|  |  |
| --- | --- |
|  |  |

**RT.DataVision**

**Инструкция администратора**

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |

**Оглавление**

[1 Введение 7](#_Toc165042015)

[1.1 Наименование и условное обозначение 7](#_Toc165042016)

[1.2 Область применения 7](#_Toc165042017)

[1.3 Термины, определения и сокращения 7](#_Toc165042018)

[1.4 Требования к квалификации администратора 8](#_Toc165042019)

[2 Подготовка к работе 8](#_Toc165042020)

[2.1 Требования к доступам 8](#_Toc165042021)

[2.2 Порядок проверки работоспособности 8](#_Toc165042022)

[3 Настройка RT.DataVision 9](#_Toc165042023)

[3.1 Запуск на HTTP-сервере 10](#_Toc165042024)

[3.2 Конфигурация балансировщика 10](#_Toc165042025)

[3.3 Пользовательская конфигурация OAuth2 11](#_Toc165042026)

[3.4 Настройки приложения Flask 13](#_Toc165042027)

[3.5 Флаги функций 13](#_Toc165042028)

[3.6 Согласование временных интервалов 14](#_Toc165042029)

[3.7 Дополнительные сетевые настройки 14](#_Toc165042030)

[3.7.1 CORS 14](#_Toc165042031)

[3.7.2 Разделение домена 14](#_Toc165042032)

[3.7.3 Промежуточное ПО 15](#_Toc165042033)

[3.8 Кэширование 15](#_Toc165042034)

[3.8.1 Кэширование графических изображений 16](#_Toc165042035)

[3.9 Регистрация событий 17](#_Toc165042036)

[3.9.1 Ведение статистики 18](#_Toc165042037)

[3.10 Обновление RT.DataVision 18](#_Toc165042038)

[3.11 Асинхронные запросы через Celery 19](#_Toc165042039)

[3.11.1 Celery 19](#_Toc165042040)

[3.11.2 Celery Flower 21](#_Toc165042041)

[3.12 Оповещения и отчеты 21](#_Toc165042042)

[3.12.1 Требования 21](#_Toc165042043)

[3.12.2 Подробная конфигурация 22](#_Toc165042044)

[3.12.3 Пользовательский файл Docker 25](#_Toc165042045)

[3.12.4 Краткое описание действий по включению оповещений и отчетов 26](#_Toc165042046)

[3.12.5 Компоновка Docker 26](#_Toc165042047)

[3.12.6 Резюме 29](#_Toc165042048)

[3.13 Шаблоны SQL 29](#_Toc165042049)

[3.13.1 Шаблоны Jinja 29](#_Toc165042050)

[3.13.2 Доступные макросы 31](#_Toc165042051)

[3.14 Создание пользовательских плагинов Viz 34](#_Toc165042052)

[3.14.1 Установите Yeoman и генератор пакетов RT.DataVision 35](#_Toc165042053)

[3.14.2 Установите RT.DataVision 35](#_Toc165042054)

[3.14.3 Установить Sup RT.DataVision- UI 35](#_Toc165042055)

[3.14.4 Создайте свой «Привет, мир» 35](#_Toc165042056)

[3.14.5 Добавьте свой плагин в RT.DataVision (с ссылкой NPM) 37](#_Toc165042057)

[3.14.6 Отображене плагина 38](#_Toc165042058)

[3.15 Использование плагина 39](#_Toc165042059)

[3.15.1 Читайте комментарии и документы 39](#_Toc165042060)

[3.15.2 Элементы управления 39](#_Toc165042061)

[3.15.3 Создайте идеальный запрос 39](#_Toc165042062)

[3.15.4 Стиль с графическими значками (Emotion) 39](#_Toc165042063)

[3.15.5 Построение 40](#_Toc165042064)

[3.15.6 Тестирование 40](#_Toc165042065)

[3.15.7 Развертывание пользовательской визуализации в рабочей среде 40](#_Toc165042066)

[3.16 Запуск в Kubernetes 40](#_Toc165042067)

