

 citius 21 февраля 2017 в 15:55

Кластер PostgreSQL высокой надежности на базе Patroni, Naproxy, Keepalived

Системное администрирование, Администрирование баз данных, DevOps

Tutorial

Привет, Хабр! Встала передо мной недавно задача: настроить максимально надежный кластер серверов PostgreSQL версии 9.6.

По задумке, хотелось получить кластер, который переживает выпадение любого сервера, или даже нескольких серверов, и умеет автоматически вводить в строй сервера после аварий.

Планируя кластер я проштудировал много статей, как из основной документации к PostgreSQL, так и различных howto, в том числе с Хабра, и пробовал настроить стандартный кластер с RepMgr, экспериментировал с pgpool.

В целом оно заработало, но у меня периодически всплывали проблемы с переключениями, требовалось ручное вмешательство для восстановления после аварий, и т.д. В общем я решил поискать еще варианты.

В итоге где-то (уже не вспомню точно где) нашел ссылку на прекрасный проект [Zalando Patroni](#), и все заверте...

Введение

Patroni — это демон на python, позволяющий автоматически обслуживать кластеры PostgreSQL с различными типами репликации, и автоматическим переключением ролей.

Его особенная красота, на мой взгляд в том, что для поддержания актуальности кластера и выборов мастера используются распределенные DCS хранилища (поддерживаются Zookeeper, etcd, Consul).

Таким образом кластер легко интегрируется практически в любую систему, всегда можно выяснить кто в данный момент мастер, и статус всех серверов запросами в DCS, или напрямую к Patroni через http.

Ну и просто это красиво :)

Я потестировал работу Patroni, пробовал ронять мастера и другие сервера, пробовал наливать разные базы (~25 Гб база автоматически поднимается с нуля на 10Гб сети за несколько минут), и в целом мне проект Patroni очень понравился. После полной реализации описанной ниже схемы я проводил тестирование простым бенчером, который ходил в базу по единому адресу, и переживал падения всех элементов кластера (мастер сервера, haproxy, keepalived).

Задержка при передаче роли новому мастеру составляла пару секунд. При возвращении бывшего мастера в кластер, или добавлении нового сервера, смены ролей не происходит.

Для автоматизации разворачивания кластера и добавления новых серверов, решено было использовать привычный Ansible (я дам ссылки на получившиеся роли в конце статьи). В качестве DCS выступает уже применяемый у нас Consul.

У статьи две основные цели: показать пользователям PostgreSQL что есть такая прекрасная штука как Patroni (упоминаний в рунете вообще и на Хабре в частности, практически нет), и заодно немного поделиться опытом использования Ansible на простом примере, тем кто только начинает с ним работать.

Я постараюсь разъяснить все действие сразу на примере разбора Ansible ролей и плейбуков. Те, кто не использует Ansible, смогут перенести все действия в любимое средство автоматизированного управления серверами, либо выполнить их же вручную.

Поскольку большая часть уaml скриптов будет длинной, я буду заворачивать их в спойлер.

Рассказ будет разделен на две части — подготовка серверов и разворачивание непосредственно кластера.

Тем кто хорошо знаком с Ansible первая часть интересна не будет, поэтому рекомендую перейти сразу ко второй.

Часть I

Для этого примера я использую виртуальные машины на базе Centos 7. Виртуалки разворачиваются из шаблона который

периодически обновляется (ядро, системные пакеты), но эта тема выходит за рамки данной статьи.

Отмечу только, что никакого прикладного или серверного софта на виртуалках заранее не установлено. Также вполне подойдут любые облачные ресурсы, например с AWS, DO, vScale, и т.п. Для них есть скрипты динамического инвентаря и интеграции с Ansible, либо можно прикрутить Terraform, так что весь процесс создания и удаления серверов с нуля может быть автоматизирован.

Для начала нужно создать инвентарь используемых ресурсов для Ansible. Ansible у меня (и по умолчанию) расположен в /etc/ansible. Создаем инвентарь в файле /etc/ansible/hosts:

```
[pgsql]
cluster-pgsql-01.local
cluster-pgsql-02.local
cluster-pgsql-03.local
```

У нас используется внутренняя доменная зона .local, поэтому у серверов такие имена.

Далее нужно подготовить каждый сервер к установке всех необходимых компонентов, и рабочих инструментов.

Для этой цели создаем плейбук в /etc/ansible/tasks:

[/etc/ansible/tasks/essentialsoftware.yml](#)

```
---

- name: Install essential software
  yum: name={{ item }} state=latest
  tags: software
  with_items:
    - ntpdate
    - bzip2
    - zip
    - unzip
    - openssl-devel
    - mc
    - vim
    - atop
    - wget
    - mytop
    - screen
    - net-tools
    - rsync
    - psmisc
    - gdb
    - subversion
    - htop
    - bind-utils
    - sysstat
    - nano
    - iptraf
    - nethogs
    - ngrep
    - tcpdump
    - lm_sensors
    - mtr
    - s3cmd
    - psmisc
    - gcc
    - git
    - python2-pip
    - python-devel

- name: install the 'Development tools' package group
  yum:
    name: "@Development tools"
    state: present
```

Набор пакетов Essential служит для создания на любом сервере привычного рабочего окружения. Группа пакетов Development tools, некоторые библиотеки -devel и python нужны pip-у для сборки Python модулей к PostgreSQL.

Мы используем виртуальные машины на базе VmWare ESXi, и для удобства администрирования в них нужно запускать агент vmware.

Для этого мы запустим открытый агент vmttoolsd, и опишем его установку в отдельном плейбуке (поскольку не все сервера у нас виртуальные, и возможно для каких-то из них этот task не понадобится):

[/etc/ansible/tasks/open-vm-tools.yml](#)

Для того чтобы завершить подготовку сервера к установке основной части софта, в нашем случае, понадобятся следующие шаги:

- 1) настроить синхронизацию времени с помощью ntp
- 2) установить и запустить zabbix агент для мониторинга
- 3) накатить требуемые ssh ключи и authorized_keys.

Чтобы не слишком раздувать статью деталям не относящимися к собственно кластеру, я кратко процитирую ansible плейбуки, выполняющие эти задачи:

NTP:

[/etc/ansible/tasks/ntpd.yml](#)

Вначале проверяется, не выставлена ли для сервера персональная таймзона, и если нет, то выставляется Московская (таких серверов у нас большинство).

Мы не используем ntpd из-за проблем с уплыванием времени на виртуалках ESXi, после которого ntpd отказывается синхронизировать время. (И tinker panic 0 не помогает). Поэтому просто запускаем кронем ntp клиент раз 5 минут.

