

Установка в режиме высокой доступности (HA)

- 1. Подготовка кластера Kubernetes.
 - Использование в Платформе
 - Отказоустойчивость
 - Установка (Bare-Metal)
 - Системные требования
 - Особенности конфигурации
 - Логирование и мониторинг
 - Масштабируемость
 - Развертывание кластера
- 2. Установка смежных зависимостей.
 - 2.1. MongoDB
 - Критерии выбора
 - Использование в Платформе
 - Отказоустойчивость
 - Установка
 - Системные требования
 - Сетевая доступность
 - Конфигурационные файлы
 - Масштабируемость
 - Руководство по установке от разработчика
 - 2.2. Redis
 - Использование в Платформе
 - Отказоустойчивость
 - Установка (Bare-Metal)
 - Системные требования
 - Сетевая доступность
 - Конфигурационные файлы
 - Масштабируемость
 - Руководство по установке от разработчика
 - 2.3. RabbitMQ
 - Использование в Платформе
 - Отказоустойчивость
 - Установка (Bare-Metal)
 - Системные требования
 - Сетевая доступность
 - Конфигурационные файлы
 - Руководство по установке от разработчика
 - 2.4. File Storage
 - Использование в Платформе
 - Установка
 - Отказоустойчивость
 - Руководство по установке от разработчика Minio
- 3. Установка Платформы
 - 3.1 Миграция Low-Code ресурсов платформы
 - 3.2 Установка сервисов платформы с помощью Helm
 - Установка Kubectrl и Helm
 - Добавление репозитория с чартами платформы
 - Получение Docker образов сервисов
 - Корректировка Helm-Values для развертывания
 - Описание Values
 - Установка и обновление платформы
 - Удаление приложения

1. Подготовка кластера Kubernetes.

Использование в Платформе

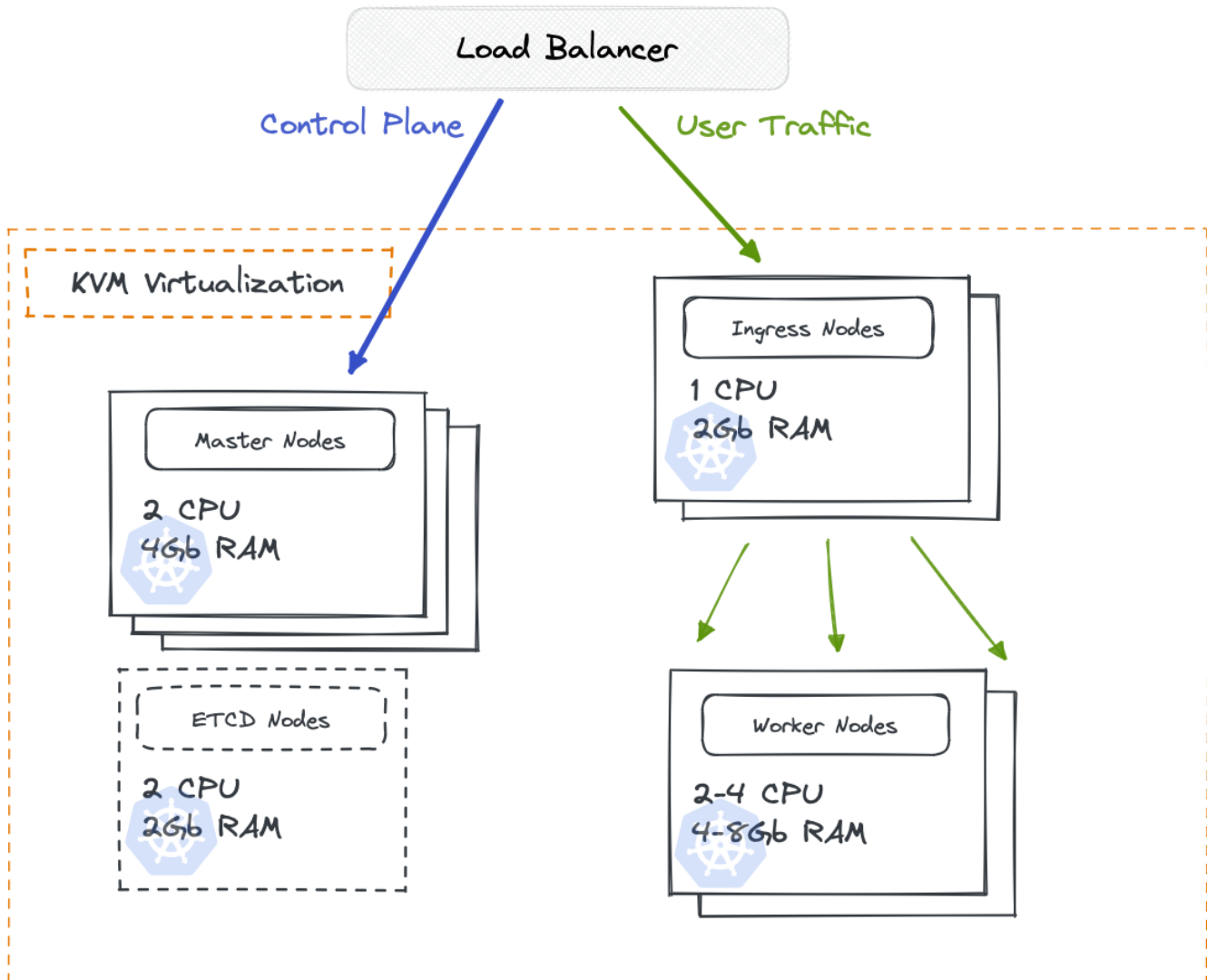
Базовые сервисы Платформы развертываются в Docker-контейнерах, в качестве оркестрации контейнеризированных приложений используется Kubernetes.

