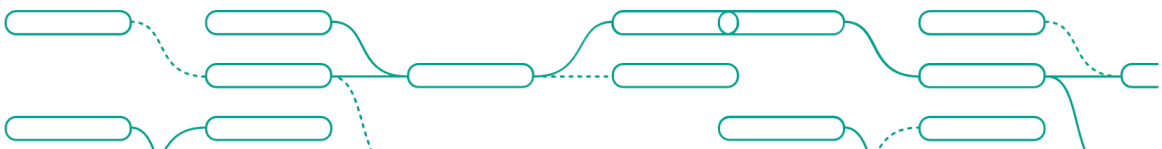
Arenadata Catalog

Arenadata Catalog

Инструкция по установке модуля ADC.DQF

Москва 2026



Arenadata Catalog

Оглавление

1 Введение 3

1.1 Термины и определения 3

1.2 Сокращения и обозначения 4

1.3 Общие положения 4

2 Назначение ПО 6

3 Требования к установке 7

4 Подготовка к установке 8

5 Установка ПО 9

5.1 Изменение общих настроек системы 11

5.2 Типовая конфигурация DQF под внешние БД сервисов 11

5.2.1 Настройка SSL/TLS для PostgreSQL 11

5.3 Типовая конфигурация DQF под внешнюю Kafka 12

5.3.1. Настройка SSL/TLS для Kafka 14

5.4 Настройка внешнего Keycloak 15

5.5 Настройка интеграции DQF с настроенным ADC 16

6 Проверка доступности сервиса 21

7 Обновление ПО 22

8 Логирование сервисов модуля ADC.DQF 23

9 Мониторинг модуля ADC.DQF 24

9.1 Метрики 24

10 Компоненты дистрибутива 25

11 Контакты технических специалистов 26

Arenadata Catalog

1 Введение

1.1 Термины и определения

Термин	Значение
Application Programming Interface (API)	Программный интерфейс приложения, интерфейс прикладного программирования – описание способов (набор классов, процедур, функций, структур или констант), которыми одна компьютерная программа может взаимодействовать с другой программой.
Data Quality Framework (DQF)	Программное обеспечение, позволяющее производить проверки данных с использованием настраиваемых правил, сконфигурированных в соответствии с особенностями предметной области и бизнес-логики заказчика.
Kafka	Apache Kafka — распределённый программный брокер сообщений с открытым исходным кодом
Keycloak	Инструмент с открытым исходным кодом, используемый для аутентификации пользователей в организации, с возможностью настройки single sign-on и управления доступом
SQL	Декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей СУБД
База данных	Совокупность данных, хранимых в соответствии со схемой, манипулирование которыми выполняют в соответствии с правилами средств моделирования данных
Задача	Процесс создания проверок по группе правил для выявления несоответствий или подтверждения качества исходных данных .
Правило (Rule)	Структурированная последовательность шагов (базовых алгоритмов, механизмов управления потоком и других правил), где каждый шаг возвращает статус выполнения и оперирует данными через контекст исполнения для решения конкретной бизнес-задачи.
Проверка	Процесс исполнения заданного правила над исходными данными для выявления несоответствий или подтверждения их качества.
Сервис	Независимый модуль, реализующий конкретный набор услуг/функций в рамках модуля ADC.DQF
Сервис каталога правил (СКП)	Сервис, позволяющий пользователю управлять правилами и группами правил.
Сервис исполнения правил (СИП)	Сервис для реализации выполнения задач по автоматической верификации данных с использованием заданного правила.
Сервис планирования задач (СПЗ)	Сервис создания задач для проверки набора данных и формирующий сообщения-запросы для Сервиса исполнения правил.
Сервис расписания задач (СРЗ)	Сервис для планирования и создания задач для проверки данных по определенному расписанию.

1.2 Сокращения и обозначения

Сокращение	Наименование
ПО	Программное обеспечение
СУБД	Система управления базами данных
БД	База данных
AD.C	Arenadata Catalog
ADC.DQF	Модуль Data Quality Framework (DQF) Arenadata Catalog
УЗ	Учетная запись
ТП	Техническая поддержка

1.3 Общие положения

Инструкция предназначена для должностных лиц, осуществляющих установку программного обеспечения модуля Arenadata Catalog Data Quality Framework (ADC.DQF). Документ содержит пошаговое описание действий для установки специального программного обеспечения системы.

Модуль ADC.DQF обеспечивает возможность развертывания с использованием контейнерной архитектуры. Предоставляемый клиентам пакет дистрибутива модуля ADC.DQF включает набор компонентов необходимый для полноценного развертывания решения. Состав пакета приведён в таблице 1.