Zabbix-agent:

[/etc/ansible/tasks/zabbix.yml](#)

```
---

- name: set zabbix ip external
  set_fact:
    zabbix_ip: 132.xx.xx.98
  tags: zabbix

- name: set zabbix ip internal
  set_fact:
    zabbix_ip: 192.168.xx.98
  when: ansible_all_ipv4_addresses | ipaddr('192.168.0.0/16')
  tags: zabbix

- name: Import Zabbix3 repo
  yum: name=http://repo.zabbix.com/zabbix/3.0/rhel/7/x86_64/zabbix-release-3.0-1.el7.noarch.rpm state=present
  tags: zabbix

- name: Remove old zabbix
  yum: name=zabbix2* state=absent
  tags: zabbix

- name: Install zabbix-agent software
  yum: name={{ item }} state=latest
  tags: zabbix
  with_items:
    - zabbix-agent
    - zabbix-release

- name: Creates directories
  file: path={{ item }} state=directory
  tags:
    - zabbix
```

```

- zabbix-mysql
with_items:
  - /etc/zabbix/externalscripts
  - /etc/zabbix/zabbix_agentd.d
  - /var/lib/zabbix

- name: Copy scripts
  copy: src=/etc/ansible/templates/zabbix/{{ item }} dest=/etc/zabbix/externalscripts/{{ item }} owner=zabbix
x group=zabbix mode=0755
  tags: zabbix
  with_items:
    - netstat.sh
    - iostat.sh
    - iostat2.sh
    - iostat_collect.sh
    - iostat_parse.sh
    - php_workers_discovery.sh

- name: Copy .my.cnf
  copy: src=/etc/ansible/files/mysql/.my.cnf dest=/var/lib/zabbix/.my.cnf owner=zabbix group=zabbix mode=07
00
  tags:
  - zabbix
  - zabbix-mysql

- name: remove default configs
  file: path={{ item }} state=absent
  tags: zabbix
  with_items:
    - /etc/zabbix_agentd.conf
    - /etc/zabbix/zabbix_agentd.conf

- name: put zabbix-agentd.conf to default place
  template: src=/etc/ansible/templates/zabbix/zabbix_agentd.tpl dest=/etc/zabbix_agentd.conf owner=zabbix gr
oup=zabbix force=yes
  tags: zabbix

- name: link zabbix-agentd.conf to /etc/zabbix
  file: src=/etc/zabbix_agentd.conf dest=/etc/zabbix/zabbix_agentd.conf state=link
  tags: zabbix

- name: zabbix-agent start and enable
  service: name=zabbix-agent state=restarted enabled=yes
  tags: zabbix

```

При установке Zabbix конфиг агента накатывается из шаблона, нужно поменять только адрес сервера.

Сервера расположенные в пределах нашей сети ходят на 192.168.x.98, а сервера не имеющие в нее доступа, на реальный адрес этого же сервера.

Перенос ssh ключей и настройка ssh вынесена в отдельную роль, которую можно найти, например, на [ansible-galaxy](#).

Вариантов там много, а суть изменений достаточно тривиальна, поэтому цитировать весь ее контент здесь я смысла не вижу.

Настала пора накатить на сервера созданную конфигурацию. Вообще я выполняю установку всех компонентов и самого кластера в один шаг, уже при наличии полного конфига, но мне кажется что для целей данного tutorials будет лучше разделить это на два шага, соответственно по главам.

Создаем плейбук для группы серверов:

[/etc/ansible/cluster-pgsql.yml](#)

Запускаем обработку всех серверов:

[Скрытый текст](#)

Если вы полностью скачали мой пример из гитхаб репозитория, то у вас будет также в наличии и роль Patroni, которую нам пока

отрабатывать не нужно.

Аргумент `--skip-tags` заставляет Ansible пропустить шаги помеченные этим тегом, поэтому роль `ansible-role-patroni` выполняться сейчас не будет.

Если же ее на диске нет, то ничего страшного и не произойдет, Ansible просто проигнорирует этот ключ.

Ansible у меня заходит на сервера сразу пользователем `root`, а если вам потребуется пускать ansible под непривилегированного пользователя, стоит дополнительно добавить в шаги требующие рутовых прав специальный флаг `«become: true»`, который побудит ansible использовать вызовы `sudo` для этих шагов.

Подготовка закончена.

Часть II

Приступаем к разворачиванию непосредственно кластера.

Поскольку для настройки кластера требуется много работы (установить PostgreSQL и все компоненты, залить для них индивидуальные конфиги), я выделил весь этот процесс в отдельную роль.

Роли в Ansible позволяют сгруппировать наборы смежных задач, и тем упрощают написание скриптов и поддержку их в рабочем состоянии.

Шаблон роли для установки Patroni я взял тут: <https://github.com/gitinsky/ansible-role-patroni>, за что спасибо его автору. Для своих целей я переработал имеющийся и добавил свои плейбуки `haproxy` и `keepalived`.

Роли у меня лежат в каталоге `/etc/ansible/roles`. Создаем каталог для новой роли, и подкаталоги для ее компонентов:

```
~# mkdir /etc/ansible/roles/ansible-role-patroni/tasks
~# mkdir /etc/ansible/roles/ansible-role-patroni/templates
```

Помимо PostgreSQL наш кластер будет состоять из следующих компонентов:

- 1) `haproxy` для отслеживания состояния серверов и перенаправления запросов на мастер сервер.
- 2) `keepalived` для обеспечения наличия единой точки входа в кластер — виртуального IP.

Все плейбуки выполняемые данной ролью перечисляем в файле, запускаемом ansible по умолчанию:

[/etc/ansible/roles/ansible-role-patroni/tasks/main.yml](#)

Далее начинаем описывать отдельные задачи.

Первый плейбук устанавливает PostgreSQL 9.6 из родного репозитория, и дополнительные пакеты требуемые Patroni, а затем скачивает с GitHub саму Patroni:

[/etc/ansible/roles/ansible-role-patroni/tasks/postgres.yml](#)

Кроме установки ПО данный плейбук также заливает конфигурацию для текущего сервера Patroni, и `systemd` юнит для запуска демона в системе, после чего запускает демон Patroni. Шаблоны конфигов и `systemd` юнит должны лежать в каталоге `templates` внутри роли.

Шаблон конфига Patroni:

[/etc/ansible/roles/ansible-role-patroni/templates/postgres.yml.j2](#)

Поскольку для каждого сервера кластера требуется индивидуальная конфигурация Patroni, его конфиг лежит в виде шаблона `jinja2` (файл `postgres0.yml.j2`), и шаг `template` заставляет ansible транслировать этот шаблон с заменой переменных, значения из которых берутся из отдельного описания для каждого сервера.

Переменные, общие для всего кластера укажем в прямо в инвентаре, который примет теперь следующий вид:

[/etc/ansible/hosts](#)

Расшифрую для чего нужны некоторые переменные:

patroni_scope — название кластера при регистрации в Consul

patroni_node_name — название сервера при регистрации в Consul

patroni_rest_password — пароль для http интерфейса Patroni (требуется для отправки команд на изменение кластера)

patroni_postgres_password: пароль для юзера postgres. Он устанавливается в случае создания patroni новой базы.

patroni_replicator_password — пароль для юзера replicator. От его имени осуществляется репликация на слейвы.

Также в этом файле перечислены некоторые другие переменные, используемые в других плейбуках или ролях, в частности тот может быть настройка ssh (ключи, пользователи), таймзона для сервера, приоритет сервера в кластере keepralived, и.т.п.

Конфигурация для остальных серверов аналогична, соответственно меняется имя сервер и приоритет (например 99-100-101 для трех серверов).

Установка и настройка haproxy:

[/etc/ansible/roles/ansible-role-patroni/tasks/haproxy.yml](#)

Нароуху устанавливается на каждом хосте, и содержит в своем конфиге ссылки на все сервера PostgreSQL, проверяет какой сервер сейчас является мастером, и отправляет запросы на него.

