ООО «Цифровое проектирование»

Инструкция по установке программы для ЭВМ «Система моделирования и оптимизации «ОНЕГА»

2026

ЦИФРОВОЕ ПРОЕКТИРОВАНИЕ

АННОТАЦИЯ

Настоящий документ содержит информацию, необходимую для установки программы для ЭВМ «Система моделирования и оптимизации «ОНЕГА» (далее также - ОНЕГА).

Оглавление

1. Общие сведения

Полное наименование программы для ЭВМ: Система моделирования и оптимизации «ОНЕГА».

Сокращённое наименование программы для ЭВМ: ОНЕГА.

2. Подготовка серверов

2.1. Описание поставки

Настоящее решение поставляется в виде Docker контейнеров сервисов, необходимых для развёртывания в изолированной среде.

Для получения доступа к сервису необходима настройка DNS сервера. Не требуется в случае доступа с хоста развертывания.

Сервис состоит из четырех компонентов:

- NGINX веб-сервер, используется для предоставления доступа к сервису
- Frontend-сервис, используется для отображения графического интерфейса сервиса
- Backend-сервис, используется для выполнения необходимых вычислений, управления пользователями и для решения иных административных задач
- PostgreSQL СУБД, используется для хранения данных
- Redis СУДБ используется для назначения вычислений расчетчикам.
- Объектное хранилище MiniO используется для хранения полученных промежуточных вычислений
- Система авторизации и аутентификации Keycloak используется для реализации централизованного управления доступом к приложению

Сервис поставляется в виде импортов образов Docker, необходимых для запуска и работы сервиса, архива volume содержащего БД, файла оркестратора контейнеров docker-compose.yaml и файла переменных окружения, необходимых для работы сервиса.

Настоящая инструкция написана по результатам развертывания на ОС Windows 11.

2.2. Системные требования

1) Установленный Docker Desktop (Инструкция была воспроизведена на хосте с Docker Desktop version 4.68.0, 223695).
2) Установленный плагин Docker Compose (Инструкция была воспроизведена на хосте с Docker Compose version v5.1.1).
3) Доступ к DNS серверу, способному разрешать хосты в демонстрационном домене. Не требуется при полностью локальном запуске.
4) Доступный для подключения клиентов порт 3000, в случае развертывания без дополнительных прокси.
5) 40 Гб свободного места на диске.
6) 16 Гб оперативной памяти.
7) 4 ядра CPU.

2.3. Порядок развертывания

1) Ниже подробное описание файлов, входящих в дистрибутив:
   - Получение архивов образов, необходимых для запуска сервиса.
     backend.tar.gz
     frontend.tar.gz
     postgres.tar.gz
     redis.tar.gz
     minio.tar.gz
     minio-mc.tar.gz
     keycloak.tar.gz
   - Получение файлов конфигурации развертывания и переменных окружения
     ./certs – директория с предподготовленными сертификатами
     ./keycloak – директория с конфигурацией realm
     .env – файл глобальных переменных Docker compose
     docker.env – файл переменных запуска Docker compose
     README.md – настоящая инструкция
     frontend.nginx.conf – файл конфигурации NGINX встроенного в образ frontend
     v1.json – пример для расчета
2) Данная инструкция предусматривает установку, развертывание всех компонентов и доступ к ним с локального ПК, однако также можно развернуться на различных хостах, для обеспечения работоспособности такого развертывания необходимо настроить DNS сервер для предоставления доступа к сервису клиентам. В настоящей инструкции рассмотрено создание демонстрационной зоны на сервере имен BIND9. В случае использования иных серверов имен следует использовать этот пункт, как референсный.

- Домен для демо-стенда: demo-onega.loc
- На сервере BIND9 создать зону:
    - sudo nano /etc/bind/zones/db.demo-onega.loc

      ;
      ;BIND data file for local loopback interface
      ;
      $TTL 604800
      @ IN SOA YOUR_DNS_SRV.DOMAIN.NAME. admin.DOMAIN.NAME. (
      1 ; Serial
      604800 ; Refresh
      86400 ; Retry
      2419200 ; Expire
      604800) ; Negative Cache TTL
      ;
      @ IN NS YOUR_DNS_SRV.DOMAIN.NAME.
      @ IN NS YOUR_2ND_DNS_SRV.DOMAIN.NAME.
      YOUR_DNS_SRV.DOMAIN.NAME. IN A YOUR_DNS_IP
      YOUR_2ND_DNS_SRV.DOMAIN.NAME. IN A YOUR_2_DNS_IP
      sigma-front.demo-onega.loc. IN A SERVICE_IP
      sigma-api.demo-onega.loc. IN A SERVICE_IP
    - sudo nano /etc/bind/named.conf.local

      zone "demo-onega.loc" {
      type master;
      file "/etc/bind/zones/db.demo-onega.loc";
      };
    - sudo systemctl restart named
- Проверить доступность зоны как с хоста, на котором планируется развертывание так и с хостов клиентов
    - nslookup sigma-front.demo-onega.loc (Ответ должен содержать IP адрес сервера, на котором будут доступны веб-интерфейсы сервиса)

3) На выбранном для развертывания сервере разрешить доступ по порту 3000
4) В предлагаемом варианте развертывания файлы размещаются в директории numdes, в случае необходимости директорию развертывания можно изменить, соответственно скорректировав настоящую инструкцию
5) Предполагается, что файл distributive.tar доставлен в директорию $HOME пользователя от имени которого проводится развертывание на целевом сервере.
6) Выполнить следующие действия. Для ОС на базе Linux действия представляют собой команды bash, пояснения к ним выделены при помощи символа #

   # Создние директории
   sudo tar -xvf $HOME/distributive.tar -C /
   sudo chown -R $USER:$USER /numdes
   cd /numdes
   #Процедура загрузки образов

docker load < backend.tar.gz
docker load < frontend.tar.gz
docker load < postgres.tar.gz
docker load < redis.tar.gz
docker load < minio.tar.gz

docker load < minio-mc.tar.gz
docker load < keycloak.tar.gz
Экспорт тэгов Docker образов как переменные
export SERVICE_IMAGE=docker-monthly.numdes.com/onega/nd_onega_sigma_optimizator-backend_master:211074
export FRONTEND_IMAGE=docker-monthly.numdes.com/onega/nd_onega_sigma_optimizator-frontnd_master:210730

Для ОС на базе Windows процесс выглядит схожим образом:
В созданной директории numdes распаковать содержимое файла distributives.tar
В CLI терминале прейти в созданную директорию, убедиться что движок Docker d Docker Desktop находится в статусе «Запущен», выполнить
docker load < backend.tar.gz
docker load < frontend.tar.gz
docker load < postgres.tar.gz
docker load < redis.tar.gz
docker load < s3.tar.gz
docker load < keycloak.tar.gz

В файл hosts (C:\WINDOWS\System32\etc\drivers\ или /etc/hosts в зависимости от ОС необходимо внести и сохранить запись)
127.0.0.1 keycloak

Запуск контейнеров
docker-compose up --wait --wait-timeout 360 --detach

Проверка состояния контейнеров
docker ps

7) Доступ к сервису
    - Пользовательский интерфейс: http://localhost:3000

3. Запуск и работа

3.1. Подготовка окружения

Заведение пользователя системы

Открыть в веб-браузере http://localhost:8080.

