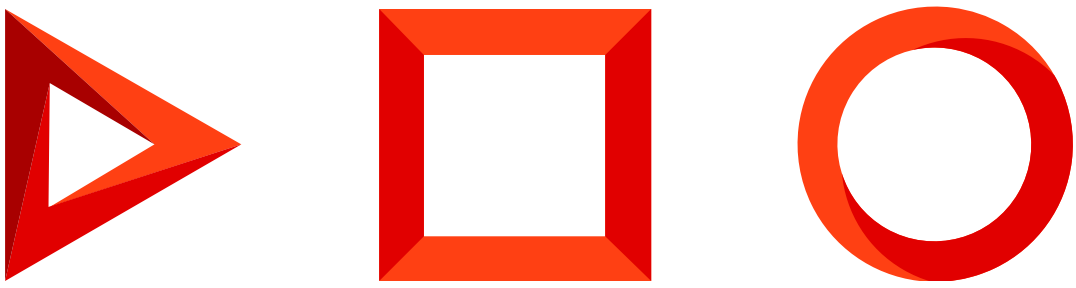
Creatio Academy

Сервер кэширования

Версия 8.0



Эта документация предоставляется с ограничениями на использование и защищена законами об интеллектуальной собственности. За исключением случаев, прямо разрешенных в вашем лицензионном соглашении или разрешенных законом, вы не можете использовать, копировать, воспроизводить, переводить, транслировать, изменять, лицензировать, передавать, распространять, демонстрировать, выполнять, публиковать или отображать любую часть в любой форме или посредством любые значения. Обратный инжиниринг, дизассемблирование или декомпиляция этой документации, если это не требуется по закону для взаимодействия, запрещены.

Информация, содержащаяся в данном документе, может быть изменена без предварительного уведомления и не может гарантировать отсутствие ошибок. Если вы обнаружите какие-либо ошибки, сообщите нам о них в письменной форме.

Содержание

Общий порядок настройки сервера кэширования данных (Redis)

ПРОДУКТЫ: ВСЕ ПРОДУКТЫ

Использование сервера кэширования данных Redis позволит упростить выполнение трудоемких запросов к базе данных. Это ускоряет работу системы и снижает затраты ресурсов.

Пакет Redis доступен в стандартных репозиториях Debian. Ниже описана установка Redis на Debian и производных дистрибутивах, таких как Ubuntu и Linux Mint. Чтобы установить Redis:

1. Войдите в систему как администратор (root):

```
sudo su
```

2. Обновите список пакетов:

```
apt-get update
```

3. Установите Redis:

```
apt-get install redis-server
```

4. Настройте Redis таким образом, чтобы он запускался как системная служба **systemd**. Для этого:

   a. Откройте **redis.conf** в текстовом редакторе от имени пользователя root. Например, для этого можно использовать текстовый редактор Nano:

   ```
   nano /etc/redis/redis.conf
   ```

   b. Найдите запись “**supervised no**”. Замените запись на “**supervised systemd**”.

   c. Сохраните изменения и закройте текстовый редактор.

   d. Перезагрузите сервер Redis:

   ```
   systemctl restart redis-server
   ```

   e. Выйдите из root-сессии:

   ```
   exit
   ```

Настроить Redis Sentinel

ПРОДУКТЫ: ВСЕ ПРОДУКТЫ

Важно. В Creatio версии 7.18.3 будет прекращена поддержка устаревшего механизма Redis Sentinel. Рекомендуем после обновления Creatio до версии 7.18.0 перейти на механизм Redis Cluster.

Отказоустойчивость хранилищ Redis, работающих с Creatio, обеспечивается при помощи механизма Redis Sentinel. Он предоставляет следующие возможности:

- **Мониторинг**. Sentinel следит за тем, чтобы главный (master) и подчиненные (slave) экземпляры Redis работали корректно.
- **Предупреждение**. Sentinel может уведомлять администратора о проблемах с экземплярами Redis.
- **Автоматическое восстановление работоспособности** (failover). Если master-экземпляр Redis не работает как ожидается, то Sentinel может назначить один из slave-экземпляров главным, а другие slave-экземпляры переконфигурировать на работу с новым master. При этом приложения Creatio, использующие Redis, оповещаются о новом адресе соединения с Redis.

