Установка в режиме высокой доступности (HA)

1. Подготовка кластера Kubernetes.

Использование в Платформе

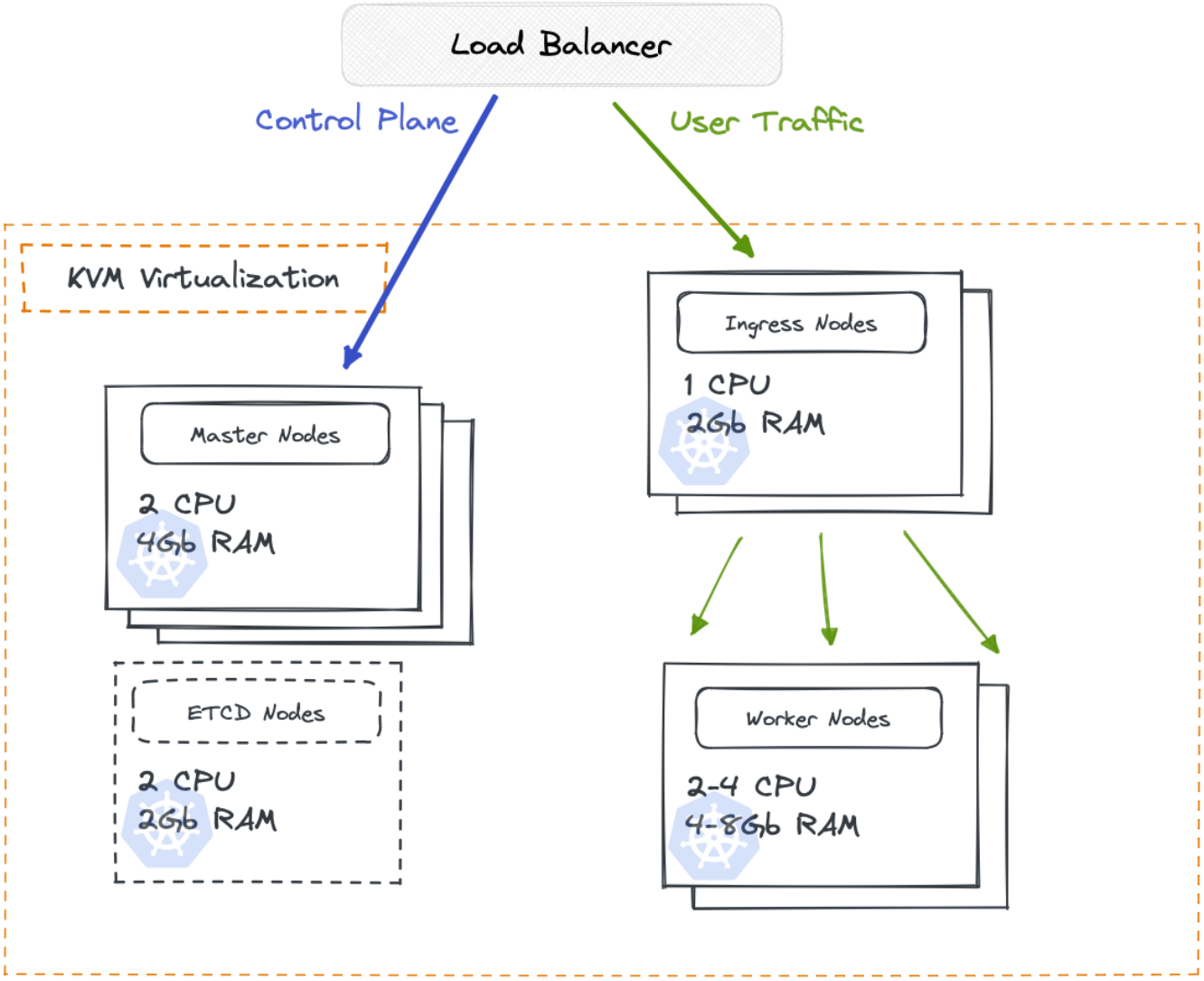
Базовые сервисы Платформы развертываются в Docker-контейнерах, в качестве оркестрации контейнеризированных приложений используется Kubernetes.