Отказоустойчивость

Отказоустойчивость кластера Kubernetes обеспечивается наличием 3 мастер нод, расположенных на 3 разных виртуальных машинах, которые в свою очередь расположены в разных зонах доступности (дата центрах) облачного хостинг-провайдера. Рабочие ноды кластера также располагаются в разных зонах доступности, что позволяет исключить одновременную недоступность всех нод, в случае проблем в одном из ДЦ. При развертывании on-premise рекомендуется разворачивать кластер на разных физических серверах. Отказоустойчивость сервисов обеспечивается за счет механизмов самоконтроля кластера Kubernetes и планировщика (kube-scheduler), который перераспределяет поды сервисы между нодами.

Установка (Bare-Metal)

Системные требования



Минимальная конфигурация нод:

Тип Ноды	Число Нод	CPU, cores	RAM, Gb	Диски, Gb	Комментарий
Master	3	2	4(2)	20GB	Рекомендуется использовать 3 и более. Нечетное количество нод. В случае вынесения ETCD отдельными нодами RAM может быть уменьшена до 2GB
Ingress	3	1	2	20GB	Ноды с Ingress-Controller, используемые для обработки запросов извне кластера.
Worker	4	4	8	20GB SYS + 30GB Data	
ETCD	3	2	2	20GB	

Особенности конфигурации

- Для обеспечения равномерной загрузки нод в кластере рекомендуется организовывать балансировку трафика с помощью внешнего LoadBalancer. Например Nginx, который настраивается на маршрутизацию клиентского трафика на Ingress ноды, а служебного трафика на Master ноды.
- В качестве Ingress-Controller устанавливается Nginx Ingress Controller
- В качестве сетевого плагина используется Calico

- Для обеспечения отказоустойчивости при высокой нагрузке на кластер рекомендуется устанавливать резервирование для kube и системных процессов на Worker нодах в размере 10-15% от CPU и RAM

Логирование и мониторинг

Логирование событий в кластере осуществляется с помощью стека Elasticsearch + Filebeat + Kibana
Для сбора логов в отдельном неймспейсе с помощью Helm разворачивается DaemonSet [Filebeat](#)

Для мониторинга используется Helm пакет [kube-state-metrics](#), включающий в себя компоненты мониторинга Prometheus + необходимые экспортеры.

Масштабируемость

При развертывании сервисов в Kubernetes предусмотрена возможность автоматического горизонтального масштабирования (HPA) на основании объема ресурсов, потребляемых подами.

Развертывание кластера

В качестве системы развертывания production-ready кластера Kubernetes можно воспользоваться open-source решением [Kubespray](#), руководствуясь прилагаемыми в репозитории инструкциями.

2. Установка смежных зависимостей.

При установке платформы вы можете выбрать установку баз данных непосредственно в кластер Kubernetes, либо установить их самостоятельно в качестве независимых компонентов на отдельных серверах либо виртуальных машинах. В случае использования внешних установок можно использовать приведенные ниже рекомендации.

2.1. MongoDB

Критерии выбора

Главный критерий выбора СУБД MongoDB - особенность low-code разработки. Мы заранее не знаем конечную схему БД, которую описывают аналитики при реализации информационной системы на базе Платформы. Данная СУБД имеет динамическую схему БД, которая в совокупности с graphql-схемой позволяют одновременно и иметь легкий и гибкий процесс описания и внесения изменений в схему БД, так и иметь строгую типизацию хранимых данных.

Помимо этого важными критериями являются:

- мощный и богатый язык для формирования агрегации данных(map-reduce)
- наличие мультидокументной ACID-транзакционности
- простота организации горизонтального масштабирования
- простота организации шардирования
- стабильность работы с большими данными
- активное и быстрорастущее community, имеет поддерживаемые opensource-решения для работы с СУБД(в том числе в связке с graphql)

Использование в Платформе

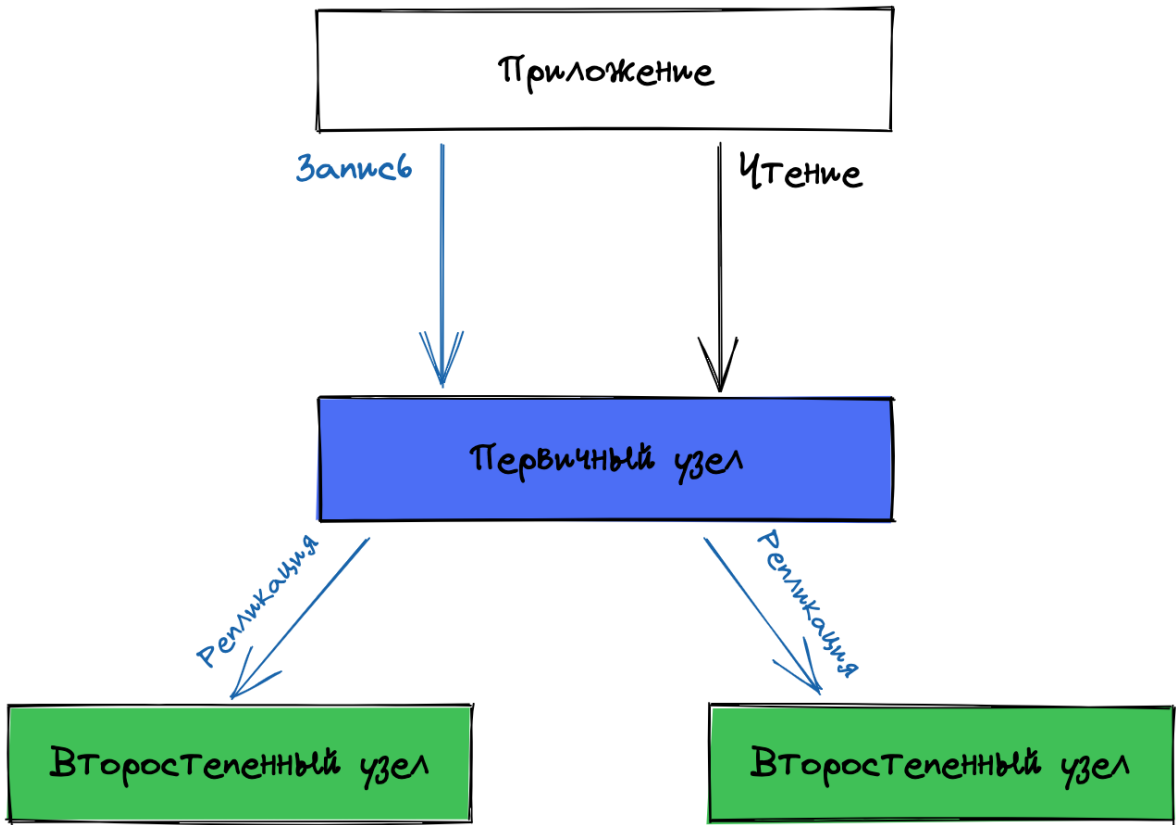
Используется как основная СУБД для проекта, в ней хранятся:

- все данные, которые пишут и используют сервисы Eftch.Factory:
 - состояния жизненных циклов, история переходов (workflow)
 - данные аутентификации, данные по пользователям, организациям, ролям, полномочиям (api-proxy)
 - данные по рассылкам (notification)
- все пользовательские данные, формируемые информационной системой, реализованной на базе Eftch.Factory.

Основная причина, почему решено хранить не только пользовательские, но и данные самих сервисов, а также данные пользователей, организаций, в MongoDB - возможность изменения схемы этих данных аналитиками инструментами lowcode. Мы позволяем аналитику гибко настраивать любой компонент системы.

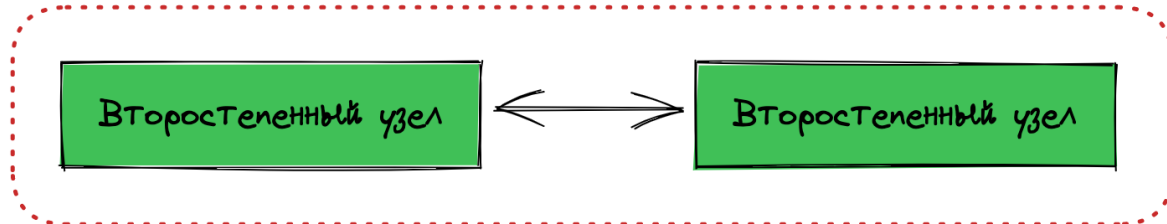
Отказоустойчивость

В базовом сценарии развертывания Платформы, поднимается набор реплик MongoDB (replica set), состоящий из 3 нод. При подключении из программного кода к СУБД указываются все сервера, состоящие в replica set, который самостоятельно определяет какая из нод в данный момент является мастером, а какая репликой. При падении ведущей ноды, второстепенные узлы, при наличии кворума, определяют мастер-ноду, поэтому не происходит прерывания в процессе работы с кластером MongoDB.

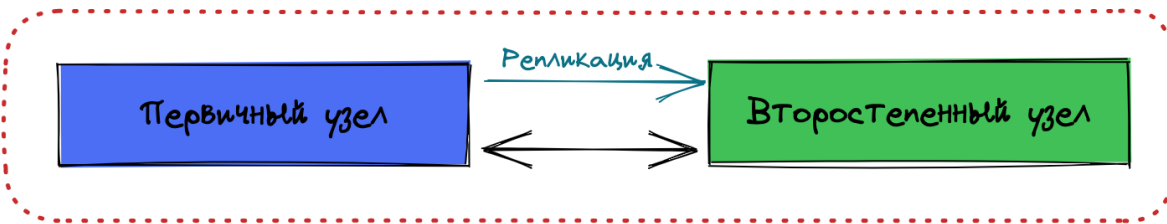




Выборы нового первичного узла



Выбран новый первичный узел



Установка

Для MongoDB требуется развёртывание ReplicaSet из трёх узлов (конфигурация Primary-Secondary-Secondary):

Системные требования

Нода	ПО	CPU, cores	RAM, Gb	SSD, Gb
mongodb-node-1	MongoDB 6	4	8	100
mongodb-node-2		4	8	100
mongodb-node-3		4	8	100

Сетевая доступность

Правила файрвола разрешают входящие подключения к следующим портам:

Порт	Назначение
21017	Для подключения сервисов Платформы к СУБД MongoDB
9100	node_exporter
9216	mongo_exporter
22	ssh

Конфигурационные файлы

Конфигурационный файл идентичен для всех узлов ReplicaSet:

mongo.conf

```
# mongod.conf
systemLog:
destination: file
logAppend: true
path: /var/log/mongodb/mongod.log

# Where and how to store data.
storage:
dbPath: /var/lib/mongo
journal:
enabled: true

# how the process runs
processManagement:
fork: true # fork and run in background
pidFilePath: /var/run/mongodb/mongod.pid # location of pidfile
timeZoneInfo: /usr/share/zoneinfo

# network interfaces
net:
port: 27017
bindIp: 0.0.0.0 # Enter 0.0.0.0,:: to bind to all IPv4 and IPv6 addresses or, alternatively, use the net.
bindIpAll setting.

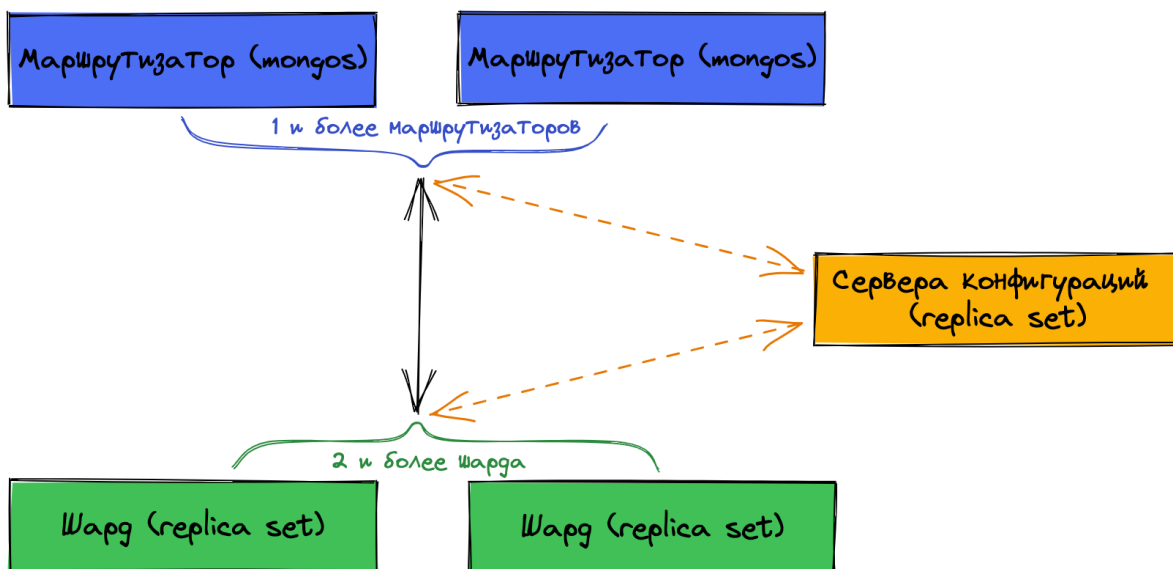
security:
keyFile: /opt/mongo/mongo-keyfile
authorization: enabled

replication:
replSetName: replicaset-01
```

