EVDOKIMOVA DARYA 21205
HAPROXY SERVER aka ОТКАЗОУСТОЙЧИВЫЙ КЛАСТЕР

РЕКОМЕНДАЦИЯ: Всё делать с использованием мобильного интернета (короче говоря, телефон - точка доступа). Иначе сделаете из-под домашнего роутера, придёте в нгу - и там все слетит, потому что ip адреса другие, сеть другая. А с мобилой всегда одни и те же.

Здесь рассказываю про идею ha кластера. Все команды - делать по инструкции. При возникновении проблем - гугол + может какие-то советы из моего дока помогут.

Оборудование: VirualBox (я делала на ubuntu, вот инструкция по скачиванию);
3 сервера:

- Node1 на debian (amd64)
- Node2 на debian (amd64)
- DNSServak на ubuntu server (потому что почему-то с debian пошло что-то не так, но может у вас норм будет).

Тип - подключения - сетевой мост(!!!) (иначе у нас 3 машины будут отображаться на какой-то один адрес, который потом принимает/отдает данные.
При клонировании установленной первой машины генерация новых mac-адресов - обязательна!!! (иначе все 3 машины будут с одним и тем же ip-адресом, а нам этого не надо).

Установка машины (если чо) - как в инструкции (пункт “c. Установка OS”)).

Важно! Нужно прописать статические адреса на узлах (Node1 and Node2).

ТЕРМИНОЛОГИЯ

- Кластер — группа серверов (вычислительных единиц), объединенных каналами связи.
- Отказоустойчивость (Fault Tolerance, FT) — способность системы к дальнейшей работе после выхода из строя какого-либо её элемента.
- Высокая доступность (High Availability, HA) — в случае выхода из строя узла пользователь какое-то время не будет получать услугу, однако восстановление системы произойдёт автоматически; время простоя минимизируется.
- КВД — кластер высокой доступности, HA-кластер.

Что значит “кластер высокой доступности”?
Принцип работы ha cluster.

Про corosync and pacemaker

- **Corosync** — программный продукт, который позволяет создавать единый кластер из нескольких аппаратных или виртуальных серверов.
  Corosync отслеживает и передает состояние всех участников (нод) в кластере.

Этот продукт позволяет:

  - мониторить статус приложений;
  - оповещать приложения о смене активной ноды в кластере;
  - отправлять идентичные сообщения процессам на всех нодах;
  - предоставлять доступ к общей базе данных с конфигурацией и статистикой;
  - отправлять уведомления об изменениях, произведенных в базе.
  -

- **Pacemaker** - менеджер ресурсов кластера. Он позволяет использовать службы и объекты в рамках одного кластера из двух или более нод. Вот вкратце его достоинства:
  - позволяет находить и устранять сбои на уровне нод и служб;
  - не зависит от подсистемы хранения: можем забыть общий накопитель, как страшный сон;
  - не зависит от типов ресурсов: все, что можно прописать в скрипты, можно кластеризовать;
  - поддерживает STONITH (Shoot-The-Other-Node-In-The-Head), то есть умершая нода изолируется и запросы к ней не поступают, пока нода не отправит сообщение о том, что она снова в рабочем состоянии;
  - поддерживает кворумные и ресурсозависимые кластеры любого размера;
  - поддерживает практически любую избыточную конфигурацию;
  - может автоматически реплицировать конфиг на все узлы кластера — не придется править все вручную;
  - можно задать порядок запуска ресурсов, а также их совместимость на одном узле;
  - поддерживает расширенные типы ресурсов: клоны (когда ресурс запущен на множестве узлов) и дополнительные состояния (master/slave и подобное) — актуально для СУБД (MySQL, MariaDB, PostgreSQL, Oracle);
  - имеет единую кластерную оболочку CRM с поддержкой скриптов.

ИДЕЯ

(*Здесь рассказываю про идею ha кластера. Все команды - делать по инструкции. При возникновении проблем - гугол + может какие-то советы из моего дока помогут. Ну и еще уточнения команд будут, зачем что делали и тд).

Идея заключается в том, что с помощью Corosync мы построим кластер (грубо говоря, “объединим” хосты), а следить за его состоянием будем с помощью Pacemaker (его главная задача — достижение максимальной доступности ресурсов, которыми он управляет, и защита их от сбоев).

Пример из жизни сисадмина, поможет понять, зачем нужен ha cluster.

