на windows (можно даже дурацкий PUTTY, он тоже неудобный, кстати).
И вообще сохраните себе нервы😊

l) После манипуляций с сетью необходимо сделать **ping ip_address** друг друга на машинах, чтобы проверить, что они могут кинуть icmp-пакет друг другу. Ещё нужно проверить выход в Интернет на машинах, иначе репозитории не будут видимы для загрузки новых пакетов.
Используем **ping google.com**

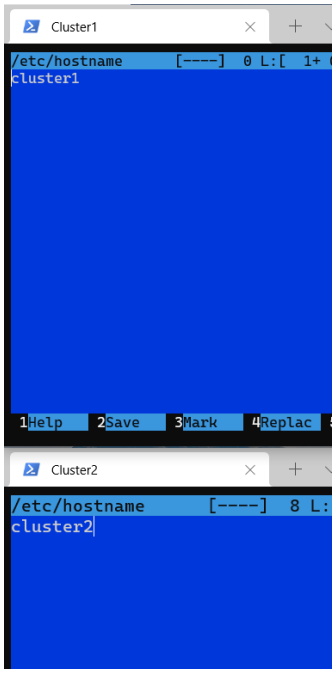
2) Настройка кластера

a) Для удобства можно установить **mc** – удобный файловый менеджер
apt-get install mc

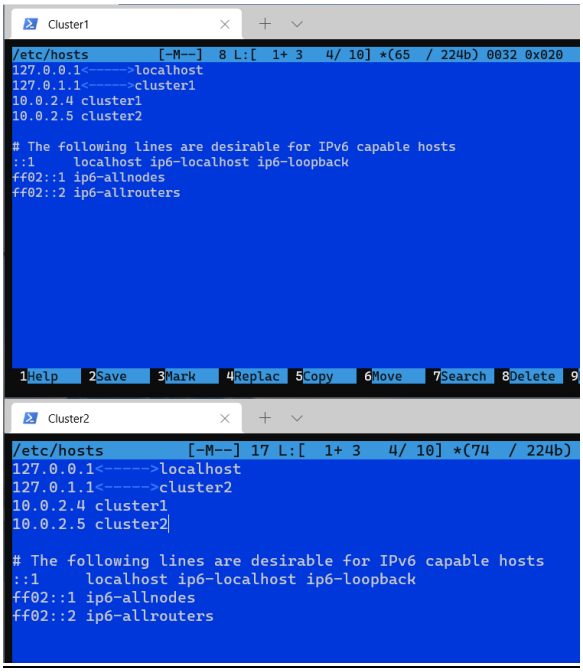
b) Если вторую виртуальную машину получили копированием первой, либо при установке не меняли имя хоста, или просто хотите изменить имя машин, то открываем файлы и меняем имена. В hosts нужно прописать две новые строчки.

Далее вместо nano будет mcedit, он поприятнее будет, не зря же mc ставили 😊

mcedit /etc/hostname



mcedit /etc/hosts



c) На обе машины устанавливаем corosync и pacemaker.

apt-get install corosync pacemaker

На обе машины устанавливаем haproxy

if (вы установили Debian Wheezy) {

Далее инструкция для другой версии Debian:

Если устанавливался Debian Wheezy { для установки haproxy надо добавить репозиторий:

echo deb http://httpredir.debian.org/debian wheezy-backports-sloppy main |
sed 's/\(.\)-sloppy \(.*\)/&@\1 \2/' | tr @ '\n' | tee /etc/apt/sources.list.d/backports.list*

обновить:

apt-get update

установить:

apt-get install haproxy -t wheezy-backports-sloppy

}

else{

Просто устанавливаем haproxy

apt-get install haproxy

}

d) Конфигурируем **corosync** (выполняет функции коммуникатора-транспортера в нашем виртуальном кластере, т.е. обеспечивает возможность коммуникации между нодами кластера):

i) Узнаем ip адрес машины (В настройках VMware можно менять ip. А мы вообще сделали свою сеть NAT в VirtualBox):

ifconfig

ii) Открываем конфигурационный файл corosync.conf (на каком-то одном ноде, например на cluster1)

mcedit /etc/corosync/corosync.conf

Ищем строку **bindnetaddr**: и меняем на адрес сети нашего кластера (ip данной машины, только последняя цифра 0) (у меня не было **bindnetaddr, я дописал сам bindnetaddr: ip_addres**)

