Auraesm

Рекомендации развертыванию и эксплуатации

Введение

Программное обеспечение Auraesm представляет собой распределенную микросервисную систему, использующую распространённый стек технологий. В связи с этим, основные рекомендации по эксплуатации связаны с корректным развертыванием и взаимодействием её компонентов.

При работе с программным обеспечением могут быть полезны следующие рекомендации и решения:

1. Запускайте Auraesm с использованием систем контейнеризации, таких как Docker(docker-compose) - это значительно упростит развертывание системы и ее обновление.
2. Auraesm состоит из нескольких независимых компонентов (frontend, backend-сервисы, БД, брокер сообщений), которые в случае запуска в Docker представляют собой отдельные контейнеры, взаимодействующие через сеть.

Кроме того, наша компания окажет полное содействие по запуску и настройке системы, любому клиенту заказавшему наше программное обеспечение.

Пример запуска

Ниже приведены примеры файла docker-compose.yml для запуска системы (открытый контур), а также файла конфигурации системы

---------- docker-compose.yml ----------

```yaml
services:
  rabbitmq:
    image: rabbitmq:management
    container_name: rabbitmq
    environment:
      RABBITMQ_DEFAULT_USER: admin
      RABBITMQ_DEFAULT_PASS: uGh2xie3oh
    ports:
      - "5672:5672"
      - "15672:15672"
    volumes:
      - rabbitmq_data:/var/lib/rabbitmq
    restart: unless-stopped

  postgres:
    container_name: postgres
    image: postgres:15
    volumes:
      - pg-data:/var/lib/postgresql/data
    environment:
      POSTGRES_USER: ${PG_USER}
      POSTGRES_PASSWORD: ${PG_PASS}
      POSTGRES_DB: ${PG_DB}
    ports:
      - 5432:5432

  mssql_intellect:
    image: mcr.microsoft.com/mssql/server:2022-latest
    container_name: mssql-intellect
    ports:
      - "1433:1433"
    environment:
      SA_PASSWORD: "YourStrong!Passw0rd"
      ACCEPT_EULA: "Y"
    volumes:
      - mssql_data:/var/opt/mssql
    restart: always

  soo-integration-service:
    image: docker.direct.farm/soo/integration-service:dev
    container_name: soo-integration-service
    volumes:
      - uploads:/uploads
    environment:
      INTELLECT_MSSQL_URL: ${INTELLECT_MSSQL_URL}
      PG_URL: ${PG_URL}
      RMQ_URL: ${RMQ_URL}
      APP_ENV: ${APP_ENV}
      WAZUH_URL: ${WAZUH_URL}
      WAZUH_USER: ${WAZUH_USER}
      WAZUH_PASS: ${WAZUH_PASS}
      WAZUH_VERIFY_TLS: ${WAZUH_VERIFY_TLS}
    ports:
```

```
    - 8070:8080
  depends_on:
    - postgres
    - mssql_intellect
    - rabbitmq

soo-api-service:
  image: docker.direct.farm/soo/api-service:dev
  container_name: soo-api-service
  volumes:
    - uploads:/uploads
  environment:
    PG_URL: ${PG_URL}
    RMQ_URL: ${RMQ_URL}
    AUTH_HASH_KEY: ${AUTH_HASH_KEY}
    APP_ENV: ${APP_ENV}
  ports:
    - 8080:8080
  depends_on:
    - postgres
    - rabbitmq

soo-incident-service:
  image: docker.direct.farm/soo/incident-service:dev
  container_name: soo-incident-service
  environment:
    PG_URL: ${PG_URL}
    RMQ_URL: ${RMQ_URL}
  ports:
    - 8090:8080
  depends_on:
    - postgres
    - rabbitmq

soo-report-service:
  image: docker.direct.farm/soo/report-service:dev
  container_name: soo-report-service
  environment:
    PG_URL_ASYNC: ${PG_URL_ASYNC}
    RMQ_URL: ${RMQ_URL}
    STATIC_PATH: ${STATIC_PATH}
    SIGN_CERT: ${SIGN_CERT}
    SIGN_KEY: ${SIGN_KEY}
    SIGN_KEY_PASSWORD: ${SIGN_KEY_PASSWORD}
    SMTP_PORT: ${SMTP_PORT}
    SMTP_HOST: ${SMTP_HOST}
    SMTP_USER: ${SMTP_USER}
    SMTP_PASSWORD: ${SMTP_PASSWORD}
    SMTP_FROM: ${SMTP_FROM}
  volumes:
    - uploads:/app/static
  depends_on:
    - postgres
    - rabbitmq

soo-telegram-service:
  image: docker.direct.farm/soo/telegram-service:dev
  container_name: soo-telegram-service
  environment:
```

```
    TELEGRAM_TOKEN: ${TELEGRAM_TOKEN}
    PG_URL_ASYNC_TG: ${PG_URL_ASYNC_TG}
    PG_URL_TG: ${PG_URL_TG}
    RMQ_URL: ${RMQ_URL}
    REVIEW_CHAT_ID: ${REVIEW_CHAT_ID}
  depends_on:
    - postgres
    - rabbitmq

 soo-tg-network-service:
  image: docker.direct.farm/soo/tg-network-service:dev
  container_name: soo-tg-network-service
  environment:
    TELEGRAM_TOKEN: ${TELEGRAM_TOKEN_NETWORK}
    PG_URL_ASYNC_TG: ${PG_URL_ASYNC_TG}
    PG_URL_TG: ${PG_URL_TG}
    RMQ_URL: ${RMQ_URL}
  depends_on:
    - postgres
    - rabbitmq
    - soo-telegram-service

 soo-front:
  image: docker.direct.farm/soo/front:dev
  container_name: soo-front
  ports:
    - "3000:3000"
  environment:
    VITE_BACKEND_URL: ${VITE_BACKEND_URL}
    APP_ENV: ${APP_ENV}

volumes:
 pg-data:
 mssql_data:
 rabbitmq_data:
 uploads:
  driver: local
  driver_opts:
    o: bind
    type: none
    device: /home/soo/uploads
```

