УДК 004.056

Пестов Игорь Евгеньевич кандидат технических наук, доцент
Санкт-Петербургский государственный университет телекоммуникаций имени профессора М. А. Бонч-Бруевича Россия, г. Санкт-Петербург
Жвакин Алексей Владимирович студент
5 курс, Санкт-Петербургский государственный университет телекоммуникаций имени профессора М. А. Бонч-Бруевича
Россия, г. Санкт-Петербург
Чезлов Кирилл Сергеевич студент
5 курс, Санкт-Петербургский государственный университет телекоммуникаций имени профессора М. А. Бонч-Бруевича
Россия, г. Санкт-Петербург

ПОСТРОЕНИЕ ОТКАЗОУСТОЙЧИВОГО КЛАСТЕРА СУБД POSTGRESQL НА ОСНОВЕ PATRONI И ETCD

Аннотация: *Статья посвящена построению отказоустойчивого кластера СУБД PostgreSQL с использованием инструментов Patroni и etcd. Рассматривается архитектура кластера, обеспечивающая автоматическое переключение ролей между узлами и минимизацию времени простоя. Подробно описаны этапы развертывания и настройки компонентов, включая установку PostgreSQL, настройку распределенного хранилища etcd и фреймворка управления Patroni. Представлены конфигурационные параметры сервисов и приведены примеры их реализации.*

Ключевые слова: *отказоустойчивость, кластер, СУБД PostgreSQL, Patroni, etcd, автоматическое переключение, высокая доступность.*

Annotation: *The article is devoted to building a fault-tolerant PostgreSQL DBMS cluster using Patroni and etcd tools. The cluster architecture is considered, which provides automatic role switching between nodes and minimizes downtime. The stages of deployment and configuration of components are described in detail, including the installation of PostgreSQL, configuration of the distributed storage*

etcd, and the management framework Patroni. Configuration parameters of the services are presented with examples of their implementation.

Key words: *fault tolerance, cluster, PostgreSQL DBMS, Patroni, etcd, automatic failover, high availability.*

Введение*. Современные информационные системы требуют высокой доступности и отказоустойчивости, особенно когда речь идет о базах данных, являющимися критически важным компонентом любой инфраструктуры. Кластеризация СУБД PostgreSQL с использованием таких инструментов, как Patroni и etcd, позволяет обеспечить автоматическое переключение ролей между узлами. В данной статье рассматривается процесс развертывания отказоустойчивого кластера PostgreSQL на основе Patroni и etcd с пояснениями конфигураций компонентов, настройки сервисов кластера. Актуальность темы обусловлена растущими требованиями к бесперебойной работе баз данных в условиях высокой нагрузки и возможных сбоев.*

Архитектура кластера СУБД. *Отказоустойчивость кластера базы данных — способность продолжать работу при сбоях отдельных узлов, обеспечивая минимальное время простоя и сохранность данных [4, 6, 7]. В современных распределенных системах для достижения этой цели используются различные специализированные инструменты, такие как Patroni и etcd.*

Patroni — это фреймворк для управления кластерами PostgreSQL, который автоматизирует процессы выбора лидера, переключения ролей и мониторинга состояния узлов. Он взаимодействует с распределенными системами хранения конфигурации (DCS), такими как etcd, ZooKeeper или Consul, для координации работы узлов. Patroni обеспечивает:

- *Автоматическое восстановление после сбоев;*
- *Динамическое управление конфигурацией PostgreSQL;*
- *Интеграцию с инструментами мониторинга и балансировки нагрузки.*

Etcd — это распределенное хранилище ключ-значение, используемое для хранения конфигурации кластера и состояния узлов. Оно обеспечивает:

- *Высокую доступность за счет репликации данных между узлами;*
- *Консистентность данных благодаря алгоритму консенсуса Raft;*
- *Быстрое обнаружение сбоев и обновление конфигурации.*

Принципы работы кластера:

1. *Репликация данных: PostgreSQL использует потоковую репликацию для синхронизации данных между master и replica узлами;*
2. *Функция автоматического восстановления: при сбое ведущий узел Patroni инициирует выбор нового лидера среди реплик, минимизируя время простоя;*
3. *Распределенное управление: Etcd хранит информацию о состоянии узлов, что позволяет кластеру принимать согласованные решения даже в условиях сетевых сбоев.*