Важно. Работа с кластером Redis в Creatio версий 7.17.4 и ниже не поддерживается.

Redis Sentinel является распределенной системой, которая предназначена для работы нескольких экземпляров Sentinel, взаимодействующих друг с другом. Преимущества такой системы:

- Факт отказа подтверждается только когда несколько экземпляров Sentinel, составляющих кворум, соглашаются с недоступностью master-экземпляра Redis. Это уменьшает количество ложных отказов.
- Механизм Sentinel работает, даже если некоторые экземпляры Sentinel неработоспособны. Это повышает отказоустойчивость системы.

Особенности Sentinel

- Для обеспечения отказоустойчивости необходимо не менее трех экземпляров Sentinel, запущенных на разных физических или виртуальных компьютерах. При этом должна быть четкая уверенность, что их отказы вызваны разными причинами.
- Из-за асинхронной репликации распределенная система Sentinel + Redis не гарантирует, что все данные будут сохранены во время отказа.
- Отказоустойчивость настроенной конфигурации должна регулярно проверяться и подтверждаться тестовыми отказами.
- При использовании Docker невозможны некоторые процессы Sentinel, т. к. Docker может перенаправлять порты (описано в блоке "Sentinel, Docker, NAT, and possible issues" документации Sentinel).

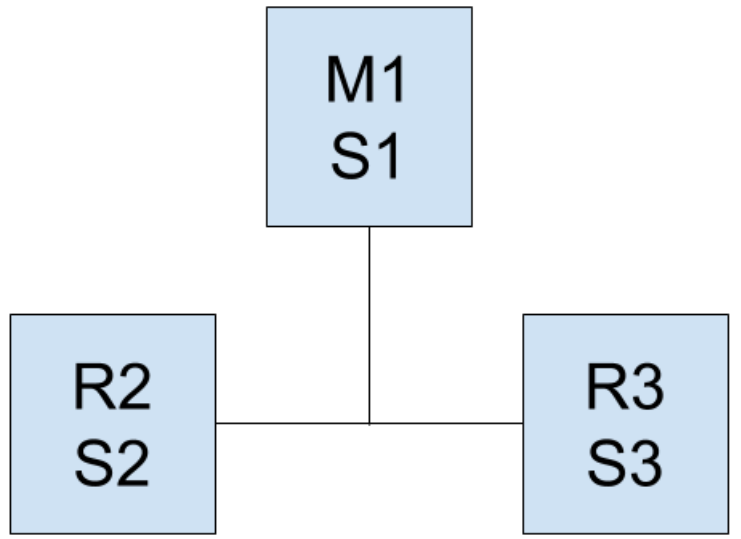
Минимальная отказоустойчивая конфигурация Redis Sentinel

Условные обозначения:

- M1, M2 — master-экземпляры Redis.
- R1, R2, R3 — slave-экземпляры Redis (от Replica).
- S1, S2, S3 — экземпляры Sentinel.
- C1 — клиентское приложение (Creatio).
- [M2] — экземпляр, сменивший свою роль (например из slave на master).

Рекомендуется использовать конфигурацию минимум с тремя экземплярами Redis и тремя экземплярами Sentinel (подробнее о конфигурации читайте в блоке "Example 2: basic setup with three boxes" документации Sentinel). Эта конфигурация основана на трех узлах (физических или виртуальных компьютерах), каждый из которых содержит запущенные экземпляры Redis и Sentinel (Рис. 1). Два экземпляра Sentinel (S2 и S3) составляют кворум (quorum) — количество экземпляров, необходимых для определения отказа текущего master-экземпляра Redis.