Для этой проверки используется прекрасная фишка Patroni — REST интерфейс.

При обращении на url server:8008 (8008 это порт по умолчанию) Patroni возвращает отчет по состоянию кластера в json, а также отражает кодом ответа http является ли данный сервер мастером. Если является — будет ответ с кодом 200. Если же нет, ответ с кодом 503.

Очень советую обратиться в документацию на Patroni, http интерфейс там достаточно интересный, позволяет также принудительно переключать роли, и управлять кластером.

Аналогично, это можно делать при помощи консольной утилиты patronictl.py, из поставки Patroni.

Конфигурация haproxy достаточно простая:

[/etc/ansible/roles/ansible-role-patroni/templates/haproxy.cfg](#)

В соответствии с этой конфигурацией haproxy слушает порт 5000, и отправляет трафик с него на мастер сервер.

Проверка статуса происходит с интервалом в 1 секунду, для перевода сервера в даун требуется 3 неудачных ответа (код 500), для переключения сервера назад — 2 удачных ответа (с кодом 200).

В любой момент времени можно обратиться непосредственно на любой haproxy, и он корректно запроксирует трафик на мастер сервер.

Также в комплекте с Patroni есть шаблон для настройки демона confd, и пример его интеграции с etcd, что позволяет динамически менять конфиг haproxy при удалении или добавлении новых серверов.

Я же пока делаю достаточно статичный кластер, лишняя автоматизация в данной ситуации, имхо, может привести к непредвиденным проблемам.

Нам хотелось, чтобы на клиентах особые изменения логики, отслеживание серверов на живости и т.д. не требовались, поэтому мы делаем единую точку входа в кластер с помощью keepralived.

Демон keepralived работает по протоколу vrrp со своими соседями, и в результате выборов одного из демонов как главного (приоритет указан в конфиге, и шаблонизирован в переменную keepralived_priority в host_vars для каждого сервера), он поднимает у себя виртуальный ip адрес.

Остальные демоны терпеливо ждут. Если текущий основной сервер keepralived по какой-то причине умрет либо просигналил соседям аварию, произойдут перевыборы, и следующий по приоритету сервер заберет себе виртуальный ip адрес.

Для защиты от падения haproxy демоны keepralived выполняют проверку, запуская раз в секунду команду «killall -0 haproxy». Она возвращает код 0 если процесс haproxy есть, и 1 если его нет.

Если haproxy исчезнет, демон keepralived просигналил аварию по vrrp, и снимет виртуальный ip.

Виртуальный IP сразу же подхватит следующий по приоритету сервер, с живым haproxy.

Установка и настройка keepralived:

[/etc/ansible/roles/ansible-role-patroni/tasks/keepalived.yml](#)

Кроме установки keepalived, этот плейбук также копирует простой скрипт для отправки алертов через телеграм. Скрипт принимает сообщение в виде переменной, и просто дергает curl-ом API телеграма.

В этом скрипте только нужно указать свои токен и ID группы telegram для отсылки оповещений.

Конфигурация keepalived описана в виде jinja2 шаблона:

[/etc/ansible/roles/ansible-role-patroni/templates/keepalived.conf.j2](#)

```
global_defs {
    router_id {{ patroni_node_name }}
}

vrrp_script chk_haproxy {
    script "killall -0 haproxy"
    interval 1
    weight -20
    debug
    fall 2
    rise 2
}

vrrp_instance {{ patroni_node_name }} {
    interface ens160
    state BACKUP
    virtual_router_id 150
    priority {{ keepalived_priority }}
    authentication {
        auth_type PASS
        auth_pass secret_for_vrrp_auth
    }
    track_script {
        chk_haproxy weight 20
    }
    virtual_ipaddress {
        {{ cluster_virtual_ip }}/32 dev ens160
    }
    notify_master "/usr/bin/sh /usr/local/sbin/alert.sh '{{ patroni_node_name }} became MASTER'"
    notify_backup "/usr/bin/sh /usr/local/sbin/alert.sh '{{ patroni_node_name }} became BACKUP'"
    notify_fault "/usr/bin/sh /usr/local/sbin/alert.sh '{{ patroni_node_name }} became FAULT'"
}
```

В переменные patroni_node_name, cluster_virtual_ip и keepalived_priority транслируются соответствующие данные из host_vars.

Также в конфиге keepalived указан скрипт для отправки сообщений о смене статуса в telegram канал.

Накатываем полную конфигурацию кластера на сервера:

```
~# ansible-playbook cluster-pgsql.yml
```

Поскольку Ansible идиомпотентен, т.е. выполняет шаги только если они не были выполнены ранее, можно запустить плейбук без дополнительных параметров.

Если же не хочется дольше ждать, или вы уверены что сервера полностью готовы, можно запустить ansible-playbook с ключом -t patroni.

Тогда будут выполнены только шаги из роли Patroni.

Отмечу что я не указываю отдельно роли серверов — мастер или слейв. Данная конфигурация создаст пустую базу, и мастером просто станет первый сконфигурированный сервер.

При добавлении новых серверов Patroni увидит через DCS что мастер кластера уже есть, автоматически скопирует с текущего мастера базу, и подключит к нему слейв.

В случае запуска слейва отставшего на какое-то время от мастера, Patroni автоматически вольет изменения при помощи pg_rewind.

Убеждаемся что все сервера запустились и выбрали себе роли:

```
~# journalctl -f -u patroni
```

Сообщения со слейва (сервер cluster-pgsql-01):

[спойлер](#)

Сообщения с мастера (в данном случае это сервер cluster-pgsql-02):

[спойлер](#)

По логам ясно видно что каждый сервер постоянно мониторит свой статус и статус мастера. Попробуем остановить мастер:

```
~# systemctl stop patroni
```

[спойлер](#)

А вот что в этот момент произошло на слейве:

[спойлер](#)

Этот сервер перехватил роль мастера на себя.

А теперь вернем сервер 2 обратно в кластер:

```
~# systemctl start patroni
```

[Заголовок спойлера](#)

Patroni обнаружила что подключается к кластеру с имеющимся мастером, и обновив базу до текущего состояния, корректно приняла на себя роль слейва.

Попробуем создать ошибку на другом слое кластера, остановив haproxy на основном сервере keepalived.

По приоритету, эту роль у меня принимает второй сервер:

```
[root@cluster-pgsql-02 ~]# ip a
2: ens160: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP qlen 1000
link/ether 00:50:56:a9:b8:7b brd ff:ff:ff:ff:ff:ff
inet 192.xx.xx.121/24 brd 192.168.142.255 scope global ens160
valid_lft forever preferred_lft forever
inet 192.xx.xx.125/32 scope global ens160 <---- виртуальный адрес кластера
valid_lft forever preferred_lft forever
inet6 fe80::xxx::4895:6d90/64 scope link
valid_lft forever preferred_lft forever
```

Остановим haproxy:

```
~# systemctl stop haproxy ; journalctl -fl
```



```
Feb 18 00:18:54 cluster-pgsql-02.local Keepalived_vrrp[25018]: VRRP_Script(chk_haproxy) failed
Feb 18 00:18:56 cluster-pgsql-02.local Keepalived_vrrp[25018]: VRRP_Instance(cluster_psql_02) Received higher
prio advert
Feb 18 00:18:56 cluster-pgsql-02.local Keepalived_vrrp[25018]: VRRP_Instance(cluster_psql_02) Entering BACKUP
STATE
Feb 18 00:18:56 cluster-pgsql-02.local Keepalived_vrrp[25018]: VRRP_Instance(cluster_psql_02) removing protocol
VIPs.
Feb 18 00:18:56 cluster-pgsql-02.local Keepalived_vrrp[25018]: Opening script file /usr/bin/sh
Feb 18 00:18:56 cluster-pgsql-02.local Keepalived_healthcheckers[25017]: Netlink reflector reports IP
192.xx.xx.125 removed
```

Keepalived отловил проблему, и убрал с себя виртуальный адрес, а также просигналил об этом соседям.