Развертывание кластера СУБД. *Установим пакеты PostgreSQL, создадим директории для данных компонентов кластера и изменим владельца на postgres, чтобы запущенные от его имени сервисы имели права на созданные директории и файлы [1, 5] (рис. 1 2).*

```
root@astra-claster1:~# apt install postgresql postgresql-contrib bind9
Чтение списков пакетов… Готово
Построение дерева зависимостей
Чтение информации о состоянии… Готово
Уже установлен пакет bind9 самой новой версии (1:9.11.5.P4+dfsg-5.1+deb10u10+ci202404131956+astra2).
Будут установлены следующие дополнительные пакеты:
  libllvm11 libz3-4 postgresql-11 postgresql-client-11 postgresql-client-common postgresql-common sysstat
Предлагаемые пакеты:
  postgresql-doc postgresql-doc-11 libjson-perl isag
Следующие НОВЫЕ пакеты будут установлены:
  libllvm11 libz3-4 postgresql postgresql-11 postgresql-client-11 postgresql-client-common postgresql-common postgresql-contrib sysstat
Обновлено 0 пакетов, установлено 9 новых пакетов, для удаления отмечено 0 пакетов, и 0 пакетов не обновлено.
Необходимо скачать 42,2 MB архивов.
После данной операции объём занятого дискового пространства возрастёт на 165 MB.
Хотите продолжить? [Д/н]
```

Рисунок 1. Установка PostgreSQL

```
root@astra-claster1:~# mkdir -p /var/lib/pgsql/etcd && chown -R postgres:postgres /var/lib/pgsql
root@astra-claster1:~#
```

Рисунок 2. Создание директории данных

Скачаем из открытых репозиториев архив etcd, распакуем и перенесем исполняемые файлы в директорию «/bin» [3]. После этого напишем сервис systemd для того, чтобы системна могла управлять этим приложением (рис. 3). В разделе «Unit» впишем описание сервиса и укажем «network.target» в

атрибуте «After», чтобы сервис запускался только после того, как будет запущена сетевая подсистема. В разделе «Service» укажем пользователя, от имени которого будет запущен сервис, атрибут «Type» со значением «notify», чтобы etcd уведомлял systemd о завершении запуска, команду запуска с указанием конфигурационного файла, автоматический перезапуск с паузой 10 секунд, увеличим лимит открытых файловых дескрипторов. В разделе «Install» укажем режим «multi-user» для запуска сервиса.

```
[Unit]
Description= etcd service
After=network.target


[Service]
User=postgres
Type=notify
ExecStart=/bin/etcd --config-file /etc/etcd.conf
Restart=always
RestartSec=10s
LimitNOFILE=40000


[Install]
WantedBy=multi-user.target
```

Рисунок 3. Сервис etcd

Теперь создадим конфигурацию службы etcd и запустим ее (рис. 4). Основные параметры etcd:

1. *Общие настройки:*

a. *«name: astra-claster1» - имя текущего узла в кластере;*

b. *«embblo-v2: true» - включает поддержку API v2;*

c. *«data-dir: /var/lib/opsql/etcd» - директория для хранения данных etcd, созданная ранее;*

2. *Настройки кластера:*

a. *«initial-cluster-token: cluster1» - уникальный идентификатор кластера;*

b. *initial-cluster - список всех узлов в кластере с их адресами:*

i. *astra-claster1=http://10.10.110.73:2380;*

ii. *astra-claster2=http://10.10.110.74:2380;*

iii. *test-ankeykud=http://10.10.110.76:2380;*

iv. ankey=http://10.10.110.75:2380;

3. Сетевые настройки:

a. «initial-advertise-peer-urls: http://10.10.110.73:2380» - URL для общения между узлами кластера (peer communication);

b. «listen-peer-urls: http://10.10.110.73:2380» - адрес и порт для прослушивания peer-соединений;

c. «listen-client-urls:http://10.10.110.73:2379,http://localhost:2379» - адреса для клиентских подключений;

d. «advertise-client-urls: http://10.10.110.73:2379» - URL, который сообщается клиентам для подключения.