Рис. 1 — Конфигурация “трех узлов”: quorum = 2



При нормальной работе конфигурации клиентское приложение (Creatio) записывает свои данные в master-экземпляр M1. Затем эти данные асинхронно реплицируются в slave-экземпляры R2 и R3.

Если master-экземпляр Redis M1 становится недоступным, то экземпляры Sentinel S1 и S2 совместно решают, что произошел отказ и начинают процесс восстановления работоспособности (failover). Они назначают один из slave-экземпляров Redis (R2 или R3) новым master-экземпляром, с которым продолжает работать клиентское приложение.

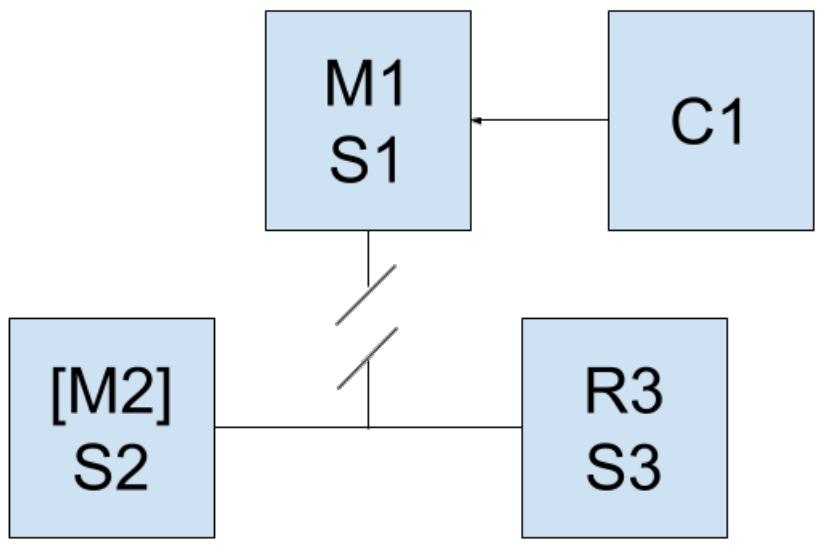
Важно. В любой конфигурации Sentinel, в которой данные реплицируются асинхронно, существует риск потери записей. Это возможно, поскольку данные могут не попасть на slave-экземпляр Redis, ставший новым master-экземпляром.

На заметку. Другие возможные отказоустойчивые конфигурации описаны в документации Sentinel.

Проблема разделения сети

При потере сетевого соединения существует риск, что клиентское приложение C1 продолжит работать со старым master-экземпляром Redis M1, в то время как будет назначен новый master-экземпляр [M2] (Рис. 2).

Рис. 2 — Разделение сети



Подобной ситуации можно избежать, включив опцию остановки записи данных, если master-экземпляр обнаруживает уменьшение количества slave-экземпляров. Для этого можно, например, установить следующие значения в конфигурационном файле redis.conf master-экземпляра Redis:

```
min-slaves-to-write 1
min-slaves-max-lag 10
```

В результате master-экземпляр Redis M1 перестанет принимать данные через 10 секунд, если не сможет их передать как минимум одному slave-экземпляру. После восстановления работоспособности системы экземплярами Sentinel, составляющими кворум (S2 и S3), клиентское приложение C1 будет переконфигурировано для работы с новым master-экземпляром [M2].

Важно. После остановки master-экземпляр не сможет автоматически продолжить работу, если сеть будет восстановлена. Если оставшийся slave-экземпляр Reids (R3) также станет недоступным, то система полностью прекратит работу.

Системные требования

Redis является размещаемой в памяти базой данной (in-memory database). Поэтому основным требованием является скорость работы и объем оперативной памяти. Кроме того, поскольку Redis является однопоточным приложением и нагружает только одно ядро процессора, то для работы одного экземпляра Reids необходим узел (физический или виртуальный компьютер) с двухъядерным процессором, как минимум. Экземпляры Sentinel потребляют относительно немного ресурсов и могут работать на одном узле с Redis.

