Рекомендательная платформа MOODREC

Поддержание жизненного цикла ПО

Содержание

Требования к персоналу

Требования к специалистам по сопровождению сервиса

Специалисты, осуществляющие мониторинг работоспособности сервиса, администрирование и поддержку инфраструктуры заказчика, должны обладать следующими знаниями и навыками:

- Техническое образование
- Знание работы протоколов TCP/IP
- Знание основных принципов работы баз данных
- Навыки администрирования ОС Linux не менее 1 года
- Знание функциональных возможностей сервиса
- Навыки работы с MongoDB, Kvrocks, ClickHouse
- Знание средств восстановления баз данных и мониторинга производительности серверов

Устранение программных ошибок осуществляется только специалистами ООО «Альткрафт» после предоставления диагностической информации.

Процессы сопровождения сервиса

Консультации пользователей и администраторов

Консультации осуществляет ООО «Альткрафт».
Поддержка предоставляется через персональный чат, email или телефон в зависимости от уровня поддержки.

Регулярные обновления сервиса

Заказчику предоставляется право получения бесплатных обновлений на весь срок использования платформы.

Устранение программных ошибок

Исправления выполняются специалистами ООО «Альткрафт» после анализа предоставленных логов.
Диагностическая информация передается через чат поддержки или на moodrec@altcraft.com.

Модификация сервиса

Создание и модернизация компонентов выполняются только специалистами ООО «Альткрафт».
Возможна доработка функционала по согласованному техническому заданию.
Контакт: customers.department@altcraft.com.

Развертывание сервиса

В данном руководстве показан процесс установки для Debian-подобной ОС (например, Astra Linux или Ubuntu).

Для non-Debian ОС процесс выглядит похожим образом, кроме установки некоторых зависимостей.

Предварительная подготовка

- Перед началом установки настоятельно рекомендуем ознакомиться с техническими требованиями).
- Все дальнейшие действия требуют наличия привилегий пользователя `root`.
- Для паролей и секретных комбинаций используется набор из символов – `abcdefghijklmnopqrstuvwxyzABCDEF`. Обязательно сгенерируйте более сложный пароль, состоящий из не менее 32-х символов.

Загрузка архива

Всем клиентам предоставляется архив с расширением `.tar.gz`, распакуйте его и переместите содержимое в каталог `/opt`:

```
{
  tar xzf <архив>.tar.gz
  mv moodrec /opt/moodrec
}
```

Проверка времени на сервере

Синхронизация времени между серверами критична: различия могут привести к нарушению работы и ошибочной статистике. Настройте NTP-сервер и проверьте синхронизацию:

```
timedatectl status
```

Установка системных лимитов

Добавьте следующие лимиты в файл `/etc/security/limits.conf`:

```
*               -       nofile          1048576
root            -       nofile          1048576
root            -       memlock         unlimited
```

Затем переподключитесь и проверьте, что лимиты применились:

```
ulimit -n
```

Установка зависимостей

Установите следующие зависимости:

```
{
  apt update
  apt install --no-install-recommends gcc g++ libc6-dev
}
```

Установка необходимых компонентов

Установка Kafka

Kafka используется как основной брокер сообщений для работы сервиса.

Установка происходит согласно инструкциям:

- **Быстрый старт**
- **С чего начать**

Установка MongoDB

TIP

MongoDB используется для хранения системных настроек и пользовательских данных. Ниже приведена инструкция по установке из архива.

Дополнительные материалы:

- Установка MongoDB Community Edition на Ubuntu
- Описание файла конфигурации
- Конфигурация базы данных

Перейдите на страницу загрузки, выберите версию 8.0, вашу версию ОС и архив `.tgz`. Скопируйте ссылку и скачайте архив:

```
wget https://fastdl.mongodb.org/linux/mongodb-linux-x86_64-ubuntu2404-8.0.17.tgz
```

Распакуйте архив в `/usr/local` и создайте символическую ссылку:

```
{
  tar xzf mongodb-linux-x86_64-ubuntu2404-8.0.17.tgz -C /usr/local
  ln -s /usr/local/mongodb-linux-x86_64-ubuntu2404-8.0.17 /usr/local/mongodb
}
```

Установите зависимости:

```
{
  apt update
  apt install -y gnupg curl
}
```

Создайте пользователя, группу и рабочие каталоги:

```
{
  groupadd mongodb
  useradd -r -g mongodb -s /bin/false mongodb
  mkdir -p /var/{lib,log}/mongodb
  chown -R mongodb:mongodb /var/{lib,log}/mongodb
}
```

INFO

По умолчанию в MongoDB отключена аутентификация. Для повышения безопасности вы можете включить аутентификацию с использованием механизма SCRAM (аутентификация по паролю).

Создайте файл конфигурации `/etc/mongod.conf`:

```
# mongod.conf

# for documentation of all options, see:
#   http://docs.mongodb.org/manual/reference/configuration-options/

storage:
  dbPath: /var/lib/mongodb
  wiredTiger:
    engineConfig:
      directoryForIndexes: true

systemLog:
  destination: file
  logAppend: true
  logRotate: reopen
  path: /var/log/mongodb/mongod.log

net:
  port: 27017
  bindIp: 127.0.0.1
```

INFO

Вы можете включить поддержку TLS для подключения к MongoDB. Для этого добавьте в файл `/etc/mongod.conf` следующие параметры:

```
net:
  tls:
    mode: requireTLS
    certificateKeyFile: <путь до файла с приватным ключом и сертификатом сервера (PEM)>
    CAFile: <путь до сертификата центра сертификации (CA)>
```

После перезапуска сервиса, MongoDB будет принимать подключения только от клиентов, использующих TLS.