Смотрим что произошло на втором сервере:

```
Feb 18 00:18:56 cluster-pgsql-01.local Keepalived_vrrp[41190]: VRRP_Instance(cluster_psql_01) forcing a new
MASTER election
Feb 18 00:18:56 cluster-pgsql-01.local Keepalived_vrrp[41190]: VRRP_Instance(cluster_psql_01) forcing a new
MASTER election
Feb 18 00:18:56 cluster-pgsql-01.local Keepalived_vrrp[41190]: VRRP_Instance(cluster_psql_01) forcing a new
MASTER election
Feb 18 00:18:56 cluster-pgsql-01.local Keepalived_vrrp[41190]: VRRP_Instance(cluster_psql_01) forcing a new
MASTER election
Feb 18 00:18:57 cluster-pgsql-01.local Keepalived_vrrp[41190]: VRRP_Instance(cluster_psql_01) Transition to
MASTER STATE
Feb 18 00:18:58 cluster-pgsql-01.local Keepalived_vrrp[41190]: VRRP_Instance(cluster_psql_01) Entering MASTER
STATE
Feb 18 00:18:58 cluster-pgsql-01.local Keepalived_vrrp[41190]: VRRP_Instance(cluster_psql_01) setting protocol
VIPs.
Feb 18 00:18:58 cluster-pgsql-01.local Keepalived_vrrp[41190]: VRRP_Instance(cluster_psql_01) Sending gratuitous
ARPs on ens160 for 192.xx.xx.125
Feb 18 00:18:58 cluster-pgsql-01.local Keepalived_vrrp[41190]: Opening script file /usr/bin/sh
Feb 18 00:18:58 cluster-pgsql-01.local Keepalived_vrrp[41190]: VRRP_Instance(cluster_psql_01) Received lower prio
advert, forcing new election
Feb 18 00:18:58 cluster-pgsql-01.local Keepalived_vrrp[41190]: VRRP_Instance(cluster_psql_01) Sending gratuitous
ARPs on ens160 for 192.xx.xx.125
Feb 18 00:18:58 cluster-pgsql-01.local Keepalived_healthcheckers[41189]: Netlink reflector reports IP
192.xx.xx.125 added
Feb 18 00:18:58 cluster-pgsql-01.local Keepalived_vrrp[41190]: VRRP_Instance(cluster_psql_01) Received lower prio
advert, forcing new election
Feb 18 00:18:58 cluster-pgsql-01.local Keepalived_vrrp[41190]: VRRP_Instance(cluster_psql_01) Sending gratuitous
ARPs on ens160 for 192.xx.xx.125
Feb 18 00:19:03 cluster-pgsql-01.local Keepalived_vrrp[41190]: VRRP_Instance(cluster_psql_01) Sending gratuitous
ARPs on ens160 for 192.xx.xx.125
```

Дважды произошли перевыборы (потому что третий сервер кластера успел отправить свой анонс до первых выборов), сервер 1 принял на себя роль ведущего, и выставил виртуальный IP.

Убеждаемся в этом:

```
[root@cluster-pgsql-01 log]# ip a
2: ens160: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP qlen 1000
link/ether 00:50:56:a9:f0:90 brd ff:ff:ff:ff:ff:ff
inet 192.xx.xx.120/24 brd 192.xx.xx.255 scope global ens160
valid_lft forever preferred_lft forever
inet 192.xx.xx.125/32 scope global ens160 <---- виртуальный адрес кластера присутствует!
valid_lft forever preferred_lft forever
inet6 fe80::1d75:40f6:a14e:5e27/64 scope link
valid_lft forever preferred_lft forever
```

Теперь виртуальный IP присутствует на сервере, не являющимся мастером репликации. Однако это не имеет значения, поскольку в базу мы обращаемся через `haproxy`, а она мониторит состояние кластера независимо, и отправляет запросы всегда на мастер.

При возврате в строй `haproxy` на втором сервере снова происходят пере выборы (`keepalived` с бОльшим приоритетом встает в строй), и виртуальный IP возвращается на свое место.

В редких случаях бывает что слейв не может догнаться до мастера (например он упал очень давно и `wal` журнал успел частично удалиться). В таком случае можно полностью очистить директорию с базой на слейве:

~~«rm -rf /var/lib/postgresql/9.6/data», и перезапустить Patroni. Она сольет базу с мастера целиком.~~
~~(Осторожно с очисткой «ненужных» баз, внимательно смотрите на каком сервере вы выполняете команду!!!)~~

В таком случае нужно воспользоваться утилитой `patronictl`. Команда `reinit` позволяет безопасно очистить конкретный узел кластера, на мастере она выполняться не будет.

Спасибо за дополнение @CyberDemon.

Сама утилита `patronictl` позволяет увидеть текущую ситуацию с кластером через командную строку, без обращений в DCS, и управлять кластером.

Пример отчета о состоянии кластера:

`/opt/patroni/patronictl.py -c /etc/patroni/postgres.yml list cluster-pgsql:`

Cluster	Member	Host	Role	State	Lag in MB
cluster-pgsql	cluster_psql_01	192.xxx.xxx.120	Leader	running	0.0
cluster-pgsql	cluster_psql_02	192.xxx.xxx.121	Sync standby	running	0.0
cluster-pgsql	cluster_psql_03	192.xxx.xxx.122		creating replica	33712.0

В данном случае наливается третья нода, ее отставание от мастера составляет 33 Гб.

После завершения этого процесса она также переходит в состояние `Running` с нулевым лагом.

Также можно обратить внимание что поле `State` у нее пустое. Это потому, что кластер в моем случае работает в синхронном режиме. Для уменьшения лага синхронной репликации, один слейв работает в синхронном режиме, а другой в обычном асинхронном. В случае пропадания мастера роли сместятся, и второй слейв перейдет в синхронный режим к ставшему мастером, первому слейву.

Послесловие

Единственное чего этому кластеру, на мой взгляд, не хватает для счастья — это пулинг коннектов и проксирование запросов на чтение на все слейвы для повышения производительности чтения, а запросов на вставки и обновления только на мастер.

В конфигурации с асинхронной репликацией, раскладывание нагрузки на чтение может привести к непредвиденным ответам, если слейв отстанет от мастера, это нужно учитывать.

Стриминговая (асинхронная) репликация не обеспечивает консистентности кластера в любой момент времени, и для этого нужна синхронная репликация.

В этом режиме мастер сервер будет ждать получения подтверждений о копировании и применении транзакций на слейвы, что замедлит работу базы. Однако если потери транзакций недопустимы (например какие-то финансовые приложения), синхронная репликация это ваш выбор.