Масштабируемость

При увеличении нагрузки на СУБД, предусмотрен механизм включения шардирования данных по необходимым ключам. В такой схеме кластер СУБД состоит из:

- 3 серверов конфигураций (config server), которые хранят метаданные по шардированию,
- необходимого количества шардов, каждый из которых представляет из себя replica set из 3 нод,
- маршрутизатор (mongos), который принимает запросы и отправляет их по шардам.



Руководство по установке от разработчика

2.2. Redis

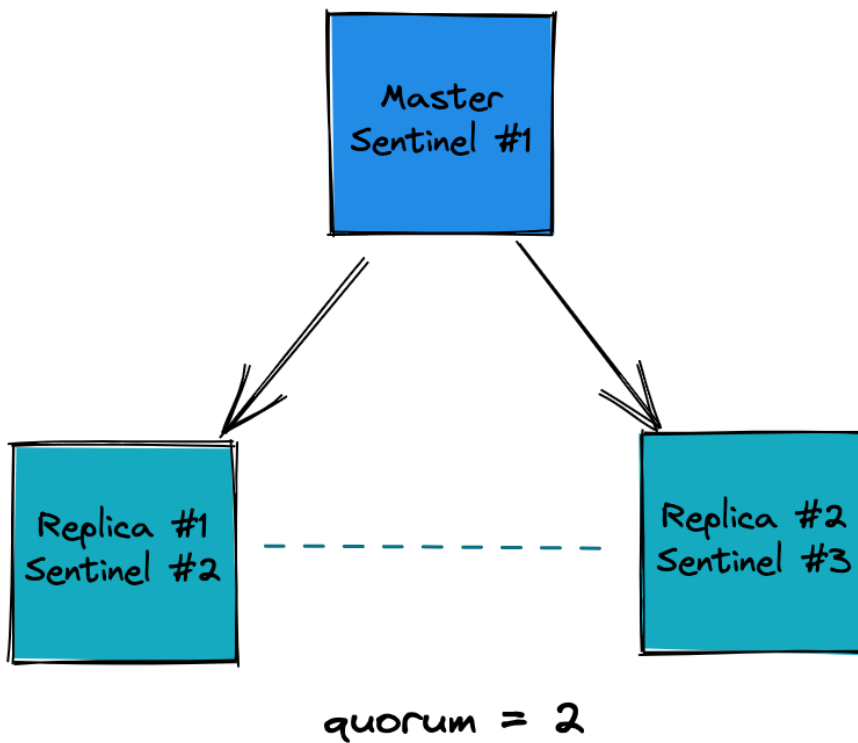
Использование в Платформе

Используется для:

- хранения кеша
- lock-ключей
- задач отложенного выполнения

Отказоустойчивость

Высокую доступность Redis в базовом варианте обеспечивает Redis Sentinel, с развертыванием на 3 нодах – 1 мастер и 2 реплики, с установкой Sentinel на каждую ноду. Если мастер (Master) выйдет из строя, Sentinel #2 и Sentinel #3 подтвердят сбой и перевыберут новый мастер (т.к. кворум в этом случае = 2), что позволит клиентам продолжить работу.



Установка (Bare-Metal)

Системные требования

Нода	ПО	CPU, cores	RAM, Gb	Storage, Gb
redis-master-1	Redis 7.0.5	2	4	50
redis-replica-1		2	4	50
redis-replica-2		2	4	50

Сетевая доступность

Правила файрвола разрешают входящие подключения к следующим портам:

Порт	Назначение
26379	Для подключения сервисов Платформы к Базе данных через Sentinel
6379	Для доступа к API Redis, а также для доступа к внутренним метрикам мониторинга
9100	Метрики Prometheus Node Exporter

Конфигурационные файлы

Конфигурация идентична для всех узлов:

Основной конфиг сервиса:

```
# General
daemonize yes
protected-mode no
pidfile "/var/run/redis/redis.pid"
dir "/var/lib/redis"
port 6379
timeout 0
tcp-keepalive 0
tcp-backlog 511
loglevel notice
logfile ""
syslog-enabled yes
syslog-ident "redis_6379"
syslog-facility user
databases 16

# Snapshotting
save 900 1
save 300 10
save 60 10000
stop-writes-on-bgsave-error yes
rdbcompression yes
rdbchecksum yes
dbfilename "dump.rdb"

# Replication
replica-serve-stale-data yes
replica-read-only yes
repl-disable-tcp-nodelay no
replica-priority 100
min-replicas-max-lag 10
# Security
# Limits
maxclients 10000
maxmemory 3500000000
maxmemory-policy noeviction

# Append Only Mode
appendonly yes
appendfilename "appendonly.aof"
appendfsync everysec
no-appendfsync-on-rewrite no
auto-aof-rewrite-percentage 100
auto-aof-rewrite-min-size 64mb

# Lua
busy-reply-threshold 5000

# Slow Log
slowlog-log-slower-than 10000
slowlog-max-len 128

# Event Notification
notify-keyspace-events ""

# Advanced
hash-max-listpack-entries 512
hash-max-listpack-value 64
set-max-intset-entries 512
zset-max-listpack-entries 128
zset-max-listpack-value 64
activerehashing yes
```



```
client-output-buffer-limit normal 0 0 0
client-output-buffer-limit replica 256mb 64mb 60
client-output-buffer-limit pubsub 32mb 8mb 60
hz 10
aof-rewrite-incremental-fsync yes
```