Создайте systemd-сервис по пути `/etc/systemd/system/mongod.service`:

```
[Unit]
Description=MongoDB Database Server
Documentation=https://docs.mongodb.org/manual
After=network.target

[Service]
User=mongodb
Group=mongodb
ExecStart=/usr/local/mongodb/bin/mongod --config /etc/mongod.conf
PIDFile=/run/mongod.pid
Restart=always
LimitFSIZE=infinity
LimitCPU=infinity
LimitAS=infinity
LimitNOFILE=256000
LimitNPROC=256000
LimitMEMLOCK=infinity
TasksMax=infinity
TasksAccounting=false

[Install]
WantedBy=multi-user.target
```

Добавьте сервис в автозагрузку, запустите и проверьте статус:

```
{
  systemctl enable mongod.service
  systemctl start mongod.service
  systemctl status mongod.service
}
```

Скачайте и установите MongoDB Shell, подключитесь к базе данных и создайте пользователя в базе `admin`:

```
use admin;
db.createUser({
    "user": "moodrec",
    "pwd": "abcdefghijklmnopqrstuvwxyzABCDEF",
    "roles": [{
        "role": "root",
        "db": "moodrec"
    }]
})
```

Обновите параметры подключения в конфигурации `configs/release/main_config.yaml`:

```
  mongo: &mongo_config
  database: "moodrec"
  username: "moodrec"
  password: "abcdefghijklmnopqrstuvwxyzABCDEF"
  auth_source: "admin"
  host: "127.0.0.1"
  port: "27017"
```

Установка ClickHouse

> **TIP**
>
> База данных ClickHouse используется для хранения истории, статистики и формирования пользовательских отчетов. Ниже представлена инструкция по установке из DEB-пакетов.

Установите зависимости и добавьте официальный репозиторий:

```
{
  apt update
  apt install -y apt-transport-https ca-certificates dirmngr gnupg
  apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv 8919F6BD2B48D754
  echo "deb https://packages.clickhouse.com/deb stable main" | tee
/etc/apt/sources.list.d/clickhouse.list
}
```

Установите ClickHouse:

```
{
  apt update
  apt install -y clickhouse-server clickhouse-client
}
```

Добавьте сервис в автозагрузку и запустите:

```
{
  systemctl enable clickhouse-server.service
  systemctl start clickhouse-server.service
  systemctl status clickhouse-server.service
}
```

Обновите параметры подключения в конфигурации `configs/release/main_config.yaml`:

```
  clickHouse: &clickHouse_config
  host: "127.0.0.1"
  port: 9000
  user: "moodrec"
  password: "moodrec"
  db_name: "moodrec"
```

Установка Kvrocks

TIP

Kvrocks — альтернатива Redis, построенная на RocksDB. В отличие от Redis, он сохраняет данные на диск, что делает его подходящим для хранения больших объемов данных.

Установите зависимости для сборки:

```
{
  apt update
  apt install -y git build-essential cmake libtool python3
}
```

Перейдите на страницу релизов проекта на GitHub, скачайте последнюю версию и распакуйте архив:

```
{
  wget https://github.com/apache/kvrocks/archive/refs/tags/v2.12.1.tar.gz
  tar xzf v2.12.1.tar.gz
}
```

Выполните сборку и установку:

```
{
  cd kvrocks-2.12.1
  ./x.py build -DPORTABLE=1 -DCMAKE_BUILD_TYPE=Release -j $(nproc)
  mv build/kvrocks /usr/local/bin/
}
```

Создайте пользователя и каталоги:

```
{
  groupadd kvrocks
  useradd -r -g kvrocks -s /bin/false kvrocks
  mkdir -p /var/{lib,log}/kvrocks /etc/kvrocks
  chown -R kvrocks:kvrocks /var/{lib,log}/kvrocks
}
```

Создайте файл конфигурации `/etc/kvrocks/kvrocks.conf`:

```
# kvrocks.conf
bind 0.0.0.0
port 6666
db-name moodrec.db
dir /var/lib/kvrocks
log-dir /var/log/kvrocks
log-retention-days 7
backup-dir /var/lib/kvrocks/backup
requirepass abcdefghijklmnopqrstuvwxyzABCDEF
supervised systemd
workers 8
```

Создайте systemd-сервис по пути `/etc/systemd/system/kvrocks.service`:

```
[Unit]
Description=kvrocks SSD key-value database
Documentation=https://github.com/apache/kvrocks
Wants=network-online.target
After=network-online.target

[Service]
User=kvrocks
Group=kvrocks
Type=notify
ExecStart=/usr/local/bin/kvrocks -c /etc/kvrocks/kvrocks.conf
WorkingDirectory=/var/lib/kvrocks
Restart=on-failure
ExecStop=/bin/kill -s TERM $MAINPID
RestartSec=10s
LimitNOFILE=100000
LimitNPROC=4096
TimeoutSec=300
NoNewPrivileges=yes

[Install]
WantedBy=multi-user.target
Alias=kvrocks.service
```

Добавьте сервис в автозагрузку и запустите:

```
{
  systemctl enable kvrocks.service
  systemctl start kvrocks.service
  systemctl status kvrocks.service
}
```

Обновите параметры подключения в конфигурации `configs/release/main_config.yaml`:

```
  redis: &redis_config
    host: "127.0.0.1"
    port: 6666
    user: "moodrec"
    password: "abcdefghijklmnopqrstuvwxyzABCDEF"
    read_timeout: 10
    write_timeout: 10
    pool_size: 100
```

Запуск сервиса

Перед запуском сервиса убедитесь, что вы заполнили файл конфигурации `configs/release/main_config.yaml`

Пример файла конфигурации на следующей странице.