Таблица 1. Компоненты модуля ADC.DQF

Функциональный блок	Компоненты продукта
Сервис исполнения правил (СИП)	DQF Engine Standalone
Сервис проверки правил (СПП)	DQF Engine Standalone (профиль probe)
Сервис расписаний задач (СРЗ)	DQF Scheduler
Сервис планирования задач (СПЗ)	DQF Planner
Сервис каталога правил (СКП)	DQF CAS
Сервис результатов	DQF Aggregator
Сервис дашбордов	DQF Dashboards Storage
Сервис отчетов	DQF Reporter
Веб интерфейс	DQF Web App
Сервис авторизации	DQF Auth
Адаптер результатов для модуля сохранения	DQF Saver Adapter

Модули экосистемы Arenadata Catalog, которые дополняют ADC.DQF и обеспечивают дополнительную функциональность, описаны в таблице 2.

Таблица 2. Компоненты экосистемы ADC

Функциональный блок	Компоненты продукта
Модуль сохранения подробных результатов*	Result Saver
*обеспечивает функцию сохранения результатов в пользовательское s3 совместимое хранилище.	

Arenadata Catalog

Инфраструктурные компоненты, необходимые для эксплуатации ADC.DQF, описаны в таблице 3.

Таблица 3. Дополнительные компоненты для среды модуля ADC.DQF

Функциональный блок	Компонент среды функционирования
Перенаправление логов	Vector - 0.28.0
Служебная база данных	Postgres 14 или любая совместимая с ней СУБД
Брокер сообщений	Kafka (предпочтительнее), RabbitMQ
Средство аутентификации	KeyCloak - 23.0
Перенаправление логов	Vector - 0.28.0
Хранение и поиск логов	Loki - 2.6.0, Grafana - 9.5.5
Сбор метрик	Prometheus - 2.41.0, Prometheus Graphite Exporter - 0.13.1
Единая точка входа	Nginx
Служба координации**	Zookeeper - 3.9
** используется для распространения и проверки лицензий	

2 Назначение ПО

Модуль Data Quality Framework ADC (далее – ADC.DQF) – высокопроизводительный сервис выполнения проверок качества данных по настраиваемым правилам с консолидацией результатов и формированием инцидентов качества.

Функциональные возможности ADC.DQF позволяют с помощью специальных правил и алгоритмов производить верификацию проверяемого объекта и его атрибутов и выявлять ошибки, а также настраивать проверки данных в соответствии с предметной областью и заданной конфигурацией (бизнес-логики) исполняемой проверки без доработки программного обеспечения.

ADC.DQF позволяет обеспечить:

- соблюдение критериев качества данных;
- исполнение требований регуляторов;
- поддержку процессов цифровизации с применением методик Data Quality на новых платформах;
- соблюдение критериев качества данных при миграции и консолидации данных, предобработке данных;
- проактивное реагирование на инциденты качества и аномалии данных;
- централизацию проверок, реализацию методологий Data Quality и Data Governance в рамках одного интерфейса.

3 Требования к установке

Для установки ADC.DQF по данной инструкции необходимо выполнение следующих требований:

1. Операционная система– Ubuntu 22.04.4 LTS, Ред ОС 7.3, ALT Server 10.1 (Mendelevium), Astra Linux Воронеж 1.7.
2. ПО Docker version 25.0.0, и выше
3. ПО Docker compose version 1.29.2, и выше
4. Пользователь с правами sudo.(или пользователь добавленный в группу docker)

Перед началом установки следует проверить версию установленного Docker Compose на соответствие требованиям.

> Важно!
> В неподходящей версии Docker compose сборка не произойдет, так как используется версия контейнера 3.9. В случае отсутствия необходимо произвести установку согласно инструкции по ссылке https://docs.docker.com/engine/install/ubuntu/

4 Подготовка к установке

Локальная среда настраивается в формате docker compose (docker-compose.yaml)

Для установки модуля ADC.DQF необходимо подключиться к репозиториям repo.arenadc.io, либо запросить архив с образами у службы поддержки Arenadata Catalog, направив соответствующий запрос на адрес info@arenadc.io.

Пример команды shell:

```
docker login repo.arenadc.io
```

Далее необходимо загрузить файл Docker compose и environment файлы по ссылке предоставленной технической поддержкой.

Общий вид подготовленного каталога стенда должен иметь вид:

```
.
├── docker
│   ├── abstract
│   ├── cookbook
│   ├── modules
│   └── profile-stand
```

Запросить у службы поддержки лицензию **dqf.license**. Указанный файл требуется положить (`.docker/modules/zk/license/es.adc.license`).

Полученный архив с образами должен быть расположен в каталоге: `docker/profile-stand/image/`

5 Установка ПО

Для установки ПО модуля ADC.DQF выполните скрипт инициализации стенда.

Перейдите в каталог `cd ./docker/profile-stand` и запустите скрипт `./init_stand.sh`.

Скрипт запросит название стенда и IP-адрес стенда, например:

Название стенда: `demo`

IP-адрес стенда: `10.80.0.70`

После этого скрипт будет работать с каталогом `stand-<stand_name>`. Например, для стенда `demo` это каталог `stand-demo`. Если каталог `stand-<stand_name>` уже существует, скрипт предложит перезаписать конфигурационные файлы стенда. Если пользователь подтвердит перезапись, скрипт заново создаст шаблонные конфигурационные файлы в этом каталоге. Ранее внесённые изменения в этих файлах могут быть потеряны. Если пользователь откажется от перезаписи, существующие конфигурационные файлы стенда останутся без изменений. При этом остальные действия скрипта инициализации будут выполнены.