Конфигурация Sentinel

```
daemonize yes
protected-mode no
dir "/var/lib/redis/sentinel"
pidfile "/var/run/redis/sentinel.pid"
port 26379
bind 0.0.0.0
sentinel resolve-hostnames yes

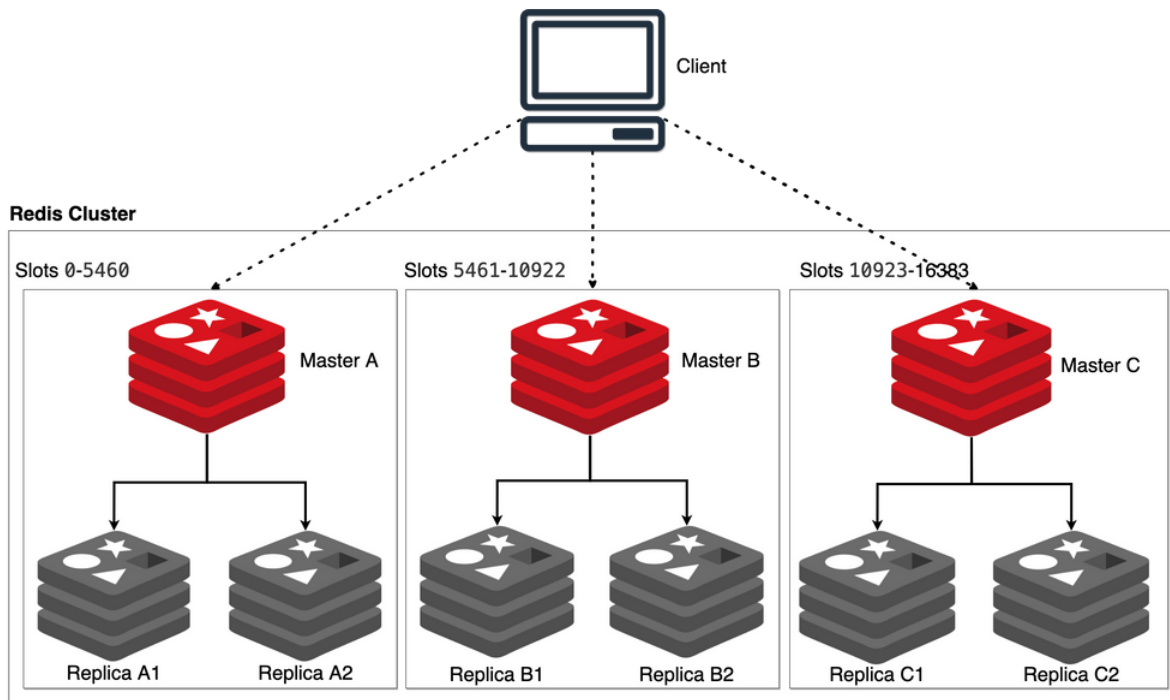
# Security
requirepass <PASSWORD_IF_NEEDED>

sentinel monitor redis-master-1 <IP_MASTER> 6379 2
sentinel auth-pass redis-master-1 <PASSWORD_IF_NEEDED>
sentinel down-after-milliseconds redis-master-1 5000

sentinel failover-timeout redis-master-1 30000
logfile ""
syslog-enabled yes
syslog-ident "sentinel_26379"
syslog-facility user
```

Масштабируемость

При увеличении нагрузки есть возможность развернуть Redis Cluster. В базовом варианте он состоит минимум из 6 нод – 3 мастера и по одной реплике для каждого мастера. Кластер организован по принципу шардирования, т.е. каждый мастер хранит свой набор key-value, основанный на id, поэтому для обеспечения отказоустойчивости и скорости работы, каждый мастер имеет минимум по одной реплике (на изображении вариант с двумя репликами). В данной реализации у каждого мастера может быть любой набор подчиненных серверов. В случае выхода из строя мастера, кворум подчиненных серверов выбирает нового мастера, и кластер продолжает работу без прерываний в предоставлении сервиса.



Руководство по установке от разработчика

Руководство по установке доступно на сайте разработчика по адресу <https://redis.io/docs/getting-started/installation/>