Описание всех параметров доступно на странице Параметры конфигурации.

```
databases:
  mongo: &mongo_config
    database: "moodrec"
    username: "root"
    password: "abcdefghijklmnopqrstuvwxyzABCDEF"
    auth_source: "admin"
    host: "127.0.0.1"
    port: "27017"
    max_pool_size: 100
    min_pool_size: 1
    max_idle_time_ms: 60000
    auth_mechanism: "SCRAM-SHA-1"
    timeout: 60000
    connect_timeout: 60000

    MONGODB_UPDATER_BATCH_SIZE:
    MONGODB_UPDATER_PERIOD_MS:
    MONGODB_SOCKET_TIMEOUT_MIN:
    MONGODB_SERVER_SELECTION_TIMEOUT:

  redis: &redis_config
    host: "127.0.0.1"
    port: 6666
    user: "moodrec"
    password: "abcdefghijklmnopqrstuvwxyzABCDEF"
    read_timeout: 10
    write_timeout: 10
    pool_size: 100

  clickHouse: &clickHouse_config
    host: "127.0.0.1"
    port: 9000
    user: "moodrec"
    password: "moodrec"
    db_name: "moodrec"

security: &security_setting
  http_token: "abcdefghijklmnopqrstuvwxyzABCDEF"
  grpc_token: "abcdefghijklmnopqrstuvwxyzABCDEF"

license: &license
  die_when_license_fails: true
  notification_license_warn_days: 10
  proxy: ""
  is_debug: false

kafka:
  seeds: &kafka_seeds
```

```
    - "127.0.0.1:29092"
    - "127.0.0.1:29093"
    - "127.0.0.1:29094"
  default_topic_config: &default_kafka_topic_config
    #Время хранения сообщений в миллисекундах
    "retention.ms": "43200000"
    #Лимит размера хранения топика.
    "retention.bytes": "-1"
    #Макс. размер лог-файла сегмента
    "segment.bytes": "1073741824"
    #Что делать со старыми сообщениями
    "cleanup.policy": "delete"
    #Макс размер одного сообщения в байтах
    "max.message.bytes": "10242880"
    #Минимум реплик, которые должны подтвердить запись.(для надежности = количество
брокеров)
    "min.insync.replicas": "3"
    #Тип сжатия сообщений
    "compression.type": "uncompressed"
    #Тип метки времени сообщения
    "message.timestamp.type": "CreateTime"
    #Задержка перед удалением устаревших сегментов в миллисекундах
    "file.delete.delay.ms": "30000"
    #Уровень компрессии для LZ4
    "compression.lz4.level": "9"
    #Уровень компрессии GZIP
    "compression.gzip.level": "1"
    #Уровень компрессии ZSTD
    "compression.zstd.level": "3"
    #Ограничение пропускной способности репликации лидера
    "leader.replication.throttled.replicas": ""
    #Ограничение пропускной способности репликации фолловера
    "follower.replication.throttled.replicas": ""
    #Включает возможность понижающей конверсии сообщений (например, для старых
клиентов)
    "message.downconversion.enable": "false"
    #Версия формата сообщений Kafka
    "message.format.version": "3.0-IV1"
    #Максимальное время перед flush
    "flush.ms": "10000"
    #Максимальное количество сообщений перед flush
    "flush.messages": "10000"
    #Минимальное время задержки перед compaction
    "min.compaction.lag.ms": "60000"
    #Если true, Kafka заранее выделяет место для сегментов логов, что может снизить
фрагментацию диска
    "preallocate": "true"
    #Определяет, сколько «грязных» данных должно накопиться перед компакцией
сегментов. Значение 0.5 означает, что компакция начнется, когда 50% сегмента станет
```

```
устаревшим
    "min.cleanable.dirty.ratio": "0.4"
    #Интервал байтов между индексируемыми сообщениями. Более высокое значение
уменьшает размер индекса, но увеличивает время поиска
    "index.interval.bytes": "8192"
    #Разрешить небезопасное лидерство при отказах
    "unclean.leader.election.enable": "false"
    #Время хранения удалённых сообщений в log compaction
    "delete.retention.ms": "43200000"
    #Максимальное разрешенное отклонение временной метки сообщения в будущем
    "message.timestamp.after.max.ms": "86400000"
    #Максимальное разрешенное отклонение временной метки сообщения в прошлом
    "message.timestamp.before.max.ms": "86400000"
    #Максимальное время жизни сегмента перед созданием нового
    "segment.ms": "259200000"
    #Добавляет случайное смещение к segment.ms, предотвращая массовую ротацию
сегментов
    "segment.jitter.ms": "1000"
    #Максимальное различие между временной меткой сообщения и текущим временем
    "message.timestamp.difference.max.ms": "86400000"
    #Максимальный размер индекса для одного сегмента
    "segment.index.bytes": "10242880"
    #максимальное время (в миллисекундах), в течение которого сообщение остается
без компакции, даже если оно устарело
    "max.compaction.lag.ms": "86400000"
    #Использование удаленного хранилища
    #"remote.storage.enable": "false"
    #Копирование логов в удаленное хранилище
    #"remote.log.copy.disable": "true"
    #Удаление логов при отключении удаленного хранилища
    #"remote.log.delete.on.disable": true"
## Services
##
offers:
  license:
    *license
  #В секундах
  gracefully_shutdown_timeout: 60
  #Размер batch с продуктами при импорте
  batch_size_offers: 100
  #Максимальный размер фида в Гб
  max_size_limit_feed: 2
  #Время ожидания скачки фида в минутах
  http_request_timeout: 5
  servers_settings:
    grpc_server:
      host: "localhost"
      port: "1111"
    #Другие сервисы с которыми нужно взаимодействие
```

```
    grpc_clients:
      tasks:
        - "127.0.0.1:2222"
      vectors:
        - "127.0.0.1:4444"
    #Таймаут запроса к другому сервису в секундах
    grpc_timeouts:
      tasks: 60
      vectors: 60