Скрипт инициализации выполняет следующие шаги:

- Генерирует ключ и сертификат стенда по пути `./certs/stand`.
- Скачивает необходимые драйвера для работы с БД.
- Подготавливает необходимые директории для маунтов (БД, логи).
- Создаёт файлы:
  - Настроенные переменные docker compose: `stand-<stand_name>/.env`.
    Например: `stand-demo/.env`
  - docker-compose-файл со специфичными для стенда сервисами: `stand-<stand_name>/docker-compose-stand-specific.yaml`
    он позволяет как добавлять новые сервисы, так и перекрывать уже существующие (добавлением профиля, можно их исключать из запуска).
    Например: `stand-demo/docker-compose-stand-specific.yaml`
  - Пустой файл конфигурации, куда можно добавить специфичные для стенда настройки:
    `stand-<stand_name>/spring-common/application.yaml`
    Например: `stand-demo/spring-common/config/application.yaml`
  - Markdown-файл с валидными ссылками на сервисы: `stand-<stand_name>/services.md`
    Например: `stand-demo/services-demo.md`.
  - Локальный env-файл для запуска docker compose для конкретного стенда: `.env`
    Используется в точке запуска: `redeploy-stand.sh`

Для всех сервисов модуля ADC.DQF в соответствии с docker compose определены специфичные конфигурационные файлы в соответствующих каталогах сервисов и общие для всех сервисов в каталоге `stand-<stand_name>/spring-common` (`stand-demo/spring-common`):

- application.yaml;
- application-dqf-kafka.yaml;
- application-keycloak.yaml.

В состав поставки ADC.DQF входит свой инстанс Keycloak. В случае необходимости подключения модуля ADC.DQF к уже используемому Keycloak или Avanpost требуется добавить настройки подключения к соответствующему Keycloak в сформированный файл: **docker/profile-stand/stand-<stand_name>/spring-common/config/application.yaml**.

Например:

```
oauth:
  keycloak:
    enabled: true
    username: admin
    password: p@$$w0rd
    client-id: ${KEYCLOAK_BACKEND_CLIENT}
    client-secret: ${KEYCLOAK_BACKEND_CLIENT_SECRET}
    url: ${KEYCLOAK_BASE_URL}/realms/${KEYCLOAK_REALM}/protocol/openid-
connect/token
```

Внимание!
Файл **application-keycloak.yaml** изменять не нужно!
(docker/profile-stand/stand-demo/spring-common/config/application-keycloak.yaml)
При подключении к уже используемому Keycloak требуется создать/загрузить в realm данные по пользователям и атрибутам доступа для аутентификации и авторизации в модуле ADC.DQF используя файл Dqf.json (docker/modules/keycloak/Dqf.json)

После отработки скрипта инициализации формируется новая точка запуска `redeploy-stand.sh -<флаг>`. Справочная информация по флагам:

1. **-p** — перед запуском загрузить последние версии Docker-образов
2. **-s** — перед запуском остановить контейнеры
3. **-r** — перезапустить контейнеры без выполнения процедуры запуска (игнорирует флаги -p и -s)
4. **-f <compose_file>** — использовать указанный Compose-файл вместо `docker-compose-main.yaml`

При старте среды в Сервис каталога правил (DQF CAS) загружаются демо-правила (docker/modules/cas-updater/algorithms/demo).

Установка сервисов модуля ADC.DQF выполняется в пределах нескольких минут.

Web-интерфейс (DQF Web App) запускает фронтенд графического конструктора проверок, а также Nginx, выполняющий роль обратного прокси (reverse proxy). Это позволяет предоставлять пользователям единый веб-интерфейс, в котором фронтенд-приложение и бэкенд-сервисы работают и доступны через один домен.

Если после установки UI модуля ADC.DQF недоступен, то, возможно, не успел запуститься nginx или Keycloak. В этом случае необходимо дождаться запуска контейнеров.

Сервисы ADC.DQF используют внутреннюю БД PostgreSQL (версия 14.x). Разворачивание БД сконфигурировано в Docker compose файле.

Если нужно просто обновить контейнеры на новые образы, достаточно запустить `redeploy-stand.sh -ps`

В случае необходимости изменить конфигурации сервисов ADC.DQF необходимо остановить контейнер и снова запустить Docker compose файл с измененной конфигурацией командой `redeploy-stand.sh -ps`

5.1 Изменение общих настроек системы

Изменение общих настроек приложения выполняется стандартным для **Spring Boot** способом — через конфигурационные файлы сервисов.