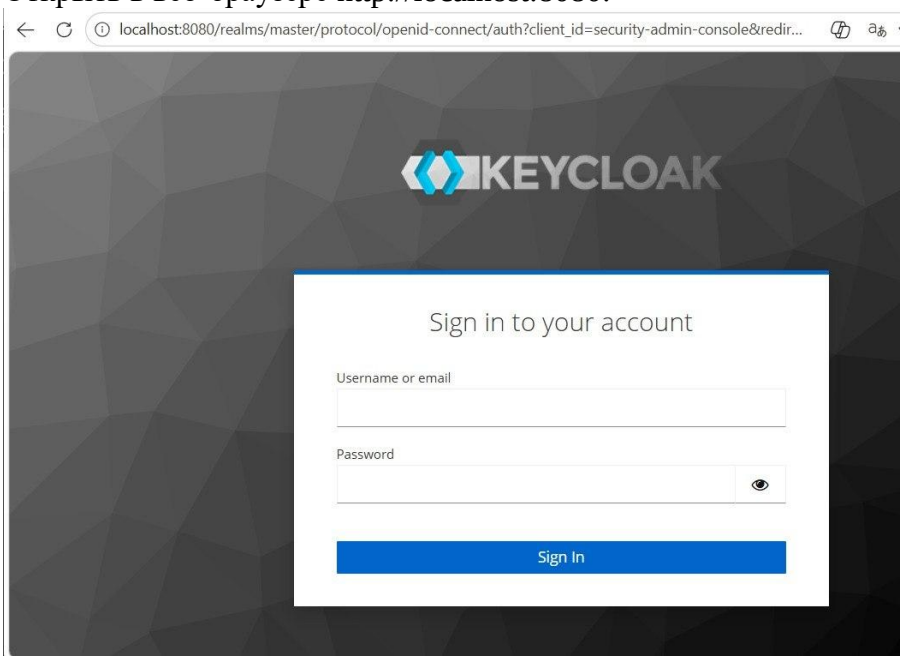

Рисунок 1 – Ввод логина и пароля.

Ввести логин/пароль (admin/admin).

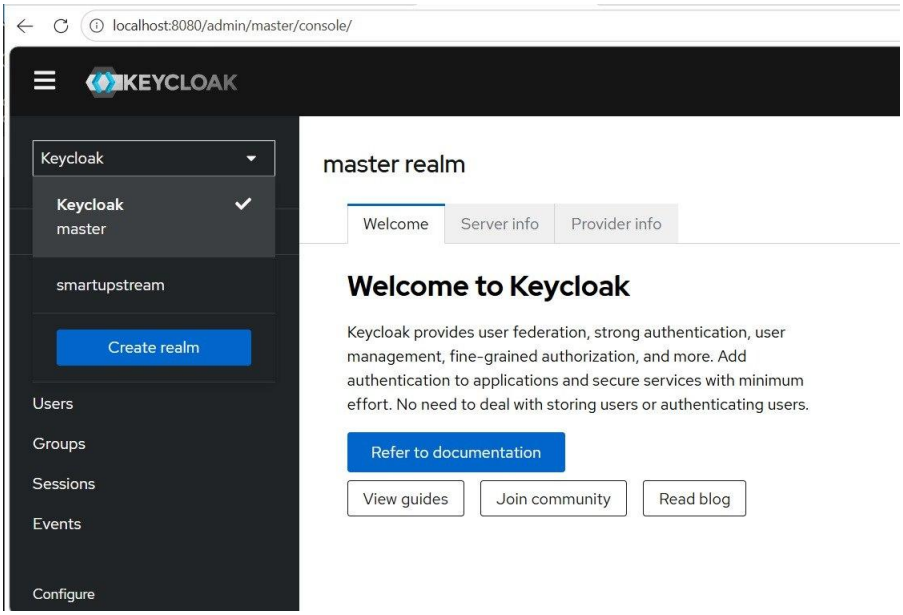

Рисунок 2 – Раздел realm.

Перейти в realm smartupstream.

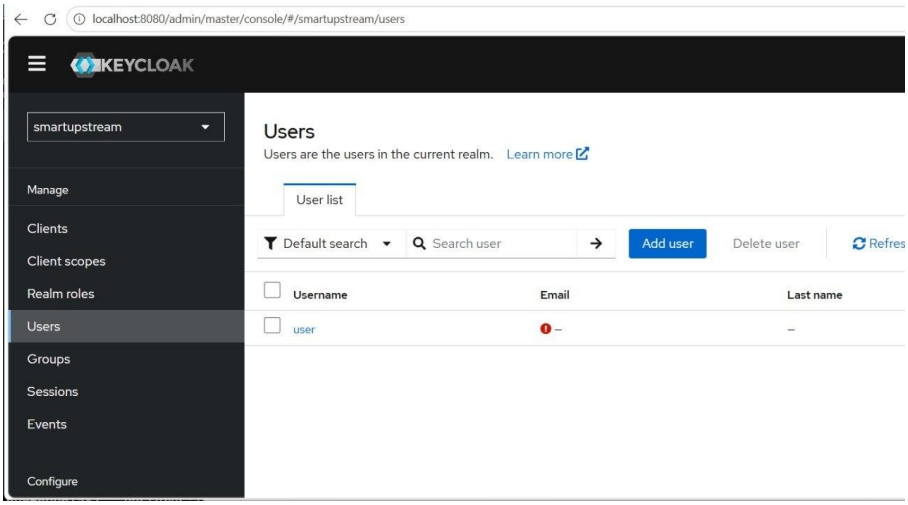

Рисунок 3 – Раздел Users.

Перейти в раздел Users нажать на Add user.

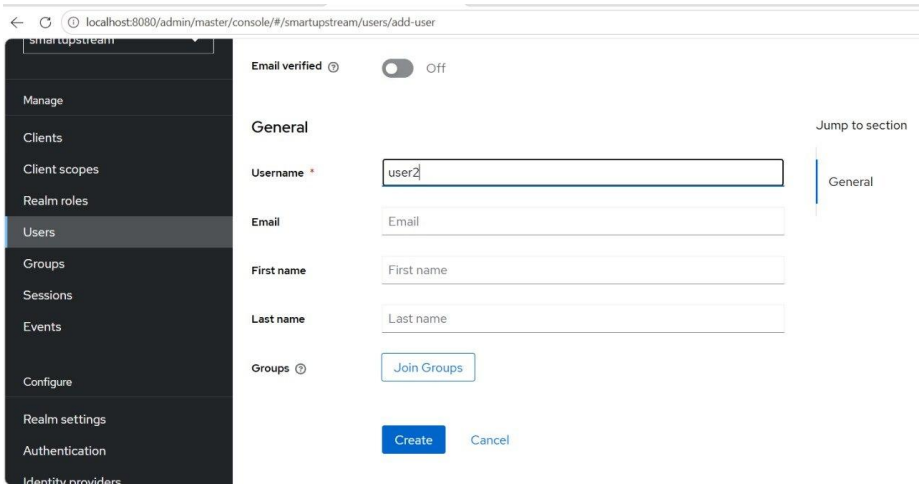

Рисунок 4 – Добавление пользователя.