    #Количество соединений к серверу по rpc
    grpc_pool_size:
      tasks: 10
      vectors: 60
  logger:
    level: "trace"
  security:
    *security_setting
  databases:
    mongo:
      *mongo_config
    redis:
      *redis_config
  brokers:
    kafka:
      seeds: *kafka_seeds
      retry_timeout_consumer: 5
      retry_timeout_producer: 5
      retry_max_count_producer: 5
      #Топики в которые будет писать сервис. И их первоначальные настройки
      topics:
        update_offers_in_index:
          replication_factor: 3
          partitions: 3
          configs:
            *default_kafka_topic_config
##
tasks:
  #В секундах
  gracefully_shutdown_timeout: 60
  servers_settings:
    grpc_server:
      host: "localhost"
      port: "2222"
    #Другие сервисы с которыми нужно взаимодействие
    grpc_clients:
      audience:
        - "127.0.0.1:7777"
      offers:
        - "127.0.0.1:1111"
```

```
      catboss:
        - "127.0.0.1:5555"
    #Таймаут запроса к другому сервису в секундах
    grpc_timeouts:
      audience: 60
      offers: 60
      catboss: 60
    #Количество соединений к серверу по rpc
    grpc_pool_size:
      audience: 10
      offers: 10
      catboss: 10
  security:
    *security_setting
  logger:
    level: "trace"
  databases:
    mongo:
      *mongo_config

##
trans:
  #Директория с библиотеками
  lib_dir_path: "/opt/moodrec/lib/"
  #Директория с моделью
  model_path: "/opt/moodrec/models/model-example"
  model_file_name: "model.onnx"
  #todo для тестирования подготовки текста для модели
  full_clean_text: false
  #В секундах
  gracefully_shutdown_timeout: 60
  #На скольких ядрах будет выполняться векторизация
  count_goroutines_for_vectorization: 4
  servers_settings:
    grpc_server:
      host: "localhost"
      port: "1234"
  security:
    *security_setting
  logger:
    level: "trace"
  databases:
    mongo:
      *mongo_config
    redis:
      *redis_config
  brokers:
    kafka:
      # Указываем название топика и сколько горутин с одинаковым groupId будут
```

```
читать топик
      subscribe_count_goroutines:
        update_offers_in_index: 3
      seeds: *kafka_seeds
      retry_timeout_consumer: 5
      retry_timeout_producer: 5
      retry_max_count_producer: 5
      #Топики в которые будет писать сервис. И их первоначальные настройки
      topics:
        save_offers_to_index:
          replication_factor: 3
          partitions: 3
          configs:
            *default_kafka_topic_config

##
vectors:
  #Количество воркеров для индекса
  count_workers_index: 120
  #Конфиг для запуска задачи для обновления векторной базы. В минутах
  vectors_update_interval: 1
  #В секундах
  gracefully_shutdown_timeout: 60
  #Размерность вектора
  vector_dimension: 384
  #Название файла индекса
  index_file_dir: "/opt/moodrec/index"
  databases:
    redis:
      *redis_config
  servers_settings:
    grpc_server:
      host: "localhost"
      port: "4444"
    grpc_clients:
      offers:
        - "127.0.0.1:1111"
      catboss:
        - "127.0.0.1:5555"
    #Таймаут запроса к другому сервису в секундах
    grpc_timeouts:
      offers: 60
      catboss: 60
    #Количество соединений к серверу по rpc
    grpc_pool_size:
      offers: 10
      catboss: 10
  security:
    *security_setting
```

```
  logger:
    level: "trace"
  brokers:
    kafka:
      # Указываем название топика и сколько горутин с одинаковым groupId будут
читать топик
      subscribe_count_goroutines:
        save_offers_to_index: 3
      seeds: *kafka_seeds
      retry_timeout_consumer: 5
      retry_timeout_producer: 5
      retry_max_count_producer: 5

##
events:
  security:
    *security_setting
  # Использовать ли доп сортировку по названию категории при выборе топ категорий
пользователя
  withSortByCatName: true
  # TODO: Если используем свои топики. то нужно описать шаблон сообщения
  # Кастомный список названий топиков для кафки (если используют запись напрямую в
kafka)
  # Указываем в каком формате будут лежать сообщения в топиках (json или proto)
  # Если формат = proto. то нужно указать еще название сообщения в proto файле
(пример :message Event {})
  custom_kafka_events_topics:
    json_events_topic:
      schema: "json"
      # Сколько горутин с одинаковым groupId будут читать топик
      subscribe_count_goroutines: 3
    proto_events_topic:
      schema: "proto"
      # Сколько горутин с одинаковым groupId будут читать топик
      subscribe_count_goroutines: 3
      message_name: "Event"

  # В секундах
  gracefully_shutdown_timeout: 60
  # Сколько топ категорий у сессии будем хранить
  session_limit_top_categories: 3
  # ttl для событий по сессии (в минутах)
  session_events_ttl: 900
  # Сколько топ категорий у сессии будем хранить
  uid_limit_top_categories: 3
  # ttl для событий по сессии (в днях)
  uid_events_ttl: 90
  servers_settings:
    #Другие сервисы с которыми нужно взаимодействие
```

```
    grpc_clients:
      audience:
        - "127.0.0.1:7777"
      offers:
        - "127.0.0.1:1111"
    #Количество соединений к серверу по rpc
    grpc_pool_size:
      audience: 100
      offers: 100
    grpc_timeouts:
      audience: 60
      offers: 60
    grpc_server:
      host: "localhost"
      port: "9999"
  logger:
    level: "trace"
  databases:
    redis:
      *redis_config
  brokers:
    kafka:
      seeds: *kafka_seeds
      # Указываем название топика и сколько горутин с одинаковым groupId будут
читать топик
      subscribe_count_goroutines:
        events: 3
        uid_reassign: 3
      retry_timeout_consumer: 1
      retry_timeout_producer: 1
      retry_max_count_producer: 1
      #Сколько воркеров будет использоваться при асинхронной отправке
      async_producer_worker_count: 50
      #Топики в которые будет писать сервис. И их первоначальные настройки
      topics:
        events:
          replication_factor: 3
          partitions: 3
          configs:
            *default_kafka_topic_config

##
catboss:
  security:
    *security_setting
  host_url: "https://example.com"
  sdk_url: "https://example.com/static/moodrec-tracker.js"
  #В секундах
  gracefully_shutdown_timeout: 60
```

```
  servers_settings:
    grpc_server:
      host: "localhost"
      port: "5555"
    #Другие сервисы с которыми нужно взаимодействие
    grpc_clients:
      tasks:
        - "127.0.0.1:2222"
      vectors:
        - "127.0.0.1:4444"
      events:
        - "127.0.0.1:9999"
      offers:
        - "127.0.0.1:1111"
      audience:
        - "127.0.0.1:7777"
      auth:
        - "127.0.0.1:8888"
    #Таймаут запроса к другому сервису в секундах
    grpc_timeouts:
      tasks: 60
      events: 60
      vectors: 60
      offers: 60
      audience: 60
      auth: 60
    #Количество соединений к серверу по rpc
    grpc_pool_size:
      tasks: 10
      events: 10
      vectors: 10
      offers: 10
      audience: 10
      auth: 10
  logger:
    level: "trace"
  brokers:
    kafka:
      # Указываем название топика и сколько горутин с одинаковым groupId будут
читать топик
      subscribe_count_goroutines:
        uid_reassign: 3
      seeds: *kafka_seeds
      retry_timeout_consumer: 5
      retry_timeout_producer: 5
      retry_max_count_producer: 5
      #Сколько воркеров будет использоваться при асинхронной отправке
      async_producer_worker_count: 50
      #Топики в которые будет писать сервис. И их первоначальные настройки
```

```
      topics:
        impressions: #Топик читает CH
          replication_factor: 3
          partitions: 3
          configs:
            *default_kafka_topic_config
  databases:
    mongo:
      *mongo_config
    redis:
      *redis_config
    clickHouse:
      *clickHouse_config

##
audience:
  security:
    *security_setting
  #В секундах
  gracefully_shutdown_timeout: 60
  servers_settings:
    grpc_server:
      host: "localhost"
      port: "7777"
    #Другие сервисы с которыми нужно взаимодействие
    grpc_clients:
      tasks:
        - "127.0.0.1:2222"
      catboss:
        - "127.0.0.1:5555"
    #Таймаут запроса к другому сервису в секундах
    grpc_timeouts:
      tasks: 60
      catboss: 60
    #Количество соединений к серверу по rpc
    grpc_pool_size:
      tasks: 10
      catboss: 10
  logger:
    level: "trace"
  databases:
    redis:
      *redis_config
    mongo:
      *mongo_config
  brokers:
    kafka:
      seeds: *kafka_seeds
      # Указываем название топика и сколько горутин с одинаковым groupId будут
```

```
читать топик
      subscribe_count_goroutines:
        events: 3
      retry_timeout_consumer: 1
      retry_timeout_producer: 1
      retry_max_count_producer: 1
      #Сколько воркеров будет использоваться при асинхронной отправке
      async_producer_worker_count: 50
      #Топики в которые будет писать сервис. И их первоначальные настройки
      topics:
        audience_mapping:
          replication_factor: 3
          partitions: 3
          configs:
            *default_kafka_topic_config
        uid_reassign:
          replication_factor: 3
          partitions: 3
          configs:
            *default_kafka_topic_config
        hard_ids_mapping:
          replication_factor: 3
          partitions: 3
          configs:
            *default_kafka_topic_config

##
gateway:
  #В секундах
  gracefully_shutdown_timeout: 60
  #Максимальное время жизни запроса
  request_timeout: 2
  servers_settings:
    http_server:
      host: "0.0.0.0"
      port: "80"
    #Другие сервисы с которыми нужно взаимодействие
    grpc_clients:
      events:
        - "127.0.0.1:9999"
      offers:
        - "127.0.0.1:1111"
      catboss:
        - "127.0.0.1:5555"
      audience:
        - "127.0.0.1:7777"
      auth:
        - "127.0.0.1:8888"
    #Количество соединений к серверу по rpc
```

```
    grpc_pool_size:
      events: 100
      offers: 100
      catboss: 100
      audience: 100
      auth: 10
    grpc_timeouts:
      catboss: 60
      events: 60
      offers: 60
      audience: 60
      auth: 60
  security:
    *security_setting
  logger:
    level: "trace"
  databases:
    mongo:
      *mongo_config

##
auth:
  #Ссылка приглашения
  invite_url: "https://example.com/invite?token="
  #В секундах
  gracefully_shutdown_timeout: 60
  servers_settings:
    # Другие сервисы с которыми нужно взаимодействие
    grpc_clients:
      catboss:
        - "127.0.0.1:5555"
    # Количество соединений к серверу по rpc
    grpc_pool_size:
      catboss: 100
    grpc_timeouts:
      catboss: 60
    grpc_server:
      host: "localhost"
      port: "8888"
  security:
    *security_setting
  logger:
    level: "trace"
  databases:
    mongo:
      *mongo_config
```