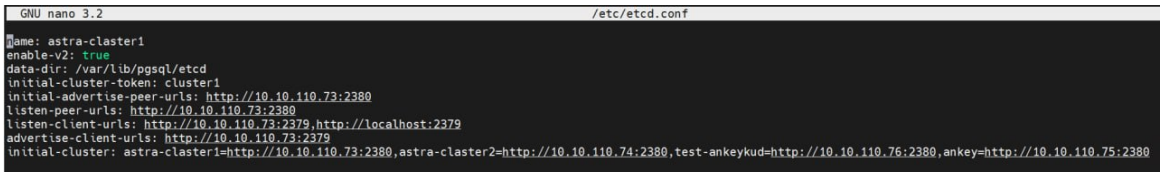

Рисунок 4. Конфигурация службы etcd

Далее нужно установить Patroni, для этого нужно установить пакеты «python3», «gcc», «libpq-dev» и модули Python «setuptools», «psycopg2-binary», «patroni[etcd]» [2].

Конфигурация Patroni содержит параметры для настройки взаимодействия с etcd (рис. 5) и PostgreSQL (рис. 6). Также необходимо указать параметры конфигурации СУБД (рис. 7).

Основные разделы конфигурации Patroni:

1. Общие настройки кластера:

a. «scope: astra-pg-cluster» - имя кластера PostgreSQL;

b. «name: astra-claster1» - имя текущего узла в кластере;

2. Настройки REST API:

a. «listen: 10.10.110.73:8008» - IP-адрес и порт для прослушивания REST API;

b. «connect_address: 10.10.110.73:8008» - адрес, по которому другие узлы могут подключиться к REST API;

3. Настройки etcd:

a. «host: 127.0.0.1:2379» - адрес для подключения к etcd;

4. Настройки кластера DCS (distributed control system), в качестве которого мы используем etcd:

a. «ttl: 30» - время жизни лидера в секундах;

b. «loop_wait: 10» - интервал проверки состояния в секундах;

c. «retry_timeout: 10» - таймаут перед повторной попыткой выполнения операции;

d. «maximum_lag_on_failover: 1048576» - максимальное отставание реплики для автоматического восстановления в байтах.

5. Конфигурация «pg_hba» - строки правил доступа;

6. Пользователи PostgreSQL – определение пользователей PostgreSQL;

7. Сервис PostgreSQL:

a. «data_dir: /var/lib/jatoba/5/data» - путь к директории данных;

b. «bin_dir: /usr/jatoba-5/bin/» - путь к исходным файлам;

8. Конфигурация «postgresql» - настройки СУБД, с учетом репликации.

```
scope: astra-pg-cluster
name: astra-claster1

restapi:
  listen: 10.10.110.73:8008
  connect_address: 10.10.110.73:8008

etcd:
  host: 127.0.0.1:2379

bootstrap:
  dcs:
    ttl: 30
    loop_wait: 10
    retry_timeout: 10
    maximum_lag_on_failover: 1048576
```

