Cook

Выпуск 0.6.4

Artyom Smirnov <artyom.smirnov@red-soft.ru>

окт. 10, 2024

Содержание:

1 О Cook **1**

2 Быстрый старт **3**
2.1 Подготовка к работе . . . 3
2.2 Запуск . . . 4

3 Инсталляция **7**
3.1 Инсталляция из исходников . . . 7
3.2 Инсталляция из бинарного пакета . . . 8

4 Конфигурация **9**
4.1 Файл конфигурации . . . 9
4.2 Использование SSL/TLS сертификатов . . . 15
4.3 Fencing . . . 18
4.4 Пример конфигурации сервиса systemd . . . 19
4.5 Использование реверсивных прокси . . . 19

5 Типовые конфигурации отказоустойчивого кластера и покрываемые ими угрозы **21**
5.1 Общий вид кластера и его составляющие . . . 21
5.2 Конфигурации кластера . . . 25

6 Обновление СУБД Ред База Данных в кластере **35**
6.1 Общий порядок обновления . . . 35
6.2 Процесс обновления Ред Базы Данных на узле . . . 37
6.3 Процесс отката Ред Базы Данных на узле . . . 37

7 Утилиты **39**
7.1 cookd . . . 39
7.2 cookctl . . . 39
7.3 cookdsvc.exe (только Windows) . . . 49

8 Режим обслуживания **51**
8.1 О режиме обслуживания . . . 51
8.2 Пример использования режима обслуживания . . . 53

9 Образ Docker **55**

10 REST API **57**

10.1 Внешнее REST API 57
10.2 Внутреннее REST API 58

11 Ограничения **61**

12 История изменений **63**
12.1 RU 63
12.2 EN 66

Глава 1

О Cook

Cook - это сервис построения отказоустойчивого кластера на основе Ред Базы Данных. В основе cook лежит кроссплатформенный движок написанный на Python. Узел cook берет на себя всю работу о настройке СУБД для репликации: создание конфигурационных файлов Ред Базы Данных, настройка репликации, остановка и запуск сервера, создание базы данных, перемещение архивов репликации, инициализация новых реплик и так далее.

Конфигурация кластера хранится в распределенной Key-Value базе данных Consul. Также Consul используется в качестве менеджера распределенных блокировок, для принятия решений, требующих согласия нескольких узлов, например инициализация кластера, выбор нового master, управление TTL.

При запуске cook читает локальную конфигурацию, подключается к Consul и проверяет существующую конфигурацию кластера. Если ее нет, переходит в режим инициализации и создает новый кластер и базу данных. Создав или загрузив конфигурацию, cook проверяет возможность перевода репликации в режим master, для свежесозданного кластера - это происходит автоматически, а при подключении к существующему кластеру, возможность перехода в master оценивается на основе текущей позиция данных в локальной БД, сохраненной позиции в конфигурации кластера и позиции данных БД остальных узлов кластера.

Узел может быть не в состоянии создать новый, кластер, например, если нет начальной конфигурации или возможность запрещена для данного узла. В этом случае cook ждет, когда любой другой узел создаст кластер.

При невозможности перехода в режим master, наличие другого master-узла в кластере, дает возможность остальным узлам начать работу, как slave-узлы в режиме асинхронной репликации. При этом оценивается позиция данных в локальной БД, если это возможно - скачиваются и применяются недостающие архивы репликации, и узел переходит в режим slave, постоянно получая актуальные архивы с текущего master. master-узлы также используются для инициализации новых slave-узлов.

Если slave-узел замечает, что в кластере больше нет master-узла, то выполняется попытка перевода в режим master, описанная ранее.

Consul выступает не только, как хранилище состояния кластера, но и как инструмент обнаружения сервисов. Узлы анонсируют свое состояние в Consul, и другие сервисы могут узнать, как к ним подключаться, например проксируя пишущие запросы к master-узлу или распределяя читающие запросы к slave-узлам.

Глава 2

Быстрый старт

В данном разделе кратко описывается запуск кластера из двух узлов Cook.

Предупреждение: Cook сам управляет процессом RedDatabase, другими словами rdbserver - дочерний процесс cookd. Важно, чтобы любые init-скрипты/сервисы/юниты/службы были отключены, а еще лучше - удалены, во избежание запуска rdbserver без управления cookd.

Предупреждение: Данное руководство не описывает установку в production.

2.1 Подготовка к работе

Для начала необходимо:

1. Подготовить две физические или виртуальные машины, например с IP 192.168.0.1 и 192.168.0.2
2. Установить **Cook** любым из способов (см. *Инсталляция из бинарного пакета* и *Инсталляция из исходников*)
3. Установить **RedDatabase** версии не ниже 3.0.8
4. Установить **Consul** версии не ниже 1.8
5. Настроить общий ресурс для передачи логов или архивов (например SMB или NFS). Данный ресурс должен гарантировать fsync, например быть смонтирован, как WRITETHROUGH (smb)

2.2 Запуск

Запустим кластер consul из двух узлов:

На **192.168.0.1**:

```
consul agent -server -bootstrap-expect 2 -ui -data-dir=data -client 0.0.0.0 -retry-
↪join=192.168.0.2
```

На **192.168.0.2**:

```
consul agent -server -data-dir=data -client 0.0.0.0 -retry-join=192.168.0.1
```

Создадим конфигурацию первого узла кластера в файле **cookd.yml** на 192.168.0.1:

```
cluster_name: testcluster
node_name: node1

cook:
    listen: 0.0.0.0:5030
    advertise: 192.168.0.1:5030

rdb:
    listen: 0.0.0.0:3050
    advertise: 192.168.0.1:3050
    pidfile: /opt/RedDatabase/rdb.pid
    home: /opt/RedDatabase
    password: masterkey
    replication_dir: /shared_dir/replication
    slave_log_directory: /local_dir/slave_logs

init:
    rdb:
        databases:
            - database: /data/database.fdb
              alias: mydatabase
```

Здесь **/shared_dir/replication** - общий ресурс, а **/local_dir/slave_logs** - локальный каталог.

В **init** описана начальная кластерная конфигурация. В данном случае указана только база данных и ее alias. В дальнейшем ее можно закомментировать или убрать.

Запустим первый узел:

```
cookd -c cookd.yml
```

После того как кластер и узел инициализируются в логе будет следующее сообщение:

> [node1] INFO:2022-04-21 11:19:57,666 - cook.mind:437 - Processing locked cluster as master. Lock owned by node1

Создадим конфигурацию **cookd.yml** следующего узла **node2** на **192.168.0.2**:

```
cluster_name: testcluster
node_name: node2

```

(continues on next page)

(продолжение с предыдущей страницы)

```
cook:
    listen: 0.0.0.0:5030
    advertise: 192.168.0.2:5030
    loggers:
        cook: INFO

rdb:
    listen: 0.0.0.0:3050
    advertise: 192.168.0.2:3050
    pidfile: /opt/RedDatabase/rdb.pid
    home: /opt/RedDatabase/rdb
    password: masterkey
    replication_dir: /shared_dir/replication
    slave_log_directory: /local_dir/slave_logs

init:
    slave_database_mapping:
      mydatabase: /local_dir/data/database.fdb
```

Здесь локальная конфигурация практически идентичная, но в **init** описана локальное расположение БД для инициализации slave. В дальнейшем секцию **init** можно закомментировать или убрать.

Также начиная с *v0.4.0* можно использовать регулярные выражения, для сопоставления базы данных в кластере с локальным файлом на узле. Псевдоним, для интерпретации, как регулярного выражения при этом обрамляется в /<выражение>/. Результаты группы захвата, можно использовать в качестве имени базы, используя синтаксис str.format() Python 3

```
init:
    slave_database_mapping:
      /db-(\d+)/: /databases/db/db-{0}.fdb
```

Запустим второй узел:

```
cookd -c cookd.yml
```

После того как узел подключится к кластеру и инициализируются в логе будет следующее сообщение:

> [node2] INFO:2022-04-21 11:23:57,127 - cook.mind:437 - Processing locked cluster as slave. Lock owned by node1

После этого архивы репликации, созданные на node1, по мере поступления будут применяться на node2.

Статус кластера можно посмотреть с помощью cookctl:

```
cookctl -c node1.yml status

testcluster (9682539a-5a36-4828-ac32-2e0b88ece6e1, 0.2.0) generation 1 (3e033d00-5aae-
↪4bd2-bfc5-eaf0e388dd7e), locked by node1

node1 (master) API URL http://192.168.0.1:5030, RDB 3.0.8.0 on 3050 and AUX 3060 (192.
↪168.0.1/3050), failover enabled
    /local_dir/data/database.fdb (mydatabase, 68825f33-f8ff-4cf0-94c1-023bd5d1ac8a) 1:1␣
↪192.168.0.1/3050:mydatabase
node2 (slave) API URL http://192.168.0.2:5031, RDB 3.0.8.0 on 3051 and AUX 3060 (192.168.
```

(continues on next page)

(продолжение с предыдущей страницы)

```
↪0.2/3051), failover enabled
   /local_dir/data/database.fdb (mydatabase, 68825f33-f8ff-4cf0-94c1-023bd5d1ac8a) 1:1␣
↪192.168.0.2/3051:mydatabase
```

Глава 3

Инсталляция

3.1 Инсталляция из исходников

Перед началом установки, в системе должны быть установлены следующие пакеты:

- **python3** >= 3.8
- **poetry** >= 1.3
- **python3-devel** >= 3.8 (в Linux)
- **gcc** (в Linux)

Скачиваем или разархивируем исходники:

```
git clone http://git.red-soft.biz/red-database/cook.git
```

Устанавливаем Cook:

```
cd cook
pip3 install poetry==1.3
poetry build
pip3 install dist/cook-0.6.4-py3-none-any.whl
```

Утилиты при этом будут установлены в /usr/local/bin.

3.2 Инсталляция из бинарного пакета

Перед началом установки, в системе должны быть установлены следующие пакеты:

- **python3** >= 3.8
- **python3-devel** >= 3.8 (в Linux)
- **gcc** (в Linux)

При установке cook из пакета wheel зависимости установятся автоматически:

```
python3 -m pip install -U path/to/cook-0.6.4-py3-none-any.whl
```

Дистрибутив для Windows самодостаточен и содержат в себе Python и все зависимости, поэтому его достаточно разархивировать в удобную директорию.

Глава 4

Конфигурация

4.1 Файл конфигурации

Конфигурация сервиса cookd и утилиты cookctl описывается в формате yaml:

```
# Имя кластера (обязательное значение)
cluster_name: testcluster

# Уникальное имя узла (обязательное значение)
node_name: node0

# Локальные опции cook (необязательная секция)
#cook:
    # Адрес и порт API которые слушает cookd (по-умолчанию 0.0.0.0:5030)
    #listen: 0.0.0.0:5030

    # Адрес и порт API к которым предлагает подключиться cookd (по-умолчанию␣
↪определяется автоматически)
    # Если машина имеет несколько IP данный параметр должен быть указан
    #advertise: 192.168.0.1:5030

    # Проверять доступность порта API перед запуском (по-умолчанию True)
    #check_port: True

    # Пользователь API (необязательное значение)
    #user: cook

    # Пароль API (необязательное значение)
    #password: cook

    # Использовать SSL или нет для коммуникации между узлами cook (по-умолчанию False)
    #ssl: False
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    # Использовать определенные алгоритмы шифрования TLS. (по-умолчанию системные)
    # Возможные значения и значения по-умолчанию зависят от системы и версии Python.
    # Приведены значения по-умолчанию для Python 3.12
    #ciphers:
    # - TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
    # - TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256
    # - TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
    # - TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
    # - TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
    # - TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
    # - TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
    # - TLS_RSA_WITH_AES_256_GCM_SHA384
    # - TLS_RSA_WITH_AES_128_GCM_SHA256
    # - TLS_RSA_WITH_AES_256_CBC_SHA256
    # - TLS_RSA_WITH_AES_128_CBC_SHA256
    # - TLS_RSA_WITH_AES_256_CBC_SHA
    # - TLS_RSA_WITH_AES_128_CBC_SHA

    # Путь к SSL сертификату (необязательное значение)
    #cert: /path/to/cert

    # Путь к ключу сертификата (необязательное значение)
    #key: /path/to/key

    # Проверять сертификат входящих подключений (для сервера API, по-умолчанию False)
    #verify_incoming: False

    # Проверять сертификат сервера (для клиента, по-умолчанию False)
    #verify_outgoing: False

    # Путь к корневому сертификату (необязательное значение)
    #cacert: /path/to/cacert

    # Путь к файлу лога, если пустое, то логировать в stdout (необязательное значение)
    #log_file: cookd.log

    # Настройка уровней логирования (по-умолчанию INFO)
    # Можно установить уровень логирования для любого логгера по его имени
    #loggers:
        #cook: WARNING
        #cook.api: DEBUG
        #urllib3: WARNING
        #werkzeug: WARNING

    # Использовать цветной лог (по-умолчанию True)
    #color_log: True

    # Изменить имя хоста для TLS (по-умолчанию значение из node_name)
    # server_name: node0

# Настройка подключения к Consul (необязательная секция)
```

(continues on next page)

(продолжение с предыдущей страницы)

```
#consul:
    # Адрес подключения к Consul (по-умолчанию http://localhost:8500)
    #url: http://localhost:8500

    # Проверять сертификат сервера (по-умолчанию False)
    #verify: False

    # Путь к корневому сертификату (необязательное значение)
    #cacert: /path/to/cacert

    # Токен API Consul (необязательное значение)
    #token: secret


# Локальные настройки RedDatabase (обязательная секция)
rdb:
    # Путь к каталогу с RedDatabase (по-умолчанию /opt/RedDatabase)
    #home: /opt/RedDatabase

    # Хост и порт на котором работает RedDatabase (по-умолчанию 0.0.0.0:3050)
    #listen: 0.0.0.0:3050

    # Хост и порт RedDatabase к которому предлагается подключиться (по-умолчанию␣
↪определяется автоматически)
    # Если машина имеет несколько IP данный параметр должен быть указан
    #advertise: 192.168.0.100:3050

    # Порт подключения клиентов к событиями RedDatabase (по-умолчанию 3060)
    #aux_port: 3060

    # Пароль для подключения к БД (обязательное значение)
    password: masterkey

    # PID-файл сервера (обязательное значение)
    pidfile: /path/to/rdbserver.pid

    # Общий ресурс (каталог) для хранения логов и архивов репликации (обязательное␣
↪значение)
    #
    # Внимание, данный ресурс должен гарантировать fsync, например быть смонтирован, как␣
↪WRITETHROUGH (smb)
    replication_dir: /path/to/replication_dir

    # Каталог для временного хранения логов на slave перед применением (обязательное␣
↪значение)
    slave_log_directory: /path/to/slave_log_directory

    # Опция архивирования логов master
    # Значение по-умолчанию для Windows: copy $(logpathname) $(archpathname)
    # Значение по-умолчанию для Linux: test ! -f $(archpathname) && cp $(logpathname)
↪$(archpathname).tmp && mv $(archpathname).tmp $(archpathname)
    #archive_command: test ! -f $(archpathname) && cp $(logpathname) $(archpathname).tmp␣
```

(continues on next page)