Для управления микросервисами MoodRec используется исполняемый файл `moodrec`. Справка по командам:

```
Usage:
  moodrec [flags]
  moodrec [command]

Available Commands:
  list          List platform information
  start         Start platform services
  stop          Stop platform services
  restart       Restart platform services
  reload-nginx  Reload nginx configuration
  help          Help about any command
  completion    Generate the autocompletion script for the specified shell
  status        Gets status of microservices

Flags:
      --appname         show application full name
      --config string   config path (default
"/opt/configs/release/main_config.yaml")
      --debug           debug mode (for development purposes)
  -h, --help            help for moodrec
  -v, --version         version for moodrec

Use "moodrec [command] --help" for more information about a command.
```

На этом установка завершена, сервис готов к запуску и дальнейшей настройке. Для продолжения настройки, следуйте Пользовательскому руководству.

Обновление сервиса

Для обновления сервиса необходимо заменить **бинарные файлы** из архива с новым релизом.
Дополнительных действий от пользователя не требуется, сервис готов к запуску.

Обновление сервиса

Для обновления сервиса необходимо заменить **бинарные файлы** из архива с новым релизом. Дополнительных действий от пользователя не требуется, сервис готов к запуску. Все побочные миграции проходят автоматически, если иное не указано в release notes.

Мониторинг работы сервиса

В зависимости от используемой в организации системы мониторинга, специалист должен выбрать способ отслеживания состояния процессов сервиса.

На странице описаны все процессы, которые необходимо поддерживать в рабочем состоянии для полноценной работы сервиса.

Рекомендуется использовать проверку на существование процесса по имени встроенными средствами системы мониторинга либо с помощью `pidof`.

В обязательном порядке по хосту (или виртуальной машине) должна собираться такая информация:

- потребление и количество свободного ОЗУ;
- потребление и количество свободного времени CPU;
- количество свободного места и информация по утилизации дисков (скорость, задержки, SMART);
- количество открытых файлов и соединений.

Базы данных и сервисы

Сервис	IP адрес по умолчанию	Порт	Важность	Доступ	Уведомление
Интерфейс платформы MoodRec	0.0.0.0	443 (HTTPS)	Чрезвычайная	Публичный или через VPN / allowlist	Недоступен веб-интерфейс
Публичный API (события, рекомендации, импорт)	0.0.0.0	443 (HTTPS)	Чрезвычайная	Публичный или в сетевом периметре клиента	Недоступны API-методы
MongoDB	127.0.0.1	27017 (TCP)	Чрезвычайная	Только из внутренней сети	Невозможно установить TCP-соединение
ClickHouse	127.0.0.1	9000 (TCP)	Чрезвычайная	Только из внутренней сети	Невозможно установить TCP-соединение
Apache Kafka	127.0.0.1	9092, 9093 (TCP)	Чрезвычайная	Только из внутренней сети	Невозможно установить TCP-соединение
KV Rocks	127.0.0.1	6666 (TCP)	Чрезвычайная	Только из внутренней сети	Невозможно установить TCP-соединение
Сервис работы с продуктами	127.0.0.1	1111 (TCP)	Высокая	Только из внутренней сети	Сервис недоступен по TCP
Сервис управления задачами	127.0.0.1	2222 (TCP)	Высокая	Только из внутренней сети	Сервис недоступен по TCP

Сервис	IP адрес по умолчанию	Порт	Важность	Доступ	Уведомление
Сервис расчёта продуктовых представлений	127.0.0.1	3333 (TCP)	Высокая	Только из внутренней сети	Сервис недоступен по TCP
Сервис обращений к векторному индексу	127.0.0.1	4444 (TCP)	Высокая	Только из внутренней сети	Сервис недоступен по TCP
Сервис выдачи рекомендаций	127.0.0.1	5555 (TCP)	Чрезвычайная	Только из внутренней сети	Сервис недоступен по TCP
Сервис аудиторий	127.0.0.1	7777 (TCP)	Высокая	Только из внутренней сети	Сервис недоступен по TCP
Сервис работы с пользователями	127.0.0.1	8888 (TCP)	Высокая	Только из внутренней сети	Сервис недоступен по TCP
Сервис обработки событий	127.0.0.1	9999 (TCP)	Чрезвычайная	Только из внутренней сети	Сервис недоступен по TCP
SMTP / SMTPS (оповещения)	внешний SMTP-сервер	25 или 465	Средняя	Исходящее соединение к SMTP-провайдеру	Невозможно установить соединение с SMTP

Дополнительно по каждому процессу

Можно собирать дополнительные данные: потребление памяти, количество открытых файлов и соединений, использование CPU и т.д. Пример — в разделе «Диагностика проблем», блок «Сбор информации для запроса в поддержку».

Процессорное время

CPU time: https://en.wikipedia.org/wiki/CPU_time

Тип метрики	Описание	Важность	Уведомление
CPU system time	Использование CPU процессом в процентах (system)	Информация	-
CPU iowait time	Использование CPU процессом в процентах (iowait)	Высокая	> 15% * количество ядер CPU
CPU user time	Использование CPU процессом в процентах (user)	Информация	-
CPU utilization	Использование CPU процессом в процентах (total)	Высокая	> 50% * количество ядер CPU

Память

RSS: https://en.wikipedia.org/wiki/Resident_set_size SWAP: https://en.wikipedia.org/wiki/Paging#Unix_and_Unix-like_systems

Тип метрики	Описание	Важность	Уведомление
RSS (resident set size)	Память процесса (RSS)	Высокая	> 20% от общего количества памяти
SWAP	Использование SWAP	Высокая	> 5% от общего объема SWAP

Сеть

Рекомендуется мониторить количество соединений по каждому состоянию:
https://en.wikipedia.org/wiki/Transmission_Control_Protocol#Protocol_operation