Если для изменяемого сервиса в каталоге стенда отсутствует конфигурация, необходимо:

1. Создать каталог сервиса в директории стенда (например, engine/config/).
2. Добавить в него конфигурационный файл `application.yaml` с требуемыми изменениями.
3. Подключить созданный каталог в контейнер Config Server, добавив соответствующее монтирование в файл `docker/profile-stand/stand-<имя_стенда>/docker-compose-stand-specific.yaml`.

Пример подключения конфигурационных файлов:

```
config-server:
  volumes:
    - ./${STAND_DIR:-.}/engine/config/:/native-files/engine-
standalone/overrides/:ro
```

5.2 Типовая конфигурация DQF под внешние БД сервисов

Развертка DQF с отдельной внешней БД под каждый сервис выполняется через переопределение конфигурации подключения к PostgreSQL в `application.yaml` соответствующего сервиса: для каждого сервиса указываются `spring.datasource.url`, `spring.datasource.username` и `spring.datasource.password`, где в URL задаются адрес, порт и имя внешней БД. Измененные директории сервисов с такими конфигурациями монтируются в **config service** через `docker-compose-stand-specific.yaml` разворачиваемого контура. Примеры готовых файлов конфигурации сервисов находятся в директории **docker/cookbook/externalpostgre-sandbox** — в подпапках `aggregator`, `cas`, `dashboards-storage`, `planner`, `reporter`, `scheduler`.

Пример подключения внешнего PostgreSQL находится в файле
`docker-compose-external-postgres-server.yaml`
А пример стендового compose-файла с монтированием конфигураций — в
`docker-compose-stand-specific.yaml`
Описание конфигурирования БД сервисов приведено в
`docker/cookbook/externalpostgre-sandbox/README.md`

5.2.1 Настройка SSL/TLS для PostgreSQL

Для использования защищенного соединения между приложением и PostgreSQL необходимо выполнить следующие действия.

Подготовка сертификатов

Подготовьте комплект SSL-сертификатов, включающий:

- server.crt — сертификат сервера PostgreSQL;

- server.key — приватный ключ сервера;
- ca.crt — сертификат удостоверяющего центра (CA), которым подписан сертификат сервера.

При использовании режима sslmode=verify-full сертификат PostgreSQL должен быть выпущен на IP-адрес или DNS-имя, используемое приложением для подключения к базе данных.

Настройка приложения

1. Разместите файл ca.crt внутри контейнера приложения или Config Server.
2. Добавьте параметры SSL в строку подключения к PostgreSQL.

```
spring:
  datasource:
    url:
jdbc:postgresql://10.7.0.35:5422/planner?ssl=true&sslmode=verify-
full&sslrootcert=/app/certs/postgres-ca.crt
    username: planner-admin
    password: planner-pass
```

При отсутствии необходимости проверки имени узла допускается использовать режим `sslmode=verify-ca`.

3. Разместите обновленную конфигурацию сервиса и подключите её через `docker-compose-stand-specific.yaml`.

```
config-server:
  volumes:
    - ./${STAND_DIR:-.}/planner/config/:/native-
files/planner/overrides/:ro
```

5.3 Типовая конфигурация DQF под внешнюю Kafka

Для подключения DQF к внешней Kafka необходимо переопределить параметры подключения приложения к Kafka и отключить встроенную Kafka, если она входит в состав стандартного compose-профиля стенда.

Примеры готовых файлов конфигурации сервисов находятся в директории **docker/cookbook/externalkafka-sandbox**

Конфигурация подключения DQF к внешней Kafka без SSL

Настройки подключения к Kafka задаются в файле `spring-common/config/application-dqf-kafka.yaml`

Для подключения без SSL используется PLAINTEXT-порт внешней Kafka:

```
spring:
  kafka:
    bootstrap-servers: <KAFKA_HOST>:<KAFKA_PORT>
```

Где:

- `<KAFKA_HOST>` — IP-адрес или DNS-имя внешней Kafka;
- <`KAFKA_PORT>` — порт внешнего PLAINTEXT listener'a Kafka.

Для подключения без SSL в конфигурации DQF не должны быть указаны параметры:

```
spring:
  kafka:
    security:
      protocol: SSL
    ssl:
      ...
```

Отключение встроенной Kafka в составе DQF

Если в составе стенда DQF по умолчанию поднимается внутренняя Kafka, её нужно отключить через stand-specific override.

Пример фрагмента docker-compose-stand-specific.yaml:

```
services:
  kafka-0:
    profiles:
      - noop

  kafka-ui:
    profiles:
      - noop
```

После этого сервисы DQF будут использовать Kafka, указанную в `spring-common/config/application-dqf-kafka.yaml`.

Создание топиков Kafka для DQF

В случае отсутствия топиков Kafka их необходимо создать.

В стандартной конфигурации DQF перечень топиков по списку:

- `read_q`,
- `write_q`,
- `planner_data_topic`,
- `write_error_q`,
- `result_write_q`,
- `result_error_q`.