---------- .env ----------

```
PG_URL=postgres://user:pass@postgres:5432/postgres
RMQ_URL=amqp://admin:uGh2xie3oh@rabbitmq:5672/

INTELLECT_MSSQL_URL=sqlserver://sa:YourStrong!Passw0rd@mssql_intellect:1433?database=master&en-
crypt=disable

PG_USER=user
PG_PASS=pass
PG_DB=postgres

PG_URL_ASYNC=postgresql+asyncpg://user:pass@postgres:5432/postgres

STATIC_PATH=static

SIGN_CERT=
```

```
SIGN_KEY=
SIGN_KEY_PASSWORD=

PG_URL_TG=postgresql://user:pass@postgres:5432/telegram
PG_URL_ASYNC_TG=postgresql+asyncpg://user:pass@postgres:5432/telegram

TELEGRAM_TOKEN=your_telegram_token_here
TELEGRAM_TOKEN_NETWORK=your_telegram_network_token_here
REVIEW_CHAT_ID=your_chat_id_here

SMTP_HOST=smtp.mail.ru
SMTP_PORT=465
SMTP_USER=your_email@mail.ru
SMTP_PASSWORD=your_email_password
SMTP_FROM=your_email@mail.ru

AUTH_HASH_KEY=your_auth_hash_key_here

VITE_BACKEND_URL=http://localhost:8080

APP_ENV=development

WAZUH_URL=https://wazuh_host:9200
WAZUH_USER=wazuh_user
WAZUH_PASS=wazuh_password
WAZUH_VERIFY_TLS=false
```

Для запуска системы на таком примере необходимо

1. Выполнить команду docker-compose up -d
2. Проверить доступность системы на портах 3000 (фронтенд) и 8080 (бэкенд АПИ)
1. В случае возникновения проблем, выполнить первичную диагностику с использованием команд docker logs для соответствующих сервисов (soo-api-service, soo-integration-service, soo-front и др.)

Рекомендации по эксплуатации

Как сказано во введении, система не имеет специфических проблем и рекомендаций, в тоже время стоит обозначить общие рекомендации:

1. Для снижения риска потери данных в случае сбоев инфраструктуры – рекомендуется выполнять периодическое резервное копирование данных системы. Для этого стоит использовать встроенные средства СУБД, такие как pg_dump для PostgreSQL. Также стоит выполнять резервное копирование пользовательских файлов системы, которые в примере выше располагаются в каталоге /home/soo/uploads
2. Для выявления практически любых ошибок в работе системы могут быть использованы команды docker logs для соответствующих сервисов (soo-api-service, soo-integration-service, soo-front и др.)