2.3. RabbitMQ

Использование в Платформе

Используется

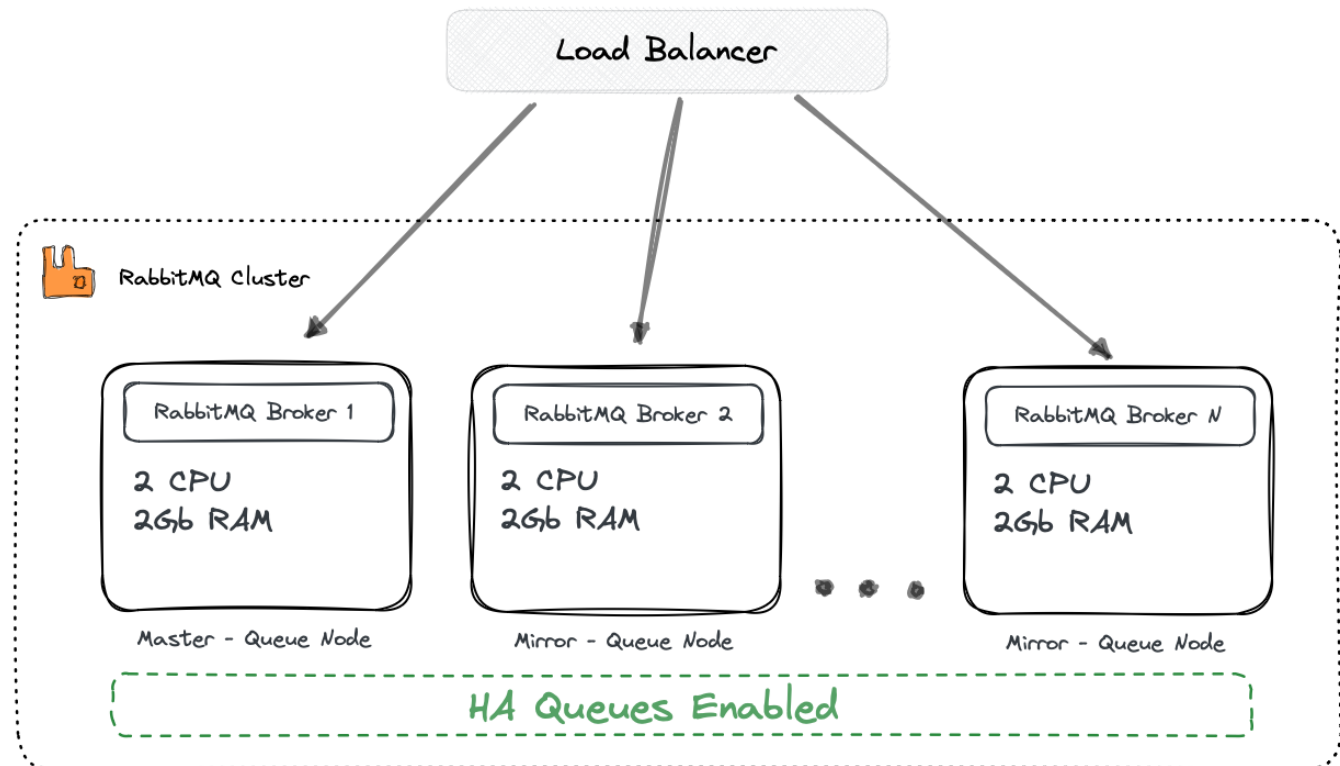
- для организации пользовательских рассылок
- в качестве шины данных для сервисов платформы

Отказоустойчивость

Брокер сообщений RabbitMQ в базовом варианте разворачивается из 3 нод с полностью синхронизируемыми очередями. Для обеспечения отказоустойчивости перед кластером устанавливается балансировщик, который по заложенному алгоритму балансировки распределяет запросы между нодами кластера. Выход из строя любой ноды не приводит к прерыванию в работе сервисов.

Установка (Bare-Metal)

Системные требования



Тип Ноды	ПО	CPU, cores	RAM, Gb	Storage, Gb
rabbitmq-node-1	RabbitMQ 3.10	2	2	20
rabbitmq-node-2	Erlang 1:24.3.4.1-1	2	2	20
rabbitmq-node-3		2	2	20

Сетевая доступность

Правила файрвола разрешают входящие подключения к следующим портам:

--	--

Порт	Назначение
5672	Для подключения сервисов Платформы к очередям RabbitMQ
25672	Для взаимодействия нод внутри кластера. Должен быть закрыт извне кластера.
4369	Порт для epmd discovery демона
35672-35682	Для взаимодействия нод внутри кластера при использовании динамического диапазона портов. Должны быть закрыты извне кластера.
15672	Порт доступа при использовании RabbitMQ HTTP API и Web клиента.
15692	Порт для получения встроенных метрик Prometheus
22	ssh

Конфигурационные файлы

Настройка HA очередей

```
rabbitmqctl set_policy ha-exactly-three ".*" '{"ha-mode":"exactly","ha-params":3,"ha-sync-mode":"automatic"}'
```

/etc/systemd/system/rabbitmq-server.service.d/limits.conf

```
[Service]
LimitNOFILE=130000
```

Руководство по установке от разработчика

Руководство по установке доступно на сайте разработчика по адресу <https://www.rabbitmq.com/download.html>

2.4. File Storage

Использование в Платформе

Используется для хранения пользовательских файлов в сервисе file. Варианты хранения:

- S3 - объектное хранилище (по умолчанию)
- Local - использует примонтированную директорию (Persistent Volume) в контейнере. За счет Kubernetes Container Storage Interface есть возможность использовать разнообразные драйверы для различных поставщиков, список поддерживаемых драйверов - <https://kubernetes-csi.github.io/docs/drivers.html>

Установка

Возможно использование любое S3-совместимое облачное хранилище, либо использование Self-Hosted решения, например [Minio](#).

Отказоустойчивость

Отказоустойчивость в случае хранения в объектном хранилище обеспечивается за счет поставщика услуг. Например, объекты в Yandex Object Storage хранятся в нескольких географически распределённых зонах доступности. При редактировании, создании или удалении объектов меняется каждая копия.

В случае примонтированной директории отказоустойчивость обеспечивается за счет файловой системы либо дисковой подсистемы, подключаемого хранилища.

Руководство по установке от разработчика Minio

Руководство по установке доступно на сайте разработчика по адресу <https://min.io/docs/minio/linux/operations/installation.html>

3. Установка Платформы

3.1 Миграция Low-Code ресурсов платформы

Конфигурация и функционал платформы определяется Low-Code ресурсами конкретного пакета установки. Ресурсы мигрируются из NPM пакета в БД Redis, развернутую на предыдущих этапах.