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 развертывание 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, использующиеся 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 рекоммендуется организовывать балансировку трафика с помощью внешнего LoadBalancer. Например HaProxy, 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

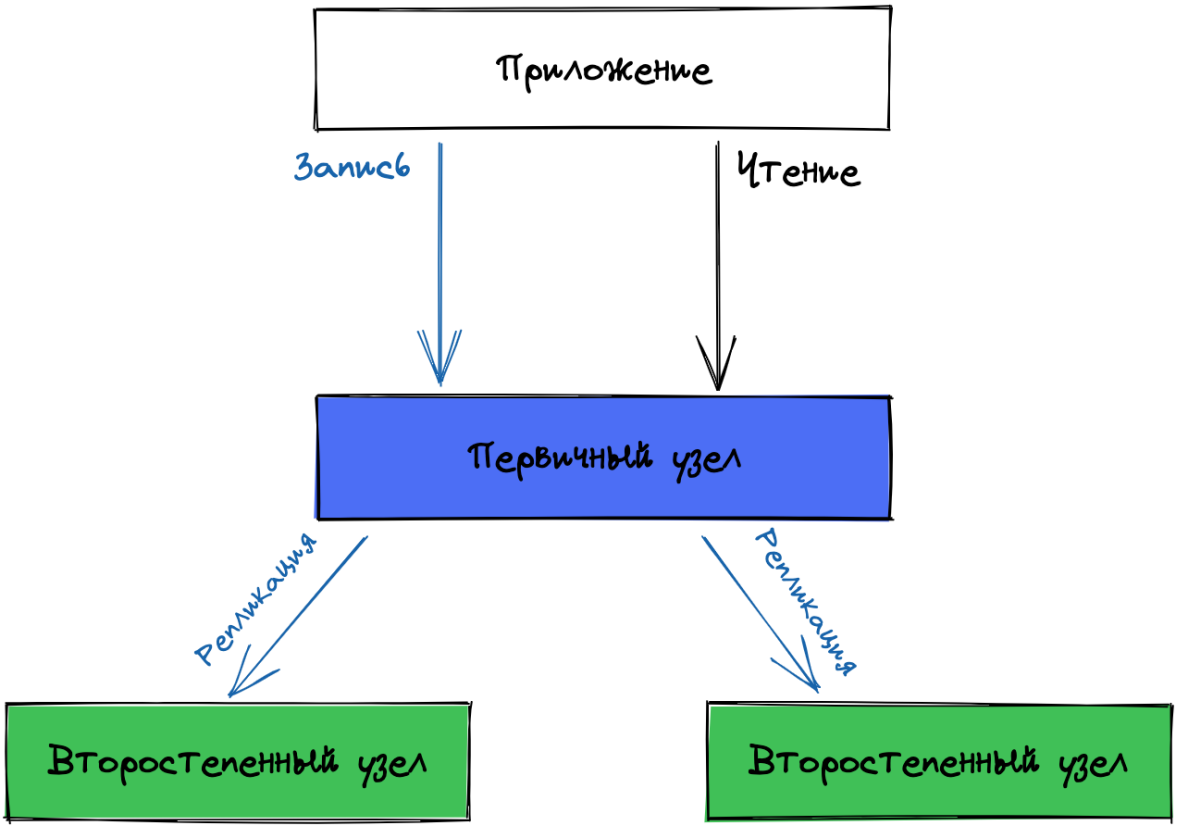
Используется как основная СУБД для проекта, в ней хранятся:

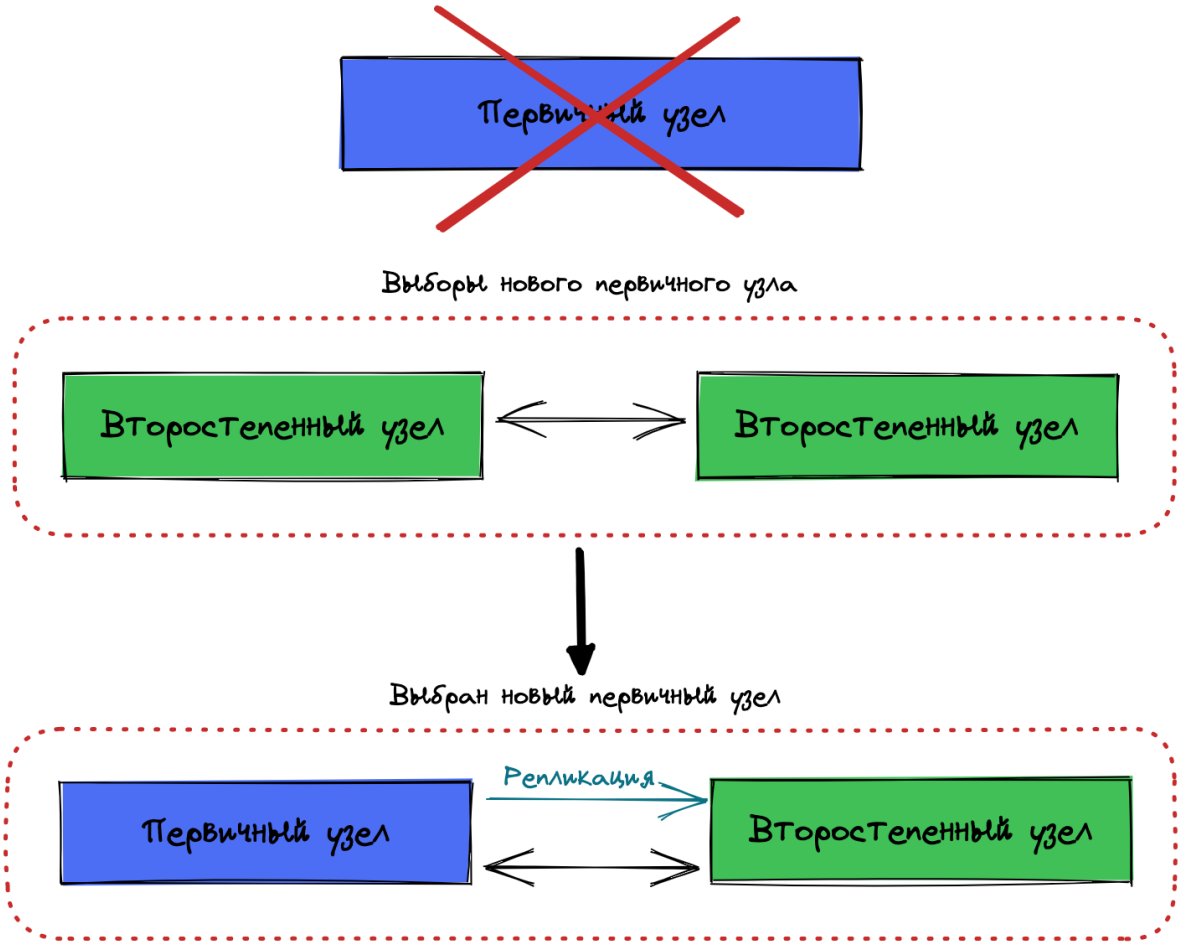
- все данные, которые пишут и используют сервисы Eftech.Factory:
    - состояния жизненных циклов, история переходов (workflow)
    - данные аутентификации, данные по пользователям, организациям, ролям, полномочиям (api-proxy)
    - данные по рассылкам (notification)
- все пользовательские данные, формируемые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


Приложение
Запись
Чтение
Первичный узел
Репликация
Репликация
Второстепенный узел
Второстепенный 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 файервола 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:

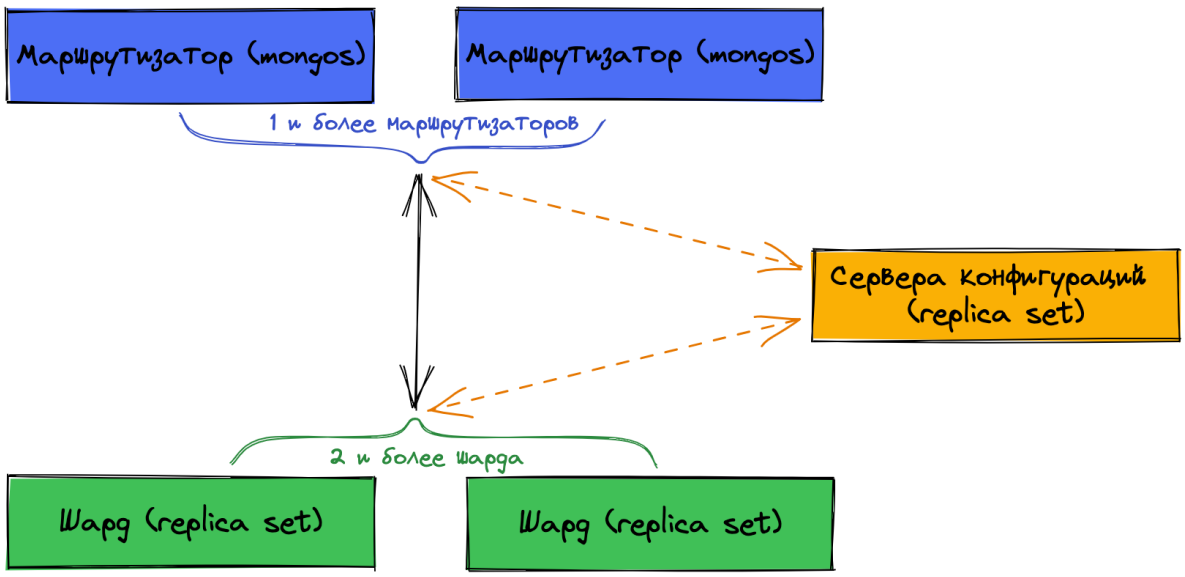
mongo.conf

```
# 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

Руководство по установке доступно на сайте разработчика по адресу https://www.mongodb.com/docs/manual/installation/

2.2. Redis

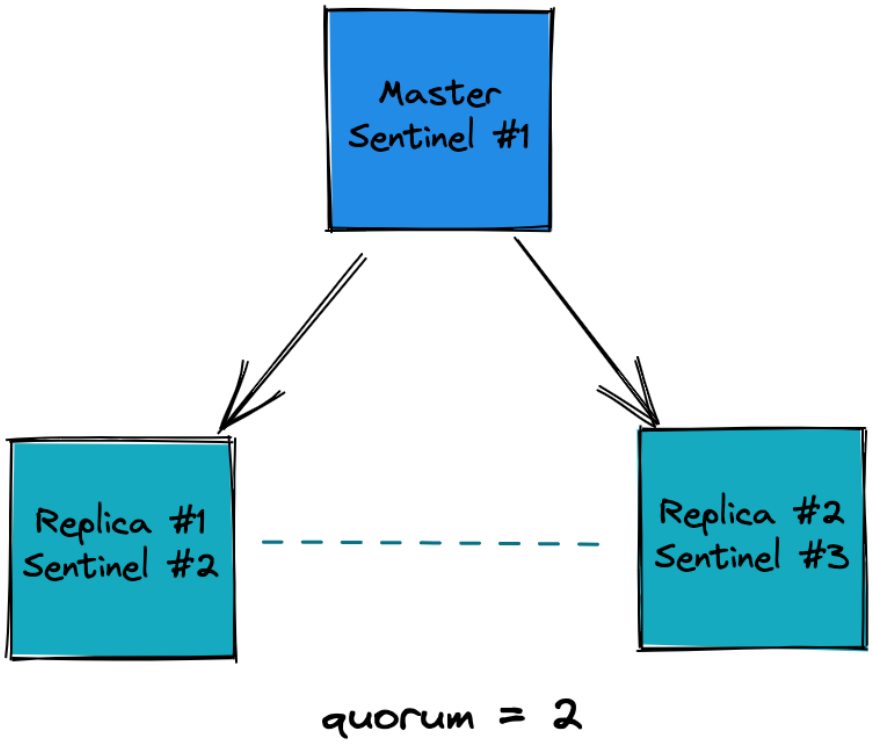
Использование в Платформе

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 файервола 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