(продолжение с предыдущей страницы)

```
↪&& mv $(archpathname).tmp $(archpathname)

    # Печатать firebird.log и replication.log в логи cook при ошибках репликации (по-
↪умолчанию False)
    #dump_logs: False

    # Количество строк для dump_logs (по-умолчанию 50)
    #dump_lines: 50

# Локальные настройки fencing для master (необязательная секция)
#fencing:
    # Какой модуль fencing активировать
    #use: linux_watchdog

    # Настройки Linux Watchdog
    #linux_watchdog:
        # Устройство Watchdog (по-умолчанию /dev/watchdog)
        #device: /dev/watchdog

    # Настройки IPMIUtil
    #ipmi_watchdog:
        # Путь к ipmiutil (по-умолчанию ipmiutil)
        #ipmiutil_bin: ipmiutil

    # Пользовательский класс fencing
    #my.custom.fencing.Class:
    #    option_1: value_1
    #    option_2: value_2
    #    option_3: value_3


# Начальные настройки конфигурации кластера (необязательная секция)
# Узел с данной секцией инициализирует новый кластер
#init:
    # Настройки cook
    #cook:
        # Время жизни сессии в Consul (по-умолчанию 30)
        #ttl: 30

        # Таймаут для повторения провалившихся задач (по-умолчанию 10)
        #retry_timeout: 10

        # Максимальное время ожидания между циклами узла кластера (по-умолчанию 10)
        #loop_wait: 10

        # Максимальный лаг архивов slave (по-умолчанию 0)
        #max_lag: 0

        # Время ожидания slave перед окончательным завершением master (по-умолчанию 10)
        #master_wait_timeout: 10

        # Регистрировать сервис в Consul или нет (по-умолчанию True)
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        #register_service: True

        # Максимальное количество накопленных архивов (по-умолчанию 100)
        #max_archives: 100

        # Дополнительные опции узлов
        #hints:
            # Имя узла кластера
            #node_name:
                # Разрешить переключение этого узла в master
                #allow_failover: True

    # Настройки RedDatabase
    #rdb:
        # Список баз данных для инициализации кластера
        # databases:
        #     Путь к файлу базы данных
        #   - database: /path/to/database1.fdb
        #     Псевдоним базы данных
        #     alias: myalias
        #     Опции базы данных в databases.conf (имена опций соответствуют опциям в␣
↪RedDatabase)
        #     properties:
        #       #option: value
        #     Опции базы данных replication.conf (имена опций соответствуют опциям в␣
↪RedDatabase)
        #     replication_options:
        #       buffer_size: 1048576
        #       compress_records: False
        #       include_filter: table1
        #       exclude_filter: table2
        #       exclude_without_pk: False
        #       log_segment_size: 16777216
        #       log_segment_count: 8
        #       log_archive_timeout: 60
        #       log_group_flush_delay: 0
        #       log_warnings: False

        # Опции firebird.conf (имена опций соответствуют опциям в RedDatabase)
        #properties:
            #option: value

        # Глобальные опции replication.conf (имена опций соответствуют опциям в␣
↪RedDatabase)
        #replication_options:
            #buffer_size: 1048576
            #compress_records: False
            #include_filter: table1
            #exclude_filter: table2
            #exclude_without_pk: False
            #log_segment_size: 16777216
            #log_segment_count: 8
```

(continues on next page)

(продолжение с предыдущей страницы)

```
            #log_archive_timeout: 60
            #log_group_flush_delay: 0
            #log_warnings: False

        # Глобальные опции firebird.conf (имена опций соответствуют опциям в RedDatabase)
        #properties:
            #option: value

        # Глобальные опции replication.conf (имена опций соответствуют опциям в␣
↪RedDatabase)
        #replication_options:
            #buffer_size: 1048576
            #compress_records: False
            #include_filter: table1
            #exclude_filter: table2
            #exclude_without_pk: False
            #log_segment_size: 16777216
            #log_segment_count: 8
            #log_archive_timeout: 60
            #log_group_flush_delay: 0
            #log_warnings: False

    # Способ инициализации реплики (экспериментальная возможность, по-умолчанию nbackup)
    # nbackup - копирование базы данных с master через nbackup
    # as_is - использование уже существующей базы данных на slave
    # shell - использование команды для копирования, указанной в slave_init_command
    # disable - не инициализировать реплики
    #slave_init: nbackup

    # Команда для инициализации реплики
    #slave_init_command:

    # Сопоставление кластерной БД и файла на слейве для инициализации реплики
    #slave_database_mapping:
        # Можно задать как точный псевдоним базы в кластере и соответствующий локальный␣
↪путь
        #myalias: /path/to/database1.fdb

        # Так и использовать регулярные выражения
        #/myalias-(\d+)/: /path/to/database-{0}.fdb
        #/anotheralias-(\d+)/: /another/path/to/database-{0}.fdb

    # Анализировать firebird.log и replication.log на узлах на наличие критических␣
↪ошибок (по-умолчанию True)
    # При выявлении ошибки в логах узел будет остановлен
    # log_analyzer: True

    # Генерация конфигурации Gobetween в Consul (необязательная секция))
    #gobetween:
        # Разрешить генерацию (по-умолчанию True)
        #generate_config: True
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        # Имя сервера в конфигурации генерацию (по-умолчанию rdb)
        #server_name: rdb

        # URL Consul (по-умолчанию http://localhost:3050)
        #consul_host: http://localhost:3050

        # Хост и порт для прослушивания Gobetween (по-умолчанию 0.0.0.0:3050)
        #bind: 0.0.0.0:3050

        # Вспомогательный хост и порт для прослушивания Gobetween (по-умолчанию 0.0.0.
↪0:3060)
        #bind_aux: 0.0.0.0:3060

        # Интервал для опроса состояния в Consul (по-умолчанию 10)
        #interval: 10
```

4.2 Использование SSL/TLS сертификатов

По-умолчанию обмен данными между узлами Cook, сервисами Consul и Gotween производится по незащищенному протоколу HTTP. В зависимости от настройки каждого из сервисов можно использовать защищенный протокол, как с односторонней проверкой (верифицируется только сервер, к которому подключается клиент), так и с двусторонней (two-way TLS), где и сервер и клиент верифицируют друг друга.

Для корректно настройки TLS нужно получить сертификаты для каждого узла либо из доверенного сертификационного сервера, либо выработать свой корневой сертификат и сгенерировать ключи узлам.

Пример генерации ключей, сертификатов и конфигурация узлов Cook и сервисов использующих их приведена ниже.

4.2.1 Генерация ключа и корневого сертификата

Ключ корневого сертификата /ca/cook_cluster_CA.key

```
openssl genrsa -des3 -out /ca/cook_cluster_CA.key 2048
```

Корневой сертификат /ca/cook_cluster_CA.crt

```
openssl req -x509 -new -subj '/C=RU/ST=/O=/localityName=/commonName=red-soft.ru' -nodes -
↪key /ca/cook_cluster_CA.key -sha256 -days 1825 -out /ca/cook_cluster_CA.crt
```

4.2.2 Генерация ключа и сертификата узлов, подписанного корневым

Данная процедура повторяется для каждого узла Cook, Consul и Gobetween с указанием необходимых доменов.

Ключ узла /cert/node0.cook.key

```
openssl genrsa -out /cert/node0.cook.key 2048
```

CSR /cert/node0.cook.csr

```
openssl req -new -subj '/C=RU/ST=/O=/localityName=/commonName=node0.cook.red-soft.ru' -
↪key /cert/node0.cook.key -out /cert/node0.cook.csr`
```

EXT-файл /cert/node0.cook.ext

```
cat << EOF > /cert/node0.cook.ext
authorityKeyIdentifier=keyid,issuer
basicConstraints=CA:FALSE
keyUsage = digitalSignature, nonRepudiation, keyEncipherment, dataEncipherment
subjectAltName = @alt_names

[alt_names]
DNS.1 = node0.cook.red-soft.ru
EOF
```

Сертификат узла /cert/node0.cook.crt

```
openssl x509 -req -in /cert/node0.cook.csr -CA /ca/cook_cluster_CA.crt -CAkey /ca/cook_
↪cluster_CA.key -CAcreateserial -out /cert/node0.cook.crt -days 825 -sha256 -extfile /
↪cert/node0.cook.ext
```

4.2.3 Пример конфигурации, использующей SSL

cookd.yml

```
cluster_name: testcluster
node_name: node0.cook.red-soft.ru

cook:
    listen: 0.0.0.0:5030
    advertise: node0.cook.red-soft.ru:5030
    ssl: True
    cert: /cert/node0.cook.crt
    key: /cert/node0.cook.key
    cacert: /ca/cook_cluster_CA.crt
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    verify: True

consul:
    url: https://server.dc1.consul.red-soft.ru:8501
    cacert: /ca/cook_cluster_CA.crt
    cert: /cert/node0.cook.crt
    key: /cert/node0.cook.key
    verify: True

rdb:
    listen: 0.0.0.0:3050
    advertise: node0.cook.red-soft.ru:3050
    pidfile: /opt/RedDatabase/rdb.pid
    home: /opt/RedDatabase
    password: masterkey
    replication_dir: /replication
    slave_log_directory: /slave_logs

init:
    rdb:
        databases:
            - database: /cook/database1.fdb
              alias: testdb1
```

4.2.4 Пример конфигурации Consul, использующей TLS

consul.json

```
{
  "log_level": "info",
  "node_name": "server",
  "domain": "consul.red-soft.ru",
  "tls": {
    "defaults": {
      "verify_incoming": true,
      "verify_outgoing": true,
      "ca_file":"/ca/cook_cluster_CA.crt",
      "cert_file":"/cert/server.dc1.consul.crt",
      "key_file":"/cert/server.dc1.consul.key"
    },
    "internal_rpc": {
      "verify_server_hostname": true
    }
  },
  "enable_agent_tls_for_checks": true,
  "ports": {
    "https": 8501
  }
}
```

4.2.5 Пример конфигурации Gobetween, использующей TLS

gobetween.toml

```
[servers.rdb]
bind = "0.0.0.0:3050"
protocol = "tcp"
balance = "iphash"
max_connections = 0

[servers.rdb.discovery]
kind = "consul"
interval = "10s"
consul_host = "server.dc1.consul.red-soft.ru:8501"
consul_service_name = "testcluster"
consul_service_tag = "master"
consul_service_passing_only = true
consul_datacenter = "dc1"

consul_tls_enabled = true
consul_tls_cert_path = "/cert/gobetween.crt"
consul_tls_key_path = "/cert/gobetween.key"
consul_tls_cacert_path = "/ca/cook_cluster_CA.crt"
```

4.3 Fencing

Cook предлагает решение на аппаратном watchdog в качестве I/O-fencing. Когда узел стартует в качестве master, он запускает аппаратный таймер, который обнуляется не реже чем 2 раза за время жизни блокировки в Consul (loop_wait таким образом должен быть не больше половины TTL). При потере блокировки, узел перестает обновлять таймер. Если за время TTL/2 узел не сможет корректно выключиться, а затем выключить аппаратный таймер, то его принудительно выключит или перезагрузит watchdog.

Пример конфигурации для Linux Hardware Watchdog (https://www.kernel.org/doc/Documentation/watchdog/watchdog-api.txt):

```
fencing:
    use: linux_watchdog
    linux_watchdog:
        device: /dev/watchdog
```

Пример кроссплатформенной конфигурации на ipmiutil (http://ipmiutil.sourceforge.net/):

```
fencing:
    use: ipmi_watchdog
    ipmi_watchdog:
        ipmiutil_bin: c:/ipmiutil/ipmiutil.exe
```

Для тестирования конфигурации можно использовать опцию **–test-fencing**.

4.4 Пример конфигурации сервиса systemd

```
[Unit]
Description=Red Database cluster cooker
After=syslog.target network.target

[Service]
Type=simple
User=cook
Group=firebird
EnvironmentFile=-/etc/cook/cookd.env
ExecStart=/usr/local/bin/cookd -c /etc/cook/cookd.yml
ExecReload=/bin/kill -s HUP $MAINPID
KillMode=process
TimeoutSec=60
Restart=no

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

4.5 Использование реверсивных прокси

Для подключения к текущему master кластера Ред Базы Данных можно использовать реверсивные прокси или балансировщики, например Gobetween или HAProxy.

4.5.1 Автоматическая генерация конфигурации Gobetween

Добавлено в *v0.4.0*.

По-умолчанию, при инициализации кластера cook генерирует конфигурацию для Gobetween в K/V-хранилище Consul, которую можно использовать напрямую из сервиса Gobetween. Например:

```
gobetween from-consul localhost:8500 --key=service/testcluster/gobetween --scheme=http -
↪f toml
```

Если позже конфигурацию Gobetween нужно изменить, то это нужно сделать это непосредственно в K/V-хранилище.

4.5.2 Пример конфигурации Gobetween

```
#----------------------------------------
# RedDatabase cluster orchestrated by Cook
#----------------------------------------
[servers.rdb]
bind = "0.0.0.0:3050"
protocol = "tcp"
balance = "iphash"
max_connections = 0
```

(continues on next page)

(продолжение с предыдущей страницы)

```
[servers.rdb.discovery]
kind = "consul"
interval = "10s"
consul_host = "consul:8500"
consul_service_name = "cook-cluster"
consul_service_tag = "master"
consul_service_passing_only = true
consul_datacenter = "dc1"
failpolicy = "setempty"

[servers.rdb_events]
bind = "0.0.0.0:3060"
protocol = "tcp"
balance = "iphash"
max_connections = 0

[servers.rdb_events.discovery]
kind = "consul"
interval = "10s"
consul_host = "consul:8500"
consul_service_name = "cook-cluster"
consul_service_tag = "master_aux"
consul_service_passing_only = true
consul_datacenter = "dc1"
failpolicy = "setempty"
```

4.5.3 Пример конфигурации HAProxy