Redis и Sentinel рекомендуется разворачивать на операционных системах Linux.

В таблице приведены рекомендуемые системные требования для одного узла (физической или виртуальной машины) в зависимости от количества пользователей Creatio.

Количество пользователей	**ЦПУ (CPU)**	**Оперативная память**
1-300	Intel Xeon E3-1225v5	2 Гб
300-500		4 Гб
500-1000		6 Гб
1000-3000		12 Гб
3000-5000		18 Гб
5000-7000		26 Гб
7000-10000		36 Гб

Установить и настроить Redis Sentinel

Redis Sentinel поставляется вместе с дистрибутивом Redis. Процесс установки описан в документации Redis. Для установки следует использовать новейшую версию Redis на дату релиза Creatio.

Пример настройки конфигурации Redis Sentinel приведен в разделе "A quick tutorial" документации Sentinel.

Настроить Creatio для работы с Redis Sentinel

Адаптированные библиотеки и настройка ConnectionStrings.config

Для получения адаптированных библиотек обратитесь в службу технической поддержки Creatio.

В строке соединения **"redis"** укажите:

- sentinelHosts — перечисленные через запятую адреса и порты экземпляров Sentinel в формате

<адрес>:<порт>. Количество может быть любым.

- masterName — имя master-экземпляра Redis.

Пример строки соединения:

```
<add name="redis" connectionString="sentinelHosts=localhost:26380,localhost:26381,localhost:2638
```

Web.config

Убедитесь, что в секции **appSettings** используются рекомендуемые параметры:

- Feature-UseRetryRedisOperation — включает внутренний механизм Creatio для повтора операций с Redis, которые завершились с ошибкой.
- SessionLockTtlSec — срок действия ключа блокировки сессии.
- SessionLockTtlProlongationIntervalSec — период, на который продлевается срок действия ключа блокировки сессии.

Настройки должны иметь следующие значения:

```
<add key="Feature-UseRetryRedisOperation" value="true" />
<add key="SessionLockTtlSec" value="60" />
<add key="SessionLockTtlProlongationIntervalSec" value="20" />
```

Убедитесь, что в секции **redis** используются рекомендуемые параметры:

- enablePerformanceMonitor — включает мониторинг времени выполнения операций с Redis. Рекомендуется включать для отладки и поиска проблем. Значение по умолчанию: По умолчанию выключен т. к. влияет на производительность приложения.
- executionTimeLoggingThresholdSec — операции с Redis, которые выполнялись дольше указанного времени, будут записаны в лог. По умолчанию 5 секунд.
- featureUseCustomRedisTimeouts — включает использование таймаутов, указанных в конфигурационном файле. Значение по умолчанию: “отключена”.
- clientRetryTimeoutMs — завершенные с ошибкой операции с Redis повторяются с тем же клиентом и соединением до истечения периода, заданного этим параметром. Используется для устранения ошибок, вызванных кратковременными перерывами в работе сети. При этом нет необходимости получать из пула нового клиента. По умолчанию 4000 миллисекунд.
- clientSendTimeoutMs — время, выделяемое на отправку запроса серверу Redis. По умолчанию 3000 миллисекунд.
- clientReceiveTimeoutMs — время, выделяемое на получение ответа от сервера Redis. По умолчанию 3000 миллисекунд.
- clientConnectTimeoutMs — время, выделяемое на установку сетевого соединения с сервером Redis. По умолчанию 100 миллисекунд.
- deactivatedClientsExpirySec — задержка удаления сбойных Redis-клиентов после их удаления из пула. 0

означает немедленное удаление. По умолчанию 0.