Patroni поддерживает все варианты, и если синхронная репликация вам подойдет больше, вам всего лишь понадобится изменить значение нескольких полей в конфигах Patroni.

Вопросы разных методов репликации прекрасно разобраны в документации к Patroni.

Кто-то наверное предложит использовать `pgpool` который сам, по сути, покрывает весь функционал этой системы. Он может и мониторить базы, и проксировать запросы, и выставлять виртуальный IP, а также осуществляет пулинг коннектов клиентов.

Да, он все это может. Но на мой взгляд схема с Patroni гораздо прозрачнее (конечно это только мое мнение), и во время

экспериментов с pgpool я ловил странное поведение с его вочдогом и виртуальными адресами, которое не стал пока слишком глубоко дебажить, решив поискать другое решение.

Конечно возможно, что проблема тут только моих в руках, и позже я к тестированию pgpool планирую вернуться.

Однако, в любом случае, pgpool не сможет полностью автоматически управлять кластером, вводом новых и (особенно) возвратом сбойных серверов, работать с DCS. На мой взгляд это самый интересный функционал Patroni.

Спасибо за внимание, буду рад увидеть предложения по дальнейшему улучшению этого решения, и ответить на вопросы в комментариях.

Огромное спасибо Zalando за Patroni, и авторам исходного проекта Governor, который послужил основой для Patroni, а также Алексу Чистякову за шаблон роли для Ansible.

Полный код плейбуков и шаблонов Ansible, описанных в статье [лежит тут](#). Буду благодарен за доработки от гуру Ansible и PostgreSQL. :)

Основные использованные статьи и источники:

Несколько вариантов кластеров PostgreSQL:

- <https://habrahabr.ru/post/301370/>
- <https://habrahabr.ru/post/213409/>
- <https://habrahabr.ru/company/etagi/blog/314000/>
- Пост о Patroni в блоге Zalando
- Проект Patroni
- [ansible-role-patroni](#) Алекса Чистякова
- Governor — к сожалению разработка давно заморожена.
- Книга Ansible for Devops — прекрасный учебник с кучей примеров применения Ansible.

Теги: postgresql, haproxy, patroni, ansible, keepalived, linux


 +34



 312

 44,6k

 69



16,0

Карма

0,1

Рейтинг

13

Подписчики

4

Подписки

Илья Рубинчик @citius

Системный администратор

Поделиться публикацией

ПОХОЖИЕ ПУБЛИКАЦИИ

13 августа 2019 в 10:27

Astra Linux 1.6 (Смоленск). Готова ли система к работе с простыми пользователями? Примеры костылей

 +21  19,9k  44  52

25 июня 2019 в 09:45











Отказоустойчивый кластер PostgreSQL + Patroni. Опыт внедрения

 +27  6,2k  86  21

22 мая 2019 в 11:39

Как мы построили надёжный кластер PostgreSQL на Patroni

 +35  8,6k  86  15

Вопросы и ответы		Гостер
 PostgreSQL  Простой	Что не правильно в запросе PostgreSQL?	2 ответа
 Python  Простой	Как оформить запрос в БД при помощи Django ORM?	1 ответ
 PostgreSQL  Простой	Почему PgAgent вставляет текст в неправильной кодировке и как это исправить?	1 ответ
 PostgreSQL  Средний	Как лучше организовать схему экспорта из OSM OpenStreetMap в PostgreSQL?	1 ответ
 Автоматизация  Средний	Упростить воркфлов для конфигураций сервисов при деплоimente в различные среды?	1 ответ
Все вопросы Задать вопрос		

Комментарии 69

- 
alexkuzko
21 февраля 2017 в 16:58


↑ 0 ↓
- Отличная статья. Спасибо.
 Попробую параллельно поднять ваш конфиг и сравнить.
- 
citius
21 февраля 2017 в 17:05




↑ 0 ↓
- Благодарю! Сравнить будете с более традиционным кластером?
 Поделитесь потом впечатлением, если не сложно. :)
- 
arzonus
21 февраля 2017 в 17:18


↑ 0 ↓
- Спасибо за статью!
 Хотел сам написать, да как то руки не доходили :)
 Мы использовали patroni, засунув его с postgresql в Docker контейнер.
- 
citius
21 февраля 2017 в 17:19




↑ 0 ↓
- А мы как-то пока базы в контейнеры не решаемся.
 Кстати, какие проблемы в эксплуатации выявили? Интересноет именно аспект контейнеризации.
- 
arzonus
21 февраля 2017 в 17:26




↑ 0 ↓
- Так ничего же страшного нету. Папка с БД выносится за контейнер, а процесс БД в контейнере работает. Разницы в работе нет никакой. БД в контейнере можно запустить с тем же юзером, что и на хост системе, что позволяет не волноваться за то, кто является владельцем и что будет происходить с папками БД.
- 
citius
21 февраля 2017 в 17:31




↑ 0 ↓
- Страшного то нет, меня скорее беспокоит то, что с контейнерами меньше возможностей управлять ресурсами, например гарантировать процессорное или дисковое время.
 Мне кажется что для кустов микросервисов контейнеры подходят больше чем для таких тяжелых вещей.
 Хотя в целом у меня с контейнерами пока и опыта богатого нет. :)
- 
arzonus
21 февраля 2017 в 17:52




↑ 0 ↓
- Тут зависит от задач. Так как если у вас небольшая база и вы не хотите управлять процессорным и дисковым временем, то вопрос где запускается БД особо не стоит. Однако если вам уже важны эти времена (я думаю на больших БД), то я думаю, что и Patroni тут уже не будет использоваться, а что то посерьезнее :)
- 
nulled
21 февраля 2017 в 18:59




↑ 0 ↓
- На данный момент юзаем в контейнере postgresql, как уже сказали проблем нет. Но управление ресурсами это уже не совсем про контейнеры. Хотя положительные подвижки в этом аспекте есть.

 PutPixel 21 февраля 2017 в 19:36 # 📖 🔗 🔄

↑ 0 ↓

Много где написано, что если монтировать docker volume на хост начинаются проблемы. У нас это приводило к порче всех файлов docker и сам демон не мог даже стартовать. Какие то особые настройки для docker?

 arzonus 21 февраля 2017 в 20:56 # 📖 🔗 🔄

↑ 0 ↓

Никогда не сталкивался с такой проблемой. Обычная команда `docker run -v /path:/path repo:tag`. Я только пробовал подключать docker volume как Azure File System. Однако из-за отсутствия поддержки симлинков в Azure File System, постгрес не хотел работать :)

 nulled 21 февраля 2017 в 21:54 # 📖 🔗 🔄

↑ 0 ↓

Подобные проблемы возникали на старых версиях docker, просто умирал dm в котором был rootfs контейнера. Fedora 25, последнее доступное ядро + данные, персистентность которых нужно обеспечить, пробрасываются через -v. Ну и сам докер держим последний.