Ввести имя пользователя, нажать Create.

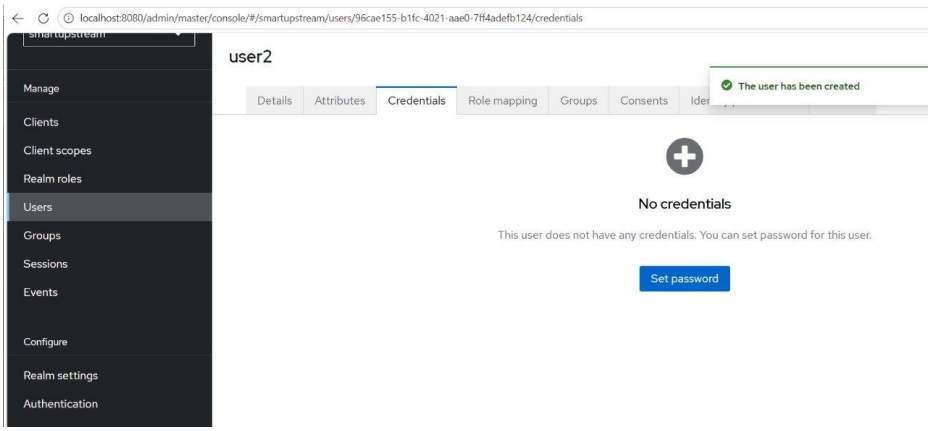

Рисунок 5 – Добавление пользователя.

Установить пароль.

Инициализация приложения

В веб-браузере в режиме «Инкогнито» открыть http://localhost:3000
Ввести учетные данные, продолжить вход.
Закрыть окно веб-браузера.

Вход в приложение

В обычном режиме веб-браузера открыть http://localhost:3000
Ввести учетные данные, продолжить вход.

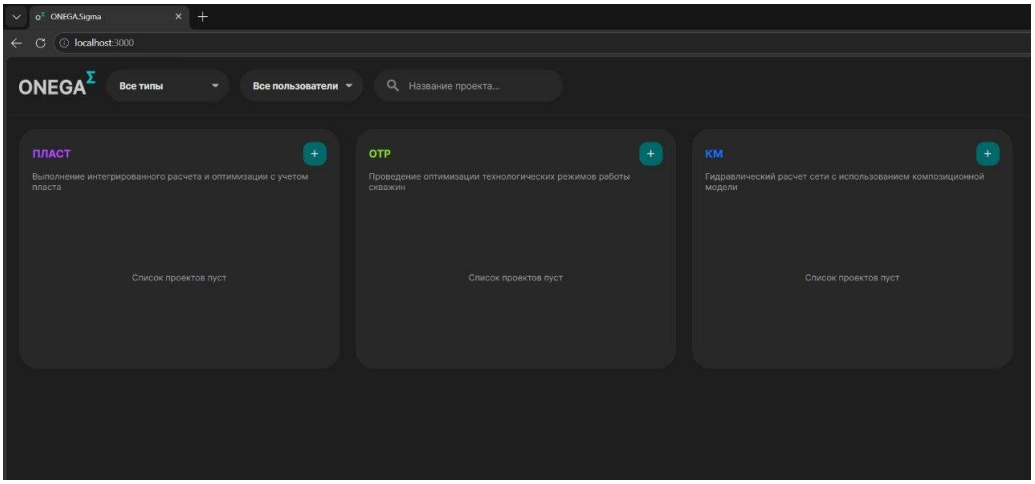

Рисунок 6 – Запуск приложения.

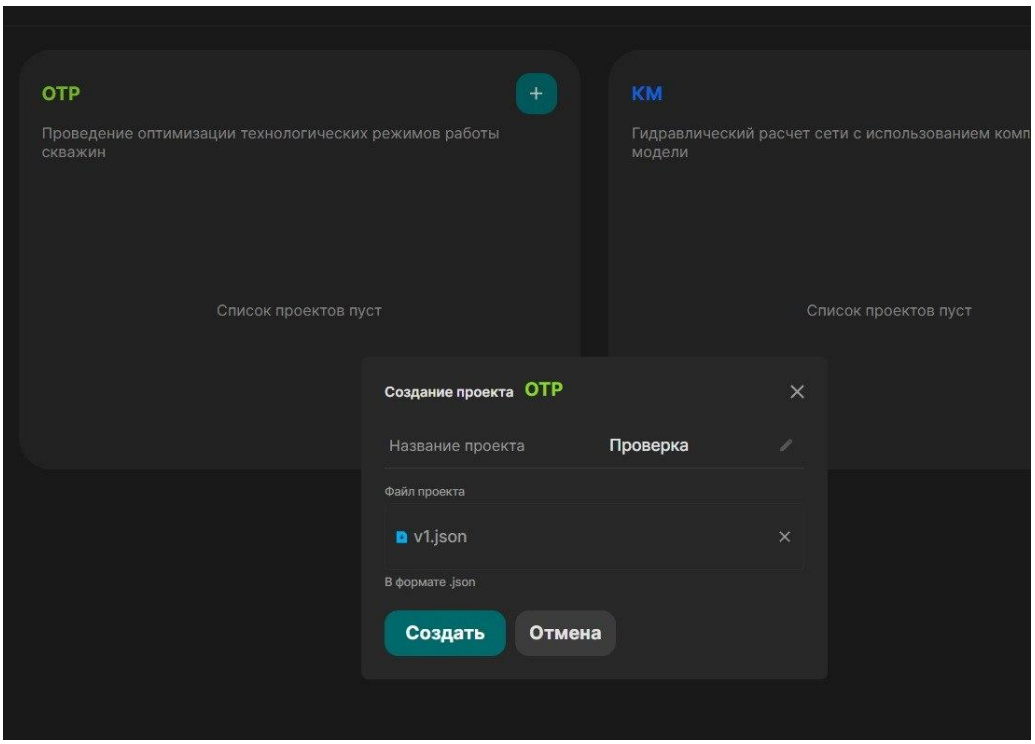

Рисунок 7 – Создание проекта.

Для проверки работы системы в дистрибутив включен файл v1.json В разделе ОТР выбрать (+). Добавить название, файл и нажать «Создать».