[4 Подключение к базам данных 49](#_Toc165042068)

[4.1 Установка драйверов 49](#_Toc165042069)

[4.1.1 Поддерживаемые базы данных и зависимости 49](#_Toc165042070)

[4.2 Добавление новых драйверов базы данных в Docker 49](#_Toc165042071)

[4.2.1 Определите драйвер, который вам нужен 50](#_Toc165042072)

[4.2.2 Установите драйвер 50](#_Toc165042073)

[4.2.3 Подключиться к MySQL 50](#_Toc165042074)

[4.3 Использование пользовательского интерфейса подключения к базе данных 51](#_Toc165042075)

[4.3.1 Как настроить предпочтительные параметры базы данных и изображения 52](#_Toc165042076)

[4.3.2 Установка изображений 52](#_Toc165042077)

[4.3.3 Как сделать доступными новые механизмы баз данных 52](#_Toc165042078)

[4.3.3.1 MySQL 53](#_Toc165042079)

[4.3.3.2 Oracle 54](#_Toc165042080)

[4.3.3.3 Postgres 54](#_Toc165042081)

[4.3.3.4 SQL- сервер 54](#_Toc165042082)

[4.4 Дополнительные настройки БД 55](#_Toc165042083)

[4.4.1 Более глубокая интеграция 55](#_Toc165042084)

[4.4.2 Схемы 55](#_Toc165042085)

[4.4.3 Внешнее хранилище паролей для соединений 55](#_Toc165042086)

[4.4.4 SSL-доступ к базам данных 56](#_Toc165042087)

[5 Безопасность 56](#_Toc165042088)

[5.1 Роли 56](#_Toc165042089)

[5.2 Предоставленные роли 56](#_Toc165042090)

[5.3 Роль Admin 56](#_Toc165042091)

[5.4 Роль Alpha 56](#_Toc165042092)

[5.5 Роль Gamma 57](#_Toc165042093)

[5.6 Роль sql\_lab 57](#_Toc165042094)

[5.7 Public 57](#_Toc165042095)

[5.8 Управление доступом к источникам данных для Gamma -ролей 57](#_Toc165042096)

[5.9 Настройка разрешений 58](#_Toc165042097)

[5.10 Разрешения 58](#_Toc165042098)

[5.11 Ограничение доступа к подмножеству источников 58](#_Toc165042099)

[5.12 Безопасность на уровне строк 59](#_Toc165042100)

[6 Разное 59](#_Toc165042101)

[6.1 Визуализация карты страны 59](#_Toc165042102)

[6.1.1 Включенные карты 60](#_Toc165042103)

[6.1.2 Добавление новой страны 60](#_Toc165042104)

[6.2 Импорт и экспорт источников данных 61](#_Toc165042105)

[6.2.1 Экспорт источников данных в YAML 61](#_Toc165042106)

[6.2.2 Импорт источников данных из YAML 62](#_Toc165042107)

[7 Подключение клиентских приложений к Dremio 63](#_Toc165042108)

[7.1 Клиенты 63](#_Toc165042109)

[7.1.1 RT.DataVision 63](#_Toc165042110)

[7.1.1.1 Поддерживаемые версии​ 64](#_Toc165042111)

[7.1.1.2 Поддерживаемые методы аутентификации​ 64](#_Toc165042112)

[7.1.1.3 Предварительные условия​ 64](#_Toc165042113)

[7.1.1.4 Создание соединения​ 64](#_Toc165042114)

[7.1.1.5 Свойства SSL-соединения 65](#_Toc165042115)

[7.1.1.6 Расширенные свойства​ 67](#_Toc165042116)

[7.2 Драйвера 67](#_Toc165042117)

[7.2.1 Драйвер JDBC для Arrow Flight SQL 67](#_Toc165042118)

[7.2.1.1 Предварительные условия для использования драйвера JDBC для Arrow Flight SQL​ 68](#_Toc165042119)