Тип метрики	Описание	Важность	Уведомление
CLOSE	Закрыт. Сокет не используется.	Информация	-
CLOSE_WAIT	Удаленная сторона отключилась; ожидание закрытия сокета.	Средняя	Количество соединений > 2500
CLOSING	Сокет закрыт, затем удаленная сторона отключилась; ожидание подтверждения.	Информация	-
ESTABLISHED	Соединение установлено.	Средняя	Количество соединений > 5000
FIN_WAIT1	Сокет закрыт; отключение соединения.	Информация	-
FIN_WAIT2	Сокет закрыт; ожидание отключения удаленной стороны.	Информация	-
LAST_ACK	Удаленная сторона отключилась, затем сокет закрыт; ожидание подтверждения.	Информация	-
LISTEN	Ожидает входящих соединений.	Информация	-
SYN_RECV	Идет начальная синхронизация соединения.	Информация	-
SYN_SENT	Активно пытается установить соединение.	Высокая	Количество соединений > 5000
TIME_WAIT	Сокет закрыт, но ожидает пакеты в сети для обработки.	Высокая	Количество соединений > 5000

Сбор и хранение логов приложения

Сбор логов сервиса может быть организован разными способами, в зависимости от инфраструктуры. Ниже описана типовая схема, в которой используется связка `systemd — journald — Filebeat — Logstash — OpenSearch`.

Архитектура хранения и обработки

Логи приложения хранятся в **OpenSearch** — это основное хранилище для долгосрочного хранения и индексации.

Для поиска, анализа и построения дашбордов используется **OpenSearch Dashboards**.

Сбор логов с серверов, контейнеров или подов выполняется агентом **Filebeat**.

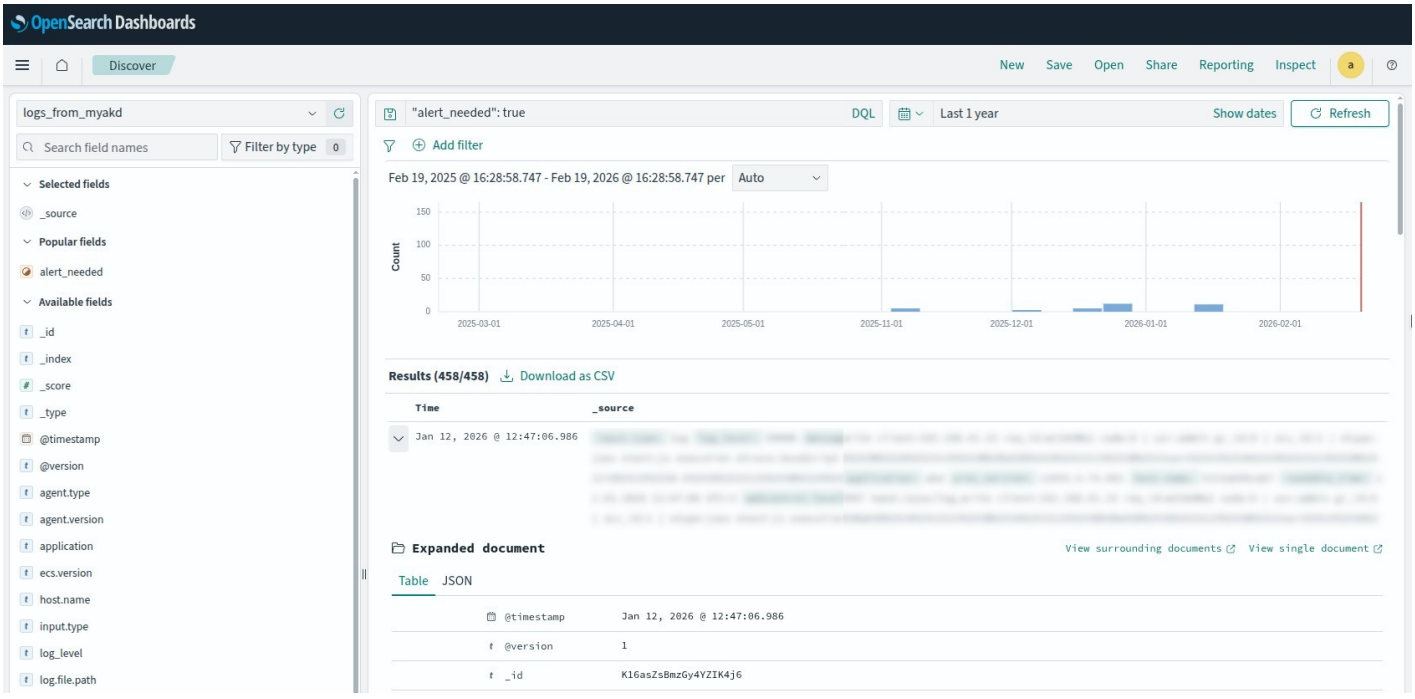
Обработка, парсинг, обогащение и нормализация логов выполняется в **Logstash**.

Типовая цепочка движения логов:

```
Приложение — journald — Filebeat — Logstash — OpenSearch — OpenSearch Dashboards
```

Все компоненты рекомендуется использовать в совместимых версиях (желательно в рамках одной ветки OpenSearch или OSS-совместимых релизов).

Итоговый результат может выглядеть так:



Логирование сервиса через systemd

Для каждого бинарного файла микросервиса скрипт-установщик создаёт unit-файл `systemd`. Пример:

```
[Unit]
Description=moodrec-offers
Wants=network-online.target
After=network-online.target

[Service]
Type=simple
ExecStart=/opt/moodrec/bin/offers
StandardOutput=journal
StandardError=journal
SyslogIdentifier=moodrec-offers
Restart=always
RestartSec=2
KillSignal=SIGTERM
KillMode=control-group
TimeoutStopSec=20

[Install]
WantedBy=multi-user.target
```

Сервис пишет логи в `journald` через `stdout` и `stderr`.

Чтение логов из journald через Filebeat

Filebeat может напрямую читать события из `journald` и отправлять их в Logstash.