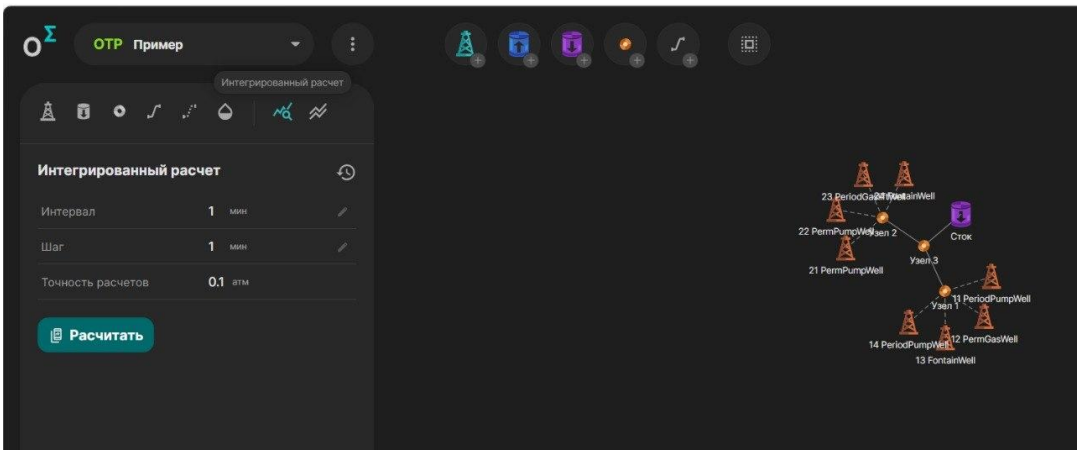

Рисунок 8 – Запуск расчёта.

Для запуска расчетов следует перейти на вкладку «Интегрированный расчет» и выбрать «Рассчитать».

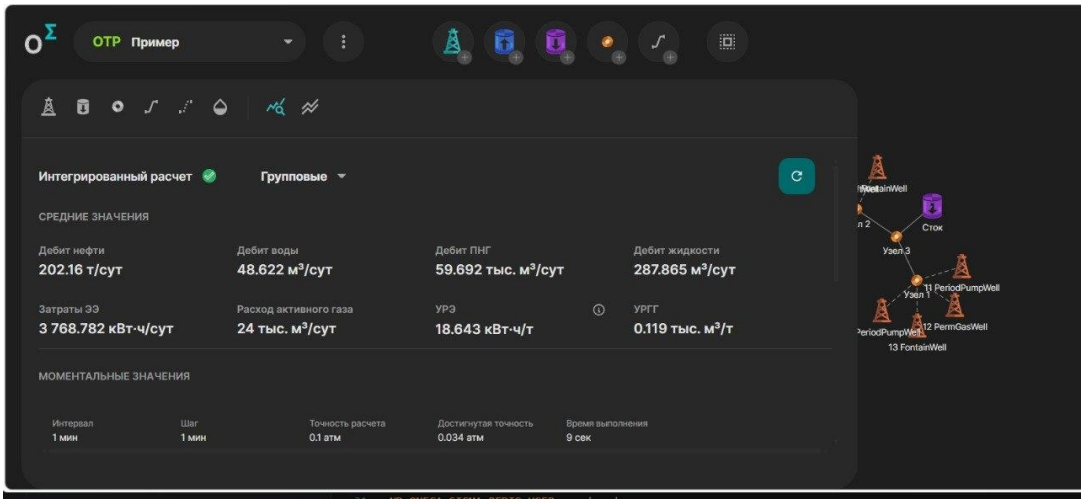

Рисунок 9 – Результаты расчёта.

Аналогично следует поступить и для расчета «Оптимизация режима».