ТЕПЕРЬ ДЕТАЛЬКИ

Про corosync: Смотреть презентацию в обсуждении задания к лабе.
Вот здесь хорошо написано.
Corosync uses the totem protocol for "heartbeat" like monitoring of the other node's health. A token is passed around to each node, the node does some work (like acknowledge old messages, send new ones), and then it passes the token on to the next node. This goes around and around all the time. Should a node not pass it's token on after a short time-out period, the token is declared lost, an error count goes up and a new token is sent. If too many tokens are lost in a row, the node is declared lost/dead.
Once the node is declared lost, the remaining nodes reform a new cluster. If enough nodes are left to form quorum, then the new cluster will continue to provide services. In two-node clusters, quorum is disabled so each node can work on it's own.

В директиве totem изменяем (либо если его нет, то создаем) interface на следующий:

```
        interface {
                ringnumber: 0
                bindnetaddr: 192.168.106.0
                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: 192.168.106.104
}
node {
        name: node2
        nodeid: 2
        ring0_addr: 192.168.106.105
}
```

ring0_addr – это ip-адреса 1 и 2 узла

*Все настройки corosync -все по инструкции.
!При копировании ключа authkey возможна ошибка permission denied.
Совет: скопировать из-под рута (с Node1) на user (Node2) в корневую папку. А потом уже в нужную (/etc/corosync/).

Once the node is declared lost, the remaining nodes reform a new cluster. If enough nodes are left to form quorum, then the new cluster will continue to provide services. In two-node clusters, quorum is disabled so each node can work on it's own. (Это как раз наш случай)

*тут плавненько перейдем к настройке Pacemaker

Читаем про кворум.
Что такое кворум? Говорят, что кластер имеет кворум при достаточном количестве «живых» узлов кластера. Достаточность количества «живых» узлов определяется по формуле: n > N/2, где n – количество живых узлов, N – общее количество узлов в кластере.
Читаем про команды pacemaker-a.
—----------

В режиме конфигурации “crm configure’ пишем следующие команды:

- *property no-quorum-policy=ignore*
  Это означает, что если если нода может достучаться более, чем до 50% нод, то нужно убить другую группу нод. В данном случае при отказе 1 ноды вторая не будет работать, т.к. не набрано 50%.
  В кластере, состоящем из двух узлов, при сбое на одном из 2-х узлов не будет кворума. По умолчанию, если нет кворума, Pacemaker останавливает ресурсы. Чтобы этого избежать, нужно при настройке Pacemaker указать ему, чтобы наличие или отсутствие кворума не учитывалось.
- *property stonith-enabled=false*
  STONITH = Shoot-TheOther-Node-In-The-Head
  When pacemaker is told that membership has changed because a node died, it looks to see what services might have been lost. Once it knows what was lost, it looks at the rules it's been given and decides what to do.

  Generally, the first thing it does is "stonith" (aka "fence") the lost node. This is a process where the lost node is powered off, called power fencing, or cut off from the network/storage, called fabric fencing. In either case, the idea is to make sure that the lost node is in a known state. If this is skipped, the node could recover later and try to provide cluster services, not having realized that it was removed from the cluster. This could cause problems from confusing switches to corrupting data.

  В нашем случае каждая нода будет пытаться сделать STONITH другой ноде. Ничего работать не будет, потому что все ноды умрут.
- 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. По-умолчанию кластер не обеспечивает работоспособность ресурсов, т.е. не следит после запуска, жив ли ресурс. Чтобы это происходило, нужно добавлять операцию monitor, тогда кластер будет следить за состоянием ресурса. Параметр interval этой операции - интервал с каким делать проверку.
- 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. При отказе одной ноды, вторая должна иметь необходимые ресурсы для продолжения работы, поэтому необходимо разместить ресурсы на *каждой* ноде.
Иногда нужно, чтобы некоторый набор ресурсов выполнялся на одном узле, но не все эти ресурсы являются примитивами, поэтому их нельзя сгруппировать. В этом случае используется colocation - 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)
- commit
  Сохраняем настройки.
- /sbin/reboot
  Перезапускаем машину.

*Когда установим pacemaker на Node2 и напишем ‘crm configure status’, то будут те же данные, что и на Node1 (на которой мы изначально задавали все настройки).
—----------

Теперь перейдем к haproxy.
Отличная статья.
HAProxy - это балансировщик нагрузки.

Команда ‘net.ipv4.ip_nonlocal_bind=1’ зачем нужна? This command allows processes to bind() to non-local IP addresses, which can be quite useful for application such as load balancer.
То есть у нас есть floating point - он же frontend - адрес 192.68.106.111 - виртуальный адрес, который мы привязываем в наш кластер.