 FireWolf2007 21 февраля 2017 в 22:03 # 📖

↑ 0 ↓

Что будет, если не программно тушить сервисы, а выдергивать сетевой кабель?

 citius 21 февраля 2017 в 22:04 # 📖 🔗 🔄

↑ +1 ↓

keeralived потеряют соседа и перевыберут мастера между собой. Patroni тоже выберет нового мастера, т.к. перестанет обновляться тикет в DCS. Текущий мастер будет изолирован, и начнет отставать по базе. После восстановления коннективности нужно будет просто перезапустить Patroni на старом мастере, и он должен штатно догнаться до слейва.

 greberj 22 февраля 2017 в 01:53 # 📖

↑ 0 ↓

Интересная статья. Хочу попробовать реализовать и потестить.
Замечание: ansible фейлит на zabbix на задаче скопировать скрипты. Их Вы не выложили в репозиторий. Если можно — выложите. Если нет, обойдется.
Спасибо за труд!

 citius 22 февраля 2017 в 01:53 # 📖 🔗 🔄

↑ 0 ↓

Да, тормозил я с этими скриптами. Надо было роль от лишнего почистить. :)
Впрочем мне не жалко, и ничего секретного там нет. Выложил в репозиторий.
Как пример пойдет.

 greberj 22 февраля 2017 в 11:09 # 📖 🔗 🔄

↑ 0 ↓

Спасибо за оперативность!
Теперь ругается на `/etc/ansible/files/mysql/.my.cnf`, так как его тоже нет. Подкиньте еще пожалуйста этот файл, чтоб не ругался больше.
Спасибо!

 citius 22 февраля 2017 в 13:01 # 📖 🔗 🔄

↑ 0 ↓

Ну это точно совсем лишняя штука в данном контексте.
Я убрал это из zabbix плейбука, и добавил темлейт конфига забикса, сейчас должно все пройти.
Если будут еще проблемы с ним, лучше дергайте меня напрямую через личку, или контакты в профиле.

 SonicGD 22 февраля 2017 в 07:38 # 📖

↑ 0 ↓

Спасибо за статью. А на Stolon не смотрели?

 citius 22 февраля 2017 в 10:13 # 📖 🔗 🔄

↑ 0 ↓

Он мне попадался, но питон мне лично гораздо проще чем Go, поэтому Patroni больше заинтересовала.
Судя по описанию тоже стоящая штука.

 CyberDem0n 22 февраля 2017 в 10:08 # 📖

↑ +4 ↓

```
«rm -rf /var/lib/pgsql/9.6/data», и перезапустить Patroni. Она сольет базу с мастера целиком.
```

Хотите повторить опыт Gitlab? :) Пожалуйста, НИКОГДА так не делайте. Специально для таких случаев мы придумали `patronictl reinit <cluster> <node>`


Эта команда абсолютна безопасна, текущий мастер просто откажется её выполнять.

Реплика-же сделает всё как нужно: Patroni вначале остановит postgres, удалит data директорию, заберёт новый pg_basebackup с мастера и снова запустит postgres.

Огромное Вам спасибо за статью от Zalando!

 citius 22 февраля 2017 в 10:11 # 0


Ох, крутяк какой. Каким-то образом я это пропустил, хотя точно помню что patronictl я ковырял.
Добавлю в статью, спасибо! :)

 past 22 февраля 2017 в 11:32 # 0

service: name=ntpd state=stopped enabled=no
Зачем Вы так жестоко ломаете ntp?

 citius 22 февраля 2017 в 12:48 # 0

А я писал в статье про проблемы с синхронизацией времени.
Полное описание есть в KB VmWare тут.
Нам пока на данный момент проще убить вообще ntpd, от vSphere мы планируем отказаться.

 NoOne 22 февраля 2017 в 11:52 # 0

Т.е. в вашей конфигурации получается, что виртуальный IP кластера может попасть на ноду слейва postgresql? И тогда при большой загрузке канала будет падать скорость, т.к. трафик удваивается (клиент<->слейв<->мастер).

Текущий мастер будет изолирован, и начнет отставать по базе.

После восстановления коннективности нужно будет просто перезапустить Patroni на старом мастере, и он должен штатно догнаться до слейва.

Что если в мастер попали данные, на слейв улететь не успели и мастер потерял сеть? Один из слейвов все равно поднимется в мастер, а бывший мастер при возврате в сеть затрет уникальные данные и станет слейвом?

 citius 22 февраля 2017 в 12:52 # 0

1) Да, это нужно учитывать. Если объем этого трафика это проблема (лаг там все-таки минимальный добавляется), то стоит либо балансеры вынести наружу, либо сделать репликацию по отдельной сети.

2) Это проблема асинхронной репликации: транзакции которые не успеют считать слейвы будут потеряны.
Именно поэтому у меня репликация синхронная, у нас такие потери недопустимы.
Синхронная репликация обеспечивает консистентность на уровне транзакций.

Недавно тут бы прекрасный был пост про CAP теорему, там эта проблема расписана в деталях.

 NoOne 22 февраля 2017 в 13:47 # 0

Да, обе проблемы ясные и понятно в какую сторону их решать. Просто всегда необходимо выбирать компромис между вариантами :)

 VolCh 22 февраля 2017 в 17:26 # 0

Синхронная репликация обеспечивает консистентность на уровне транзакций.

А если ляжет слейв, мастер продолжит выполнять транзакции?

 citius 22 февраля 2017 в 17:31 # +1

Асинхронный слейв будет переключен в синхронный режим.
Если совсем не будет слейвов, patroni отключит синхронную репликацию.

Вот цитата из документации:

On each HA loop iteration Patroni re-evaluates synchronous standby choice. If the current synchronous standby is connected and has not requested its synchronous status to be removed it remains picked. Otherwise the cluster member available for sync that is furthest ahead in replication is picked.

 VolCh 22 февраля 2017 в 18:27 # 0



Если совсем не будет слейвов, patroni отключит синхронную репликацию.

Вот это интересовало. Спасибо.


 CyberDem0n 22 февраля 2017 в 18:32      +1 

Всё верно, но скоро ещё добавим `synchronous_mode_strict`.
В этом случае мастер не будет выполнять транзакции если нет `synchronous standby`


Но не забывайте, это поведение по умолчанию, и клиент всегда может решить что ему не нужна синхронная репликация и отключить её: `SET local synchronous_commit = 'local';`

 VolCh 22 февраля 2017 в 18:34      0 



В этом случае мастер не будет выполнять транзакции если нет `synchronous standby`
гибко регулировать можно будет? Типа из пяти слейвов в кластере минимум два должны быть с синхронной репликацией, чтобы мастер принимал транзакции?

