

# Настройка отказоустойчивости компонентов KUMA

Отказоустойчивость реализована встроенным функционалом KUMA для компонентов: Коррелятор и Хранилище ([подробнее про кластер Хранилища](#)). Для компонента Коллектор отказоустойчивость системы достигается за счёт комбинации наложенных средств HAProxy, Агента KUMA или другого балансировщика.

Отказоустойчивость компонентов KUMA можно обеспечить на уровне виртуализации

## Отказоустойчивость Коррелятора и Хранилища

Предварительно устанавливается дублирующий компонент на отдельных от первого ресурсах (процесс установки не будет показан в этой инструкции). Ниже будет показан пример настройки для Коррелятора, для Хранилища процесс аналогичный.

В **Точках назначения** добавляется второй компонент Коррелятора с его API портом для работы.

Основные параметры

Дополнительные параметры

\*Название

[Example] Correlator

\*Тенант

Main

Выключено

\*Тип

correlator

\*URL

test-kuma.sales.lab:7249

URL

test-kuma2.sales.lab:7258

Копировать URL сервиса

+ URL

?

Далее необходимо перейти во вкладку дополнительные параметры, нас интересует параметр **Политика выбора URL (URL selection policy)**, ниже описаны детали каждого метода:

1. **Любой (Any)** - выбор любого доступного узла для отправки событий. Если в начале был выбран узел URL1, то события будут лететь в него до тех пор, пока он доступен, в случае выхода его из строя - будет выбран узел URL2, который тоже будет получать события пока сам не выйдет из строя. И так пока не закончатся активные узлы.
2. **Сначала первый (Prefer First) (рекомендуется для корреляторов)** - события отправляются в URL1 до тех пор, пока сервис (URL1) живой. Если он выйдет из строя, события полетят в URL2. Когда URL1 снова станет активным, поток снова направится в него.
3. **По очереди (Round robin) (рекомендуется для хранилищ и агента KUMA для балансировки коллекторов)** - отправляются события не по одному, а пачками. Количество событий в пачке почти всегда будет уникальным - пачка формируется, либо когда достигнут лимит ее размера, либо когда сработает таймер. Каждый URL получит равное количество пачек.

Параметр **Ожидание проверки работоспособности (Health check timeout)** - задает периодичность запросов Health check.

Основные параметры

Дополнительные параметры

Прокси-сервер

Размер буфера

0

?

Время ожидания

0

?

Размер дискового буфера

0

?

Политика выбора URL

Любой

Сначала первый

По очереди

?

Интервал очистки буфера

0

?

Рабочие процессы

Ожидание проверки работоспособности

0

?

Отладка

Выключить

Дисковый буфер

Включить

Фильтр

Создать

☐ Сохранить фильтр

Условия

И

+ Добавить условие

+ Добавить группу

+ Добавить фильтр

Подробнее про устройство кластера хранилища и компонент keeper - [тут](#)

## Отказоустойчивость Коллектора

Предварительно устанавливается на отдельную машину в качестве наложенного средства балансировщик HAProxy для своей версии ОС (процесс установки не будет показан в этой инструкции). Для Oracle Linux 8.x пакет установки HAProxy можно загрузить по ссылке - [https://rpmfind.net/linux/RPM/centos/8-stream/appstream/x86\\_64/Packages/haproxy-1.8.27-5.el8.x86\\_64.html](https://rpmfind.net/linux/RPM/centos/8-stream/appstream/x86_64/Packages/haproxy-1.8.27-5.el8.x86_64.html)

Также можно использовать агента KUMA для балансировки трафика, подробнее [тут](#).