22	ssh

Конфигурационные файлы

Конфигурация идентична для всех узлов:

Основной конфиг сервиса:

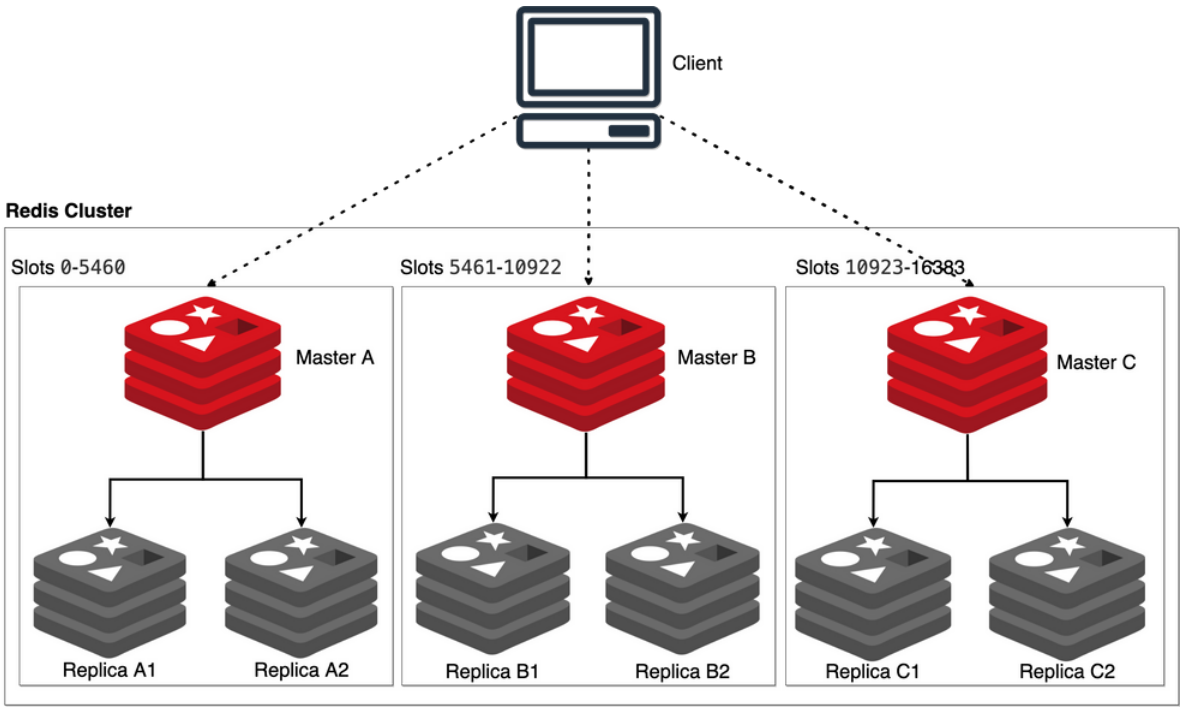
```
# General
daemonize yes
protected-mode no
pidfile "/var/run/redis/redis.pid"
dir "/var/lib/redis"
port 6379
timeout 0
tcp-keepalive 0
tcp-backlog 511
loglevel notice
logfile ""
syslog-enabled yes
syslog-ident "redis_6379"
syslog-facility user
databases 16