4. Приложения

4.1. docker-compose.yaml

```
services:
  postgres:
    image: postgres:16
    container_name: postgres_container
    environment:
      POSTGRES_USER: ${ND_ONEGA_SIGMA_POSTGRES_USER}
      POSTGRES_PASSWORD: ${ND_ONEGA_SIGMA_POSTGRES_PASSWORD}
      POSTGRES_DB: ${ND_ONEGA_SIGMA_POSTGRES_DBNAME}
    ports:
      - "${ND_ONEGA_SIGMA_POSTGRES_PORT}:5432"
    volumes:
      - postgres_data:/var/lib/postgresql/data
      - ./certs/postgres_server.crt:/var/lib/postgresql/server_certs/server.crt:ro
      - ./certs/postgres_server.key:/var/lib/postgresql/server_certs/server.key:ro
      - ./certs/ca.crt:/var/lib/postgresql/server_certs/ca.crt:ro
    entrypoint:
      - bash
      - -c
      - |
        cp /var/lib/postgresql/server_certs/server.key /tmp/server.key
        chown 999:999 /tmp/server.key
        chmod 600 /tmp/server.key
        exec docker-entrypoint.sh postgres \
          -c ssl=on \
          -c ssl_cert_file=/var/lib/postgresql/server_certs/server.crt \
          -c ssl_key_file=/tmp/server.key \
          -c ssl_ca_file=/var/lib/postgresql/server_certs/ca.crt
    healthcheck:
      test: ["CMD-SHELL", "pg_isready -U $$POSTGRES_USER -d $$POSTGRES_DB"]
      interval: 5s
      timeout: 5s
      retries: 10
      start_period: 30s
    networks:
      - onega_sigma_network

  s3:
    image: minio/minio
    container_name: minio_s3_container
    command: server /data --console-address ":9001"
    environment:
      MINIO_ROOT_USER: ${ND_ONEGA_SIGMA_S3_LOGIN}
      MINIO_ROOT_PASSWORD: ${ND_ONEGA_SIGMA_S3_PASSWORD}
    volumes:
      - s3_data:/data
    ports:
      - "9000:9000"
      - "9001:9001"
    healthcheck:
      test: ["CMD", "curl", "-f", "http://localhost:9000/minio/health/live"]
      interval: 5s
      timeout: 5s
      retries: 10
      start_period: 10s
```

```
    networks:
      - onega_sigma_network

  s3-init:
    image: minio/mc
    depends_on:
      s3:
        condition: service_healthy
    restart: on-failure
    environment:
      MINIO_USER: ${ND_ONEGA_SIGMA_S3_LOGIN}
      MINIO_PASS: ${ND_ONEGA_SIGMA_S3_PASSWORD}
      MINIO_BUCKET: ${ND_ONEGA_SIGMA_S3_BUCKET}
    entrypoint:
      - sh
      - -c
      - |
        mc alias set local http://s3:9000 "$$MINIO_USER" "$$MINIO_PASS"
        mc mb --ignore-existing "local/$$MINIO_BUCKET"
        echo "Bucket $$MINIO_BUCKET is ready."
    networks:
      - onega_sigma_network

  redis:
    image: redis:latest
    container_name: redis_onega_sigma_container
    environment:
      ND_ONEGA_SIGMA_REDIS_USER: ${ND_ONEGA_SIGMA_REDIS_USER}
      ND_ONEGA_SIGMA_REDIS_PASSWORD: ${ND_ONEGA_SIGMA_REDIS_PASSWORD}
    command:
      - redis-server
      - --databases 37
      - --user ${ND_ONEGA_SIGMA_REDIS_USER} on >${ND_ONEGA_SIGMA_REDIS_PASSWORD} ~* &* +@all
      - --port 0
      - --tls-port 6380
      - --tls-cert-file /certs/redis_server.crt
      - --tls-key-file /certs/redis_server.key
      - --tls-ca-cert-file /certs/ca.crt
      - --tls-auth-clients yes
    volumes:
      - redis_data:/data
      - ./certs:/certs:ro
    ports:
      - "${ND_ONEGA_SIGMA_REDIS_PORT}:6380"
    healthcheck:
      test: ["CMD-SHELL", "redis-cli --tls --cert /certs/redis_client.crt --key /certs/redis_client.key --cacert /certs/ca.crt -p 6380 --user $$ND_ONEGA_SIGMA_REDIS_USER --pass $$ND_ONEGA_SIGMA_REDIS_PASSWORD ping"]
      interval: 5s
      timeout: 5s
      retries: 10
      start_period: 10s
    networks:
      - onega_sigma_network

  keycloak:
    image: quay.io/keycloak/keycloak:25.0
    container_name: keycloak_container
    user: root
    command: start-dev --import-realm
    environment:
      KEYCLOAK_ADMIN: admin
```

```
    KEYCLOAK_ADMIN_PASSWORD: admin
  volumes:
    - ./keycloak:/opt/keycloak/data/import:ro
    - keycloak_data:/opt/keycloak/data/h2
  ports:
    - "8080:8080"
  healthcheck:
    test: ["CMD", "bash", "-c", "exec 3<>/dev/tcp/localhost/8080"]
    interval: 10s
    timeout: 5s
    retries: 12
    start_period: 15s
  networks:
    - onega_sigma_network

web:
  image: ${SERVICE_IMAGE}
  container_name: web_container
  env_file: docker.env
  environment:
    RUN_REST: "1"
    PUBLISH_PORT: "8000"
    N_WORKERS: "${N_WORKERS:-4}"
    ENVIRONMENT: "local"
    ROOT_PATH: ""
    ND_ONEGA_SIGMA_POSTGRES_USER: "${ND_ONEGA_SIGMA_POSTGRES_USER}"
    ND_ONEGA_SIGMA_POSTGRES_PASSWORD: "${ND_ONEGA_SIGMA_POSTGRES_PASSWORD}"
    ND_ONEGA_SIGMA_POSTGRES_DBNAME: "${ND_ONEGA_SIGMA_POSTGRES_DBNAME}"
    ND_ONEGA_SIGMA_S3_LOGIN: "${ND_ONEGA_SIGMA_S3_LOGIN}"
    ND_ONEGA_SIGMA_S3_PASSWORD: "${ND_ONEGA_SIGMA_S3_PASSWORD}"
    ND_ONEGA_SIGMA_REDIS_USER: "${ND_ONEGA_SIGMA_REDIS_USER}"
    ND_ONEGA_SIGMA_REDIS_PASSWORD: "${ND_ONEGA_SIGMA_REDIS_PASSWORD}"
  entrypoint:
    - bash
    - -c
    - |
      cp /numdes/certs/postgres_client.key /tmp/postgres_client.key
      cp /numdes/certs/redis_client.key /tmp/redis_client.key
      chmod 600 /tmp/postgres_client.key /tmp/redis_client.key
      export ND_ONEGA_SIGMA_POSTGRES_SSL_KEY_FILEPATH=/tmp/postgres_client.key
      export ND_ONEGA_SIGMA_REDIS_SSL_KEY_FILEPATH=/tmp/redis_client.key
      exec ./docker/docker-entrypoint.sh
  volumes:
    - ./certs:/numdes/certs:ro
  ports:
    - "8000:8000"
  depends_on:
    postgres:
      condition: service_healthy
    redis:
      condition: service_healthy
    s3:
      condition: service_healthy
    keycloak:
      condition: service_healthy
  healthcheck:
    test: ["CMD", "curl", "-f", "http://localhost:8000/health/live"]
    interval: 10s
    timeout: 5s
    retries: 12
    start_period: 60s
```

```
    networks:
      - onega_sigma_network

  worker:
    image: ${SERVICE_IMAGE}
    container_name: worker_container
    env_file: docker.env
    environment:
      RUN_WORKER: "1"
      OMP_NUM_THREADS: "${OMP_NUM_THREADS:-4}"
      ENVIRONMENT: "local"
      ND_ONEGA_SIGMA_POSTGRES_USER: "${ND_ONEGA_SIGMA_POSTGRES_USER}"
      ND_ONEGA_SIGMA_POSTGRES_PASSWORD: "${ND_ONEGA_SIGMA_POSTGRES_PASSWORD}"
      ND_ONEGA_SIGMA_POSTGRES_DBNAME: "${ND_ONEGA_SIGMA_POSTGRES_DBNAME}"
      ND_ONEGA_SIGMA_S3_LOGIN: "${ND_ONEGA_SIGMA_S3_LOGIN}"
      ND_ONEGA_SIGMA_S3_PASSWORD: "${ND_ONEGA_SIGMA_S3_PASSWORD}"
      ND_ONEGA_SIGMA_REDIS_USER: "${ND_ONEGA_SIGMA_REDIS_USER}"
      ND_ONEGA_SIGMA_REDIS_PASSWORD: "${ND_ONEGA_SIGMA_REDIS_PASSWORD}"
    entrypoint:
      - bash
      - -c
      - |
        cp /numdes/certs/postgres_client.key /tmp/postgres_client.key
        cp /numdes/certs/redis_client.key /tmp/redis_client.key
        chmod 600 /tmp/postgres_client.key /tmp/redis_client.key
        export ND_ONEGA_SIGMA_POSTGRES_SSL_KEY_FILEPATH=/tmp/postgres_client.key
        export ND_ONEGA_SIGMA_REDIS_SSL_KEY_FILEPATH=/tmp/redis_client.key
        exec ./docker/docker-entrypoint.sh
    volumes:
      - ./certs:/numdes/certs:ro
    depends_on:
      postgres:
        condition: service_healthy
      redis:
        condition: service_healthy
      s3:
        condition: service_healthy
    networks:
      - onega_sigma_network

  frontend:
    image: ${FRONTEND_IMAGE}
    container_name: frontend_container
    entrypoint:
      - sh
      - -c
      - |
        echo "window.env = { API_URL: '${API_URL}', BASENAME: '/' }" > /app/env.js
        exec nginx -g 'daemon off;'
    volumes:
      - ./frontend.nginx.conf:/etc/nginx/conf.d/default.conf:ro
    ports:
      - "3000:80"
    depends_on:
      web:
        condition: service_healthy
    networks:
      - onega_sigma_network

volumes:
  postgres_data:
```

```
  s3_data:
  redis_data:
  keycloak_data:

networks:
  onega_sigma_network:
```