Настроим балансировку потока событий для двух коллекторов:

<input type="checkbox"/>	<input checked="" type="checkbox"/>	Collector	<a href="#">TEST (TCP/5577)</a>	2.0.0.306	Main
<input type="checkbox"/>	<input checked="" type="checkbox"/>	Collector	<a href="#">TEST HA (TCP/5578)</a>	2.0.0.306	Main

Конфигурация HAпроху будет следующая (путь файла конфигураций по умолчанию - `/etc/haproxy/haproxy.cfg`):

```
defaults
  timeout connect 5s
  timeout client 1m
  timeout server 1m

listen collectorTEST
  bind *:55777
  mode tcp
  balance roundrobin
  server tcp1 <IP_КОЛЛЕКТОРА_1>:5577 check
  server tcp2 <IP_КОЛЛЕКТОРА_2>:5578 backup check
```

Балансировщик прослушивает порт 55777, вы можете поменять конфигурацию порта на свое.

Опция backup используется для того, чтобы использовать этот (сервер с этой опцией) сервер в случае отказа первого.

Для запуска HAпроху:

```
haproxy -D -f /etc/haproxy/haproxy.cfg -p /var/run/haproxy.pid -sf $(cat /var/run/haproxy.pid)
```

Для остановки HAпроху:

```
pkill haproxy
```

Для типа коннектора SQL прокси не подойдет и потребуется написание скрипта для реализации следующего:

- создано 2 микросервиса коллектора SQL с одним конфигом

- установлены на сервера Coll-A и Coll-B
- один микросервис включен, второй - выключен
- у каждого из них сервисов в точке монтирования какая-то сетевая папка /opt/kaspersky/kuma/collector/<ID>/sql/ т.е. стейт (состояния) с позициями, которые уже были загружены из базы будут общими
- скрипт произаодит мониторинг за событиями аудита KUMA и если "микросервис перешел из зеленого в красный" и как только один SQL коллектор на сервере А стал красным - запускаем скриптом микросервис на сервере В и он начнет читать из базы из того же места (где остановился сервис А)

Отказоустойчивость за счет Rsyslog: Управление потоком событий с помощью rsyslog (kaspersky.com)

На компонентах всех комопнентах, где присутствует точка назначения возможно настроить буферизацию (ОЗУ и на диске), в случае если точка назначения недоступна.

## Отказоустойчивое ядро

НА ядра доступно для версии KUMA 2.1 и выше

Подробные детали по этой теме можно прочитать тут: <https://kb.kuma-community.ru/books/ustanovka-i-obnovlenie/page/ustanovka-kuma-versii-ot-21x-s-otkazoustoicivym-iadrom>

В случае выхода из строя ядра, остальные компоненты будут продолжать корректно функционировать, тк у них в памяти сохраняется конфигурация от ядра (нельзя перезагружать компоненты, когда ядро недоступно).

## Отказоустойчивые балансировщики

- Ручное резервирование LB (Load Balancer). можно создать клон виртуальной машины с LB или скопировать конфигурацию на резервный "железный" сервер и в случае необходимости переключить трафик на резерв руками или с помощью какой-либо автоматизации. Возможны длительные задержки с момента обнаружения проблемы с LB до момента переключения трафика на резервный LB. FQDN и IP LB при этом переносятся на резервный хост;

- Можно использовать службу **keepalived**, позволяющая использовать VRRP, пример настройки: [https://docs.oracle.com/en/operating-systems/oracle-linux/6/admin/section\\_sm3\\_svy\\_4r.html](https://docs.oracle.com/en/operating-systems/oracle-linux/6/admin/section_sm3_svy_4r.html). В качестве tcp балансировщиков могут выступать машины с nginx или haproxy. Переключение на резервный балансировщик при этом осуществляется автоматически. Такой подход может оказаться неприменим при размещении хостов с tcp балансировщиками в разных data-центрах, т.к. они используют VRRP;
- Для Nginx можно использовать коммерческую версию, пример - <https://www.nginx.com/products/nginx/high-availability/>;
- Отказоустойчивость компонентов также можно обеспечить функционалом систем виртуализации;
- Использование GSLB или других DNS-based балансировщиков "поверх" tcp LB.

---

Revision #16

Created 11 August 2023 10:08:51 by Boris RZR

Updated 16 January 2025 11:56:17 by Boris RZR