- operationRetryIntervalMs — если внутренний цикл повтора сбойных операций не привел к успешному выполнению операции, то такая операция откладывается на указанное время. После этого операция выполняется с новым клиентом, который, уже может иметь установленное соединение с новым master-экземпляром. По умолчанию 1000 миллисекунд.
- operationRetryCount — количество повторных попыток выполнения операции с новым Redis-клиентом. По умолчанию 10.

Настройки должны иметь следующие значения:

```
<redis connectionStringName="redis" enablePerformanceMonitor="false" executionTimeLoggingThresho
```

Настроить Redis Cluster

ПРОДУКТЫ: ВСЕ ПРОДУКТЫ

Отказоустойчивость хранилищ Redis, работающих с Creatio, обеспечивается при помощи механизма Redis Cluster.

Важно. Работа с Redis Cluster доступна в Creatio версии 7.18.0 и выше.

Отказоустойчивость Redis Cluster

Механизм Redis Cluster работает, даже если большая часть экземпляров Redis неработоспособны. Это достигается благодаря следующим свойствам:

- **Репликация данных**. Для работы Redis Cluster используется асинхронная репликация — над значениями не выполняются операции слияния. Из-за асинхронной репликации система Redis Cluster не гарантирует, что все данные будут сохранены во время отказа.
- **Мониторинг**. Для выполнения своих задач все узлы кластера подключены с помощью шины TCP: каждый экземпляр подключен ко всем остальным узлам кластера с помощью шины кластера. Узлы используют gossip-протокол для распространения информации об изменении в кластере (обнаружении новых экземпляров, проверки связи с существующими экземплярами, отправки других сообщений кластера). Шина кластера также используется для отправки сообщений Pub/Sub по кластеру и организации ручной отработки отказа по запросу пользователей. Отказоустойчивость настроенной конфигурации должна регулярно проверяться и подтверждаться тестовыми отказами.
- **Автоматическое восстановление работоспособности** (failover). Redis Cluster может функционировать, когда часть master-экземпляров не работает как ожидается при условии, что для каждого из недоступных master-экземпляров есть по крайней мере один slave-экземпляр. Благодаря миграции реплик master-экземпляры, которые больше не реплицируются ни одним slave-экземпляром, получат новый slave-экземпляр от master-экземпляра, который обслуживается несколькими slave-экземплярами. При этом приложения Creatio, использующие Redis, переконфигурируют соединение согласно состояния Redis Cluster. Для обеспечения

отказоустойчивости необходимо не менее шести экземпляров Redis, запущенных на разных физических или виртуальных компьютерах.

- **Масштабирование**. Механизм Redis Cluster позволяет добавлять и удалять узлы во время работы кластера. Redis Cluster можно масштабировать до 1000 экземпляров благодаря автоматическому шардированию данных.

Системные требования Redis Cluster

Redis Cluster разворачивается на операционных системах Linux с Redis Server версии 4.0 и выше.