4.2. .env

```
# ============================================================
# Переменные уровня запуска Compose активируются при выполнении docker compose.
# Сюда добавить секреты и значения необходимые при запуске.
# ============================================================

# --- Application images ---
SERVICE_IMAGE=docker-monthly.numdes.com/onega/nd_onega_sigma_optimizator-backend_master:211074
FRONTEND_IMAGE=docker-monthly.numdes.com/onega/nd_onega_sigma_optimizator-frontend_master:210730

# --- Фронтенд ---
# API_URL записывается в /app/env.js при запуске контейнера и читается
# браузером при открытии. URL должен быть доступен глобально за пределами Docker сети
API_URL=http://localhost:8000

# --- Настройка утилизации ресурсов ---
N_WORKERS=4
OMP_NUM_THREADS=4

# --- СУБД ---
ND_ONEGA_SIGMA_POSTGRES_USER=pt_smartupstream_features_full_access
ND_ONEGA_SIGMA_POSTGRES_PASSWORD=qwertyuiop
ND_ONEGA_SIGMA_POSTGRES_DBNAME=pt-smartupstream-features-master
ND_ONEGA_SIGMA_POSTGRES_PORT=5432

# --- Объектное хранилище (S3) ---
ND_ONEGA_SIGMA_S3_LOGIN=service
ND_ONEGA_SIGMA_S3_PASSWORD=qwertyuiop
ND_ONEGA_SIGMA_S3_BUCKET=pipetimum

# --- Redis ---
ND_ONEGA_SIGMA_REDIS_USER=smartupstream
ND_ONEGA_SIGMA_REDIS_PASSWORD=qwertyuiop
ND_ONEGA_SIGMA_REDIS_PORT=6380
```

4.3. docker.env

```
# ============================================================
# Значения переменных уровня приложений, передаются в рабочие контейнеры.
# Для разрешения имен используются названия сервисов  Docker.
# Секреты переданы через .env  не следует дублировать их тут
# ============================================================

# --- СУДБ ---
ND_ONEGA_SIGMA_POSTGRES_HOST=postgres
ND_ONEGA_SIGMA_POSTGRES_PORT=5432
ND_ONEGA_SIGMA_POSTGRES_SSL_CERTIFICATE_FILEPATH=/numdes/certs/postgres_client.crt
ND_ONEGA_SIGMA_POSTGRES_SSL_KEY_FILEPATH=/numdes/certs/postgres_client.key
ND_ONEGA_SIGMA_POSTGRES_SSL_CA_CERTIFICATES_FILEPATH=/numdes/certs/ca.crt

# --- Объектное хранилище (S3) ---
```

```
ND_ONEGA_SIGMA_S3_FOLDER=smartupstream_test
ND_ONEGA_SIGMA_S3_URL=http://s3:9000
ND_ONEGA_SIGMA_S3_BUCKET=pipetimum

# --- Keycloak ---
ND_ONEGA_SIGMA_KEYCLOAK_REALM_URL=http://keycloak:8080/realms/smartupstream
ND_ONEGA_SIGMA_KEYCLOAK_CLIENT_ID=smartupstream-app
ND_ONEGA_SIGMA_KEYCLOAK_CLIENT_SECRET=QZilZxwlgaBJUJmMOsX-jGDtfn1I4Y0bpAID_kc1f6I

# --- Redis ---
ND_ONEGA_SIGMA_REDIS_HOST=redis
ND_ONEGA_SIGMA_REDIS_PORT=6380
ND_ONEGA_SIGMA_REDIS_DB=36
ND_ONEGA_SIGMA_REDIS_QUEUE=local
ND_ONEGA_SIGMA_REDIS_SSL_CERTIFICATE_FILEPATH=/numdes/certs/redis_client.crt
ND_ONEGA_SIGMA_REDIS_SSL_KEY_FILEPATH=/numdes/certs/redis_client.key
ND_ONEGA_SIGMA_REDIS_SSL_CA_CERTIFICATES_FILEPATH=/numdes/certs/ca.crt

# --- Поведение приложения ---
ND_ONEGA_SIGMA_CORS_ORIGINS=http://localhost:3000
ND_ONEGA_SIGMA_COOKIE_SESSION_ID_SAMESITE=none
ND_ONEGA_SIGMA_COOKIE_LOGIN_SAMESITE=none
ND_ONEGA_SIGMA_IS_HTTPS=false
ND_ONEGA_SIGMA_TEST_MODE=true
```

4.4. Описание переменных

Переменная	Описание
SERVICE_IMAGE	Docker-образ backend-сервиса приложения. Используется docker compose для запуска нужной версии контейнера.
FRONTEND_IMAGE	Docker-образ frontend-сервиса приложения. Определяет версию frontend-контейнера.
API_URL	Публичный URL backend API, который записывается в /app/env.js при старте frontend-контейнера и используется браузером. Должен быть доступен клиенту вне Docker-сети.
N_WORKERS	Количество worker-процессов/потоков приложения, используемое для настройки утилизации ресурсов.
OMP_NUM_THREADS	Количество потоков OpenMP для вычислительных библиотек. Влияет на параллельность CPU-нагрузки.
ND_ONEGA_SIGMA_POSTGRES_USER	Имя пользователя PostgreSQL для подключения приложения к базе данных.
ND_ONEGA_SIGMA_POSTGRES_PASSWORD	Пароль пользователя PostgreSQL. Чувствительное значение: лучше хранить в секретах/защищённом .env, не коммитить в репозиторий.
ND_ONEGA_SIGMA_POSTGRES_DBNAME	Имя базы данных PostgreSQL, к которой подключается приложение.
ND_ONEGA_SIGMA_POSTGRES_PORT	Порт PostgreSQL на уровне запуска Compose. Может использоваться для проброса/параметризации сервиса БД.
ND_ONEGA_SIGMA_S3_LOGIN	Логин/Access Key для доступа к S3-совместимому объектному хранилищу.
ND_ONEGA_SIGMA_S3_PASSWORD	Пароль/Secret Key для S3. Чувствительное значение: не следует хранить в открытом виде.
ND_ONEGA_SIGMA_S3_BUCKET	Имя S3 bucket. В примере также дублируется в docker.env; желательно держать единый источник истины.
ND_ONEGA_SIGMA_REDIS_USER	Пользователь Redis для подключения приложения/сервисов.
ND_ONEGA_SIGMA_REDIS_PASSWORD	Пароль пользователя Redis. Чувствительное значение: хранить защищённо.
ND_ONEGA_SIGMA_REDIS_PORT	Порт Redis на уровне запуска Compose. В примере также указан в docker.env.