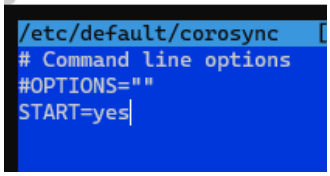
Ещё нужно добавить node в nodelist. (Обязательно сменить name и id)

```
# 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

        # 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 knet transport
        crypto_cipher: none
        crypto_hash: none
        bindnetaddr: 10.0.2.0
}

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: corosync_votequorum
}

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

        node {
                name: cluster1
                nodeid: 1
                ring0_addr: 10.0.2.4
        }
        node {
                name: cluster2
                nodeid: 2
                ring0_addr: 10.0.2.5
        }
}
```

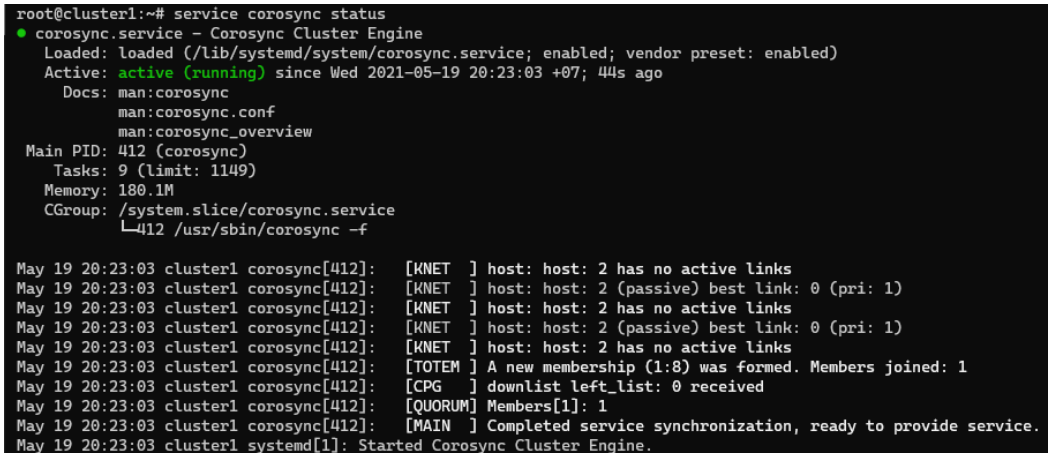
iii) Теперь копируем конфигурацию настройки на другой node (cluster2)
scp /etc/corosync/corosync.conf root@cluster2:/etc/corosync/corosync.conf
Генерируем ключ:
corosync-keygen

iv) Копируем ключ на другой компьютер
scp /etc/corosync/authkey root@cluster2:/etc/corosync/authkey

v) Добавляем corosync в автозапуск: (на двух node сразу. Причём у меня не было старт вообще)
mcedit /etc/default/corosync
Меняем START=no на START=yes



Перезагружаемся **reboot**
Поднялся corosync. Проверяем corosync на успешность загрузки:
service corosync status
Должно быть примерно вот так, но основное, что он был ACTIVE и зелёным радостно светился.



e) Теперь нужно сконфигурировать haproxy:
(на каком-то одном ноде, например на cluster1)
mcedit /etc/haproxy/haproxy.cfg
Дописываем в конце файла эти строчки:
frontend front
bind 192.168.197.11:80 (ip адрес:порт виртуального интерфейса VirtualIP P.S. любой незанятый)
default_backend back

backend back (back – имя, можно свои придумать)
balance roundrobin
server cluster1 192.168.197.128:81 (ip адрес первой машины)
server cluster2 192.168.197.129:81 (ip адрес второй машины)
cluster1/2 – название host из установки OS или /etc/hostname и /etc/hosts
80 порт для haproxy, 81 – для apache2
Порты для машин должны отличаться от порта для виртуального адреса, чтобы каждый слушал свой порт. haproxy будет слушать запросы и динамически направлять их на backend.
Мои параметры:
VirtualIP – это какой-нибудь третий ip, который должен поднять pacemaker, чтобы получать запросы из сети. Этот ip – это фактически ip сервера с вашим сайтом.
В лабе просят два сайта поднять, поэтому будет два блока будет frontend-backend, каждый блок для своего сайта как бы формально.