Описание переменных для проведения миграции:

- ENV_NAME - имя окружения, в котором поднят сервис (При установке в Kubernetes - Имя Namespace)
- REDIS_MODE=single/sentinel - (по умолчанию=single) режим работы с редис
- REDIS_HOST=localhost - (для single) адрес redis
- REDIS_PORT=6379 - (для single) порт redis
- REDIS_DB=0 - (для single) бд redis
- REDIS_SENTINELS - (для sentinel) адреса sentinel в формате: host1:port1,host2:port2
- REDIS_SENTINEL_MASTER - (для sentinel) имя мастера sentinel
- REDIS_SENTINEL_PASSWORD - (для sentinel) пароль подключения к sentinel(при необходимости)

Процесс установки:

1. Скачайте пакет необходимой версии с Low-Code ресурсами. Для скачивания потребуются предварительно установленная утилита npm



Для доступа в репозиторий пакетов необходимо пройти авторизацию в registry, используя предварительно полученные учетные данные у представителя компании.

```
npm set //nexus.effective-group.ru/repository/npm-hosted/:_authToken <REGISTRY_AUTH_TOKEN>
npm set registry http://nexus.effective-group.ru/repository/npm-hosted/
npm install @et-app/proq@<VERSION> --prefix ./
```

2. С помощью утилиты et-configurator, включенной в скачанный пакет проведите миграцию ресурсов, предварительно установив переменные для корректного подключения к БД Redis:

```
ENV_NAME=example-namespace \
REDIS_MODE=sentinel \
REDIS_SENTINELS=redis-host-1.example.com:26379,redis-host-2.example.com:26379,redis-host-3.example.com:26379 \
REDIS_SENTINEL_MASTER=redis-host-1 \
REDIS_SENTINEL_PASSWORD=<REDIS_SENTINEL_PASSWORD> \
./node_modules/.bin/et-configurator apply-app @et-app/proq@<VERSION>
```

3.2 Установка сервисов платформы с помощью Helm

Для установки сервисов Платформы в кластер Kubernetes используется пакетный менеджер Helm.



Установочные команды приведены для версии Helm v3.

Установка Kubectl и Helm

Kubectl - <https://kubernetes.io/docs/tasks/tools/>

Helm - <https://helm.sh/docs/intro/install/#from-script>

Добавление репозитория с чартами платформы

Для установки платформы с помощью Helm чартов требуется первично добавить репозиторий с чартами, пройти авторизацию, в случае если репозиторий приватный, предварительно получив имя пользователя и пароль доступа.

Репозитории могут быть 2 типов, HTTP и OCI совместимые.

Для добавления OCI репозитория достаточно пройти авторизацию с помощью команды:

Добавление OCI репозитория

```
helm registry login --username=<USERNAME> --password=<PASS> harbor.effective-group.ru
```

Для добавления HTTP-совместимого репозитория требуется использовать следующий набор команд:

Добавление HTTP репозитория

```
helm repo add factory --username=<USERNAME> --password=<PASS> https://harbor.effective-group.ru
helm repo update
```

Получение Docker образов сервисов

В процессе установки Helm должен иметь доступ до Docker образов устанавливаемых сервисов.



Перед началом установки необходимо пройти авторизацию в Docker-registry, используя предварительно полученные имя пользователя и пароль у представителя компании.

Добавление HTTP репозитория

```
docker login git.effective-team.ru:4567 -u <USERNAME> -p <PASS>
```

Корректировка Helm-Values для развертывания

После добавления репозитория, требуется осуществить предварительную настройку параметров установки платформы. Для этого с помощью Helm необходимо скопировать базовый набор настроек, и произвести корректировку в файле **factory-values.yaml** в соответствии с установленными зависимыми компонентами.

```
helm show values oci://harbor.effective-group.ru/factory-dev/factory --version 13.0.27 > factory-values.yaml
```

Обязательные параметры:

Имя переменной	Описание	Значение по умолчанию
<code>global.domain</code>	Доменное имя, по которому будет доступна платформа	<code>example.com</code>
<code>global.image.registry</code>	Адрес Docker-Registry, в котором хранятся образы сервисов платформы	<code>git.effective-team.ru:4567</code>
<code>global.image.repository</code>	Репозиторий внутри Registry, в котором хранятся образы сервисов платформы	<code>effectivex</code>
<code>global.mongodb.dsn</code>	Строка подключения к MongoDB вида: <code>mongodb://<USERNAME>:<PASSWORD>@<MONGODB_HOSTS>/<MONGODB_NAME>?replicaSet=<REPLICASET_NAME></code>	
<code>global.rabbitmq.host</code>	Адрес хоста сервиса очередей RabbitMQ	
<code>global.rabbitmq.port</code>	Порт подключения к RabbitMQ	<code>5672</code>
<code>global.rabbitmq.user</code>	Пользователь для авторизации в RabbitMQ	
<code>global.rabbitmq.password</code>	Пароль для авторизации в RabbitMQ	
<code>global.rabbitmq.vhost</code>	Имя Виртуального Хоста в RabbitMQ	
<code>global.redis.mode</code>	Тип подключения к Redis. Поддерживаются Single и Sentinel варианты подключений	<code>sentinel</code>
<code>global.redis.host</code>	При использовании Single подключения - домен или адрес сервера Redis	
<code>global.redis.sentinel.master</code>	При использовании Sentinel подключения - имя мастер инстанса Redis	

<code>global.redis.sentinel.addresses</code>	При использовании Sentinel подключения - адреса или доменные имена Redis Хостов	
<code>global.redis.sentinel.password</code>	При использовании Sentinel подключения с аутентификацией - пароль для подключения к Redis	
Следующие параметры обязательны при использовании хранилища S3: <code>file.defaultStorage=s3</code>		
<code>file.s3KeyId</code>	Идентификатор подключения к S3 хранилищу	
<code>file.s3Key</code>	Ключ подключения к S3 хранилищу	
<code>file.s3Region</code>	Регион облачного хранилища провайдера S3	ru-central1
<code>file.s3Bucket</code>	Имя S3 Бакета	
<code>file.s3Endpoint</code>	Доменное имя провайдера подключения S3 хранилища	https://storage.yandexcloud.net
<code>file.s3RootPath</code>	Путь к папке для записи файлов	/

Описание Values

Помимо основных обязательных настроек Helm чарта существует ряд дополнительных настраиваемых параметров, используемых для более тонкой настройки конкретной инсталляции.