```
#---------------------------------------
# RedDatabase cluster orchestrated by Cook
#---------------------------------------
frontend rdb
    bind *:3050
    mode tcp
    timeout client 60m
    default_backend cook_cluster

backend cook_cluster
    option tcplog
    option httpchk GET /master
    http-check expect status 200
    server node1 10.81.0.1:3050 check port 5030
    server node2 10.81.0.1:3050 check port 5030
```

Глава 5

Типовые конфигурации отказоустойчивого кластера и покрываемые ими угрозы

5.1 Общий вид кластера и его составляющие

5.1.1 Кластер Cook

Типичное решение с кластером под управлением cook содержит следующие элементы:

1. СУБД Ред База Данных
2. Сервис cook
3. Сервис consul[1][2]
4. Общий дисковый ресурс

Типичными клиентами кластера будут:

1. Приложение использующее ресурсы кластера
2. **Прокси-серверы, перенаправляющие клиентские запросы к master**
    - gobetween[3][4]
    - HAProxy[5][6]

Cook и РБД в данном случае составляют неделимый сервис, так как cook управляет конфигурацией и процессами РБД.

Consul формирует собственный кластер предоставляет cook K/V-хранилище для хранения конфигурации и функционал распределенных блокировок.

[1] Загрузка Consul - https://www.consul.io/downloads
[2] Документация Consul - https://www.consul.io/docs
[3] Загрузка Gobetween - http://gobetween.io/downloads.html
[4] Документация Gobetween - http://gobetween.io/documentation.html
[5] Загрузка HAProxy - http://www.haproxy.org/#down
[6] Документация HAProxy - http://www.haproxy.org/#docs

Дисковый ресурс используется для передачи логов репликации между узлами.

Приложение может использовать API Consul, API Cook или прокси-сервер (HAProxy, gobetween) для подключения к кластеру РБД.

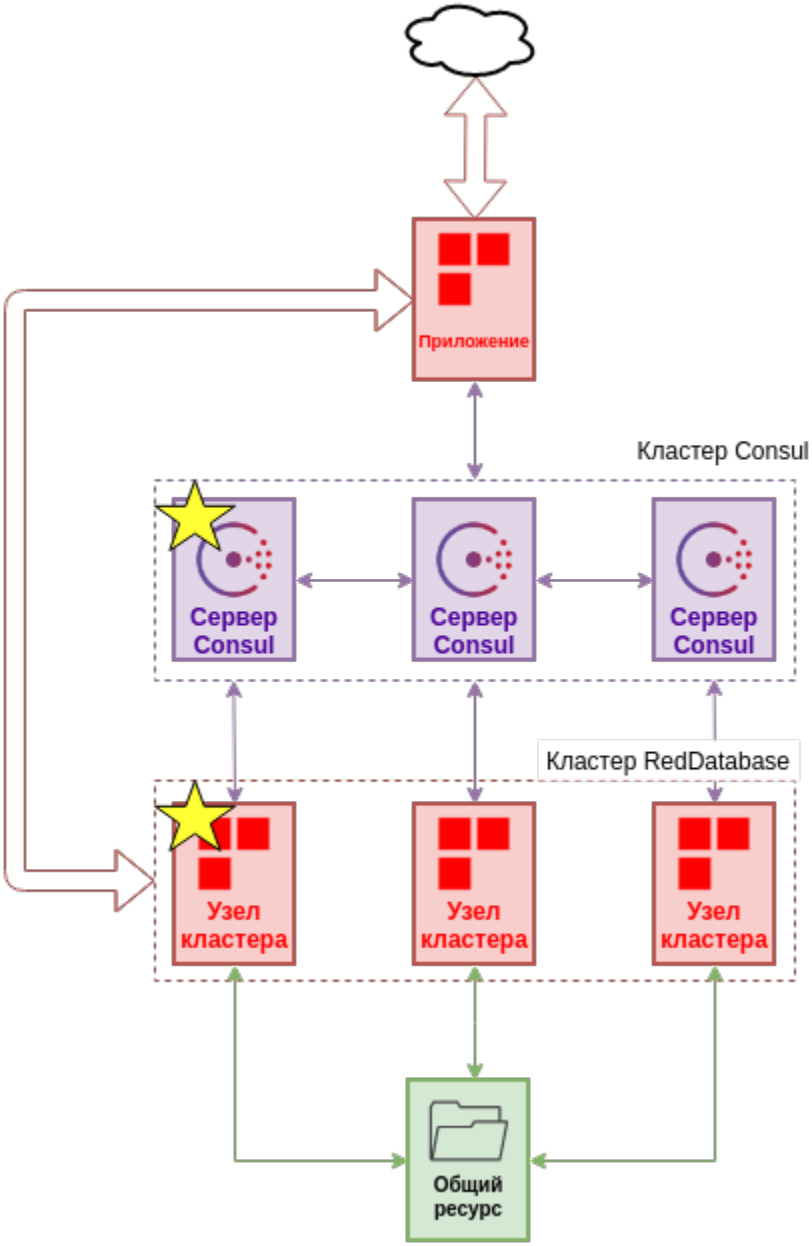

Рис. 1: Схематический вид кластера RedDatabase

В общем случае отказоустойчивый кластер решает проблему единой точки отказа, но необходимо принимать во внимание инфраструктурные особенности конкретных конфигураций.

Для поддержания безостановочной работы кластера нужно обеспечить в частности:

1. Кворум кластера Consul
2. Связность сети между cook и Consul

3. Работу общего дискового ресурса
4. Связность сети между cook и дисковым ресурсом

5.1.2 Узел Cook

Узел Cook или узел кластера RedDatabase - обособленная машина, на которой располагаются:

- Оркестратор Cook
- RedDatabase Enterprise Edition
- Файл базы данных
- Клиент Consul
- Подключение к общему ресурсу

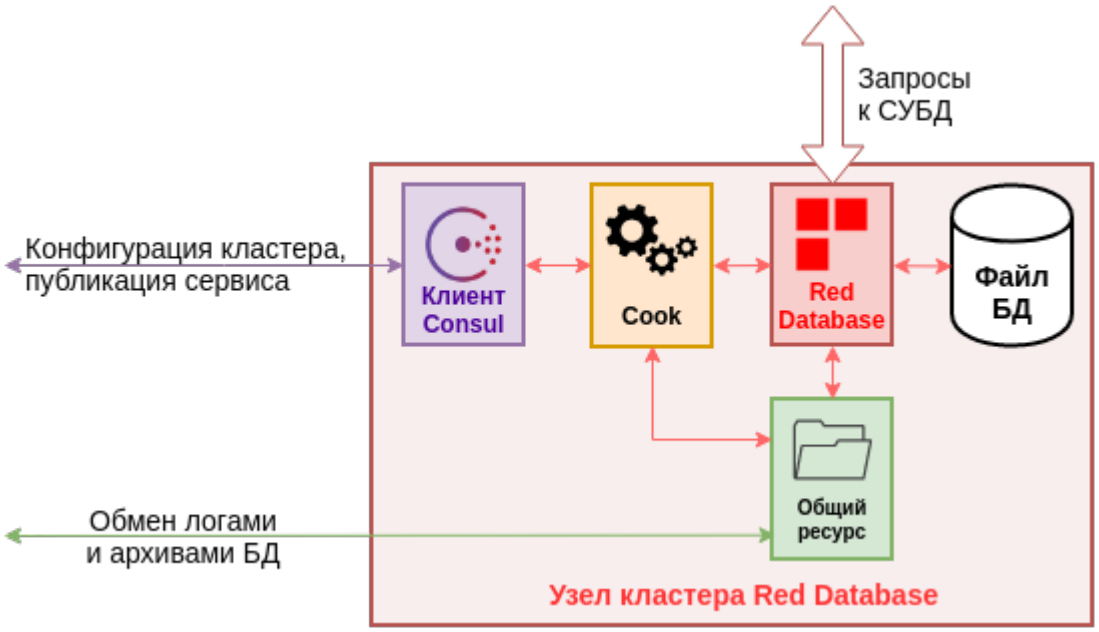

Рис. 2: Схематический вид узла кластера RedDatabase

Конфигурация Cook может выглядеть следующим образом:

- Windows 10/Server 2019 и выше или RedOS 7.2/RHEL 7/CentOS 7/Oracle Linux 7 и выше
- Если в качестве общего ресурса используется CIFS, то диски должны быть смонтированы в режиме WRITETHROUGH
- Хранилище достаточное для постоянного хранения файла базы данных
- Хранилище достаточное для временного хранения архивов базы данных (в зависимости от настроек репликации Cook)

5.1.3 Узел Consul

Обычно выделяют серверные и клиентские узлы Consul. Серверные выполняют работу поддержания кворума, репликации хранилища, связи LAN/WAN между узлами и прочее. Легковесные клиентские предоставляют доступ к API и не участвуют в кворуме. Количество узлов кластера определяется требуемой избыточностью для кворума по формуле (N/3)+1, где N - количество серверных узлов. Так, для 3 узлов кластера необходимо 2 работающих узла для поддержания кворума. В общем случае узлы Consul - отдельные машины, но для упрощения вместо клиента на узле cook может находиться непосредственно серверный узел кластера Consul, при этом необходимо помнить, что при выводе из кластера этой машины кворум Consul может быть нарушен.

5.1.4 Узел приложения

Узел приложения может быть произвольным, быть настроенным как HA ресурс или отсутствовать вовсе. В простейшем случае, приложение, которое принимает клиентские запросы извне, выполняет подключение к кластеру RedDatabase через прокси, например gobetween.

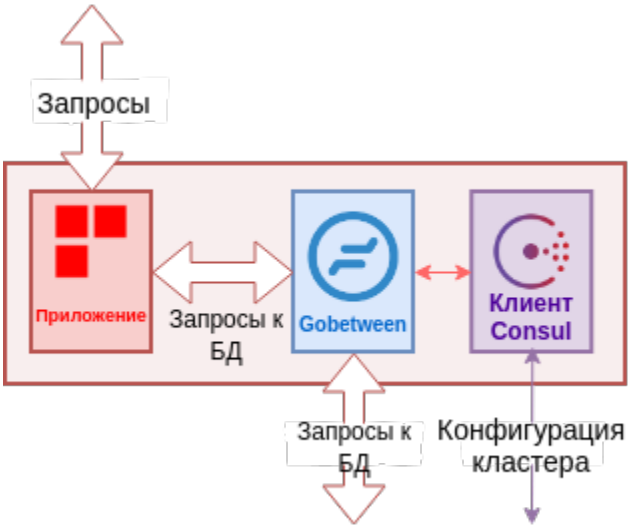

Рис. 3: Схематический вид узла приложения