В конце файла haproxy.cfg добавить строки.

frontend front
bind *:80
default_backend back

backend back
balance roundrobin
server cluster1 10.0.2.4:81 *check*
server cluster2 10.0.2.5:81 *check*

i) Проверка валидности настроек haproxy:
haproxy -f /etc/haproxy/haproxy.cfg -c

```
root@cluster1:~# haproxy -f /etc/haproxy/haproxy.cfg -c
Configuration file is valid
```

ii) Копируем эти настройки на вторую машину
scp /etc/haproxy/haproxy.cfg root@cluster2:/etc/haproxy/haproxy.cfg

iii) **service haproxy start** *полезные команды*
service haproxy stop
service haproxy reload
service haproxy status

f) **apt install crmsh** — нужно установить shell crm, через него будем управлять кластером.

g) Теперь нужно сконфигурировать **pacemaker** (управляет нашим кластером)
Заходим в режим конфигурации:
Можно выполнить на cluster1. По идее на cluster 2 настройки должны передаться автоматически. Но это не точно! Проверить можно **crm configure show** на второй машине, после **commit** с первой.
crm configure
Вводим следующее:
property no-quorum-policy="ignore"
property stonith-enabled="false"
(отключаем то, что на кластере из 2-ух узлов вызовет дедлоки)
primitive VIP ocf:heartbeat:IPaddr2 params ip="10.0.2.10" cidr_netmask="24" op monitor interval="1s"
(добавляем скрипт на виртуальный IP нашему pacemaker)
primitive HAP lsb:haproxy op monitor interval="1s"
(добавляем скрипт на haproxy, который уже сконфигурирован и будет слушать виртуальный ip на фронтенде по 80-ому порту каждую секунду)
colocation CLC inf: VIP HAP
(определяем размещение, т.е. виртуальный ip и haproxy в нашем кластере всегда будут стартовать на одной машине, иначе haproxy не увидит VIP)
order ORD inf: VIP HAP
(определяем порядок запуска: сначала VIP потом HAP, которыйслушает VIP)

i) Теперь выполняем команду:
commit

```
root@cluster1:~# crm configure
crm(live/cluster1)configure# property no-quorum-policy="ignore"
crm(live/cluster1)configure# property stonith-enabled="false"
crm(live/cluster1)configure# primitive VIP ocf:heartbeat:IPaddr2 params ip="10.0.2.10" cidr_netmask="24" op monito
r interval="1s"
crm(live/cluster1)configure# primitive HAP lsb:haproxy op monitor interval="1s"
crm(live/cluster1)configure# colocation CLC inf: VIP HAP
crm(live/cluster1)configure# order ORD inf: VIP HAP
crm(live/cluster1)configure# commit
```

ii) *exit*, *end*, *status*, *show* – полезные команды в режиме конфигурации
delete удалять примитивы и прочее. Например: delete ORD(это имя)

На второй машине **crm configure show.**

```
root@cluster2:~# crm configure show
node 1: node1
node 2: node2
primitive HAP lsb:haproxy \
        op monitor interval=1s
primitive VIP IPaddr2 \
        params ip=10.0.2.10 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
```

На первой то же самое!

reboot

Поднялся pacemaker.