[7.2.1.2 Поддерживаемые методы аутентификации​ 68](#_Toc165042120)

[7.2.1.3 Загрузка драйвера JDBC для Arrow Flight SQL​ 68](#_Toc165042121)

[7.2.1.4 Интеграция драйвера JDBC для Arrow Flight SQL​ 68](#_Toc165042122)

[7.2.1.5 Название класса​ 68](#_Toc165042123)

[7.2.1.6 Подключение к базам данных​ 68](#_Toc165042124)

[7.2.2 Драйвер ODBC для Arrow Flight SQL 69](#_Toc165042125)

[7.2.2.1 Поддерживаемые операционные системы​ 69](#_Toc165042126)

[7.2.2.2 Поддерживаемые методы аутентификации​ 69](#_Toc165042127)

[7.2.2.3 Загрузка и установка в Windows​ 70](#_Toc165042128)

[7.2.2.4 Загрузка и установка в Linux​ 70](#_Toc165042129)

[7.2.2.5 Загрузка и установка на macOS​ 70](#_Toc165042130)

[8 Интерфейс командной строки администратора в Dremio 71](#_Toc165042131)

[8.1 Синтаксис​ 71](#_Toc165042132)

[8.2 Параметры​ 71](#_Toc165042133)

[8.3 Команды​ 71](#_Toc165042134)

[8.4 Каталог журналов​ 72](#_Toc165042135)

[8.5 Детализация журнала​ 73](#_Toc165042136)

[9 Управление нагрузками заданий 73](#_Toc165042137)

[9.1 Управление очередью 73](#_Toc165042138)

[9.1.1 Классы работы​ 73](#_Toc165042139)

[9.1.2 Управление очередью запросов 74](#_Toc165042140)

[9.1.3 Управление очередью отражений​ 74](#_Toc165042141)

[9.1.4 Управление памятью запросов​ 74](#_Toc165042142)

[9.1.5 Порог запроса​ 74](#_Toc165042143)

[9.2 История работы и сведения о работе 75](#_Toc165042144)

[9.2.1 Новый интерфейс заданий​ 75](#_Toc165042145)

[9.2.1.1 Включение нового пользовательского интерфейса​ 76](#_Toc165042146)

[9.2.1.2 Взаимодействие с заданием 76](#_Toc165042147)

[9.2.1.3 Визуализатор плана запроса​ 77](#_Toc165042148)

[Приложение 1. Подключения к базам данных 78](#_Toc165042149)

# Введение

## Наименование и условное обозначение

Наименование системы: RT.DataVision.

## Область применения

RT.DataVision – это современное корпоративное веб-приложение для бизнес-аналитики. Это быстрая, легкая, интуитивно понятная платформа, позволяющая пользователям с любым набором навыков легко исследовать и визуализировать свои данные, от простых круговых диаграмм до высокодетализированных геопространственных диаграмм.

## Термины, определения и сокращения

В настоящем документе использованы и определены следующие термины и сокращения:

| *Термин/Сокращение* | *Определение* |
| --- | --- |
| Apache | Apache HTTP-сервер – [свободный](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%BE%D0%B1%D0%BE%D0%B4%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) [веб-сервер](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80).  Apache является [кроссплатформенным ПО](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%BE%D1%81%D1%81%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B5%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5), поддерживает различные операционные системы.  Основными достоинствами Apache считаются надёжность и гибкость конфигурации. Он позволяет подключать внешние модули для предоставления данных, использовать [СУБД](https://ru.wikipedia.org/wiki/%D0%A1%D0%A3%D0%91%D0%94) для [аутентификации](https://ru.wikipedia.org/wiki/%D0%90%D1%83%D1%82%D0%B5%D0%BD%D1%82%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D1%8F) пользователей, модифицировать сообщения об ошибках и т. д. Поддерживает [IPv4](https://ru.wikipedia.org/wiki/IPv4). |
| Flask | [Фреймворк](https://ru.wikipedia.org/wiki/%D0%A4%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA) для создания [веб-приложений](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5) на языке программирования [Python](https://ru.wikipedia.org/wiki/Python), использующий набор инструментов Werkzeug, а также шаблонизатор Jinja. Относится к категории так называемых [микрофреймворков](https://ru.wikipedia.org/w/index.php?title=%D0%9C%D0%B8%D0%BA%D1%80%D0%BE%D1%84%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA&action=edit&redlink=1) – минималистичных каркасов веб-приложений, сознательно предоставляющих лишь самые базовые возможности. |
| Gunicorn | [Python](https://ru.bmstu.wiki/Python) [WSGI](https://ru.bmstu.wiki/WSGI_(Web_Server_Gateway_Interface)) HTTP сервер. Gunicorn является PreFork моделью, портированой с [Ruby](https://ru.bmstu.wiki/Ruby_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)) Unicorn проекта. Сервер Gunicorn совместим с рядом веб-фреймворков, просто реализован, не требует много ресурсов сервера и достаточно быстр. |
| Jinja | [Шаблонизатор](https://ru.wikipedia.org/wiki/%D0%A8%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD%D0%B8%D0%B7%D0%B0%D1%82%D0%BE%D1%80) для языка программирования [Python](https://ru.wikipedia.org/wiki/Python_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)). Он подобен шаблонизатору [Django](https://ru.wikipedia.org/wiki/Django), но предоставляет Python-подобные выражения, обеспечивая исполнение шаблонов в [песочнице](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%81%D0%BE%D1%87%D0%BD%D0%B8%D1%86%D0%B0_(%D0%B1%D0%B5%D0%B7%D0%BE%D0%BF%D0%B0%D1%81%D0%BD%D0%BE%D1%81%D1%82%D1%8C)). Это текстовый шаблонизатор, поэтому он может быть использован для создания любого вида разметки, а также исходного кода. |
| Nginx | Программное обеспечение с открытым исходным кодом, которое позволяет создавать веб-сервер. Также его используют как почтовый сервер или обратный прокси-сервер. |
| Selenium | Selenium WebDriver – это инструмент для автоматизации действий веб-браузера. В большинстве случаев используется для тестирования [Web-приложений](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5), но этим не ограничивается. |
| SQL | (англ. structured query language, язык структурированных запросов) –декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей [системой управления базами данных](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F_%D0%B1%D0%B0%D0%B7%D0%B0%D0%BC%D0%B8_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85). |
| ОС | Операционная система. |
| Система | Система «RT.DataVision». |
| СУБД | Система управления базами данных. |

## Требования к квалификации администратора

Администратор Системы должен иметь навыки работы со следующими продуктами:

* Опыт администрирования ОС, на которой устанавливается Система.
* Опыт управление безопасностью, задание групповых политик, создание пользователей, управление удаленным доступом и т.п.
* Знание языка запросов (SQL).
* Умение работать с большим объёмом информации.
* Опыт масштабирования СУБД.
* Формирование отчетов, в том числе для конечного пользователя.
* Навыки технической поддержки пользователей.

# Подготовка к работе

## Требования к доступам

Администратор должен обладать необходимыми и достаточными доступами, в том числе сетевыми, к предустановленным в рамках Системы компонентам.

Для получения паролей к учетным записям для проведения работ на системе администратору необходимо обращаться в подразделение, осуществляющее поддержку соответствующих систем (RT.DataVision).

## Порядок проверки работоспособности

Первичная проверка работоспособности компонентов системы заключается в авторизации в RT.DataVision.