для DQF их можно создать скриптом:

```
docker exec -it kafka-0 kafka-topics --create --bootstrap-server localhost:9092
--topic read_q --partitions 20 --replication-factor 1 --if-not-exists
docker exec -it kafka-0 kafka-topics --create --bootstrap-server localhost:9092
--topic write_q --partitions 20 --replication-factor 1 --if-not-exists
docker exec -it kafka-0 kafka-topics --create --bootstrap-server localhost:9092
--topic planner_data_topic --partitions 20 --replication-factor 1 --if-not-
exists
docker exec -it kafka-0 kafka-topics --create --bootstrap-server localhost:9092
--topic write_error_q --partitions 20 --replication-factor 1 --if-not-exists
docker exec -it kafka-0 kafka-topics --create --bootstrap-server localhost:9092
--topic result_write_q --partitions 20 --replication-factor 1 --if-not-exists
docker exec -it kafka-0 kafka-topics --create --bootstrap-server localhost:9092
--topic result_error_q --partitions 20 --replication-factor 1 --if-not-exists
docker exec -it kafka-0 kafka-topics --create --bootstrap-server localhost:9092
--topic signal --partitions 20 --replication-factor 1 --if-not-exists
```

, где имя контейнера с Kafka `kafka-0`.

5.3.1. Настройка SSL/TLS для Kafka

Данный раздел описывает подключение DQF к внешней Kafka по SSL/TLS при условии, что сертификаты уже подготовлены.

Подготовленные файлы сертификатов

Для настройки SSL/TLS должны быть заранее подготовлены серверные сертификаты для Kafka и клиентские сертификаты для DQF.

На стороне DQF используются:

```
client.truststore.p12
client.keystore.p12
```

Для одностороннего SSL/TLS DQF использует только client.truststore.p12.

Для взаимной аутентификации DQF дополнительно использует client.keystore.p12.

5.4 Настройка внешнего Keycloak

Если для аутентификации используется внешний сервер Keycloak, в соответствующем Realm необходимо создать и настроить клиентов, используемых приложением DQF.

Создание клиентов

Создайте клиент dqf-web-app для фронтенда DQF.

При настройке клиента необходимо указать адрес развернутого экземпляра DQF в параметрах **Root URL**, **Home URL**, **Valid Redirect URIs** и **Web Origins**.

Создайте клиент login-app для бекендовых сервисов DQF.

После создания клиента перейдите на вкладку **Credentials** и скопируйте значение **Client Secret**.

Полученное значение необходимо указать в параметре `KEYCLOAK_BACKEND_CLIENT_SECRET` файла .env.

Настройка Client Scopes

Для клиента **dqf-web-app** назначьте Client Scope:

- openid;
- dqf.

В Scope dqf создайте **Mapper** типа **User attribute** со следующими параметрами:

- **User attribute** — `DQF_ACCESS_ATTRIBUTE`
- **JSON type** — `String`

Настройка приложения

После создания клиентов необходимо проверить параметры подключения к внешнему Keycloak в файле .env.

```
KEYCLOAK_URL=http://<keycloak-host>:<port>
KEYCLOAK_REALM=<реалм клиентов DQF>
KEYCLOAK_BACKEND_CLIENT_ID=login-app
KEYCLOAK_BACKEND_CLIENT_SECRET=<Client Secret>
KEYCLOAK_FRONTEND_CLIENT_ID=dqf-web-app
```

После изменения конфигурации необходимо перезапустить сервисы DQF.

Настройка пользователей в Keycloak выполняется в соответствии с руководством администратора.

5.5 Настройка интеграции DQF с настроенным ADC

При разворачивании DQF совместно с ADC требуется выполнить дополнительные настройки интеграции. В типовом сценарии DQF использует:

- Keycloak, развернутый на стороне ADC;
- ADC API для backend-интеграций;
- ADC API из frontend DQF;
- общий механизм авторизации через Keycloak;
- получение атрибутов пользователей/групп из ADC.

Ниже перечислены обязательные изменения, которые нужно внести в конфигурацию стенда DQF.

1. Настроить переменные окружения стенда

В stand-specific **.env** файле DQF необходимо указать адреса DQF, ADC и Keycloak.

Пример:

```
# Хост, на котором расположен DQF
DEPLOYMENT_HOST=<DQF_HOST>

# Хост, на котором расположен ADC
ADC_HOST=<ADC_HOST>

# Keycloak используется со стороны ADC
KEYCLOAK_HOST=${ADC_HOST}:<KEYCLOAK_PORT>
KEYCLOAK_BASE_URL=http://${KEYCLOAK_HOST}
KEYCLOAK_EXTERNAL_URL=http://${KEYCLOAK_HOST}
KEYCLOAK_WEB_APP_CLIENT_ROOT_URL=http://${KEYCLOAK_HOST}

KEYCLOAK_REALM=<KEYCLOAK_REALM>

# Секрет backend-клиента нужно взять из Keycloak
KEYCLOAK_BACKEND_CLIENT_SECRET=<KEYCLOAK_BACKEND_CLIENT_SECRET>

# Базовый URL ADC
ADC_BASE_URL=http://${ADC_HOST}:<ADC_PORT>

# Backend-профили DQF
DQF_SPRING_PROFILES=keycloak,dqf-kafka
```