Пример конфигурации:

```
filebeat.inputs:
  - type: journald
    id: moodrec-journald
    seek: cursor
    include_matches:
      - "_SYSTEMD_UNIT=moodrec-offers.service"

processors:
  - add_host_metadata: {}
  - rename:
      fields:
        - from: "systemd.unit"
          to: "service.name"
      ignore_missing: true
      fail_on_error: false

output.logstash:
  hosts: ["logstash:5044"]

logging.level: info
logging.to_files: true
logging.files:
  path: /var/log/filebeat
  name: filebeat
  keepfiles: 7
```

В этой схеме:

- Filebeat считывает записи из journald в реальном времени.
- Logstash принимает события, разбирает строки логов (`grok` , `dissect` и другие фильтры), добавляет метаданные (`application` , `environment` , `pod_name` и др.), нормализует формат и отправляет данные в OpenSearch.
- OpenSearch индексирует события и хранит их.
- OpenSearch Dashboards используется для поиска и визуализации логов.

Создание дашборда мониторинга в OpenSearch Dashboards

Мониторинг строится на данных, которые уже попадают в OpenSearch из цепочки `journald → Filebeat → Logstash`. Ниже — практический порядок действий.

1. Проверка поступления логов

В консоли сервера выполните:

```
moodrec status
```

Убедитесь, что сервисы активны.

В OpenSearch Dashboards откройте раздел Discover и проверьте, что создаются индексы (например, `moodrec-*` или `filebeat-*`) и в них появляются события.

Если индексов нет — проверьте подключение Filebeat к Logstash и Logstash к OpenSearch.

2. Создание Index Pattern (Data View)

Перейдите в: Stack Management → Data Views → Create data view

Укажите: Имя: `moodrec-logs` Index pattern: `moodrec-*` (или `filebeat-*`) Time field: `@timestamp`

После сохранения данные станут доступны для визуализаций.

3. Базовые поля для мониторинга

Для сервисов, логирующихся через systemd, обычно используются:

- `@timestamp`
- `service.name`
- `host.name`
- `log.level`
- `message`
- `application` (если добавляется в Logstash)
- `environment`

Если используется rename из `systemd.unit` → `service.name`, фильтрация будет идти по `service.name = moodrec-offers`.

4. Создание визуализаций

Перейдите в Visualize → Create visualization.

4.1 График количества логов во времени

Тип: Line chart Метрика: Count Ось X: Date histogram по `@timestamp` Интервал: auto

Этот график показывает всплески активности или ошибок.

4.2 Количество ошибок

Создайте фильтр: `log.level: error` или `message:*ERROR*`

Тип: Metric Метрика: Count

Так формируется индикатор текущего числа ошибок.

4.3 Ошибки по сервисам

Тип: Bar chart Метрика: Count Buckets → Split series → Terms → `service.name`

Добавьте фильтр по `log.level:error`.

Это покажет, какой микросервис генерирует больше ошибок.

4.4 Топ хостов

Тип: Data table Метрика: Count Buckets → Terms → `host.name`

Используется для выявления проблемных узлов.

5. Создание дашборда

Перейдите в Dashboard → Create dashboard → Add panels.

Добавьте:

- График логов во времени
- Индикатор ошибок
- Ошибки по сервисам
- Таблицу по хостам

Сохраните дашборд, например: MoodRec Monitoring

6. Настройка фильтров верхнего уровня

Добавьте глобальные фильтры:

`service.name: moodrec-offers ``environment: production` Или создайте несколько дашбордов для разных сред.

7. Практическая схема мониторинга

Операционный мониторинг:

- Резкий рост общего числа логов — возможен цикл ошибок
- Рост error/warn — деградация
- Отсутствие логов от сервиса — сервис не работает

Проверка состояния сервиса выполняется локально:

```
moodrec status
```

Если сервис активен, но логов нет — проверяется Filebeat и Logstash.

8. Расширенный мониторинг

Дополнительно можно:

Создать Alert в OpenSearch Alerting: условие — если `count(log.level:error)` > N за 5 минут.

Создать отдельный индекс для error-логов через Logstash pipeline.

Добавить парсинг времени выполнения запросов и строить latency-графики.

В итоге дашборд становится центральной точкой контроля: состояние сервисов, ошибки, нагрузка, распределение по хостам и средам.

Работу MoodRec обеспечивают следующие микропроцессы.

Сервисы

Сервис	Назначение
vectors	Работа с векторным хранилищем и поиск
trans	Преобразование описаний товаров в векторные представления
tasks	Выполнение отложенных заданий
offers	Управление продуктами
gateway	Проксирование HTTP (JSON) → gRPC (Proto)
events	Обработка событий
catboss	Работа с рекомендациями
auth	Авторизация
audience	Управление аудиториями

Проверка статуса всех сервисов:

```
moodrec status
```

Проверка статуса конкретного сервиса:

```
moodrec status vectors
moodrec status trans
moodrec status tasks
moodrec status offers
moodrec status gateway
moodrec status events
moodrec status catboss
moodrec status auth
moodrec status audience
```

Если сервис запущен корректно, вывод содержит `active (running)`. Если сервис остановлен — `inactive` или `failed`.

Управление конкретным сервисом выполняется отдельно для каждого микропроцесса.

Запуск:

```
moodrec start vectors
```

Остановка:

```
moodrec stop vectors
```

Перезапуск:

```
moodrec restart vectors
```

Аналогично для остальных сервисов:

```
moodrec start <service>
moodrec stop <service>
moodrec restart <service>
```

где `<service>` — один из: vectors, trans, tasks, offers, gateway, events, catboss, auth, audience.

После выполнения операций рекомендуется проверить состояние:

```
moodrec status <service>
```