iii) Чтобы зайти в командную оболочку pacemaker нужно набрать: *crm*

crm_mon или *crm status* – монитор нашего кластера (показывает состояние)

crm configure show – посмотреть конфигурацию pacemaker

```
root@cluster2:/home/dudaev# crm configure show
node cluster1
node cluster2
primitive HAP lsb:haproxy \
        op monitor interval="1s"
primitive VIP ocf:heartbeat:IPaddr2 \
        params ip="192.168.197.11" cidr_netmask="24" \
        op monitor interval="1s"
colocation CLC inf: VIP HAP
order ORD inf: VIP HAP
property $id="cib-bootstrap-options" \
        dc-version="1.1.7-ee0730e13d124c3d58f00016c3376a1de5323cff" \
        cluster-infrastructure="openais" \
        expected-quorum-votes="2" \
        no-quorum-policy="ignore" \
        stonith-enabled="false"
root@cluster2:/home/dudaev# _
```

h) Настройка **apache2**

i) Устанавливаем:

apt-get install apache2

ii) Открываем конфигурационный файл:

mcedit /etc/apache2/ports.conf

Находим и меняем, чтобы не было проблем с haproxy.

NameVirtualHost *: (меняем на порт «для apache2», который указывали ранее) Listen (аналогично)

```
NameVirtualHost *:81
Listen 81
```

iii) Создаем конфигурационный файл для нашего сайта (никакой зависимости имени файла и имени сайта нет, после этих действий доступ все равно будет осуществляться по ip):

mcedit /etc/apache2/sites-available/mysite.com.conf

И пишем туда минимальные данные:

<VirtualHost 192.168.197.128:81 > (ip *адрес машины и порт для* apache2)

ServerName mysite.com

DocumentRoot /var/www/mysite.com (*Расположение в файловой системе*)

</VirtualHost>

Cluster1

<VirtualHost 10.0.2.4:81 >.

ServerName fit.nsu

DocumentRoot /var/www/fit.nsu

</VirtualHost>

Cluster2

<VirtualHost 10.0.2.5:81 >.

ServerName fit.nsu

DocumentRoot /var/www/fit.nsu

</VirtualHost>

iv) Создаем папку для сайта
mkdir /var/www/fit.nsu

v) Создаем стартовую страницу (либо копируем дефолтную (которая лежит рядом с папкой, например через mc), и меняем для узнаваемости)
mcedit /vat/www/fit.nsu/index.html

vi) Запускаем сайт
a2ensite mysite.com.conf

vii) Запускаем apache2
service apache2 start

viii) Повторяем для второго компьютера (Только ip другой) (Имена «сайтов» могут быть разными, они никак не связаны)

ix) Теперь по VIP модно ходить на сайты с **apache2**, при этом каждый раз балансировщик будет бросать на следующий компьютер.

x) **Я куки не делал** 😊
Если привязать к кукам, то лазить надо будет с разных браузеров. Для этого надо настроить **sticky sessions**:
Открываем настройки **haproxy**:
nano /etc/haproxy/haproxy.cfg
И меняем:

balance roundrobin
server cluster1 192.168.197.128:81
server cluster2 192.168.197.129:81

На:

balance roundrobin
cookie SERVERID insert indirect nocache
server cluster1 192.168.197.128:81 cookie cluster1 check
server cluster2 192.168.197.129:81 cookie cluster2 check

3) Проверка кластера:

Открываем браузер на третьей виртуалке и вбиваем виртуальный ip в адресную строку.
При обновлении страницы содержимое будет меняться (если на серверах лежат разные сайты)
Если отключить один сервер, то сайт должен дальше работать, но отображать только сайт другого кластера
Можно ещё пробросить порт в VirtualBox написать виртуальный ip сайта и порт 80, из haproxy.
В браузере на Windows localhost:<порт проброса на хосте>.

4) Настраиваем ДНС (сайт у нас уже есть, доступный по виртуальному ip)
https://code-inside.com/prostaya-nastroyka-dns-servera-bind9-na-debian-7-wheezy/#.YKZCkFD-pPY
(сайт для ознакомления, делал по нему. Ниже команды нужные)

a) Создаем четвертую машину без графического интерфейса, либо можно использовать третью с графическим