Важно:

- `<ADC_HOST>` должен указывать на хост, где доступен ADC.
- `<ADC_PORT>` – порт UI Arenadata Catalog,
- `<ADC_BASE_URL>` должен указывать на базовый адрес ADC без /api.
- `<KEYCLOAK_BASE_URL>` и <`KEYCLOAK_EXTERNAL_URL>` должны указывать на Keycloak, используемый ADC.
- `<KEYCLOAK_BACKEND_CLIENT_SECRET>` нельзя оставлять пустым — его нужно взять из настроек backend-клиента в Keycloak.

2. Настроить общий Spring-конфиг DQF

В stand-specific конфигурации **spring-common/config/application.yaml** нужно включить интеграцию DQF с ADC.

Пример:

```
dqf:
  integration:
    adc:
      http-client:
        url: ${ADC_BASE_URL}
      enabled: true
```

Эта настройка включает backend-интеграцию DQF с ADC и передает сервисам DQF адрес ADC API через переменную `ADC_BASE_URL`.

3. Настроить Keycloak-профиль для получения атрибутов из ADC

В конфигурации spring-common/config/application-keycloak.yaml необходимо указать, что атрибуты пользователей/групп должны запрашиваться из ADC.

Пример:

```
spring:
  security:
    oauth2:
      web:
        realm: ${KEYCLOAK_REALM}
        client-id: ${KEYCLOAK_FRONTEND_CLIENT}
        url: ${KEYCLOAK_EXTERNAL_URL}
dqf:
  auth:
    attribute-provider:
      type: ADC
      adc:
        url: ${ADC_BASE_URL}/api
```

Важно: для `dqf.auth.attribute-provider.adc.url` используется URL именно до ADC API, то есть с добавлением /api.

4. Настроить Nginx для проксирования ADC API

Для работы frontend-интеграции DQF с ADC необходимо добавить в конфигурацию Nginx отдельный location для ADC API. Это нужно, чтобы web-приложение DQF обращалось к ADC через тот же origin и не ловило CORS-ошибки.

Пример блока в **nginx/templates/web-app-full.conf.template**:

```
location /adc-api {
    proxy_pass http://<ADC_HOST>:<ADC_PORT>/api;

    proxy_pass_request_body on;
    proxy_set_header X-Real-IP         $remote_addr;
    proxy_set_header Host              $host;
    proxy_set_header X-Forwarded-For   $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto $scheme;
    proxy_set_header X-Forwarded-Port  $server_port;
}
```

Если ADC развернут без SSL, Nginx-конфигурация DQF также должна учитывать этот сценарий. В таком случае для проксирования ADC API используется http://, а не https://.

В случае если nginx был запущен до конфигурирования настроек без SSL, необходимо удалить из конфигурации nginx настройки SSL:

Строки по примеру ниже лишние:

```
return 301 https://$host$request_uri;
}

server {
listen 443 ssl;
server_name $DEPLOYMENT_HOST;

ssl_certificate /etc/nginx/ssl/certificate.crt;
ssl_certificate_key /etc/nginx/ssl/private.key;
ssl_protocols TLSv1.2 TLSv1.3;
ssl_ciphers 'ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384';
ssl_prefer_server_ciphers on;
```

5. Настроить docker-compose-stand-specific.yaml

В stand-specific **docker-compose-stand-specific.yaml** нужно добавить настройки для **web-app**, **nginx, keycloak** и **config-server**.

Пример:

```
services:
  keycloak:
    profiles:
      - noop

  web-app:
    environment:
      - EXPLORER_ADC_BASE_URI=/adc-api

  nginx:
    volumes:
      - ./${STAND_DIR}/nginx/templates/web-app-
full.conf.template:/etc/nginx/sites-enabled/web-app.conf.template

  config-server:
    volumes:
      - ./${STAND_DIR:-.}/spring-common/config/:/native-files/overrides:ro
```

Назначение настроек:

- `EXPLORER_ADC_BASE_URI=/adc-api` — сообщает frontend DQF, что ADC API доступен через Nginx-префикс /adc-api;
- volume для nginx — подключает измененный шаблон Nginx-конфига;
- volume для config-server — подключает stand-specific Spring-конфиги, включая настройки интеграции с ADC.

Если для отдельных сервисов DQF используются свои override-конфиги, их также нужно смонтировать в config-server, например для auth, planner, reporter, scheduler, engine.