# 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

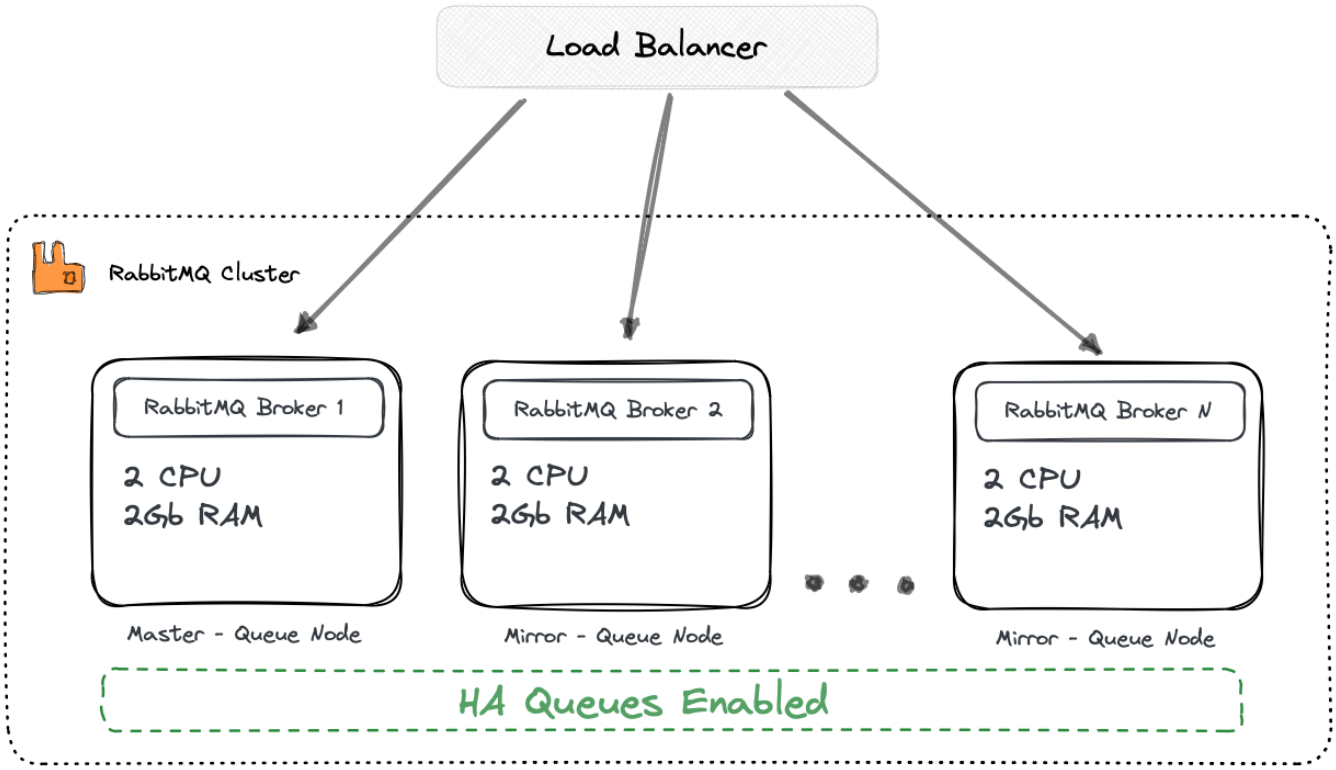
- для организации пользовательских рассылок
- в качестве шины данных для сервисов платформы

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 файервола 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 закрыт 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 облачного хранилища, либо использование Self-Hosted решения, например Minio.

