

# Redis Sentinel

Настроить Redis Sentinel

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

<b>Настроить Redis Sentinel</b>	<b>4</b>
Особенности Sentinel	4
Минимальная отказоустойчивая конфигурация Redis Sentinel	5
Проблема разделения сети	6
Системные требования	7
Установить и настроить Redis Sentinel	7
Настроить Creatio для работы с Redis Sentinel	7

# 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](#)).

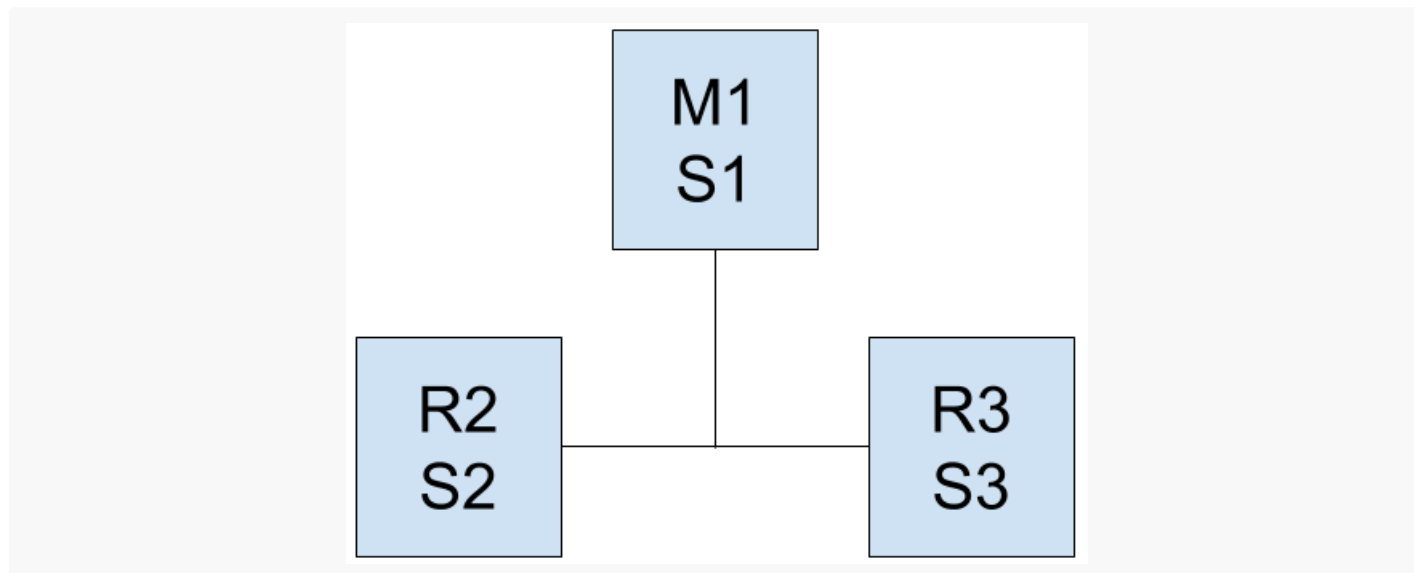
# Минимальная отказоустойчивая конфигурация Redis Sentinel

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.

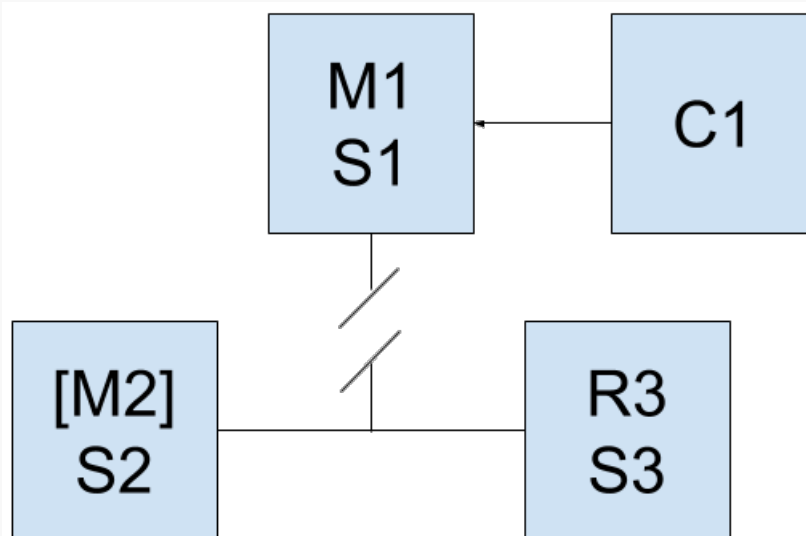
**Важно.** В любой конфигурации Sentinel, в которой данные реплицируются асинхронно, существует риск потери записей. Это возможно, поскольку данные могут не попасть на slave-экземпляр Redis, ставший новым master-экземпляром.

**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:26382"
```

## 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" executionTimeLoggingThreshc
```