6. Проверить настройки backend-клиента Keycloak

Для сервисов DQF, которые обращаются к Keycloak, должен быть настроен backend-клиент. В конфигурации сервисов, использующих Keycloak, должны быть заданы:

```
oauth:
  keycloak:
    enabled: true
    username: <KEYCLOAK_USERNAME>
    password: <KEYCLOAK_PASSWORD>
    client-id: ${KEYCLOAK_BACKEND_CLIENT}
    client-secret: ${KEYCLOAK_BACKEND_CLIENT_SECRET}
    url: ${KEYCLOAK_BASE_URL}/realms/${KEYCLOAK_REALM}/protocol/openid-
connect/token
```

Важно:

- client-secret берется из переменной `KEYCLOAK_BACKEND_CLIENT_SECRET`;
- URL должен указывать на token endpoint нужного realm;
- учетные данные администратора Keycloak должны соответствовать Keycloak, который используется ADC.

7. Чек-лист перед запуском

Перед запуском совместного стенда ADC + DQF необходимо проверить

Таблица 4. Чеклист настройки интеграции ADC+DQF

№	Проверка	Статус
1	В `.env` указан корректный `DEPLOYMENT_HOST`.	
2	В `.env` указан корректный `ADC_HOST`.	
3	`ADC_BASE_URL` указывает на ADC без суффикса `/api`.	
4	`KEYCLOAK_BASE_URL` и `KEYCLOAK_EXTERNAL_URL` указывают на Keycloak ADC.	
5	Указан корректный `KEYCLOAK_REALM`.	
6	Заполнен `KEYCLOAK_BACKEND_CLIENT_SECRET`.	
7	В `spring-common/config/application.yaml` включена интеграция: `dqf.integration.adc.enabled=true`.	
8	В `spring-common/config/application-keycloak.yaml` настроен `dqf.auth.attribute-provider.type=ADC`.	
9	В Nginx добавлен `location /adc-api`.	
10	В `web-app` добавлена переменная `EXPLORER_ADC_BASE_URI=/adc-api`.	
11	В `config-server` смонтированы stand-specific конфигурационные файлы.	
12	После изменения конфигурации перезапущены DQF-сервисы.	

8. Проверка после запуска

После запуска стенда нужно проверить:

1. DQF web-приложение открывается по адресу:
   `http://<DQF_HOST>`

или, если стенд работает через SSL:
   `https://<DQF_HOST>`

2. ADC API доступен через Nginx DQF:
   `http://<DQF_HOST>/adc-api`
3. DQF backend видит ADC по адресу:
   `${ADC_BASE_URL}/api`
4. Авторизация проходит через Keycloak ADC.
5. В DQF доступны сценарии интеграции с ADC:
   - работа с группами;
   - работа с задачами;
   - получение атрибутов из ADC;
   - создание групп через frontend DQF без CORS-ошибок.

6 Проверка доступности сервиса

Информацию о состоянии сервисов можно получить, выполнив команды:

```
docker ps
docker compose ps
docker compose logs <container name>
```

После инициализации сервисов и БД запускаем модуль ADC.DQF в web-браузере по адресу:

```
https://[server_address]
```

В случае технической необходимости остановить сервисы и/или переустановить сервисы используем следующие команды:

```
docker compose -f docker-compose-main.yaml --env-file .env down
docker compose -f docker-compose-main.yaml --env-file .env up -d
```

7 Обновление ПО

Обновление приложения может быть произведено следующим образом:

- Отключить старую версию программы выполнив:

```
docker compose -f docker-compose-main.yaml --env-file .env down
```

- Загрузить новую версию ПО (файлы окружения, образы) по предоставленным ссылкам.
- Выполнить подготовку к установке новой версии, включая настройки конфигурации, согласно раздела 5 настоящего документа.
- Выполнить установку и запустить модуль ADC.DQF, согласно раздела 6 настоящего документа.

8 Логирование сервисов модуля ADC.DQF

Сервисы модуля ADC.DQF выполняет логирование с записью логов с настроенным уровнем детализации в соответствующий файл.

Уровень логирования, может быть, один из следующих:

• TRACE- самый детальный уровень логирования:

• DEBUG – логирование всех видов событий;

• INFO – логирование ошибок, предупреждений и сообщений;

• WARN – логирование ошибок и предупреждений;

• ERROR – логирование только ошибок.

Настройка уровня логирования для всех сервисов модуля ADC.DQF выполняется в файле application.yaml в каталоге [spring-common]:

```
logging:
  level:
    io:
      arenadc:
        dqf:
          service: info
```

Логи сохраняются в директории вида :