ND_ONEGA_SIGMA_POSTGRES_HOST	Имя Docker-сервиса PostgreSQL внутри compose-сети. Используется контейнерами приложения для DNS-разрешения.
ND_ONEGA_SIGMA_POSTGRES_PORT	Внутренний порт PostgreSQL, по которому приложение подключается к сервису postgres.
ND_ONEGA_SIGMA_POSTGRES_SSL_CERTIFICATE_FILEPATH	Путь внутри контейнера к клиентскому TLS-сертификату PostgreSQL.
ND_ONEGA_SIGMA_POSTGRES_SSL_KEY_FILEPATH	Путь внутри контейнера к приватному ключу клиентского TLS-сертификата PostgreSQL. Чувствительный файл.
ND_ONEGA_SIGMA_POSTGRES_SSL_CA_CERTIFICATES_FILEPATH	Путь внутри контейнера к CA-сертификату для проверки TLS-сертификата PostgreSQL.
ND_ONEGA_SIGMA_S3_FOLDER	Префикс/папка внутри S3 bucket для хранения данных приложения.
ND_ONEGA_SIGMA_S3_URL	Внутренний URL S3-сервиса в Docker-сети. Имя s3 должно соответствовать названию сервиса в compose.
ND_ONEGA_SIGMA_S3_BUCKET	Имя S3 bucket, используемое приложением. Дублируется с .env / Compose.
ND_ONEGA_SIGMA_KEYCLOAK_REALM_URL	Внутренний URL realm Keycloak в Docker-сети. Используется backend для аутентификации/валидации токенов.
ND_ONEGA_SIGMA_KEYCLOAK_CLIENT_ID	Client ID приложения в Keycloak.
ND_ONEGA_SIGMA_KEYCLOAK_CLIENT_SECRET	Client Secret приложения в Keycloak. Чувствительное значение: хранить в секретах и регулярно ротировать при компрометации.
ND_ONEGA_SIGMA_REDIS_HOST	Имя Docker-сервиса Redis внутри compose-сети.
ND_ONEGA_SIGMA_REDIS_PORT	Внутренний порт Redis, используемый приложением.
ND_ONEGA_SIGMA_REDIS_DB	Номер логической базы Redis, которую использует приложение.
ND_ONEGA_SIGMA_REDIS_QUEUE	Имя очереди Redis/очереди задач для локального окружения.
ND_ONEGA_SIGMA_REDIS_SSL_CERTIFICATE_FILEPATH	Путь внутри контейнера к клиентскому TLS-сертификату Redis.
ND_ONEGA_SIGMA_REDIS_SSL_KEY_FILEPATH	Путь внутри контейнера к приватному ключу клиентского TLS-сертификата Redis. Чувствительный файл.
ND_ONEGA_SIGMA_REDIS_SSL_CA_CERTIFICATES_FILEPATH	Путь внутри контейнера к CA-сертификату для проверки TLS-сертификата Redis.
ND_ONEGA_SIGMA_CORS_ORIGINS	Разрешённые origins для CORS. Должны соответствовать реальному URL frontend, с которого браузер обращается к API.
ND_ONEGA_SIGMA_COOKIE_SESSION_ID_SAMESITE	SameSite-политика cookie идентификатора сессии. Значение none обычно нужно для cross-site сценариев; при HTTPS обычно требуется Secure.
ND_ONEGA_SIGMA_COOKIE_LOGIN_SAMESITE	SameSite-политика login-cookie. Значение none обычно используется при интеграциях через внешний домен/IdP.
ND_ONEGA_SIGMA_IS_HTTPS	Флаг, указывающий приложению, работает ли внешнее соединение через HTTPS. В production чаще должен быть true при TLS-терминации на proxy.
ND_ONEGA_SIGMA_TEST_MODE	Включение тестового режима приложения. Для production обычно должно быть false, если нет отдельного требования.

4.5. Примечания по безопасности

TLS-сертификаты и ключ CA — файл certs/ca.key намеренно добавлен в репозиторий. Это самоподписанный центр сертификации, используемый только внутри Docker-сети данного стека и не доверенный ни одной реальной системой или браузером. Любой, у кого есть этот файл, сможет выпустить дополнительные сертификаты, которым будет доверять этот стек, что допустимо для локального изолированного развёртывания. Не используйте эти сертификаты или этот CA в любом другом контексте.

Секрет клиента Keycloak — секрет в keycloak/smartupstream-realm.json и docker.env относится только к этому изолированному развёртыванию и не соответствует ни одному production-экземпляру Keycloak.

Пароли — все пароли сервисов (qwertyuiop) являются намеренно слабыми значениями по умолчанию для локального использования. Не открывайте порты сервисов в сеть, которую вы не контролируете.

4.6. Примечания по деплою, доступному по сети

Инструкции по переводу изолированного локального стека в деплой, доступный по сети.

DNS-имена:

Хост	Назначение
`sigma-front.demo-onega.loc`	Фронтенд
`sigma-api.demo-onega.loc`	Backend API
`sigma-api.demo-onega.loc:8080`	Keycloak (на том же хосте, другой порт)

Оба DNS-имени должны указывать на IP-адрес машины, на которой запущен Docker.

1) «.env» — API_URL для браузера

`API_URL` записывается в `window.env` внутри фронтенд-контейнера и читается **браузером пользователя**, а не Docker-сетью. Должен быть публичным адресом.

```
-API_URL=http://localhost:8000
+API_URL=http://sigma-api.demo-onega.loc:8000
```

2) «docker.env» — CORS, Keycloak realm URL, cookies

- CORS origins

```
-ND_ONEGA_SIGMA_CORS_ORIGINS=http://localhost:3000
+ND_ONEGA_SIGMA_CORS_ORIGINS=http://sigma-front.demo-onega.loc:3000
```

Если нужно временно оставить доступ и с localhost (например, для отладки), перечислить через запятую:

```
ND_ONEGA_SIGMA_CORS_ORIGINS=http://sigma-front.demo-onega.loc:3000,http://localhost:3000
```

- Keycloak realm URL для backend

Backend проверяет JWT-токены, запрашивая OIDC discovery-документ с этого адреса и сравнивая поле `iss` в токене. Значение `iss` в токене генерирует Keycloak — оно должно совпадать с realm URL.

Поскольку мы настраиваем Keycloak выдавать токены с публичным хостом (шаг 3 ниже), realm URL тоже должен указывать на публичный адрес:

```
-ND_ONEGA_SIGMA_KEYCLOAK_REALM_URL=http://keycloak:8080/realms/smartupstream
+ND_ONEGA_SIGMA_KEYCLOAK_REALM_URL=http://sigma-api.demo-onega.loc:8080/realms/smartupstream
```

Контейнеры `web` и `worker` должны уметь резолвить `sigma-api.demo-onega.loc` — для этого добавляем `extra_hosts` в `docker-compose.yml` (шаг 4).

- Cookies SameSite / HTTPS

Фронтенд (`sigma-front`) и API (`sigma-api`) — разные хосты, поэтому cookies остаются cross-site. Оставить как есть:

```
ND_ONEGA_SIGMA_COOKIE_SESSION_ID_SAMESITE=none
ND_ONEGA_SIGMA_COOKIE_LOGIN_SAMESITE=none
ND_ONEGA_SIGMA_IS_HTTPS=false
```

Важно: браузеры требуют `Secure` для cookies с `SameSite=None`. Если браузер отклоняет cookies, выставить `ND_ONEGA_SIGMA_IS_HTTPS=true` и поднять HTTPS (самоподписанный сертификат или Caddy/nginx с TLS termination).