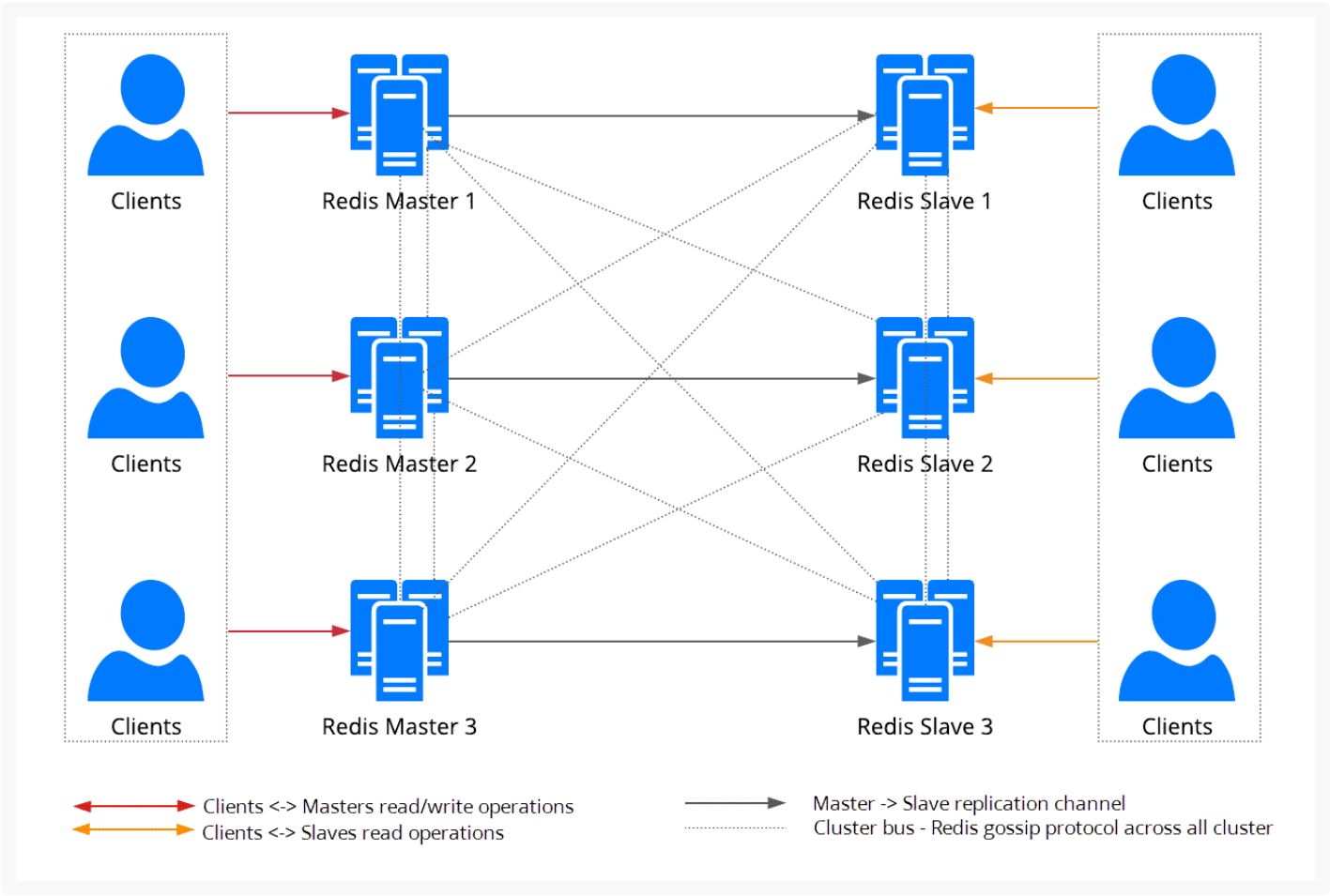
Redis является однопоточным приложением и нагружает только одно ядро процессора, поэтому для работы **одного экземпляра** Redis необходим узел (физический или виртуальный компьютер) с двухъядерным процессором, как минимум. Для обеспечения **максимальной отказоустойчивости** рекомендуется использовать количество физических машин, равное количеству экземпляров Redis, чтобы каждая связка master-slave была распределена на разные физические сервера. Для расчета требований к серверам воспользуйтесь калькулятором системных требований.

Установка и настройка Redis Cluster

Минимальная отказоустойчивая конфигурация Redis Cluster

Рекомендуется использовать конфигурацию минимум с шестью экземплярами Redis (подробнее о конфигурации читайте в блоке "Creating and using a Redis Cluster" документации Redis Cluster). Эта конфигурация основана на шести узлах (физических или виртуальных компьютерах), каждый из которых содержит запущенные экземпляры Redis (Рис. 1).

Рис. 1 — Конфигурация Redis Cluster из шести узлов



В штатном режиме slave- экземпляры работают только на чтение. Данные на них асинхронно реплицируются из master-экземпляров. Master-экземпляры работают и на чтение, и на запись. В случае если на узел попадает команда с ключом, который находится в другом узле, то возвращается информация о том, на каком узле необходимо выполнить операцию.