- docker/profile-stand/mnt/logs/<service>

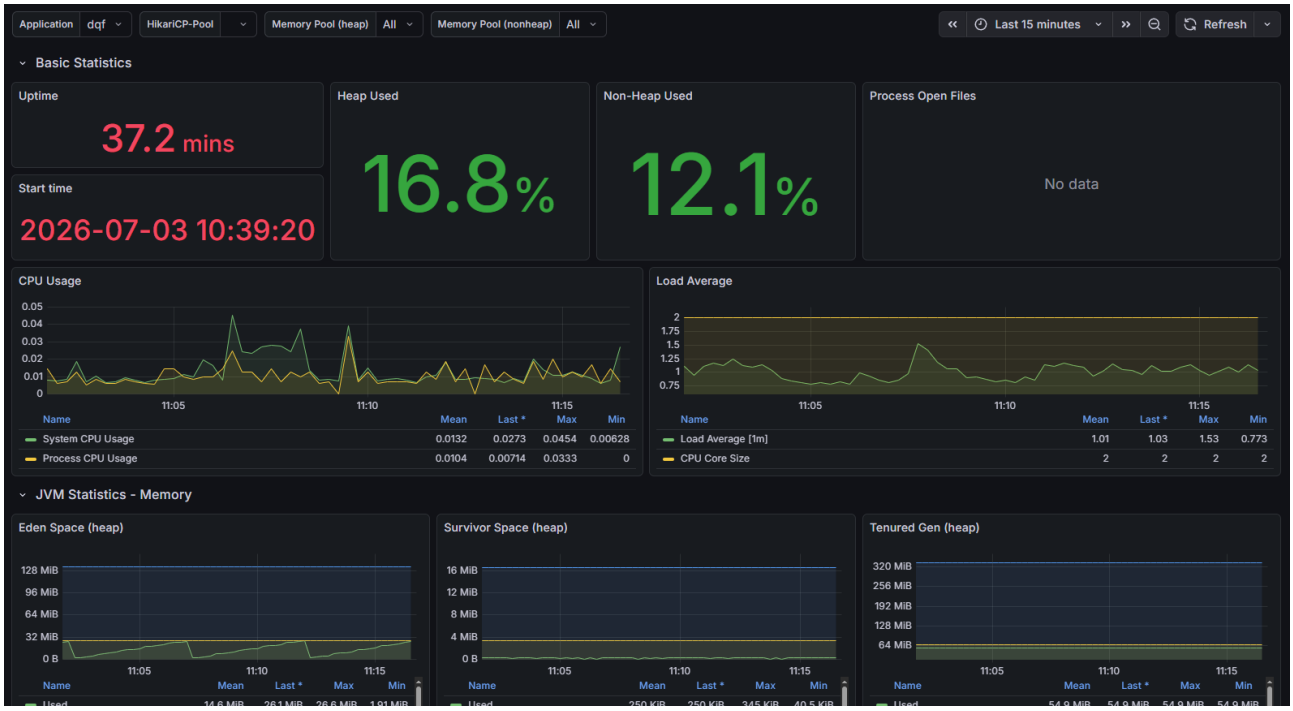
например:

- docker/profile-stand/mnt/logs/aggregator
- docker/profile-stand/mnt/logs/auth
- docker/profile-stand/mnt/logs/cas
- docker/profile-stand/mnt/logs/dashboards-storage
- docker/profile-stand/mnt/logs/engine
- docker/profile-stand/mnt/logs/engine-probe
- docker/profile-stand/mnt/logs/keycloak
- docker/profile-stand/mnt/logs/planner
- docker/profile-stand/mnt/logs/reporter
- docker/profile-stand/mnt/logs/result-saver
- docker/profile-stand/mnt/logs/result-saver-adapter
- docker/profile-stand/mnt/logs/scheduler

9 Мониторинг модуля ADC.DQF

9.1 Метрики

ADC.DQF поставляется с настроенными дашбордами в графане. Доступны стандартные метрики Prometheus Spring- дашборд Spring DQF Statistics.



Статистика отображается отдельно по каждому сервису (application
Доступны метрики:

1. JVM - Micrometer – метрики JVM – heap memory, GC, операции ввода/вывода и тд.
2. HikariCP Statistics
3. HTTP Statistics
4. Tomcat Statistics
5. Logback Statistics

9.2 Prometheus

10 Компоненты дистрибутива

Функциональный блок	Компонент платформы
Сервис исполнения правил (СИП)	DQF Engine Standalone
Сервис проверки правил (СИП)	DQF Engine Standalone (профиль probe)
Сервис расписаний задач (СРЗ)	DQF Scheduler
Сервис планирования задач (СПЗ)	DQF Planner
Сервис каталога правил (СКП)	DQF Complex Algorithms Storage Service
Сервис результатов	DQF Aggregator
Сервис дашбордов	DQF Dashboards Storage
Сервис отчетов	DQF Reporter
Веб-портал	DQF Web App
Сервис авторизации	DQF Auth Service
Адаптер результатов для модуля сохранения	DQF Saver Adapter
Модуль сохранения подробных результатов*	Result Saver
Служебная база данных	Postgres 14 или совместимая
Брокер сообщений	Kafka (предпочтительнее), RabbitMQ
Средство аутентификации	KeyCloak 23.0
Перенаправление логов	Vector - 0.28.0
Хранение и поиск логов	Loki - 2.6.0, Grafana - 9.5.5
Сбор метрик	Prometheus - 2.41.0,
	Prometheus Graphite Exporter - 0.13.1
Прокси	Nginx
Служба координации	Zookeeper - 3.9

11 Контакты технических специалистов

В случае возникновения трудностей при установке программного обеспечения, свяжитесь с технической поддержкой, используя электронный адрес info@arenadc.io.