3) «docker-compose.yml» — Keycloak hostname и extra_hosts

- Переменная окружения для Keycloak

Keycloak в режиме `start-dev` динамически берёт хост из входящего запроса и подставляет его в поле `iss` токена. Чтобы `iss` был всегда одинаковым (публичный URL), зафиксировать его явно:

```
keycloak:
  environment:
    KEYCLOAK_ADMIN: admin
    KEYCLOAK_ADMIN_PASSWORD: admin
    KC_HOSTNAME_URL: http://sigma-api.demo-onega.loc:8080   # добавить
    KC_HOSTNAME_ADMIN_URL: http://sigma-api.demo-onega.loc:8080  # добавить
```

- extra_hosts для web и worker

Контейнеры `web` и `worker` обращаются к Keycloak по `ND_ONEGA_SIGMA_KEYCLOAK_REALM_URL`, который теперь содержит публичный хост. Docker не резолвит внешние DNS внутри контейнера автоматически — добавить `extra_hosts`:

```
web:
  extra_hosts:
    - "sigma-api.demo-onega.loc:host-gateway"

worker:
  extra_hosts:
    - "sigma-api.demo-onega.loc:host-gateway"
```

`host-gateway` — специальное значение Docker Compose, которое резолвится в IP-адрес хост-машины. Работает на Docker 20.10+.

4) «keycloak/smartupstream-realm.json» — client URLs и redirects

Клиент `smartupstream-app` сейчас содержит только `localhost`-адреса. Нужно добавить публичные адреса.

Найти секцию "clientId": "smartupstream-app" и изменить следующие поля:

- rootUrl и adminUrl

```
-"rootUrl": "http://localhost:8000/",
-"adminUrl": "http://localhost:8000/",
+"rootUrl": "http://sigma-api.demo-onega.loc:8000/",
+"adminUrl": "http://sigma-api.demo-onega.loc:8000/",
```

- redirectUris — добавить новые, оставить старые (для совместимости)

```
 "redirectUris": [
   "http://localhost:8080/*",
   "http://localhost:8000/*",
```

```
   "http://localhost:3000/*",
+  "http://sigma-front.demo-onega.loc:3000/*",
+  "http://sigma-api.demo-onega.loc:8080/*",
+  "http://sigma-api.demo-onega.loc:8000/*"
 ],
```

- webOrigins — разрешённые CORS-источники для Keycloak

```
 "webOrigins": [
   "http://localhost:8000",
   "http://localhost:3000",
+  "http://sigma-api.demo-onega.loc:8000",
+  "http://sigma-front.demo-onega.loc:3000"
 ],
```

- post.logout.redirect.uris

```
-"post.logout.redirect.uris":
"http://localhost:3000/*##http://localhost:8080/*##http://localhost:8000/*",
+"post.logout.redirect.uris":
"http://localhost:3000/*##http://localhost:8080/*##http://localhost:8000/*##http://sigma-front.demo-onega.loc:3000/*##http://sigma-api.demo-onega.loc:8080/*",
```

- sslRequired — критически важно для HTTP-доступа

Текущее значение `"external"` означает, что Keycloak требует HTTPS для всех не-localhost соединений. При доступе по HTTP (`http://sigma-api.demo-onega.loc:8080`) логин будет заблокирован редиректом на HTTPS.

```
-"sslRequired": "external",
+"sslRequired": "none",
```

5) Применение изменений realm (если стек уже запускался)

Realm-конфиг импортируется Keycloak только при первом старте (пустой volume). Если `keycloak_data` volume уже существует — изменения в JSON не применятся автоматически.

Вариант A — пересоздать volume (теряются пользователи):

```
docker compose down
docker volume rm isolated_deploy_keycloak_data
docker compose up -d keycloak
# подождать health, затем поднять остальное
docker compose up --wait --wait-timeout 180 --detach
```

Вариант B — применить через Keycloak Admin UI (пользователи сохраняются):

1. Открыть `http://sigma-api.demo-onega.loc:8080` → войти как `admin` / `admin`
2. Выбрать realm **smartupstream** (dropdown слева вверху)
3. **Clients** → `smartupstream-app` → вкладка **Settings**:
   - Root URL: `http://sigma-api.demo-onega.loc:8000/`
   - Home URL: оставить пустым или `http://sigma-front.demo-onega.loc:3000/`
   - Valid redirect URIs: добавить `http://sigma-front.demo-onega.loc:3000/*` и `http://sigma-api.demo-onega.loc:8000/*`
   - Valid post logout redirect URIs: добавить `http://sigma-front.demo-onega.loc:3000/*`
   - Web origins: добавить `http://sigma-front.demo-onega.loc:3000` и `http://sigma-api.demo-onega.loc:8000`
4. **Realm settings** → вкладка **General** → **Require SSL**: выставить None

6) Проверка после запуска

```
# Все сервисы в статусе healthy / running
docker compose ps

# Backend отвечает с нового адреса
curl http://sigma-api.demo-onega.loc:8000/health/live

# Keycloak OIDC discovery — проверить поле "issuer"
curl http://sigma-api.demo-onega.loc:8080/realms/smartupstream/.well-known/openid-configuration \
  | python3 -m json.tool | grep issuer
# Ожидаемый результат:
# "issuer": "http://sigma-api.demo-onega.loc:8080/realms/smartupstream"

# Фронтенд — проверить что env.js содержит правильный API_URL
curl http://sigma-front.demo-onega.loc:3000/env.js
# Ожидаемый результат:
# window.env = { API_URL: 'http://sigma-api.demo-onega.loc:8000', BASENAME: '/' }
```

7) Итоговый список изменений

Файл	Что изменить
`.env`	`API_URL` → публичный адрес API
`docker.env`	`CORS_ORIGINS`, `KEYCLOAK_REALM_URL`
`docker-compose.yml`	`KC_HOSTNAME_URL`, `KC_HOSTNAME_ADMIN_URL` в `keycloak`; `extra_hosts` в `web` и `worker`
`keycloak/smartupstream-realm.json`	`sslRequired: none`; добавить новые URL в `redirectUris`, `webOrigins`, `post.logout.redirect.uris`; обновить `rootUrl`/`adminUrl`

8) Опциональный nginx reverse proxy (clean URLs без порта)

Если нужны URL без порта (`http://sigma-front.demo-onega.loc` вместо `:3000`), добавить в `docker-compose.yml` nginx-прокси, который слушает порт 80 на хосте:

```
  proxy:
    image: nginx:alpine
    ports:
      - "80:80"
    volumes:
      - ./proxy.nginx.conf:/etc/nginx/conf.d/default.conf:ro
    depends_on:
      - frontend
      - web
    networks:
      - onega_sigma_network
```

proxy.nginx.conf:

```
server {
    listen 80;
    server_name sigma-front.demo-onega.loc;
    location / {
        proxy_pass http://frontend:80;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }
}

server {
    listen 80;
    server_name sigma-api.demo-onega.loc;
    location / {
        proxy_pass http://web:8000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_read_timeout 120s;
    }
}
```

Keycloak оставить на порту `:8080` напрямую — проброс через path-prefix (`/auth`) требует дополнительной конфигурации KC_HTTP_RELATIVE_PATH и не рекомендован для этого случая.

При использовании proxy обновить URL без портов:

- `.env`: `API_URL=http://sigma-api.demo-onega.loc`
- `docker.env`: `ND_ONEGA_SIGMA_CORS_ORIGINS=http://sigma-front.demo-onega.loc`
- `docker.env`: `ND_ONEGA_SIGMA_KEYCLOAK_REALM_URL=http://sigma-api.demo-onega.loc:8080/realms/smartupstream`
- Realm JSON: `redirectUris` и `webOrigins` — варианты без порта