Переменная	Описание	Значение по умолчанию
<code>global.envName</code>	Имя окружения. Используется в БД Redis как префикс для хранения ключей, а так же в логировании базовых сервисов.	test
<code>global.image.pullPolicy</code>	Глобальная политика разворачивания сервисов внутри кластера Kubernetes. Документация .	
<code>global.revisionHistoryLimit</code>	Количество хранимых в памяти ReplicaSet. (Документация). Для больших инсталляций рекомендуется увеличить данный параметр до 10+ для возможности провести откат до определенной версии инсталляции в случае возникновения проблем.	3
<code>global.apiProxyService</code>	Адрес базового сервиса ApiProxy, установленного в кластере Kubernetes	http://api-proxy
<code>global.dataService</code>	Адрес базового сервиса Data, установленного в кластере Kubernetes	http://data
<code>global.workflowService</code>	Адрес базового сервиса Workflow, установленного в кластере Kubernetes	http://workflow
<code>global.fileService</code>	Адрес базового сервиса File, установленного в кластере Kubernetes	http://file
<code>global.cryptoService</code>	Адрес базового сервиса криптографии (при использовании), установленного в кластере Kubernetes либо на внешнем ресурсе.	
<code>global.captchaSiteKey</code>	Ключ использования функции Google Captcha (При наличии)	
<code>global.ingress.protocol</code>	Протокол доступа к web-ресурсам платформы (HTTP или HTTPS)	http
<code>global.ingress.clusterIssuer</code>	При использовании HTTPS и Cert-Manager - наименование сервиса типа ClusterIssuer от которого происходит выпуск сертификатов letsencrypt	letsencrypt-prod
<code>global.ingress.httpsRedirect</code>	Осуществлять ли редирект на HTTPS при обращении к HTTP ресурсам платформы	
<code>global.ingress.robotsTxt</code>	Настройки для обработки запросов Поисковых и системам сбора данных	
<code>global.hpa.enable</code>	Настройка автомасштабирования ресурсов платформы в зависимости от пользовательской нагрузки. Применяется в облачных инсталляциях или системах с динамически расширяемыми нодами кластера Kubernetes.	false
<code>global.hpa.minReplicas</code>	Минимальное количество реплик сервисов, доступное для использования.	1
<code>global.hpa.maxReplicas</code>	Максимально расширяемое количество реплик сервисов, доступных для использования при пиках нагрузки.	2

<code>global.hpa.targetMemoryUtilizationPercentage</code>	Процентное значение потребляемой оперативной памяти (от Limits) при котором будет создана дополнительная реплика сервиса	75
<code>global.hpa.targetCPUUtilizationPercentage</code>	Процентное значение потребляемопроцессорного времени (от Limits) при котором будет создана дополнительная реплика сервиса	80
<code>global.replicaCount</code>	При неиспользовании НРА задает статическое количество реплик каждого сервиса. (Значение глобальное, для каждого сервиса может устанавливаться отдельное значение)	1
<code>global.tracing.enabled</code>	регулирует включение функций трейсинга запросов внутри платформы для осуществления отладочных операций. При включенной функции будет установлен дополнительный open-source сервис Jaeger .	false
<code>global.tracing.jaegerAgentHost</code>	Адрес сервиса агента отслеживания запросов	jaeger
<code>global.tracing.jaegerAgentPort</code>	Порт сбора информации от сервисов, написанных на Node.js	6382
<code>global.tracing.jaegerAgentPortGo</code>	Порт сбора информации от сервисов, написанных на Go	6381
<code>global.timeout</code>	время таймаута запросов при проксировании от api-проху к остальным сервисам	
<code>global.configEnvs</code>	Переменные окружения, доступные для всех сервисов платформы.	
<code>global.nodeSelector</code>	Параметры выбора нод кластера kubernetes для разворачивания платформы. Более подробно в документации по Kubernetes.	
<code>global.tolerations</code>	Параметры выбора нод кластера kubernetes для разворачивания платформы. Более подробно в документации по Kubernetes.	
<code>global.configurator.enabled</code>	Определяет необходимость установки сервиса Configurator (При его наличии)	true
<code>global.configurator.domain</code>	Доменное имя, по которому будет доступен сервис Configurator	configurator.example.com
<code>file.fileQuota</code>	При использовании локального хранилища - объем запрашиваемого для PersistentVolume дискового пространства	10Gi
<code>file.storageClass</code>	Имя StorageClass в кластере Kubernetes, используемого для выделения дискового пространства (см. Подготовка кластера)	nfs
<code>notification.smtpPort</code>	Параметры сервиса отправки уведомлений. Порт SMTP почтового сервиса (см. Подготовка инфраструктуры)	
<code>notification.smtpServer</code>	Адрес или доменное имя почтового сервиса	postfix-relay.default.svc.cluster.local
<code>notification.smtpAuth</code>	Параметр включения авторизации на почтовом сервере	false
<code>notification.smtpLogin</code>	При включенной авторизации - имя пользователя почтового сервиса	
<code>notification.smtpPassword</code>	При включенной авторизации - пароль для почтового сервиса	
<code>notification.fromEmail</code>	Адрес электронной почты, от которого будет осуществляться рассылка сообщений.	
<code>notification.fromName</code>	Имя отправителя сообщений (поле От:)	

Установка и обновление платформы

Установка последней версии платформы:

```
helm upgrade --install factory oci://harbor.effective-group.ru/factory-dev/factory -n <NAMESPACE> --kubeconfig <PATH_TO_KUBECONFIG_FILE> --wait
```

Установка специфичной версии платформы следует указать версию соответствующего helm-чарта с помощью параметра `--version`. Таблица соответствия версий чартов и версий платформы приведена ниже.

```
helm upgrade --install factory oci://harbor.effective-group.ru/factory-dev/factory -n <NAMESPACE> --kubeconfig <PATH_TO_KUBECONFIG_FILE> --version <CHART_VERSION> --wait
```

Версия платформы	Версия чарта
13.0.2	13.0.27

Удаление приложения

Для удаления установленной платформы выведите список установленных релизов, а затем выполните команду удаления:

```
helm list -n <NAMESPACE> [ --kubeconfig <PATH_TO_KUBECONFIG_FILE> ]  
helm uninstall factory -n <NAMESPACE> [ --kubeconfig <PATH_TO_KUBECONFIG_FILE> ]
```