Если один из master-экземпляров Redis становится недоступным, то начинается процесс восстановления работоспособности (failover). В рамках этого процесса один из slave-экземпляров Redis становится новым master-экземпляром, с которым продолжает работать клиентское приложение.

Важно. В любой конфигурации, в которой данные реплицируются асинхронно, существует риск потери записей. Это возможно, поскольку данные могут не попасть на slave-экземпляр Redis, ставший новым master-экземпляром. Автоматическая переконфигурация, в рамках которой назначается новый master-экземпляр, может занять до 15 секунд.

Установка Redis Cluster

Redis Cluster поставляется вместе с дистрибутивом Redis. Для установки следует использовать новейшую версию Redis на дату релиза Creatio.

Процесс установки описан в документации Redis. Пример настройки конфигурации Redis Cluster приведен в разделе “Creating and using a Redis Cluster”.

Для **мониторинга состояния** кластера при подключении к одному из узлов рекомендуется выполнять

следующие проверки:

- **посмотреть конфигурацию кластера** с помощью команды `cluster nodes`.
- **проверить общую работоспособность** кластера при помощи redis-cli: `redis-cli --cluster check ClusterIP` где “ClusterIP“ — это IP-адрес одного из узлов кластера.

Настройка Creatio для работы с Redis Cluster

1. В файле `ConnectionStrings.config` необходимо указать адреса узлов Redis Cluster:

   ```
   <add name="redis" connectionString="clusterHosts=ClusterIP1,ClusterIP2,ClusterIP3,ClusterIP4,
   ```

   где ClusterIP1–ClusterIPn — это IP-адреса узлов кластера.

2. Убедитесь, что в вашем приложении включена функциональность `Feature-UseRetryRedisOperation`. Она запускает внутренний механизм Creatio для повтора операций с Redis, которые завершились с ошибкой. Эта проверка выполняется:

   a. Для приложений **Net Framework** (Windows) в файле `web.config`, который находится в корневой папке приложения;

   b. Для приложений **.NET Core** (Linux) в файле `Terrasoft.WebHost.dll.config`.

   ```
   <add key="Feature-UseRetryRedisOperation" value="true" />
   ```

3. Убедитесь, что в секции **redis** файлов `web.config` (Net Framework) и `Terrasoft.WebHost.dll.config` (.NET Core) используются рекомендуемые параметры:

   - **enablePerformanceMonitor** — включает мониторинг времени выполнения операций с Redis. Рекомендуется включать для отладки и поиска проблем. По умолчанию выключен т. к. влияет на производительность приложения.
   - **executionTimeLoggingThresholdSec** — операции с Redis, которые выполнялись дольше указанного времени, будут записаны в лог. По умолчанию 5 секунд.
   - **clientConnectTimeoutMs** — время, выделяемое на установку сетевого соединения с сервером Redis. По умолчанию 5000 миллисекунд.
   - **clientSyncTimeoutMs** — время, выделяемое на выполнение синхронных операций Redis. По умолчанию 5000 миллисекунд.
   - **clientAsyncTimeoutMs** — время, выделяемое на выполнение асинхронных операций Redis. По умолчанию 5000 миллисекунд.
   - **operationRetryIntervalMs** — если внутренний цикл повтора сбойных операций не привел к успешному выполнению операции, то такая операция откладывается на указанное время. После этого операция выполняется с новым клиентом, который уже может иметь установленное соединение с новым master-экземпляром. По умолчанию 5000 миллисекунд.
   - **operationRetryCount** — количество повторных попыток выполнения операции с новым Redis-

клиентом. По умолчанию 25.

Настройки должны иметь следующие значения:

```
<redis connectionStringName="redis" enablePerformanceMonitor="false" executionTimeLoggingThre
```