Рисунок 5. Конфигурация Patroni для взаимодействия с etcd

Рисунок 6. Конфигурация Patroni для взаимодействия с PostgreSQL

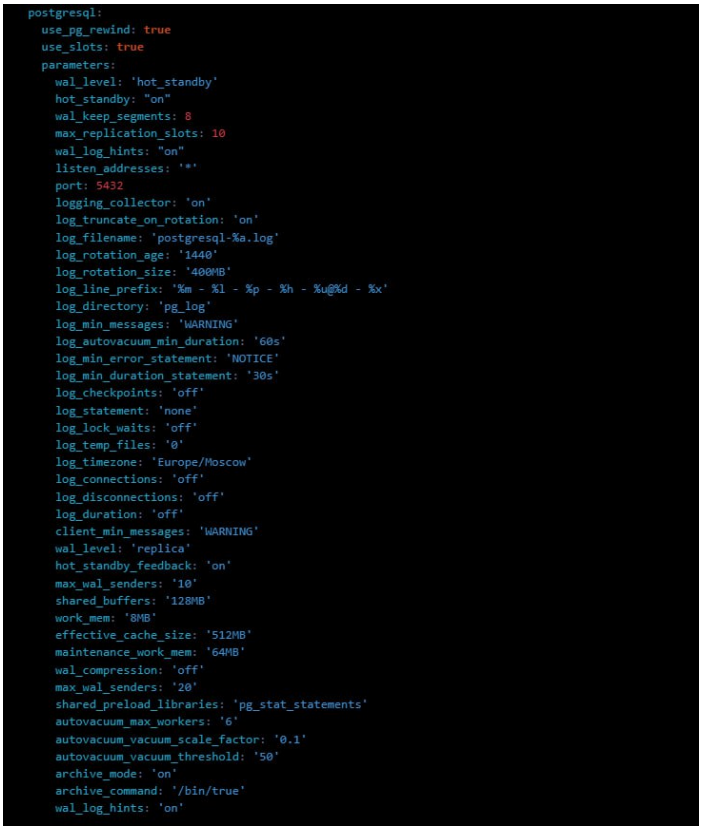

Рисунок 7. Параметры конфигурации PostgreSQL

Напишем сервис Patroni аналогично сервисы etcd и запустим его (рис. 8).

```
[Unit]
Description=Runners to orchestrate a high-availability PostgreSQL
After=etcd.service syslog.target network.target


[Service]
Type=simple
User=postgres
Group=postgres
ExecStart=/usr/local/bin/patroni /etc/patroni/patroni.yml
ExecReload=/bin/kill -s HUP $MAINPID
KillMode=process
TimeoutSec=30
Restart=no


[Install]
WantedBy=multi-user.target
```

Рисунок 8. Сервис Patroni

Повторим вышеперечисленные действия на других серверах, изменяя необходимые части конфигураций. В результате получим работающий кластер СУБД PostgreSQL под управлением Patroni. Посмотрим состояние кластера и убедимся в этом (рис. 9). По полученному отчету видим, что все экземпляры СУБД активны и распределили роли между собой, выбрав в качестве лидера сервер «astra-claster1».

```
root@astra-claster1:/etc/patroni# patronictl -c /etc/patroni/patroni.yml list
+ Cluster: astra-pg-cluster (7460117583040562467) ----+----+-----------+
| Member         | Host         | Role    | State     | TL | Lag in MB |
+----------------+--------------+---------+-----------+----+-----------+
| ankey          | 10.10.110.75 | Replica | streaming |  9 |         0 |
| astra-claster1 | 10.10.110.73 | Leader  | running   |  9 |           |
| astra-claster2 | 10.10.110.74 | Replica | streaming |  9 |         0 |
| test-ankeykud  | 10.10.110.76 | Replica | streaming |  9 |         0 |
+----------------+--------------+---------+-----------+----+-----------+
```

Рисунок 9. Просмотр состояния кластера СУБД

Заключение. *В ходе работы был успешно развернут отказоустойчивый кластер PostgreSQL с использованием Patroni и etcd. Настроены сервисы для управления узлами, обеспечено автоматическое переключение ролей в случае сбоев, а также проведена проверка работоспособности кластера. Выбранное решение позволяет достичь высокой доступности и надежности базы данных, что особенно важно для критически важных систем. Дальнейшее развитие проекта может включать оптимизацию параметров репликации, интеграцию с инструментами мониторинга и тестирование кластера в условиях*

повышенной нагрузки. Применение подобных технологий открывает новые возможности для построения устойчивых и масштабируемых инфраструктур.

Использованные источники:

1. PostgreSQL Global Development Group. PostgreSQL: официальная документация [Электронный ресурс]. URL: https://www.postgresql.org/docs/ (дата обращения: 12.05.2025).

2. Patroni: PostgreSQL HA with ZooKeeper, etcd, or Consul [Электронный ресурс]. URL: https://patroni.readthedocs.io/ (дата обращения: 12.05.2025).

3. etcd Documentation [Электронный ресурс]. URL: https://etcd.io/docs/ (дата обращения: 12.05.2025).

4. Метод передачи метрик загруженности инстансов облачной инфраструктуры в кластер обработки средствами и методами больших данных для защиты информации и обеспечения информационной безопасности / И. Е. Пестов, П. О. Федоров, С. А. Кошелева, Р. В. Алехин // I-methods. – 2022. – Т. 14, № 1 – 14 с.

5. Завадский В. П., Новиков С. В. Администрирование PostgreSQL. Практическое руководство. / Завадский В. П., Новиков С. В. // ДМК Пресс – 2022. – 416 с.

6. Глушаков И. А. Высоконагруженные приложения: архитектура, управление, эксплуатация. / Глушаков И. А. // СПб.: Питер – 2021. – 368 с.

7. Клюев В. В. Надежность и отказоустойчивость информационных систем. / Клюев В. В. // Горячая линия – Телеком – 2020. – 288 с.