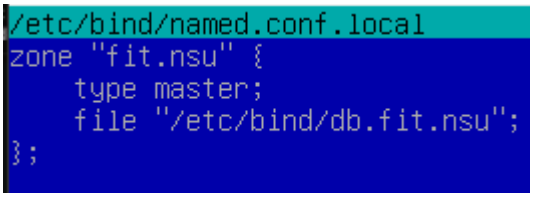
b) Устанавливаем bind9:
apt-get install bind9

c) Настройка dns-сервера
Отредактируем файл
mcedit /etc/bind/named.conf.options
Приводим файл к следующему виду:

```
acl mynetwork {10.0.2.0/24; 127.0.0.1; };
  options {
    directory "/var/cache/bind";
    auth-nxdomain no;
    forwarders {8.8.8.8; };
    listen-on-v6 { none; };
    allow-query { mynetwork; };
  };
```

d) Настройка зоны

mcedit /etc/bind/named.conf.local




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

e) Создаем новую зону

mcedit /etc/bind/db.fit.nsu

```
$TTL 30
$ORIGIN fit.nsu.
.
@ IN SOA fit.nsu. admin.fit.nsu. (
   2021052001 ; Serial
        1d ; Refresh
        1h ; Retry
        1w ; Expire
        2h ; Negative Cache TTL
   )
.
@   IN   NS   fit.nsu.
@   IN   NS   ns.provider.org.
@   IN   A    10.0.2.10 – Virtual IP of the CLUSTER
```

f) Проверка файла зон на наличие ошибок:

named-checkconf -z

g) Если никаких ошибок нет, обновим информацию о зонах:

```
rndc reload
```

5) Настройка ДНС для клиента.

Данный пункт обязателен для подключения DNS – сервера. Необходимо прописать ip этого сервера на машине, с которого будет осуществлена проверка.
https://losst.ru/nastrojka-dns-v-debian

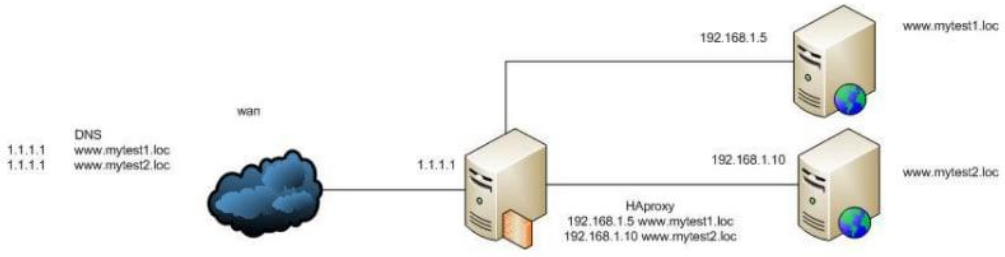
Вместо

nameserver 8.8.8.8 – это dns сервера google
nameserver 8.8.4.4

Написать в конфигурационный файл

nameserver (ip dns сервера) мой был 10.0.2.9

Как сделать два сайта? Чтобы у кластера было два виртуальных IP.
Нужно начать настройку с haproxy.cfg
https://habr.com/ru/post/244027/



При этом конфигурация самого HAProxy крайне простая (пример №1):

```
frontend http_frontend
bind *:80
mode http
option httpclose
acl is_mytest1 hdr_end(host) -i mytest1.loc
use_backend mytest1_web if is_mytest1
acl is_mytest2 hdr_end(host) -i mytest2.loc
use_backend mytest2_web if is_mytest2

backend mytest1_web
mode http
cookie SERVERID insert indirect nocache
server mytestweb1 192.168.1.5:80 check cookie mytestweb1

backend mytest2_web
mode http
cookie SERVERID insert indirect nocache
server mytestweb2 192.168.1.10:80 check cookie mytestweb2
```

Для этой штуки нодам с haproxy нужен доступ к DNS серверу, иначе он не сможет проксить по fit.nsu/fit2.nsu.

При этом, если мы так делаем, то по ip зайти уже не можем.

Я дописал ещё две строчки, где вместо fit.nsu писал ip.

Вероятно, есть способ лучше. Я не придумал, да и было это в пол2 ночи на третий день работы с лабой. Лень.

Далее нужно запилить VIP2. HAP оставить один.

DNS запись делается изи. Повторить там три шага.