Отказоустойчивость

Отказоустойчивость в случае хранении 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 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 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 12.13.10 > factory-values.yaml
```

Обязательные параметры:

Имя переменной	Описание	Значение по умолчанию
`global.domain`	Доменное имя, по которому будет доступна платформа	example.com
`global.image.registry`	Адрес Docker-Registry, в котором хранятся образы сервисов платформы	git.effective-team.ru:4567
`global.image.repository`	Репозиторий внутри Registry, в котором хранятся образы сервисов платформы	effectivex
`global.mongodb.dsn`	Строка подключения к MongoDB вида: `mongodb://<USERNAME>:<PASSWORD>@<MONGODB_HOSTS>/<MONGODB_NAME>?replicaSet=<REPLICASET_NAME>`	
`global.rabbitmq.host`	Адрес хоста сервиса очередей RabbitMQ	

global.rabbitmq.port	Порт подключения к RabbitMQ	5672
global.rabbitmq.user	Пользователь для авторизации в RabbitMQ	
global.rabbitmq.password	Пароль для авторизации в RabbitMQ	
global.rabbitmq.vhost	Имя Виртуального Хоста в RabbitMQ	
global.redis.mode	Тип подключения к Redis. Поддерживается Single и Sentinel варианты подключений	sentinel
global.redis.host	При использовании Single подключения - домен или адрес сервера Redis	
global.redis.sentinel.master	При использовании Sentinel подключения - имя мастер инстанса Redis	
global.redis.sentinel.adresses	При использовании Sentinel подключения - адреса или доменные имена Redis Хостов	
global.redis.sentinel.password	При использовании Sentinel подключения с аутентификацией - пароль для подключения к Redis	
Следующие параметры обязательны при использовании хранилища S3: `file.defaultStorage=s3`		
file.s3KeyId	Идентификатор подключения к S3 хранилищу	
file.s3Key	Ключ подключения к S3 хранилищу	
file.s3Region	Регион облачного хранилища провайдера S3	ru-central1
file.s3Bucket	Имя S3 Бакета	
file.s3Endpoint	Доменное имя провайдера подключения S3 хранилища	https://storage.yandexcloud.net
file.s3RootPath	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
global.envName	Имя окружения. Используется в БД Redis как префикс для хранения ключей, а так же в логировании базовых сервисов.	test
global.image.pullPolicy	Глобальная политика развертывания сервисов внутри кластера Kubernetes. Документация.	
global.revisionHistoryLimit	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
global.apiProxyService	Адрес базового сервиса ApiProxy, установленного в кластере Kubernetes	http://api-proxy
global.dataService	Адрес базового сервиса Data, установленного в кластере Kubernetes	http://data
global.workflowService	Адрес базового сервиса Workflow, установленного в кластере Kubernetes	http://workflow
global.fileService	Адрес базового сервиса File, установленного в кластере Kubernetes	http://file
global.cryptoService	Адрес базового сервиса криптографии (при использовании), установленного в кластере Kubernetes либо на внешнем ресурсе.	
global.captchaSiteKey	Ключ использования функции Google Captcha (При наличии)	
global.ingress.protocol	Протокол доступа к web-ресурсам платформы (HTTP или HTTPS)	http
global.ingress.clusterIssuer	При использовании HTTPS и Cert-Manager - наименование сервиса типа ClusterIssuer от которого происходит выпуск сертификатов letsencrypt	letsencrypt-prod

global.ingress.httpsRedirect	Осуществлять ли редирект на HTTPS при обращении к HTTP ресурсам платформы	
global.ingress.robotsTxt	Настройки для обработки запросов Поисковых и системам сбора данных	
global.hpa.enable	Настройка автомасштабирования ресурсов платформы в зависимости от пользовательской нагрузки. Применяется в облачных инсталляциях или системах с динамически расширяемыми нодами кластера Kubernetes.	false
global.hpa.minReplicas	Минимальное количество реплик сервисов, доступное для использования.	1
global.hpa.maxReplicas	Максимально расширяемое количество реплик сервисов, доступных для использования при пиках нагрузки.	2
global.hpa.targetMemoryUtilizationPercentage	Процентное значение потребляемой оперативной памяти (от Limits) при котором будет создана дополнительная реплика сервиса	75
global.hpa.targetCPUUtilizationPercentage	Процентное значение потребляемопроцессорного времени (от Limits) при котором будет создана дополнительная реплика сервиса	80
global.replicaCount	При неиспользовании HPA задает статическое количество реплик каждого сервиса. (Значение глобальное, для каждого сервиса может устанавливаться отдельное значение)	1
global.tracing.enabled	регулирует включение функций трейсинга запросов внутри платфомы для осуществления отладочных операций. При включенной функции будет установлен дополнительный open-source сервис Jaeger.	false
global.tracing.jaegerAgentHost	Адрес сервиса агента отслеживания запросов	jaeger
global.tracing.jaegerAgentPort	Порт сбора информации от сервисов, написанных на Node.js	6382
global.tracing.jaegerAgentPortGo	Порт сбора информации от сервисов, написанных на Go	6381
global.timeout	время таймаута запросов при проксировании от api-proxy к остальным сервисам	
global.configEnvs	Переменные окружения, доступные для всех сервисов платформы.	
global.nodeSelector	Параметры выбора нод кластера kubernetes для разворачивания платформы. Более подробно в документации по Kubernetes.	
global.tolerations	Параметры выбора нод кластера kubernetes для разворачивания платформы. Более подробно в документации по Kubernetes.	
global.configurator.enabled	Определяет необходимость установки сервиса Configurator (При его наличии)	true
global.configurator.domain	Доменное имя, по которому будет доступен сервис Configurator	configurator.example.com
file.fileQuota	При использовании локального хранилища - объем запрашиваемого для PersistentVolume дискового пространства	10Gi
file.storageClass	Имя StorageClass в кластере Kubernetes, используемого для выделения дискового пространства (см. Подготовка кластера)	nfs
notification.smtpPort	Параметры сервиса отправки уведомлений. Порт SMTP почтового сервиса (см. Подготовка инфраструктуры)	
notification.smtpServer	Адрес или доменное имя почтового сервиса	postfix-relay.default.svc.cluster.local
notification.smtpAuth	Параметр включения авторизации на почтовом сервере	false
notification.smtpLogin	При включенной авторизации - имя пользователя почтового сервиса	
notification.smtpPassword	При включенной авторизации - пароль для почтового сервиса	
notification.fromEmail	Адрес электронной почты, от которого будет осуществляться рассылка сообщений.	

notification.fromName	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
12.13.0	12.13.10

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> ]
```