Подсказка: Примеры конфигураций с Gobetween и HAProxy (*Использование реверсивных прокси*)

Подсказка: Также возможно прямое подключение к кластеру, минуя прокси, но приложение при этом должно опросить Consul или узлы Cook, чтобы узнать текущий master.

5.2 Конфигурации кластера

5.2.1 Все узлы на одном сервере

Строго говоря, данный вариант не является отказоустойчивой конфигурацией и может использоваться только в качестве демонстрации или разработки. При сбое процесса Ред Базы или cook будет выполнено аварийное переключение.

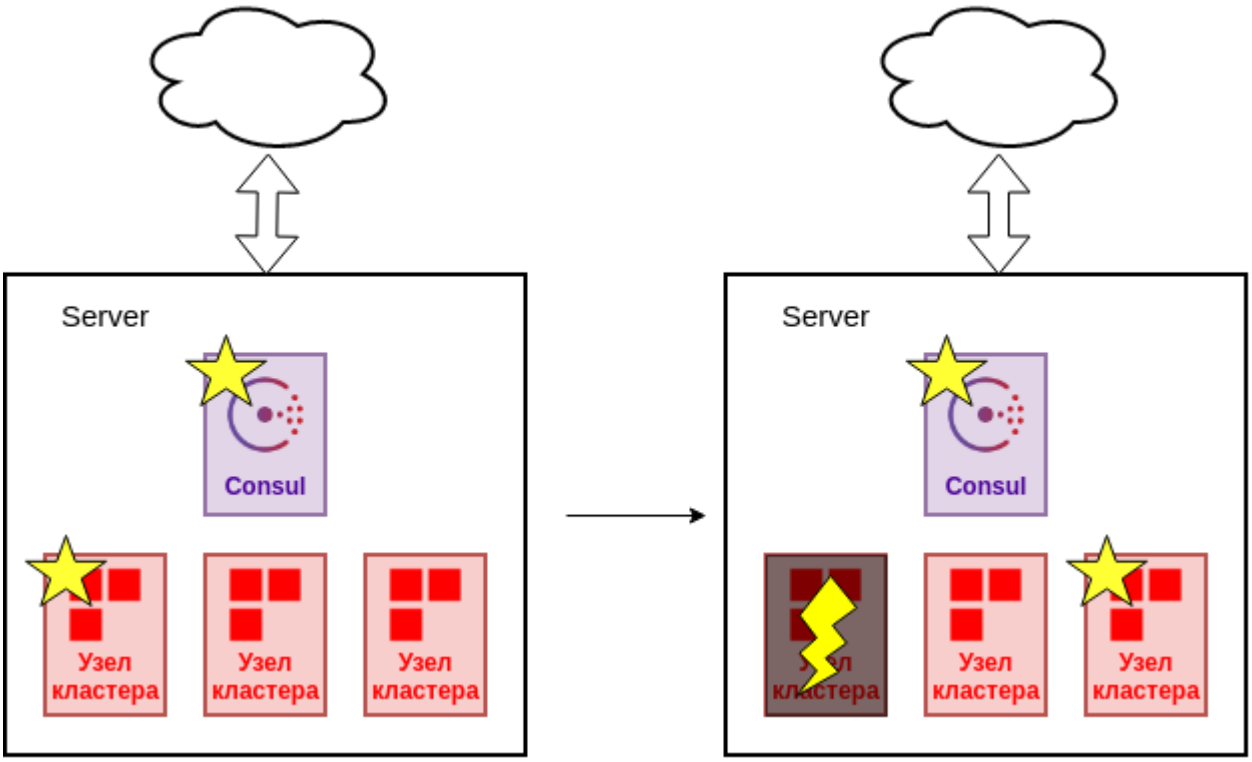

Рис. 4: Сбой процесса

Но, очевидно, что сбой сервера приведет к полной потере кластера.

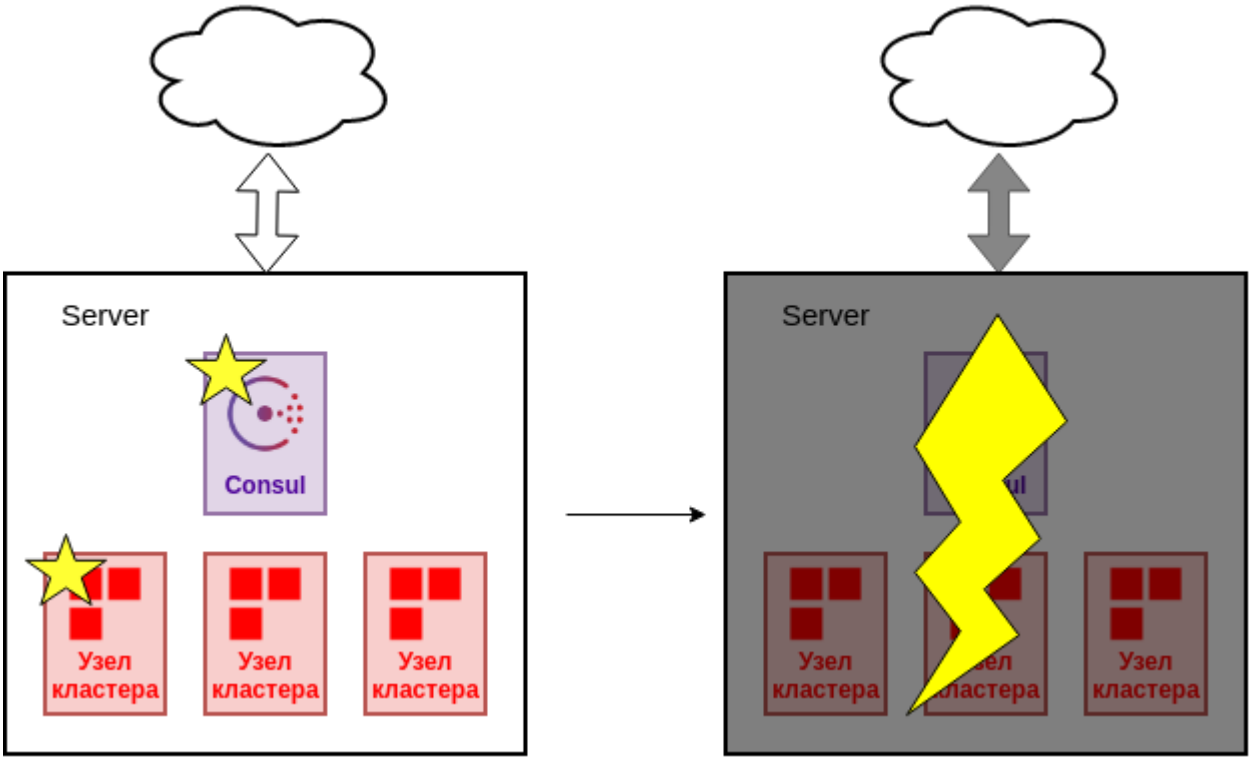

Рис. 5: Сбой сервера

5.2.2 Узлы кластера в гипервизоре

Развертывание узлов на виртуальных машинах с одной стороны более удачный вариант, например для того, чтобы уменьшить время простоя при обновлениях.

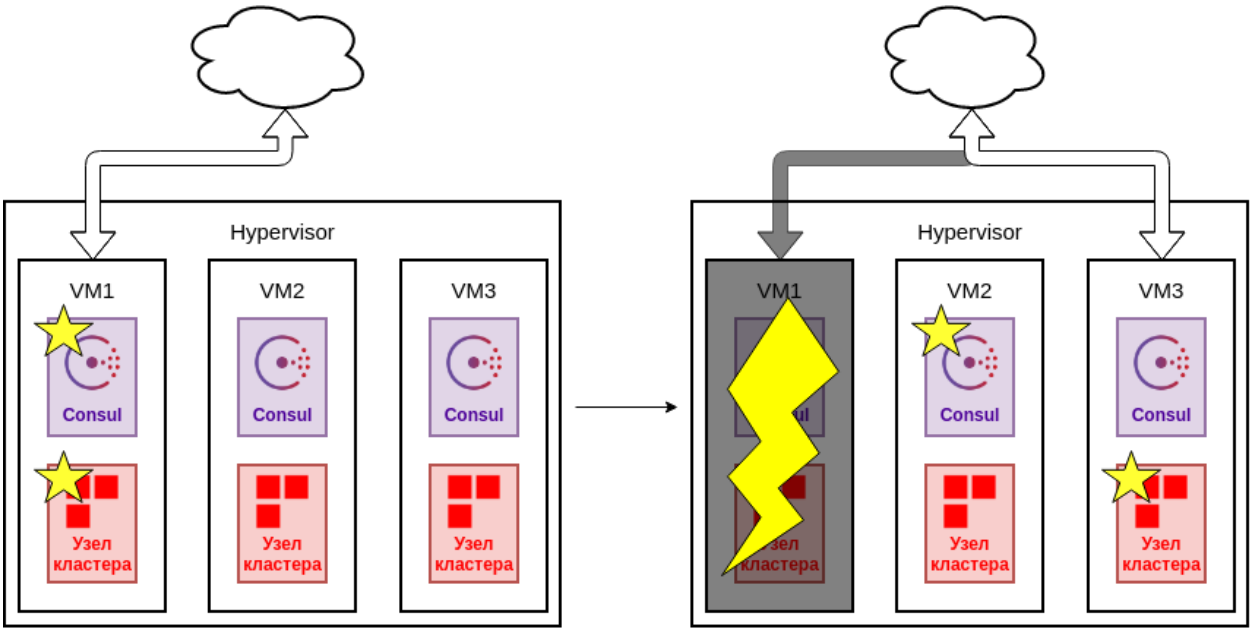

Рис. 6: Сбой виртуальной машины

Однако, когда все машины кластера являются развернуты в одном гипервизоре, точкой отказа является сам гипервизор, так как при сбое весь кластер станет недоступен.

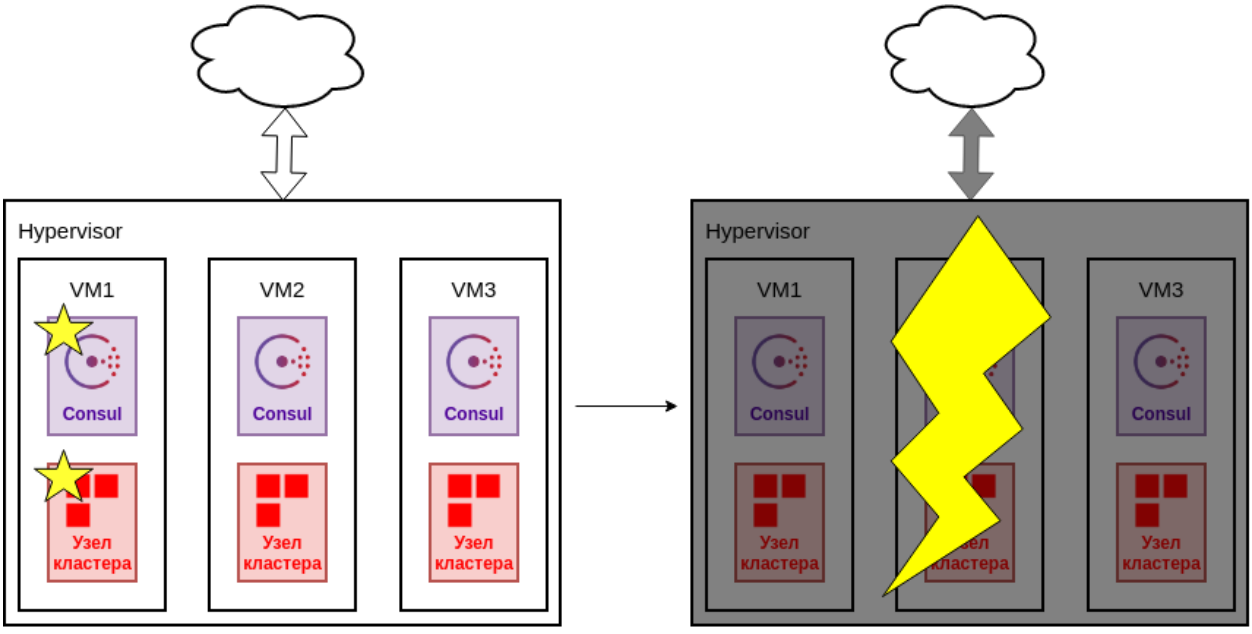

Рис. 7: Сбой гипервизора

Логичным решением в данном случае избегать расположения узлов кластера на одном гипервизоре.

Также возможно применение средств отказоустойчивости самого гипервизора, но это выходит за рамки данного документа.

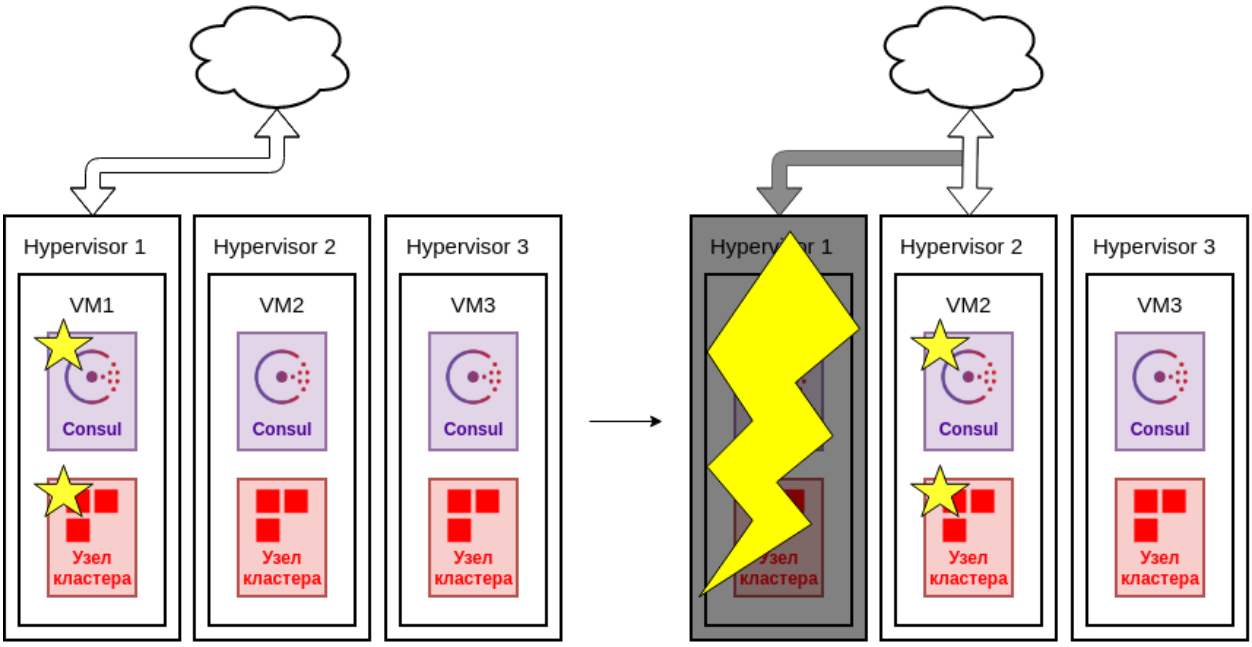

Рис. 8: Сбой гипервизора

Необходимо также помнить про поддержку кворума кластера Consul. Если узлы кластера Consul находятся на тех же машинах, что и cook, то для поддержания кворума должно быть запущено достаточное их количество.

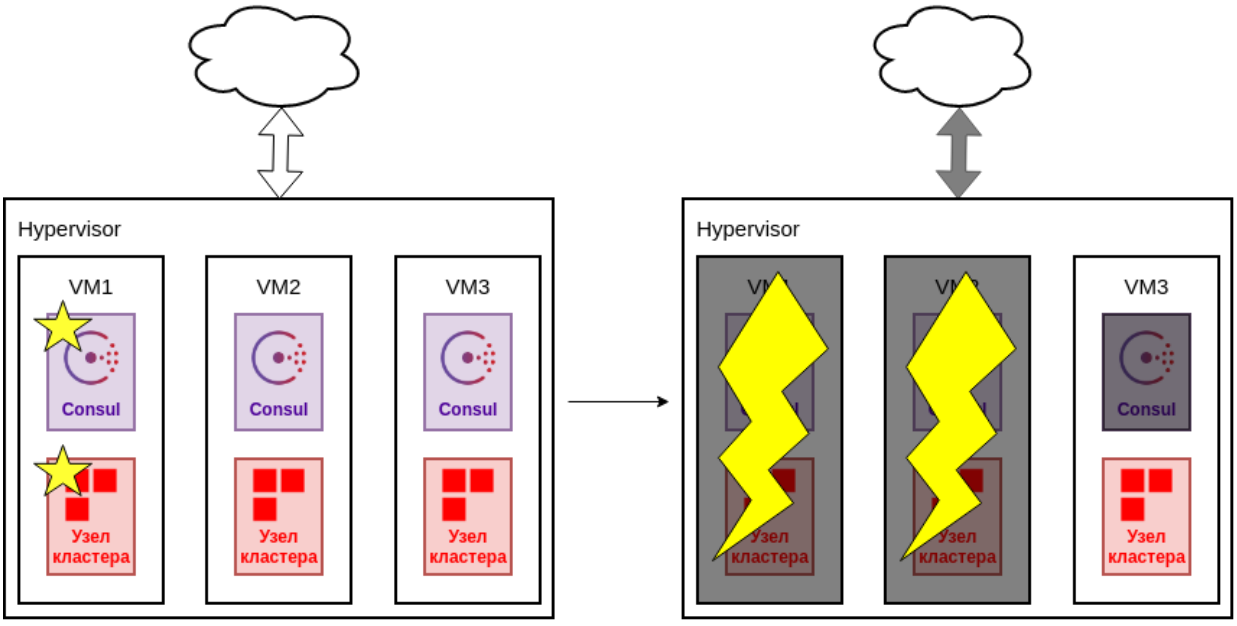

Рис. 9: Потеря кворума Consul

5.2.3 Узлы кластера на резервных площадках

При возможном географическом разнесении кластера, например резервные узлы в разных ЦОД (Data Center, DC), становится возможным защитить кластер от выхода из строя всего ЦОД, но при этом остается проблема кворума Consul.

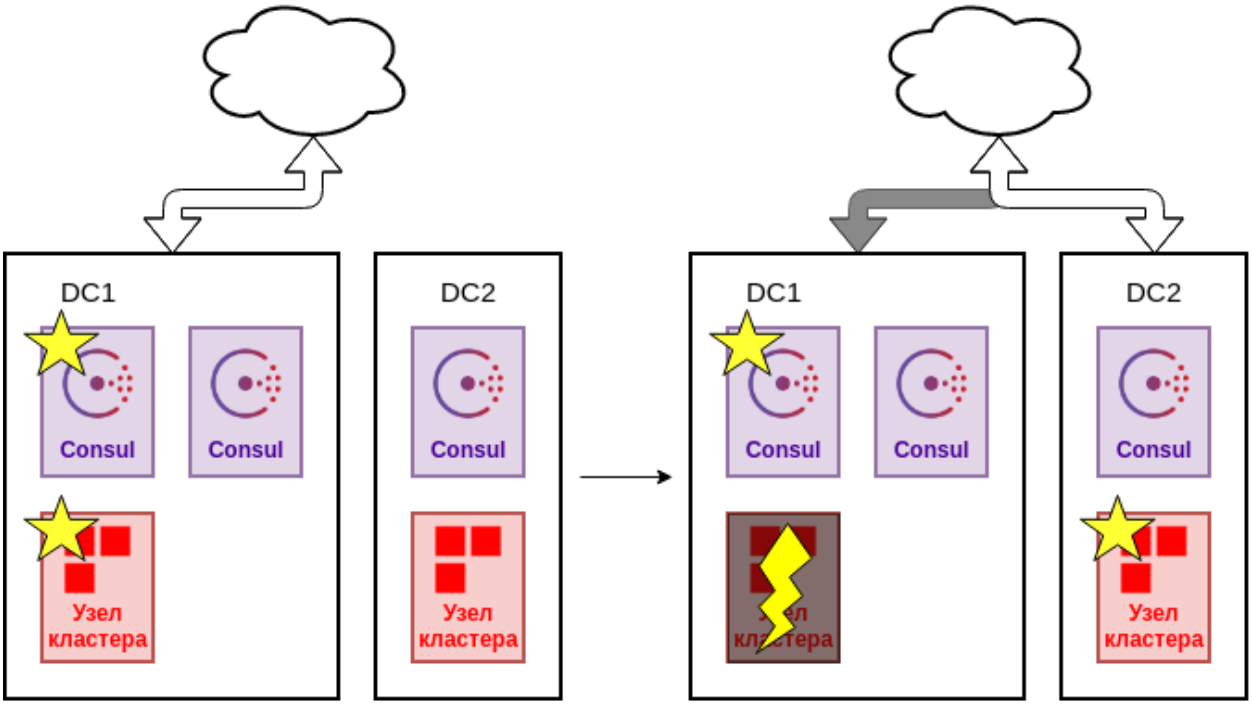

Рис. 10: Выход из строя узла на одном из ЦОД

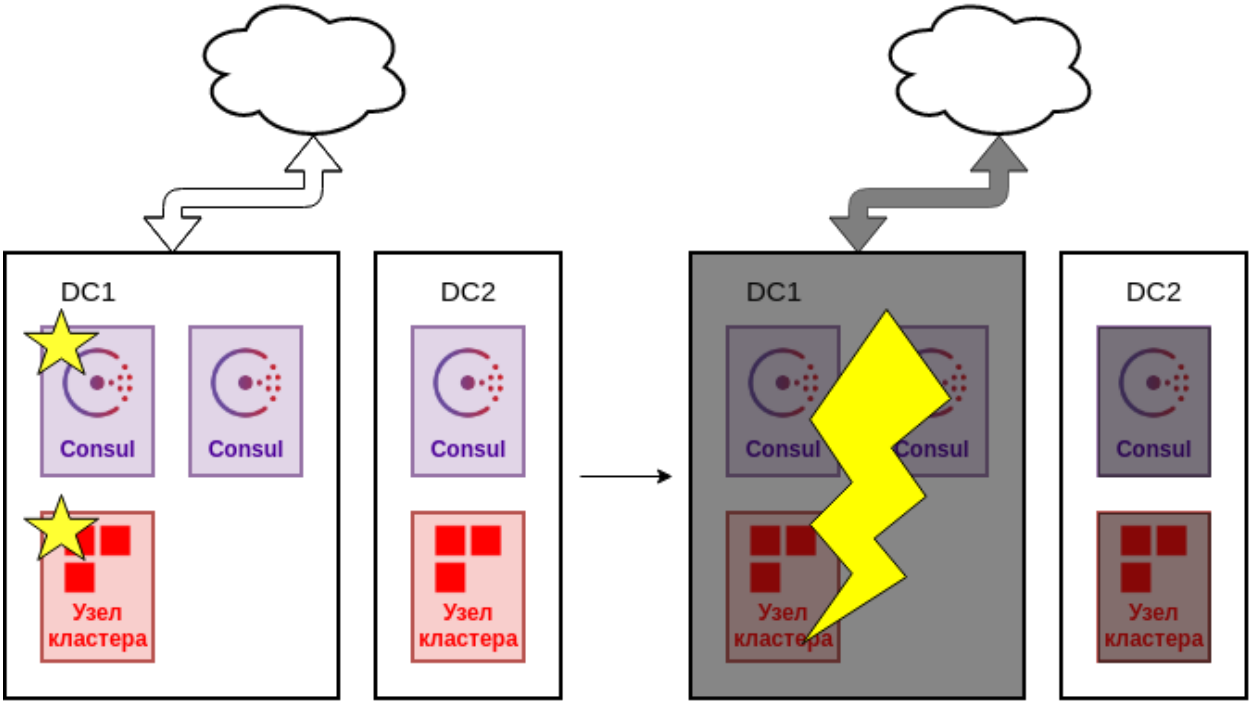

Рис. 11: Выход из строя ЦОД. Потеря кворума Consul

Для поддержания кворума один из узлов Consul может быть свидетелем на отдельной площадке

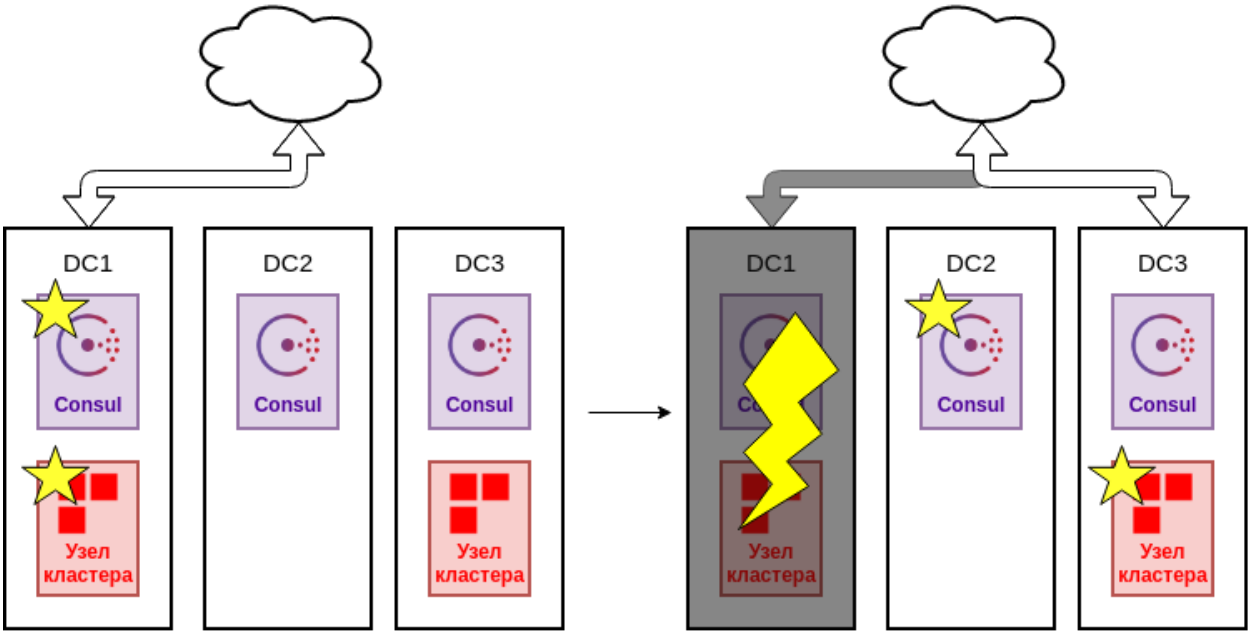

Рис. 12: Выход из строя ЦОД без потери кворума Consul

Предупреждение: Конфигурация с несколькими ЦОД в данный момент возможна только при наличии обмена между ними трафиком уровня 2 (L2)

5.2.4 Fencing

Конфигурация описана в разделе *Fencing*

При работе с данными в любой распределенной системе важна их консистентность. Когда один из узлов работает в режиме master он управляет как БД, так и текущим логом транзакций. В момент отказа этого master, мы должны быть уверены, что он изолирован и перестал выполнять любую работу с данными до того, как мы начали ввод в строй новый master. Проконтролировать это невозможно ни снаружи ни внутри, поэтому важно настроить аппаратный watchdog, который выключит узел физически, в аварийной ситуации.

Watchdog настраивается для каждого узла физической или виртуальной машины. После старта master watchdog начинает свою работу и ожидает, что его будут периодически «сбрасывать». Во время аварийной ситуации, узел пытается штатно завершить работу master, при этом watchdog перестает обновляться. В конечном счете, при хорошем исходе узел переходит в режим standalone и останавливает watchdog, но если узел не успевает корректно выполнить завершение работы, то это сделает watchdog аппаратно.

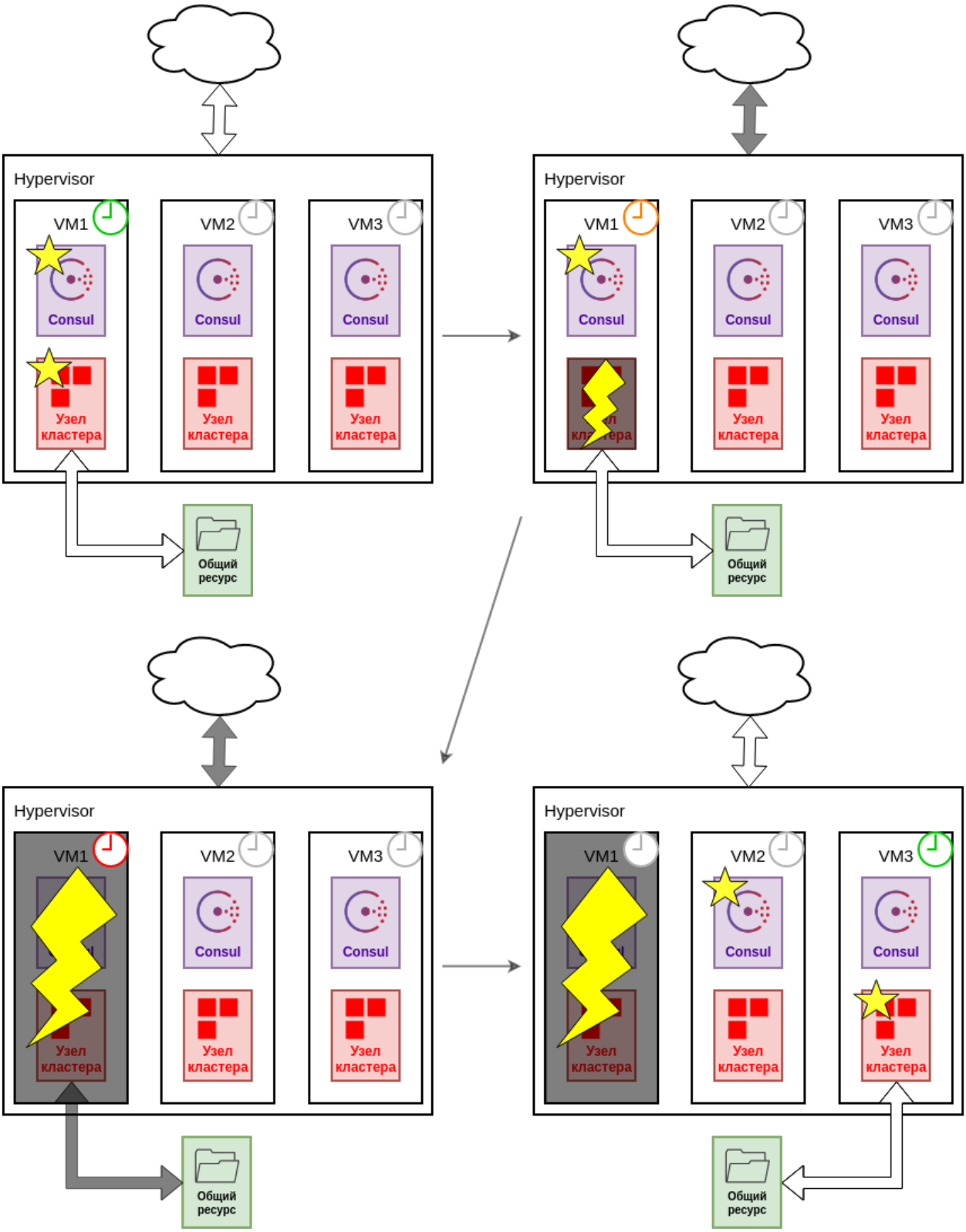

Рис. 13: Работа watchdog во время изоляции узла

Ссылки

Глава 6

Обновление СУБД Ред База Данных в кластере

6.1 Общий порядок обновления

Подсказка: Данный порядок может использоваться как для обновления узлов, так и для отката к предыдущей версии.

1. Установить нужную версию Ред Базы Данных на узлах, которые сейчас остановлены, если таковые имеются
2. Выбрать узел в состоянии slave