 CyberDem0n 23 февраля 2017 в 14:37      +1 

Начиная с 9.6 такое возможно, но Patroni пока-что так не умеет.
Если будет свободное время — сделаю, но с другой стороны мы всегда рады пулл-реквестам :)

 unnforgiven 22 февраля 2017 в 12:03    0 

Хорошая статья, спасибо автору. Я писал тоже про кластер postgres только с `repmgr`. Не рассматривали `repmgr`?
<https://habrahabr.ru/company/etagi/blog/314000/>

 citius 22 февраля 2017 в 12:54      0 

Видел, я же даже в «использованные статьи» вас добавил. :)
С Patroni подобная же схема, на мой взгляд гораздо проще и прозрачнее.

 trider 8 сентября 2017 в 11:00      0 

Ни разу не прозрачнее для тех кто не имел дело с DSC. Как я выяснил Patroni сам не заведёт Consul и etcd по которым документации с гулькин нос и надо вшиваться в DSC, чтобы понять как запустить всю эту связку

 SXN 22 февраля 2017 в 14:13    0 


Отличная статья. Спасибо. надо попробовать.

 neb0t 22 февраля 2017 в 16:40    0 


Статья обалденная, но скажите пожалуйста, что вы делаете если `ansible trigger` перезагружает мастера с которого «шарится» IP? Существует бородастый баг, когда нетворк перезагружается — `keepalive` вылетает со скоростью света. Делали здесь

 citius 22 февраля 2017 в 16:40      0 

Ссылка не вставилась, повторите плз.
У нас таких проблем не возникало.

 neb0t 22 февраля 2017 в 18:43      0 

Вот ссылка...
Я попытался сделать реализацию с 2 блб. Если на мастере перезапустить нетворк — тогда шаред IP станет недоступным.

 citius 22 февраля 2017 в 18:47      0 

Я только что попробовал перезапустить сеть на главном `keepalived`, ничего не случилось.
Пинги не пропадали, сеть осталась рабочей.

Это Centos 7.2 с ядром `kernel-ml 4.9.0`, перезапускал через `systemctl restart network`.

 neb0t 22 февраля 2017 в 18:42      0 

<https://blog.a2o.si/2013/10/08/restarting-network-with-keepalived-on-redhat-centos/>

 Seboreia 16 марта 2017 в 22:35    0 

Спасибо за труд!
Хотел бы уточнить одну вещь — в шаблоне для `haproxy` вижу такие строки:
`server {{ patroni_node_name }} {{ patroni_node_name }}.local:5432 maxconn 300 check port 8008`
`server {{ patroni_node_name }} {{ patroni_node_name }}.local:5432 maxconn 300 check port 8008`

server {{ patroni_node_name }} {{ patroni_node_name }}.local:5432 maxconn 300 check port 8008
Разве сюда не будет вставляться одно и то же значение 3 раза?

 citius 16 марта 2017 в 22:52

↑ 0 ↓

Да, пробрался косяк.

Поправил в репозитории на более явное определение серверов.

Нужно сделать строки соответствующие всем серверам кластера, чтобы хапрокси мог их простукивать и проксировать трафик на мастер:

```
backend postgres-patroni
  option httpchk

  http-check expect status 200
  default-server inter 3s fall 3 rise 2

server cluster-pgsql-01 cluster-pgsql-01.local:5432 maxconn 300 check port 8008
server cluster-pgsql-02 cluster-pgsql-02.local:5432 maxconn 300 check port 8008
server cluster-pgsql-03 cluster-pgsql-03.local:5432 maxconn 300 check port 8008
```


 Seboreia 16 марта 2017 в 22:59

↑ 0 ↓

Кстати, не пробовал сам, но видел где-то в интернете: если hostname'ы узлов совпадают с hostname_inventory, то можно записать так:

```
{{ ansible_play_hosts[0] }}
{{ ansible_play_hosts[1] }}
```


и. д.

 citius 16 марта 2017 в 23:01

↑ 0 ↓

Да в ансIBLE вообще по всякому можно, мощная штука. :)


Есть прекрасная книга с кучей примеров, советую прочесть.

 Myrddin 28 марта 2017 в 04:31

↑ 0 ↓

Спасибо за материал. Как раз изучаю вопрос.

В некоторых статьях вместе с харгоху используется rpgbouncer. Есть ли смысл добавлять его в эту схему?

 citius 28 марта 2017 в 09:08

↑ 0 ↓

От задачи зависит. Если нужен пулинг и ограничения баунсера не помешают, то конечно можно добавить.

 past 10 мая 2017 в 17:30

↑ 0 ↓

Скажите, в чем причина использования ядра 4 версии?
Чем не устроило дефолтное центосовское ядро?

 citius 10 мая 2017 в 17:32

↑ +1 ↓

Ну в чейнжлоге между 3.10 и очень 4.10 — много всего, не перечислить. :)

Вкратце — стараюсь не использовать некрософт, если это не обусловлено какими-то требованиями к совместимости.

Новые ядра, как правило, и быстрее и безопаснее.

 trider 7 сентября 2017 в 15:30

↑ 0 ↓

Я так понял в статье не раскрыта конфигурация Consul, которую требуется произвести перед запуском Patroni. Я не имел дел с Consul и etcd и не могу сориентироваться какие телодвижения требуется произвести с Consul

 citius 7 сентября 2017 в 15:48

↑ 0 ↓

В простейшем приближении никаких, демон консула просто запускается где удобно, и с ним можно сразу работать от имени клиентов.

Если нужна отказоустойчивость на его уровне, то есть кластеризация и т.д.


Советую почитать статьи по консулу, их много.

С etcd примерно также, ничего сложного там нет.

 trider 7 сентября 2017 в 16:33

↑ 0 ↓


С consul'ом я так понял нужен не только пионовский модуль

 trider 7 сентября 2017 в 16:37 #

↑ 0 ↓

С consul'ом я так понял нужен не только питоновский модуль python-consul, но и Consul server www.consul.io/downloads.html и я думал patroni с запуском и конфигурацией consul или etcd сам разберётся. Если использую Consul, ругается что не может подключиться к my_internal_ip:8500, если etcd, то говорит:

EtcdKeyNotFound: Key not found: /service/my-db-cluster/leader

 trider 7 сентября 2017 в 18:49 #

↑ 0 ↓

Подскажите что мне делать с Консулом:

```
Traceback (most recent call last):
  File "/usr/lib/python2.7/site-packages/patroni/dcs/consul.py", line 154, in refresh_session
    return self.retry(self._do_refresh_session)
  File "/usr/lib/python2.7/site-packages/patroni/dcs/consul.py", line 116, in retry
    return self._retry.copy()(*args, **kwargs)
  File "/usr/lib/python2.7/site-packages/patroni/utils.py", line 269, in __call__
    raise RetryFailedError("Exceeded retry deadline")
RetryFailedError: 'Exceeded retry deadline'
2017-09-07 18:47:05,073 INFO: waiting on consul
2017-09-07 18:47:20,057 ERROR: refresh_session
```

 trider 11 сентября 2017 в 11:50 #

↑ 0 ↓

Запустить Patroni я так и не смог, что я только не делал с Consul'ом, убил несколько дней, но Patroni кричал:

```
INFO: waiting on consul
```

Поэтому решение с Patroni достаточно мутное, хотите нормальный PostgreSQL кластер не лепите велосипед, надо брать Postgres Pro Enterprise.