```
cookctl nodes
```

```
node0.cook.red-soft.ru (master) API URL http://node0.cook.red-soft.ru:5030,␣
↪RDB 3.0.10.0 on 3050 and AUX 3060 (node0.cook.red-soft.ru/3050), failover␣
↪enabled, priority 0
    testdb1 /db/database1.fdb 1:0 node0.cook.red-soft.ru/3050:testdb1
node1.cook.red-soft.ru (slave) API URL http://node0.cook.red-soft.ru:5030,␣
↪RDB 3.0.10.0 on 3050 and AUX 3060 (node1.cook.red-soft.ru/3050), failover␣
↪enabled, priority 0
    testdb1 /db/database1.fdb 1:0 node1.cook.red-soft.ru/3050:testdb1
```

3. Остановить на нем сервис cookd

```
systemctl stop cookd
```

4. Удостовериться, что процесс rdbserver был остановлен

```
ps aux|grep rdbserver
```

5. Установить необходимую версию Ред Базы Данных (смотри *процесс обновления* или *отката*)

6. Запустить сервис cookd

```
systemctl start cookd
```

7. Теперь этот узел работает на новой версии Ред Базы Данных

```
cookctl nodes
```

```
node0.cook.red-soft.ru (master) API URL http://node0.cook.red-soft.ru:5030,␣
↪RDB 3.0.10.0 on 3050 and AUX 3060 (node0.cook.red-soft.ru/3050), failover␣
↪enabled, priority 0
    testdb1 /db/database1.fdb 1:0 node0.cook.red-soft.ru/3050:testdb1
node1.cook.red-soft.ru (slave) API URL http://node1.cook.red-soft.ru:5030,␣
↪RDB 3.0.11.0 on 3050 and AUX 3060 (node1.cook.red-soft.ru/3050), failover␣
↪enabled, priority 0
    testdb1 /db/database1.fdb 1:0 node1.cook.red-soft.ru/3050:testdb1
```

8. Повторять шаги 2-6 для каждого slave в кластере
9. Выбрать активный master
10. Перевести активный master на другой узел или остановить его и дождаться появления в кластере нового master

```
cookctl promote <node>
```

или

```
systemctl stop cookd
```

Например:

```
cookctl promote node1.cook.red-soft.ru
```

```
Asking cluster to promote node node1.cook.red-soft.ru
```

```
cookctl nodes
```

```
node0.cook.red-soft.ru (slave) API URL http://node0.cook.red-soft.ru:5030,␣
↪RDB 3.0.10.0 on 3050 and AUX 3060 (node0.cook.red-soft.ru/3050), failover␣
↪enabled, priority 0
    testdb1 /db/database1.fdb 2:0 node0.cook.red-soft.ru/3050:testdb1
node1.cook.red-soft.ru (master) API URL http://node1.cook.red-soft.ru:5030,␣
↪RDB 3.0.11.0 on 3050 and AUX 3060 (node1.cook.red-soft.ru/3050), failover␣
↪enabled, priority 0
    testdb1 /db/database1.fdb 2:0 node1.cook.red-soft.ru/3050:testdb1
```

11. Повторить шаги для 2-6 для старого master, который теперь стал slave или остановлен. После этого весь кластер будет работать на новой версии Ред Базы Данных:

```
cookctl nodes
```

```
node0.cook.red-soft.ru (slave) API URL http://node0.cook.red-soft.ru:5030,␣
↪RDB 3.0.11.0 on 3050 and AUX 3060 (node0.cook.red-soft.ru/3050), failover␣
```

(continues on next page)

(продолжение с предыдущей страницы)

```
↪enabled, priority 0
    testdb1 /db/database1.fdb 2:0 node0.cook.red-soft.ru/3050:testdb1
node1.cook.red-soft.ru (master) API URL http://node1.cook.red-soft.ru:5030,␣
↪RDB 3.0.11.0 on 3050 and AUX 3060 (node1.cook.red-soft.ru/3050), failover␣
↪enabled, priority 0
    testdb1 /db/database1.fdb 2:0 node1.cook.red-soft.ru/3050:testdb1
```

12. Если необходимо, перенести master обратно

6.2 Процесс обновления Ред Базы Данных на узле

1. Сделать бэкап баз данных, если версия, на которую необходимо обновиться предполагает бэкап/рестор
2. Сделать бэкап файла cluster.yml
3. Установить необходимую версию Ред Базы Данных
4. Запретить автозапуск или удалить сервис firebird

```
systemd disable firebird
```

5. Установить владельца и права файлов и каталогов в /opt/RedDatabase

```
cookctl fixpermissions firebird
```

6. Восстановить базы данных, если версия, на которую необходимо обновиться предполагает бэкап/рестор
7. Восстановить файл cluster.yml

6.3 Процесс отката Ред Базы Данных на узле

1. Восстановить бэкап баз данных, если версия, с которой выполняется откат предполагает бэкап/рестор
2. Установить необходимую версию Ред Базы Данных
3. Запретить автозапуск или удалить сервис firebird

```
systemd disable firebird
```

4. Установить владельца и права файлов и каталогов в /opt/RedDatabase

```
cookctl fixpermissions firebird
```

5. Восстановить базы данных, если версия, на которую необходимо обновиться предполагает бэкап/рестор
6. Восстановить файл cluster.yml

Глава 7

Утилиты

7.1 cookd

cookd - это непосредственно демон сервиса cook. Чтобы запустить его, достаточно передать конфигурационный файл: *cookd -c <config>*.

7.2 cookctl

Утилита cookctl - это утилита управления кластером cook. Она может использовать тот же конфигурационный файл, что и cook для подключения к кластеру. *cookctl -c <config> <command>*

7.2.1 status

Показывает информацию о кластере и его активные узлы

Пример:

```
cookctl -c configuration.yml status
```

Результат:

```
cook-cluster: generation 2, locked by cook-node-3

Databases:
  testdb1
  testdb2

cook-node-1 (slave) API URL http://cook-node-1:5030, RDB 3.0.8.0 on 3050 and AUX 3060␣
↪(cook-node-1/3050), failover enabled, priority 0
        testdb1 /data/database1.fdb 2:0 cook-node-1/3050:testdb1
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        testdb2 /data/database2.fdb 2:0 cook-node-1/3050:testdb2
cook-node-2 (slave) API URL http://cook-node-2:5030, RDB 3.0.8.0 on 3050 and AUX 3060␣
↪(cook-node-2/3050), failover enabled, priority 0
        testdb1 /data/database1.fdb 2:0 cook-node-2/3050:testdb1
        testdb2 /data/database2.fdb 2:0 cook-node-2/3050:testdb2
cook-node-3 (master) API URL http://cook-node-3:5030, RDB 3.0.8.0 on 3050 and AUX 3060␣
↪(cook-node-3/3050), failover enabled, priority 0
        testdb1 /data/database1.fdb 2:0 cook-node-3/3050:testdb1
        testdb2 /data/database2.fdb 2:0 cook-node-3/3050:testdb2
```

Ключ **-v** выводит дополнительную техническую информацию в **status**, а также в других командах:

Пример:

```
cookctl -c configuration.yml -v status
```

Результат:

```
cook-cluster: (eec6fb84-607a-48cc-a511-e3d04494c37d, 0.3.0), generation 2 (95abc4a4-3f44-
↪4adf-a251-822924c48bae), locked by cook-node-3

Databases:
  testdb1 (362706dd-1689-4157-a5e9-286a2f335e5d)
  testdb2 (de917dc6-a134-4f54-8da7-ff9ff5582399)

cook-node-1 (slave) API URL http://cook-node-1:5030, RDB 3.0.8.0 on 3050 and AUX 3060␣
↪(cook-node-1/3050), failover enabled, priority 0
        testdb1 /data/database1.fdb (362706dd-1689-4157-a5e9-286a2f335e5d) 2:0 cook-node-
↪1/3050:testdb1
        testdb2 /data/database2.fdb (de917dc6-a134-4f54-8da7-ff9ff5582399) 2:0 cook-node-
↪1/3050:testdb2
cook-node-2 (slave) API URL http://cook-node-2:5030, RDB 3.0.8.0 on 3050 and AUX 3060␣
↪(cook-node-2/3050), failover enabled, priority 0
        testdb1 /data/database1.fdb (362706dd-1689-4157-a5e9-286a2f335e5d) 2:0 cook-node-
↪2/3050:testdb1
        testdb2 /data/database2.fdb (de917dc6-a134-4f54-8da7-ff9ff5582399) 2:0 cook-node-
↪2/3050:testdb2
cook-node-3 (master) API URL http://cook-node-3:5030, RDB 3.0.8.0 on 3050 and AUX 3060␣
↪(cook-node-3/3050), failover enabled, priority 0
        testdb1 /data/database1.fdb (362706dd-1689-4157-a5e9-286a2f335e5d) 2:0 cook-node-
↪3/3050:testdb1
        testdb2 /data/database2.fdb (de917dc6-a134-4f54-8da7-ff9ff5582399) 2:0 cook-node-
↪3/3050:testdb2
```

7.2.2 nodes

Показывает все активные узлы кластера их статусы и IP, которые они анонсировали для подключения к БД и REST API, а также список баз с локальными путями и DSN, которыми они управляют.

Пример:

```
cookctl -c configuration.yml nodes
```

Результат:

```
cook-node-1 (slave) API URL http://cook-node-1:5030, RDB 3.0.8.0 on 3050 and AUX 3060␣
↪(cook-node-1/3050), failover enabled, priority 0
        testdb1 /data/database1.fdb 2:0 cook-node-1/3050:testdb1
        testdb2 /data/database2.fdb 2:0 cook-node-1/3050:testdb2
cook-node-2 (slave) API URL http://cook-node-2:5030, RDB 3.0.8.0 on 3050 and AUX 3060␣
↪(cook-node-2/3050), failover enabled, priority 0
        testdb1 /data/database1.fdb 2:0 cook-node-2/3050:testdb1
        testdb2 /data/database2.fdb 2:0 cook-node-2/3050:testdb2
cook-node-3 (master) API URL http://cook-node-3:5030, RDB 3.0.8.0 on 3050 and AUX 3060␣
↪(cook-node-3/3050), failover enabled, priority 0
        testdb1 /data/database1.fdb 2:0 cook-node-3/3050:testdb1
        testdb2 /data/database2.fdb 2:0 cook-node-3/3050:testdb2
```

Статус узла может содержать дополнительные «флаги»:

- **failover enabled/disabled** - флаг ручного запрета перевода узла в режим master
- **restrict master** - признак, ограниченного перевода узла в режим master (во время старта кластера, или после сбоя)
- **priority N** - приоритет при выборе узла в качестве следующего master в случае сбоя
- **MAINTENANCE** - режим обслуживания

7.2.3 history

Показывает историю перехода статуса ведущего узла в кластере и накопления архивов репликации:

Пример:

```
cookctl -c configuration.yml history
```

Результат:

```
1 by master cook-node-1
        testdb1 1:3
        testdb2 1:2
2 by master cook-node-3
        testdb1 2:?
        testdb2 2:?
```

7.2.4 promote

Формат:

```
promote <node>
```

Дает команду кластеру перевести узел <node> в режим master.

Пример:

```
cookctl -c configuration.yml promote cook-node-1
```

Состояние кластера до promote :

```
cook-cluster: generation 1, locked by cook-node-1

Databases:
  testdb1
  testdb2

cook-node-1 (master) API URL http://cook-node-1:5030, RDB 3.0.8.0 on 3050 and AUX 3060␣
↪(cook-node-1/3050), failover enabled, priority 0
        testdb1 /data/database1.fdb 1:3 cook-node-1/3050:testdb1
        testdb2 /data/database2.fdb 1:2 cook-node-1/3050:testdb2
cook-node-2 (slave) API URL http://cook-node-2:5030, RDB 3.0.8.0 on 3050 and AUX 3060␣
↪(cook-node-2/3050), failover enabled, priority 0
        testdb1 /data/database1.fdb 1:2 cook-node-2/3050:testdb1
        testdb2 /data/database2.fdb 1:1 cook-node-2/3050:testdb2
cook-node-3 (slave) API URL http://cook-node-3:5030, RDB 3.0.8.0 on 3050 and AUX 3060␣
↪(cook-node-3/3050), failover enabled, priority 0
        testdb1 /data/database1.fdb 1:2 cook-node-3/3050:testdb1
        testdb2 /data/database2.fdb 1:1 cook-node-3/3050:testdb2
```

```
1 by master cook-node-1
        testdb1 1:?
        testdb2 1:?
```

Состояние кластера после promote :

```
cook-cluster: generation 2, locked by cook-node-3

Databases:
  testdb1
  testdb2

cook-node-1 (slave) API URL http://cook-node-1:5030, RDB 3.0.8.0 on 3050 and AUX 3060␣
↪(cook-node-1/3050), failover enabled, priority 0
        testdb1 /data/database1.fdb 2:0 cook-node-1/3050:testdb1
        testdb2 /data/database2.fdb 2:0 cook-node-1/3050:testdb2
cook-node-2 (slave) API URL http://cook-node-2:5030, RDB 3.0.8.0 on 3050 and AUX 3060␣
↪(cook-node-2/3050), failover enabled, priority 0
        testdb1 /data/database1.fdb 2:0 cook-node-2/3050:testdb1
        testdb2 /data/database2.fdb 2:0 cook-node-2/3050:testdb2
cook-node-3 (master) API URL http://cook-node-3:5030, RDB 3.0.8.0 on 3050 and AUX 3060␣
```

(continues on next page)

(продолжение с предыдущей страницы)

```
↪(cook-node-3/3050), failover enabled, priority 0
        testdb1 /data/database1.fdb 2:0 cook-node-3/3050:testdb1
        testdb2 /data/database2.fdb 2:0 cook-node-3/3050:testdb2
```

```
1 by master cook-node-1
        testdb1 1:3
        testdb2 1:2
2 by master cook-node-3
        testdb1 2:?
        testdb2 2:?
```

7.2.5 allowfailover

Формат:

```
allowfailover <node> <0|1>
```

Устанавливает возможность автоматического конвертации узла slave в master при сбое (по-умолчанию 1).

Пример:

```
cookctl -c configuration.yml allowfailover cook-node-3 0
```

Состояние узлов с запретом перехода master на cook-node-3:

```
cook-node-1 (slave) API URL http://cook-node-1:5030, RDB 3.0.8.0 on 3050 and AUX 3060␣
↪(cook-node-1/3050), failover enabled, priority 10
        testdb1 /data/database1.fdb 2:0 cook-node-1/3050:testdb1
        testdb2 /data/database2.fdb 2:0 cook-node-1/3050:testdb2
cook-node-2 (slave) API URL http://cook-node-2:5030, RDB 3.0.8.0 on 3050 and AUX 3060␣
↪(cook-node-2/3050), failover enabled, priority 0
        testdb1 /data/database1.fdb 2:0 cook-node-2/3050:testdb1
        testdb2 /data/database2.fdb 2:0 cook-node-2/3050:testdb2
cook-node-3 (master) API URL http://cook-node-3:5030, RDB 3.0.8.0 on 3050 and AUX 3060␣
↪(cook-node-3/3050), failover disabled, priority 0
        testdb1 /data/database1.fdb 2:0 cook-node-3/3050:testdb1
        testdb2 /data/database2.fdb 2:0 cook-node-3/3050:testdb2
```

7.2.6 maintenance

Версия: 0.1.0

Формат:

```
maintenance <0|1> [--wait]
```

Дает команду кластеру перевести узлы в режим обслуживания (см. *Режим обслуживания*). Опционально аргумент –**wait** позволяет дождаться, когда все узлы перейдут в режим обслуживания.

Пример:

```
cookctl -c configuration.yml maintenance 1 --wait
Enabling maintenance mode
```

Состояние кластера в режиме maintenance:

```
cook-cluster: generation 2, locked by cook-node-3

Databases:
  testdb1
  testdb2

cook-node-1 (slave) API URL http://cook-node-1:5030, RDB 3.0.8.0 on 3050 and AUX 3060␣
↪(cook-node-1/3050), failover enabled, priority 0
        testdb1 /data/database1.fdb 2:0 cook-node-1/3050:testdb1
        testdb2 /data/database2.fdb 2:0 cook-node-1/3050:testdb2
cook-node-2 (slave) API URL http://cook-node-2:5030, RDB 3.0.8.0 on 3050 and AUX 3060␣
↪(cook-node-2/3050), failover enabled, priority 0
        testdb1 /data/database1.fdb 2:0 cook-node-2/3050:testdb1
        testdb2 /data/database2.fdb 2:0 cook-node-2/3050:testdb2
cook-node-3 (master) API URL http://cook-node-3:5030, RDB 3.0.8.0 on 3050 and AUX 3060␣
↪(cook-node-3/3050), failover enabled, priority 0
        testdb1 /data/database1.fdb 2:0 cook-node-3/3050:testdb1
        testdb2 /data/database2.fdb 2:0 cook-node-3/3050:testdb2
```

7.2.7 priority

Версия: 0.3.0

Формат:

```
priority  <node> <priority>
```

Устанавливает приоритет узла при выборе нового master при сбое (по-умолчанию 0).

Пример:

```
cookctl -c configuration.yml priority cook-node-1 10
```

Состояние узлов с измененным приоритетом:

```
cook-node-1 (slave) API URL http://cook-node-1:5030, RDB 3.0.8.0 on 3050 and AUX 3060␣
↪(cook-node-1/3050), failover enabled, priority 10
        testdb1 /data/database1.fdb 2:0 cook-node-1/3050:testdb1
        testdb2 /data/database2.fdb 2:0 cook-node-1/3050:testdb2
cook-node-2 (slave) API URL http://cook-node-2:5030, RDB 3.0.8.0 on 3050 and AUX 3060␣
↪(cook-node-2/3050), failover enabled, priority 0
        testdb1 /data/database1.fdb 2:0 cook-node-2/3050:testdb1
        testdb2 /data/database2.fdb 2:0 cook-node-2/3050:testdb2
cook-node-3 (master) API URL http://cook-node-3:5030, RDB 3.0.8.0 on 3050 and AUX 3060␣
↪(cook-node-3/3050), failover enabled, priority 0
        testdb1 /data/database1.fdb 2:0 cook-node-3/3050:testdb1
        testdb2 /data/database2.fdb 2:0 cook-node-3/3050:testdb2
```

7.2.8 add-database

Версия: 0.3.0

Формат:

```
add-database <alias>
```

Добавляет базу данных в кластер

На master база данных <**alias**> должна быть в секции **init.rdb.databases** в конфигурации передаваемой в cookctl, также как при инициализации кластера.

> **Предупреждение:** Данную команду можно выполнить только на активном master.
>
> При добавлении базы данных текущий master не потеряет блокировку, но остановит Ред Базу Данных во время добавления и увеличит поколение репликации. Подключенные slave в кластере также остановят репликацию для инициализации новой базы данных.

Пример:

```
cookctl -c configuration.yml add-database mydb3
```

7.2.9 rm-database

Версия: 0.3.0

Формат:

```
rm-database <alias>
```

Удаляет базу данных из кластера.

> **Предупреждение:** Данную команду можно выполнить только на активном master.
>
> При удалении базы данных текущий master не потеряет блокировку, но остановит Ред Базу Данных во время удаления и увеличит поколение репликации. Подключенные slave в кластере также остановят репликацию для удаление базы данных.

Пример:

```
cookctl -c configuration.yml rm-database mydb3
```

7.2.10 shutdown

Версия: *0.5.0*

Формат:

```
shutdown <0|1> [--wait [--reset]]
```

Останавливает все узлы Cook.

Опционально аргумент –**wait** позволяет дождаться, когда все узлы остановятся, а –**reset** - автоматически отменяет shutdown.

Предупреждение: Если не использовались флаги –**wait** –**reset**, то режим shutdown сохраняется и узлы не смогут стартовать.

Пример остановки кластера без ожидания остановки всех узлов:

```
cookctl -c configuration.yml shutdown 1
```

Пример остановки кластера с ожиданием остановки всех узлов и автоматической отмены режима shutdown:

```
cookctl -c configuration.yml shutdown 1 --wait --reset
```

Пример ручной отмены режима shutdown:

```
cookctl -c configuration.yml shutdown 0
```

7.2.11 logs

Для облегчения анализа и поиска логов и архивов cookctl предоставляет команды **ls** и **find**. Они оперируют каталогом репликации на текущем узле, поэтому результат выполнения команды может различаться на разных узлах.

logs ls

Версия: *0.5.0*

Формат:

```
logs ls [--analyze] [--database <database|GUID>] [--generation <generation>]
```

Выводит все доступные архивы и логи на текущем узле.

Аргумент **-a, --analyze** позволяет вывести номер и состояние лога.

Чтобы ограничить вывод логов необходимым поколением и базой данных, можно использовать аргументы **-g, --generation** и **-d, --database**.

Пример:

```
cookctl  -c configuration.yml logs ls -a -d testdb1 -g 2
```

```
Analyzing /tmp/cook/replication for node node0
Cluster directory /tmp/cook/replication/cce57a4f-abe8-49c7-ac1f-6c34a146093a

  Generation 2.
  Master is node1
  Archives directory: /tmp/cook/replication/cce57a4f-abe8-49c7-ac1f-6c34a146093a/
↪archives/master/2
  Logs directory: /tmp/cook/replication/cce57a4f-abe8-49c7-ac1f-6c34a146093a/logs/2
    Database testdb1. Master GUID 332BBB79-A701-4C4C-3EAD-D06C3AF48FF0
      Archives in /tmp/cook/replication/cce57a4f-abe8-49c7-ac1f-6c34a146093a/archives/
↪master/2/9413abe6-376e-4035-8d8b-51daec71ba05:
        cook_async_repl.arch-000000001 sequence: 1, state: arch
        cook_async_repl.arch-000000002 sequence: 2, state: arch
        cook_async_repl.arch-000000003 sequence: 3, state: arch
        cook_async_repl.arch-000000004 sequence: 4, state: arch
        cook_async_repl.arch-000000005 sequence: 5, state: arch
        cook_async_repl.arch-000000006 sequence: 6, state: arch
        cook_async_repl.arch-000000007 sequence: 7, state: arch
        cook_async_repl.arch-000000008 sequence: 8, state: arch
        cook_async_repl.arch-000000009 sequence: 9, state: arch
        cook_async_repl.arch-000000010 sequence: 10, state: arch
        cook_async_repl.arch-000000011 sequence: 11, state: arch
        cook_async_repl.arch-000000012 sequence: 12, state: arch
        cook_async_repl.arch-000000013 sequence: 13, state: arch
        cook_async_repl.arch-000000014 sequence: 14, state: arch
        cook_async_repl.arch-000000015 sequence: 15, state: arch
      Logs in /tmp/cook/replication/cce57a4f-abe8-49c7-ac1f-6c34a146093a/logs/2/9413abe6-
↪376e-4035-8d8b-51daec71ba05:
        cook_async_repl.log-000 sequence: 15, state: free
      Slave logs (/tmp/cook/replication/slave_logs_0/cce57a4f-abe8-49c7-ac1f-
↪6c34a146093a/2/9413abe6-376e-4035-8d8b-51daec71ba05:
        Control file {332BBB79-A701-4C4C-3EAD-D06C3AF48FF0} sequence: 15
        Logs not found
```

logs find

Версия: *0.5.0*

Формат:

```
logs find <database|GUID> <position>
```

Ищет логи и архивы репликации для указанной базы данных и позиции.

База данных может быть указана как в виде псевдонима, указанного при создании кластера или ее добавлении так и GUID, сгенерированного кластером.

Позиция указывается в формате G:S, где G - поколение, S - порядковый номер лога.

Пример:

```
cookctl  -c configuration.yml logs find testdb1 2:3
```

```
Analyzing /tmp/cook/replication for node node0
Found archive /tmp/cook/replication/cce57a4f-abe8-49c7-ac1f-6c34a146093a/archives/master/
↪2/9413abe6-376e-4035-8d8b-51daec71ba05/cook_async_repl.arch-000000003 in state arch
```

7.2.12 recreate

Версия: *0.5.0*

Формат:

```
recreate <node> [--force]
```

Удаляет и инициализирует данные узла. Все базы данных и логи реплики будут удалены, затем узел заново инициализирует их из существующего master.

Предупреждение: Для инициализации в конфигурации cookd должна присутствовать секция **init.slave_database_mapping**.

Предупреждение: По-умолчанию утилита не будет удалять данные текущего master. Чтобы принудительно виниализировать текущий master, можно использовать ключ **–force**. При этом сначала текущий master будет остановлен, а инициализация будет выполнена только после корректного выбора нового master, поэтому данный вариант не рекомендуется использовать.

Пример:

```
cookctl -c configuration.yml recreate node1
```

7.2.13 fixpermissions

Версия: *0.6.0*

Формат:

```
fixpermissions <owner> [<groups>]
```

Устанавливает владельца и права на файлы и директории, используемые cookd.

Предупреждение: Для инициализации в конфигурации cookd должна присутствовать секция **init.slave_database_mapping**.

Предупреждение: Утилита должна быть запущена от root.

Пример:

```
cookctl -c configuration.yml fixpermissions firebird
```

7.3 cookdsvc.exe (только Windows)

Служба Cook для Windows. Конфигурация cookd должна лежать в файле **cookd.yml** в одном каталоге с файлом **cookdsvc.exe**.

Инсталляция выполняется с помощью команды *cookdsvc.exe install*.

Полезно настроить логирование в файл с помощью опции **cook.log_file**, так как логирование в stdout в данном случае невозможно.

Глава 8

Режим обслуживания

Добавлено в v0.1.0.

8.1 О режиме обслуживания

Режим обслуживания позволяет временно приостановить полноценную работу кластера, при этом не теряя состояние master/slave узлов.

Это полезно при обновлениях СУБД и отладке кластера.

В режиме обслуживания выполняются следующие действия:

1. Ручная остановка rdbserver игнорируется и не приведет к demote или отключению slave от кластера
2. При остановке cook не останавливает rdbserver
3. При запуске cook не запускает rdbserver
4. cookd в режиме master игнорирует потерю блокировки в Consul
5. cookd в режиме slave игнорирует потерю master и не выполняет promote
6. cookd в режиме slave подключится к новому master, если он появится в кластере
7. cookd выполнит ручной promote и demote игнорируя любые ограничения

Для того, чтобы перевести кластер в режим обслуживания нужно выполнить команду **cookctl maintenance** с параметром 1.

Опционально можно добавить параметр –**wait**, чтобы дождаться, когда все узлы применят новую конфигурацию и войдут в режим обслуживания:

```
cookctl -c config.yml maintenance 1 --wait
Enabling maintenance mode
```

cookctl status показывает режим обслуживания для всего кластера и отдельно для узлов:

```
cookctl -c config.yml status

testcluster (c93e02c8-0e7d-49ee-b286-c7195c221b6a, 0.2.0) generation 2 (585866d6-85b0-
↪45e9-8002-a257619e8157), locked by node1, MAINTENANCE

node0 (slave) API URL http://127.0.0.1:5030, RDB 3.0.9.0 on 3050 and AUX 3060 (127.0.0.1/
↪3050), failover enabled, pending restart, MAINTENANCE
    /tmp/cook/node0/database1.fdb (testdb1, 4b2780bd-0d3d-4da3-b037-98fb3b0b9e01) 2:0␣
↪127.0.0.1/3050:testdb1
    /tmp/cook/node0/database2.fdb (testdb2, 010228a8-33d2-4058-bbe5-5b441d1acbbc) 2:0␣
↪127.0.0.1/3050:testdb2
node1 (master) API URL http://127.0.0.1:5031, RDB 3.0.9.0 on 3051 and AUX 3060 (127.0.0.
↪1/3051), failover enabled, pending restart, MAINTENANCE
    /tmp/cook/node1/database1.fdb (testdb1, 4b2780bd-0d3d-4da3-b037-98fb3b0b9e01) 2:0␣
↪127.0.0.1/3051:testdb1
    /tmp/cook/node1/database2.fdb (testdb2, 010228a8-33d2-4058-bbe5-5b441d1acbbc) 2:0␣
↪127.0.0.1/3051:testdb2
```

Узлы в режиме обслуживания добавляют (MAINTENANCE) в логах:

```
[node1 (MAINTENANCE)] INFO:2022-04-25 14:34:00,568 - cook.mind:436 - Processing locked␣
↪cluster as master. Lock owned by node1
[node1 (MAINTENANCE)] DEBUG:2022-04-25 14:34:00,569 - cook.mind:458 - Updating data␣
↪state...
[node1 (MAINTENANCE)] DEBUG:2022-04-25 14:34:00,569 - cook.rdb:535 - Updating master␣
↪data position...
[node1 (MAINTENANCE)] DEBUG:2022-04-25 14:34:00,569 - cook.rdb:995 - Selecting data from␣
↪mon$replication
[node1 (MAINTENANCE)] DEBUG:2022-04-25 14:34:00,575 - cook.rdb:538 - Master position of /
↪tmp/cook/node1/database1.fdb (testdb1, 4b2780bd-0d3d-4da3-b037-98fb3b0b9e01) is 2:0
[node1 (MAINTENANCE)] DEBUG:2022-04-25 14:34:00,576 - cook.rdb:995 - Selecting data from␣
↪mon$replication
[node1 (MAINTENANCE)] DEBUG:2022-04-25 14:34:00,581 - cook.rdb:538 - Master position of /
↪tmp/cook/node1/database2.fdb (testdb2, 010228a8-33d2-4058-bbe5-5b441d1acbbc) is 2:0
[node1 (MAINTENANCE)] DEBUG:2022-04-25 14:34:00,582 - cook.mind:462 - Updating master␣
↪lock...
[node1 (MAINTENANCE)] DEBUG:2022-04-25 14:34:00,582 - cook.arbiter:281 - Updating master␣
↪lock
[node1 (MAINTENANCE)] DEBUG:2022-04-25 14:34:00,582 - cook.arbiter:283 - Renewing consul␣
↪session...
[node1 (MAINTENANCE)] DEBUG:2022-04-25 14:34:00,585 - cook.mind:464 - Master lock and␣
↪cluster position updated
[node1 (MAINTENANCE)] DEBUG:2022-04-25 14:34:00,585 - cook.mind:197 - Updating my status␣
↪in cluster
[node1 (MAINTENANCE)] DEBUG:2022-04-25 14:34:00,606 - cook.mind:197 - Updating my status␣
↪in cluster
[node1 (MAINTENANCE)] DEBUG:2022-04-25 14:34:05,544 - cook.mind:64 - Node step
[node1 (MAINTENANCE)] DEBUG:2022-04-25 14:34:05,545 - cook.mind:197 - Updating my status␣
↪in cluster
```

Для того, чтобы перевести кластер в нормальный режим нужно выполнить команду **cookctl maintenance** с параметром 0.

```
cookctl -c config.yml maintenance 0 --wait
Disabling maintenance mode
```

8.2 Пример использования режима обслуживания

При выполнении обновления RedDatabase необходимо остановить сервер СУБД, кластера это означает остановка сервиса cook и, если обновление выполняется на master, то demote текущего и promote нового. Для того чтобы этого избежать можно:

1. Перевести cookd в режим maintenance
2. Остановить rdbserver вручную
3. Обновить дистрибутив RedDatabase
4. Запустить rdbserver вручную
5. Перевести cookd в нормальный режим

Глава 9

Образ Docker

Пример запуска Consul и узла Cook:

```
docker run -i -t --name=consul \
                 --net cook \
                 -p 8500:8500 \
                 -v consul_data:/consul/data \
                 -e CONSUL_BIND_INTERFACE=eth0 \
                 consul agent -log-level debug -server -data-dir=/consul/data -client=0.
↪0.0.0 -bootstrap -ui
```

```
docker run -i -t --name=node0 \
                 --hostname=node0 \
                 --net cook \
                 -p 3050:3050 \
                 -p 5030:5030 \
                 -e COOK_CLUSTER_NAME=cook-cluster \
                 cook
```

Глава 10

REST API

REST API используется, как внутри кластера, так и извне. Внутри кластера API в основном используется, как инструмент получения оперативной информации о репликации и получения файлов архива. Извне, такие сервисы, как HA-Proxy или Gobetween могут получать статус узла, для корректного проксирования запросов.

10.1 Внешнее REST API

10.1.1 /master

Код возврата 200, если узел находится в режиме *master* или *503*

10.1.2 /slave

Код возврата 200, если узел находится в режиме *slave* или *503*

10.1.3 /status

Возвращает json со статусом узла:

Пример:

```
{
  "allow_failover": true,
  "api_url": "http://127.0.0.1:5030",
  "databases": {
    "67a215ef-cfb7-4f58-b838-d1486f155a3f": {
      "alias": "testdb1",
      "dsn": "127.0.0.1/3050:testdb1",
```

(continues on next page)

(продолжение с предыдущей страницы)


```
      "generation": 1,
      "guid": "67a215ef-cfb7-4f58-b838-d1486f155a3f",
      "master_guid": "AAF48134-FBEE-4B27-36A4-963DCF345B44",
      "mode": "master",
      "path": "/tmp/cook/node0/database1.fdb",
      "sequence": 0
    },
    "dc76c4f2-a7b5-4863-abda-0b6021eb2482": {
      "alias": "testdb2",
      "dsn": "127.0.0.1/3050:testdb2",
      "generation": 1,
      "guid": "dc76c4f2-a7b5-4863-abda-0b6021eb2482",
      "master_guid": "700F591B-0752-4741-D69F-4568866064CA",
      "mode": "master",
      "path": "/tmp/cook/node0/database2.fdb",
      "sequence": 0
    }
  },
  "maintenance": false,
  "master": true,
  "mode": "master",
  "priority": 0,
  "rdb_aux_port": 3060,
  "rdb_endpoint": "127.0.0.1/3050",
  "rdb_host": "127.0.0.1",
  "rdb_pending_restart": false,
  "rdb_port": 3050,
  "rdb_version": "3.0.10.0",
  "restrict_master": [
    false,
    ""
  ]
}
```


10.2 Внутреннее REST API

10.2.1 /archive/<GUID>/<generation>/<sequence>

Копирование существующего архива с любого узла.

- GUID - GUID базы данных в кластере
- generation - номер поколения архивов
- sequence - порядковый номер архива

Если архив существует, возвращает бинарный поток с файлом архива репликации, иначе - *404*

10.2.2 /database/<GUID>

Копирование базы данных с master.

- GUID - GUID базы данных в кластере

Если база данных готова для копирования, возвращает бинарный поток с файлом базы данных, иначе - *503*

10.2.3 /database/add

Добавление базы данных в кластер с текущего master узла.

В POST запросе должны быть передан json для создания новой базы данных:

```
{
  "alias": "database",
  "database": "/path/to/database",
  "properties": {
    "property": "value"
  },
  "replication_options": {
    "option": "value"
  }
}
```

10.2.4 /database/<GUID>/rm

Удаление базы данных из кластера с текущего master узла.

Глава 11

Ограничения

- Доставка логов через API не гарантирует сохранности данных, так как всегда присутствует разрыв в данных master->лог->архив->slave
- Инициализация реплик выполняется с помощью нулевого уровня nbackup, поэтому в момент инициализации работа с nbackup администратору недоступна
- Инициализация реплик выполняется по-порядку, потому что на только одна сессия nbackup может быть запущена на master

Глава 12

История изменений

12.1 RU

12.1.1 v0.6.4

Выпущена: 10.10.2024

Улучшено:

- Разделенная настройка верификации для входящих и исходящих соединений API и клиента. RS-195992

12.1.2 v0.6.3

Выпущена: 12.09.2024

Улучшено:

- Теперь можно указать необходимые алгоритмы шифрования TLS для сервера API RS-193865

Исправлено:

- Настройка HTTP-таймаутов не применяется при перезагрузке конфигурации RS-179879

12.1.3 v0.6.2

Выпущена: 27.02.2024

Исправлено:

- Скаченные архивы игнорируются при попытке запроса их через REST API RS-165068
- Если в PID файле мусор, то cook не стартует RS-160653
- Падение при обработке невалидного конфига RS-157553

12.1.4 v0.6.1

Выпущена: 13.09.2023

Исправлено:

- Невозможно инициализировать узел методами «as_is» или «shell» RS-147778 RS-147781

12.1.5 v0.6.0

Выпущена: 15.08.2023

Добавлено:

- Команда fixpermissions в cookctl для установки прав на права и каталоги RS-143642

Исправлено:

- Не проверялся идентификатор кластера и хранилища при запуске RS-143641
- Не хватает прав при запуске cookd из текущего каталога RS-142574

12.1.6 v0.5.0

Выпущена: 09.06.2023

Добавлено:

- Команда *shutdown* в cookctl для остановки всех узлов RS-97153
- Команда *recreate* в cookctl для переинициализации данных на узле RS-62521
- Команда *logs ls* в cookctl для перечисления доступных архивов и логов RS-97160
- Команда *logs find* в cookctl для поиска файла архива или лога RS-97160

Исправлено:

- Некорректно завершаются процессы rdbserver при использовании архитектуры Classic RS-99685
- Не проверяется существование каталога репликации с логами master RS-97243
- Некорректно выполняется promote RS-106972 RS-107282
- Узел ошибочно участвует в выборе master, даже если не выполнил процедуру failover RS-107298
- Сломанный кластер при потере связи с Consul RS-107408
- Стандартный конфиг Gobetween оставляет подключения к последнему узлу, если в кластере не найден master RS-139845

- История кластера выводится без сортировки RS-140399

12.1.7 v0.4.0

Выпущена: 21.03.2023

Добавлено:

- Анализ логов firebird.log и replication.log на предмет фатальных для репликации ошибок и вывода узла из репликации RS-55210
- Отображение прогресса копирования базы через nbackup RS-82917
- Регулярные выражения для сопоставления кластерной базы с локальной при инициализации реплики RS-82877
- Генерация конфигурации Gobetween в Consul RS-74865

Изменено:

- Теперь узел аварийно остановится, если обнаружит файлы другого кластера RS-97506
- Теперь стандартный системный путь к конфиг используется в cookctl тоже RS-97394
- Теперь узел ждет готовность каталога репликации и не становится репликой RS-97243

Исправлено:

- Создаются лишние каталоги репликации RS-97393
- При failover к реплике применяется лишний архив RS-97499
- Узел не обновлял состояние при остановке RS-98272
- Логи могли не скачаться при отставании реплики на несколько поколений RS-98274
- Исправлена работа двустороннего TLS между узлами Cook и Consul RS-98386 RS-98385 RS-98383
- Узел пытался синхронизироваться с кластером, даже при отсутствующем Master RS-98142

12.1.8 v0.3.0

Выпущена: 02.09.2022

Добавлено:

- Добавление и удаление баз данных в инициализированном кластере RS-76629
- Приоритет узлов для назначения master при автоматическом failover RS-53643

Изменено:

- Теперь реплики мониторят блокировку master, вместо простого ожидания, для более быстрого старта процесса failover RS-82315
- Пароли из конфигурации маскируются в логах RS-83178
- Узлы останавливают работу, если невозможно восстановить данные или инициализировать реплику

Исправлено:

- Подсистема Ред Базы Данных выставляет флаг необходимости перезагрузки, только если ее параметры были изменены RS-63556

- Исправлена работа команды allowfailover в cookctl RS-82157

12.2 EN

12.2.1 v0.6.4

Released: 10.10.2024

Improved:

- Split configuring of in- and outbound connections verification for API and client. RS-195992

12.2.2 v0.6.3

Released: 12.09.2024

Improved:

- Ability to set TLS ciphers for API server RS-193865

Исправлено:

- HTTP-timeouts not adjusted on config reload RS-179879

12.2.3 v0.6.2

Released: 27.02.2024

Fixed:

- Received archives ignored when trying to download with Cook's REST API RS-165068
- Unable to start cookd if PID-file have garbage RS-160653
- Crash if local config have invalid characters RS-157553

12.2.4 v0.6.1

Released: 13.09.2023

Fixed:

- Unable to initialize node with «as_is» and «shell» methods RS-147778 RS-147781

12.2.5 v0.6.0

Released: 15.08.2023

Added:

- Command fixpermissions for cookctl to fix owner and permissions of files and directories RS-143642

Fixed:

- Check if cluster ID in storage and cluster ID obtained from config are equal. RS-143641
- Change working dir to RBD home before spawning rdbserver RS-142574

12.2.6 v0.5.0

Released: 09.06.2023

Added:

- Command shutdown for cookctl to stop whole cluster RS-97153
- Command recreate for cookctl for reinitialization node data RS-62521
- Command logs ls for cookctl to enumerate all archives and logs on node RS-97160
- Command logs find for cookctl to search log or archive file RS-97160

Fixed:

- Classic architecture processes terminated incorrectly RS-99685
- Master replication directory existence not checked RS-97243
- Promote command executed wrongly RS-106972 RS-107282
- Node can be wrongly promoted even if not properly failovered RS-107298
- Broken cluster if all nodes failed to contact Consul RS-107408
- Default Gobetween config leaving stale node connections even if cluster do not have master RS-139845
- Cluster history not sorted RS-140399

12.2.7 v0.4.0

Released: 21.03.2023

Added:

- Analysis of firebird.log and replication.log for fatal replication errors and node removal from replication RS-55210
- Display of database copying progress via nbackup RS-82917
- Regular expressions for matching a clustered database with a local one during replica initialization RS-82877
- Generation of Gobetween configuration in Consul RS-74865

Modified:

- Now the node will stop abruptly if it detects files from another cluster RS-97506
- Now the standard system path to config is used in cookctl as well RS-97394
- Now the node waits for the replication directory to be ready before becoming a replica RS-97243

Fixed:

- Unnecessary replication directories are created RS-97393
- An extra archive is applied during failover to a replica RS-97499
- Node state not updated during stop RS-98272
- Catchup can fail if slave left behind for several generations RS-98274
- Fix two-way TLS between Cook nodes and Consul RS-98386 RS-98385 RS-98383
- Standalone node tried to sync data even if no master node in cluster RS-98142

12.2.8 v0.3.0

Released: 02.09.2022

Added:

- Ability to add and remove databases from cluster RS-76629
- Ability to set slave priority for automatic failover RS-53643

Changed:

- Passwords masked in log output RS-83178
- Now slaves monitoring master key change instead simple waiting between cluster steps to speed up failover initiation RS-82315
- Stop node if unable to recover data or initialize slave

Fixed:

- RDB subsystem now gets pending restart flag only if RDB config actually changed RS-63556
- Fixed allowfailover in cookctl RS-82157

12.2.9 v0.2.7

Released: 26.08.2022

New:

- Add more node monitoring info into REST API /status output RS-85696

Fixed:

- Properly demote data during consul communication failure RS-86067

12.2.10 v0.2.6

Released: 19.08.2022

Changed:

- Allow to set exclude_without_pk = 0 for master instead hardcoded „1“ RS-85743

12.2.11 v0.2.5

Released: 12.08.2022

Fixed:

- Fixed automatic master and slave recovery if last logs are free RS-85369 RS-85368

12.2.12 v0.2.4

Released: 07.07.2022

Improved:

- Adjust buffering during database copy transmission RS-83484

Fixed:

- Avoid unnecessary recover after replica initialization RS-83484
- Sometimes nbackup unable to connect database because another engine already connected RS-83549

12.2.13 v0.2.3

Released: 24.06.2022

Changed:

- Temporary database files stored in database destination directory instead system temp RS-82923

Fixed:

- Stale rdbserver processes can be skipped and not killed in some cases RS-82862
- Default log level INFO were not applied by default RS-82916
- Slave which failed to initialize not removing databases RS-82936

12.2.14 v0.2.2

Released: 21.06.2022

Changed:

- More compact output in cookctl and expanded with -v/–verbose RS-76708

Fixed:

- Server can not start if server config bigger than 256 bytes RS-82702
- In some cases gstat can return 0 retcode, so check stderr too RS-76714
- Wrong database mode in storage file RS-82229
- Slave sometimes can fail to recover with error «Can not update position of standalone» RS-82316
- Master can fail to replay it's latest log when recovering RS-82701
- Cook can hang when dumping log if log is not file RS-82732

12.2.15 v0.2.1

Released: 06.06.2022

Fixed:

- Windows service could not start on some configurations
- Removed unwanted trace output

Changed:

- Metadata marked by version in which were changed instead current version RS-76711
- Bundle consul, gobetween and nssm into windows archive RS-71321

12.2.16 v0.2.0

Released: 25.04.2022

New:

- Multiple databases support RS-71321

Changed:

- Single lock for initialization/master to speedup init+promote and minimize racing for this locks RS-67771
- Minimize cluster metadata overhead by loading cluster history separately RS-74382
- More compact CLI output RS-66495
- Optimize last lines of logs dumping for large log files RS-73517

12.2.17 v0.1.0

Released: 21.12.2021

Fixed:

- Node will recover all possible archives and logs and will not trust data position in Consul RS-56202 RS-69931
- rdbserver's pid-file can contain wrong process PID after server or container crash RS-71040
- Sometimes new database initialization unsuccessful RS-70725 RS-68045
- Replication connection on slave do not allow remote connections RS-70612
- Unable to initialize cluster with default config RS-69889
- promote can stuck if no nodes to be new master RS-68576
- FDB driver crashing RS-67776

Added:

- Maintenance mode which allows node ignore state of cluster and rdbserver health RS-62798
- User defined way to initialize replica. Added local option **rdb.slave_init** to change replica initialization way (**disable**, **nbackup**, **as_is** and **shell**). Added local option **rdb.slave_init_command** to define shell command for slave initialization RS-68618

- Local options **cook.advertise**/**rdb.advertise** can be used to define external interface for connection RS-69933
- Dumping firebird.log and replication.log on some errors if local option **rdb.dump_logs** is **True** RS-71669
- Cook API port availability can be omitted if **cook.check_port** is **False** RS-68986
- –version argument for utilities RS-67298

Changed:

- Cook docker container now opens AUX port 3051 by default RS-67387
- Bump minimal RDB version to 3.0.8 RS-71584

12.2.18 v0.0.10

Released: 06.05.2021

Fixed:

- Avoid corruption of position file when renaming by using atomic os.rename #62005

Added:

- Cookd will configure **databases.conf** and will add alias with database filename or custom alias **rdb.alias** #62520

Changed:

- Now only master sweeps archives if **rdb.external_archive_transfer** used #59727
- API port checked for availability to listen before starting API thread and node loop #62577
- Option **init.disable** removed. Now node will initialize cluster if init section exists #62803
- New windows binaries built with cx_Freeze #55679

12.2.19 v0.0.9

Released: 13.04.2021

Fixed:

- Fixed wrong cluster position update after promote error #62594
- Fixed init lock loosing during background task #62473
- Better network errors handling #55850

Added:

- Load fbclient library from RDB_HOME variable #62574

12.2.20 v0.0.8

Released: 25.02.2021

Fixed:

- Avoid loosing data on failed master by applying latest log to master itself #60335
- Check failed master last log sequence by reading log files directly to avoid starting server #56169
- Avoid closing nbackup session prematurely #60672
- Avoid unnecessary replication by using -NODBTRIGGERS and -COPY in nbackup #60596
- Fixed replication position when downloading master logs #60592
- Support replication utilities with rdb* prefix #60517
- Fixed replication position when catching up archives of failed master #56159
- Fixed wrong cluster position after unsuccessful master promoting #55709
- Avoid connecting slaves to master before promoting finished #55744

Changed:

- Bump minimal RDB version to 3.0.5.354 #60672

Added:

- Added nbackup stderr if something was wrong during database copy #60720
- New PDF and HTML documentation #55626

12.2.21 v0.0.7

Released: 16.11.2020

Fixed:

- Try to avoid position file corruption on hard reset
- Properly create directories when catching up master
- Do not flush master logs after master failure if **rdb.external_archive_transfer** used
- Fixed leaving cluster if server stopped unexpectedly
- Start server when catching up archives (for windows) and stop after updating status
- Try to avoid incrementing data and cluster position in case failed promoting #55047
- Fixed slave initialization in windows

Changed:

- Now monitoring connection do not fire DB triggers #55063

Added:

- Register aux port (3060 by default) as consul service. Can be changed by local option **rdb.aux_port**
- Ansible based windows demo stand (misc/cook-demo-win)

12.2.22 v0.0.6

Released 28.10.2020

Fixed:

- Avoid multiple cluster initialization on simultaneous nodes start #54826

Added:

- Ansible based demo stand (misc/cook-demo) #54741

12.2.23 v0.0.5

Released 22.10.2020

Fixed:

- Always recover database after leaving cluster #54705
- Do fb_shutdown for fdb on sys.exit to avoid segfault #54651
- Implicitly close nbackup and call on_close handler if NBackupWrapper wasn't closed explicitly #54622
- Handle error response from node without json body
- Do not allow copy database if node is not master

Changed:

- Avoid mixing archives and logs from different clusters. Now all archives and logs resides in subdirectory named as cluster UUID #54628
- Use default config from /etc/cook/cookd.yml on posix systems #54627

Added:

- Allow configure loggers with environment variable: COOK_LOGGERS=cook:INFO,cook.api:DEBUG #54354
- Simple Dockerfile for building image of RDB 3.0.5.1 with cook #54354
- Sample docker-compose scenery with gobetween #54354
- Add –test-config option to cookd for testing configuration file and environment without starting node #54610
- Add sample config to misc/cookd.yml and README #54585

12.2.24 v0.0.4

Released 16.10.2020

Fixed:

- Remove stalled master lock after consul offline/node demote/consul online #54463
- Catch errors on cluster config obtaining and handle exceptions during and do not allow raise exception out of cluster step #54463
- Properly set FDBUtils initialization flag and release it before actual exit #54416
- Do not allow multiple slaves initializing to avoid replication errors #54069
- Do sys.exit on INT/TERM signals if cook not started instead endless looping #54415

Changed:

- Slave now leaves cluster on consul errors #54463
- Stop fence right after demote/stop. We do not need to wait releasing locks after stopping #54463
- Serve flask multithreaded #54457

12.2.25 v0.0.3

Released 09.10.2020

Fixed:

- Fix incomplete archives transferring on linux #54237
- Now if promote failed, cookd will try demote database and stop fencing #54212
- Fix wait for downloads from slave once again

Changed:

- slave_log_directory will be configured automatically only if **rdb.external_archive_transfer** is False #54171
- During async task will not update cluster position

Added:

- Windows service **cookdsvc.exe** #54033
- Allow override archive command with local config option **rdb.archive_command** #54237
- Allow override slave log directory with local config option **rdb.slave_log_directory** #54171
- Add cookd version to cluster metadata and check if cook too old #53274
- Show RDB version in node metadata #53275

12.2.26 v0.0.2

Released 02.10.2020

Fixed:

- Now bool options for firebird.conf and replication.conf generated as 0/1
- Fixed python 3.6 compatibility

Changed:

- Replication archives prefixed with _cook_async_repl_

Added:

- Shared directory replication mode
- Basic fencing capability based on watchdog
- Slaves initialized by nbackup streaming
- Now old archives removed from nodes automatically
- Logging to file
- HA-proxy and Gobetween configs samples

- Windows executable for cookd and cookctl
- Python packages
- Init script and systemd unit file
- Config file now is optional and it's possible to configure node with environment variables

12.2.27 v0.0.1

Released: 04.09.2020