 CyberDem0n 11 сентября 2017 в 15:04 #

↑ 0 ↓

@trider

Судя по логам очевидно что Patroni не может подключиться к Consul.
Покажи конфиг Patroni.

 trider 11 сентября 2017 в 15:15 #

↑ 0 ↓

```
# cat /etc/patroni/postgres.yml
name: db01
scope: &scope db

consul:
  host: 127.0.0.1:8500

restapi:
  listen: 0.0.0.0:8080
  connect_address: 172.16.128.70:8080
  auth: 'username:test'

bootstrap:
  dcs:
    ttl: &ttl 30
    loop_wait: &loop_wait 10
    maximum_lag_on_failover: 1048576 # 1 megabyte in bytes
  postgresql:
    use_pg_rewind: true
    use_slots: true
    parameters:
      archive_mode: "on"
      wal_level: hot_standby
      archive_command: mkdir -p ../wal_archive && cp %p ../wal_archive/%f
      max_wal_senders: 10
      wal_keep_segments: 8
      archive_timeout: 1800s
      max_replication_slots: 5
      hot_standby: "on"
      wal_log_hints: "on"
```

```
pg_hba: # Add following lines to pg_hba.conf after running 'initdb'
- host replication replicator 172.16.0.0/12 md5
- host all all 0.0.0.0/0 md5

postgresql:
  listen: 0.0.0.0:5432
  connect_address: 172.16.128.70:5432
  data_dir: /var/lib/pgsql/9.6/data
  pg_rewind:
    username: superuser
    password: test
  pg_hba:
    - host all all 0.0.0.0/0 md5
    - hostssl all all 0.0.0.0/0 md5
  replication:
    username: replicator
    password: test
    network: 172.16.0.0/12
  superuser:
    username: superuser
    password: test
  admin:
    username: admin
    password: test
  restore: /usr/bin/patroni_wale_restore
```

```
# netstat -nap | grep consul
tcp        0      0 127.0.0.1:8400      0.0.0.0:*           LISTEN      2737/consul
tcp        0      0 127.0.0.1:8500      0.0.0.0:*           LISTEN      2737/consul
tcp        0      0 127.0.0.1:8600      0.0.0.0:*           LISTEN      2737/consul
tcp6       0      0 :::8300             :::*                LISTEN      2737/consul
tcp6       0      0 :::8301             :::*                LISTEN      2737/consul
tcp6       0      0 :::8302             :::*                LISTEN      2737/consul
udp        0      0 127.0.0.1:8600      0.0.0.0:*           2737/consul
udp6       0      0 :::8301             :::*                2737/consul
udp6       0      0 :::8302             :::*                2737/consul
unix  3      [ ]          STREAM  CONNECTED  83481      2737/consul
```

Вот что journalctl говорит по поводу consul:

```
Sep 11 15:09:23 db01.localdomain consul[2737]: 2017/09/11 15:09:23 [ERR] agent: failed to sync remote state:
No cluster leader
Sep 11 15:09:27 db01.localdomain consul[2737]: 2017/09/11 15:09:27 [ERR] agent: coordinate update error: No
cluster leader
Sep 11 15:09:44 db01.localdomain consul[2737]: 2017/09/11 15:09:44 [ERR] agent: coordinate update error: No
cluster leader
Sep 11 15:09:52 db01.localdomain consul[2737]: 2017/09/11 15:09:52 [ERR] agent: failed to sync remote state:
No cluster leader
```

```
# consul members
Node           Address           Status Type    Build Protocol DC
db01.localdomain 172.16.128.70:8301 alive  server  0.6.4  2      dc1
```

Мне всё-таки очень интересно запустить этот «автомат» master-slave.

У меня такое ощущение судя по либам patroni, что он сам должен был с consul'ом разобраться

 **CyberDem0n** 11 сентября 2017 в 15:59    

 0 

Проблема не в Patroni, а в Consul, он конечно запущен (процесс живой) и даже порт слушает, но при этом неконсистентен и Patroni не может в него ничего записать ни прочитать из него.

К сожалению с кластеризацией Consul я вряд-ли смогу помочь.

 **CyberDem0n** 11 сентября 2017 в 16:05    

 0 

Рекомендую на счёт Consul почитать: <https://www.consul.io/docs/guides/bootstrapping.html> и <https://www.consul.io/intro/getting-started/join.html>

P.S. обычно выбирают тот DCS кластер которого уже настроен и работает.

 **trider** 11 сентября 2017 в 18:21     ↑ 0 ↓

Вот по такому шаблону eas.me/consul можно сконфигурировать Consul под Postgre для последующей интеграции patroni?

 **CyberDem0n** 11 сентября 2017 в 20:30     ↑ 0 ↓

Думаю что да, но есть несколько тонкостей:

1. Во первых надо запустить Consul кластер на 3 хостах (иначе не будет HA)
2. Consul agent должен работать на всех машинах где планируется запускать Patroni + Postgres. При этом этот агент не обязательно должен участвовать в кворуме.
3. Patroni использует Consul исключительно как KV Store.

Может лучше попробовать etcd? Там кластеризация в 100 раз проще:

<https://github.com/coreos/etcd/blob/master/Documentation/op-guide/clustering.md#static>

Если планируется запускать больше двух нод с Patroni+Postgres, то можно попробовать

<https://github.com/zalando/patroni/pull/375>, он не требует внешнего DCS

 **trider** 12 сентября 2017 в 10:18     ↑ 0 ↓

Да, я планирую запустить 2е ноды master-slave

 **trider** 19 сентября 2017 в 11:35     ↑ 0 ↓

Не подскажете какой DCS я могу использоваться для организации failover'a master-slave из 2х нод?

 **citius** 19 сентября 2017 в 12:42     ↑ 0 ↓


Любой из поддерживаемых patroni.

 **trider** 20 сентября 2017 в 10:52     ↑ 0 ↓

Но для работы Consul минимум 3 ноды, на 2х нодах etcd не заводится тоже пока

 **citius** 21 сентября 2017 в 18:43     ↑ 0 ↓

Прямо в репе патрони на гитхабе в ридми есть пример как на локалхосте запустить демон etcd и два инстанса патрони.


 **trider** 2 октября 2017 в 16:26     ↑ 0 ↓

Да не будет это работать, если просто по дефолту установить и запустить etcd, его нужно конфигурировать, иначе patroni выдаст:

```
EtdcKeyNotFound: Key not found : /service/postgre_cluster/leader
```

 **Seboreia** 21 сентября 2017 в 23:07     ↑ 0 ↓

Если вы хотите настоящий HA-кластер, то вам в любом случае понадобятся 3 ноды, т.к. у etcd кворумная кластеризация (т.е. для выбора нового мастера необходимо N/2+1 живых нод)

 **CyberDem0n**  11 сентября 2017 в 15:02   ↑ 0 ↓

Только полноправные пользователи могут оставлять комментарии. Войдите, пожалуйста.

Сутки

Неделя

Месяц

Узнаем паспортные данные физического лица по ФИО (если есть залоговое имущество)

↑ +76 👁 31,2k 📖 153 💬 61

Людмила Бокова предлагает деанонимизировать всех пользователей Рунета

↑ +23 👁 22,8k 📖 15 💬 216

Представлен моноблок «Эльбрус 801М» в «новом современном дизайне»

↑ +26 👁 19,9k 📖 6 💬 171

Хакеры воруют и отмывают деньги через сервисы доставки еды и бронирования отелей

↑ +36 👁 20k 📖 54 💬 67

Зарабатываем на солнечной энергии или пассивный доход в 25% годовых, практический опыт

↑ +69 👁 38,5k 📖 151 💬 218

Ваш аккаунт	Разделы	Информация	Услуги
Войти	Публикации	Правила	Реклама
Регистрация	Новости	Помощь	Тарифы
	Хабы	Документация	Контент
	Компании	Соглашение	Семинары
	Пользователи	Конфиденциальность	
	Песочница		

Если нашли опечатку в посте, выделите ее и нажмите Ctrl+Enter, чтобы сообщить автору.