# Настройка RT.DataVision

Чтобы настроить ваше приложение, вам нужно создать файл ***superset\_config.py*** и добавить его в свой ***PYTHONPATH***. Вот некоторые параметры, которые вы можете установить в этом файле:

# Конфигурация

ROW\_LIMIT = 5000

SUPERSET\_WEBSERVER\_PORT = 8088

# Конфигурация FAB

# Ваш секретный ключ

SECRET\_KEY = '\2\1thisismyscretkey\1\2\e\y\y\h'

# Строка подключения SQLAlchemy к БД

SQLALCHEMY\_DATABASE\_URI = 'sqlite:////path/to/superset.db'

# Флаг Flask-WTF для CSRF

WTF\_CSRF\_ENABLED = True

# Конечные точки, свободные от защиты CSRF

WTF\_CSRF\_EXEMPT\_LIST = []

# Токен CSRF со сроком действия 1 год

WTF\_CSRF\_TIME\_LIMIT = 60 \* 60 \* 24 \* 365

# Включение визуализаций Mapbox

MAPBOX\_API\_KEY = ''

Все параметры и значения по умолчанию, определенные в config.py, можно изменить в вашем локальном файле ***superset\_config.py***. Администраторы захотят прочитать файл, чтобы понять, что можно настроить локально, а также значения по умолчанию.

Поскольку ***superset\_config.py*** действует как модуль конфигурации Flask, его можно использовать для изменения настроек самого Flask, а также расширений Flask, таких как ***flask-wtf***, ***flask-caching***, ***flask-migrate*** и ***flask-appbuilder***. Flask App Builder, веб-фреймворк, используемый RT.DataVision, предлагает множество настроек конфигурации. Для получения дополнительной информации по его настройке, обратитесь к [документации Flask App Builder](https://flask-appbuilder.readthedocs.org/en/latest/config.html).

Обязательно измените:

* SQLALCHEMY\_DATABASE\_URI: по умолчанию он хранится в ~/.superset/superset.db.
* SECRET\_KEY: введите длинную случайную строку.

Если вам нужно исключить конечные точки из CSRF (например, если вы используете пользовательскую конечную точку постбэка аутентификации), вы можете добавить конечные точки в ***WTF\_CSRF\_EXEMPT\_LIST***:

WTF\_CSRF\_EXEMPT\_LIST = [‘’]

## Запуск на HTTP-[сервере](https://superset.apache.org/docs/installation/configuring-superset#running-on-a-wsgi-http-server)

Хотя вы можете запустить RT.DataVision на NGINX или Apache, мы рекомендуем использовать Gunicorn в асинхронном режиме. Это обеспечивает впечатляющий параллелизм и довольно прост в установке и настройке. Пожалуйста, обратитесь к документации, предпочитаемой вами технологии, чтобы настроить приложение Flask WSGI так, чтобы оно хорошо работало в вашей среде. Вот асинхронная установка, которая, как известно, хорошо работает в продакшене:

-w 10 \

-k gevent \

--timeout 120 \

-b 0.0.0.0:6666 \

--limit-request-line 0 \

--limit-request-field\_size 0 \

--statsd-host localhost:8125 \

"superset.app:create\_app()"

Обратитесь к [документации Gunicorn](https://docs.gunicorn.org/en/stable/design.html) для получения дополнительной информации. Обратите внимание, что веб-сервер разработки (*superset run* или *flask run*) не предназначен для использования на продуктивных серверах.

Если вы не используете Gunicorn, вы можете отключить использование flask-compress, установив ***COMPRESS\_REGISTER = False*** в своем ***superset\_config.py***.

## Конфигурация [балансировщика](https://superset.apache.org/docs/installation/configuring-superset#configuration-behind-a-load-balancer)

Если вы используете надмножество за балансировщиком нагрузки или обратным прокси-сервером (например, NGINX или ELB на AWS), вам может потребоваться использовать конечную точку проверки работоспособности, чтобы ваш балансировщик нагрузки знал, работает ли ваш экземпляр RT.DataVision. Это предусмотрено, при ***/health*** котором будет возвращен ответ 200, содержащий «ОК», если веб-сервер работает.

Если балансировщик нагрузки вставляет ***X-Forwarded-For/X-Forwarded-Proto*** заголовки, вы должны установить ***ENABLE\_PROXY\_FIX = True*** в файле конфигурации RT.DataVision (***superset\_config.py***) извлечение и использование заголовков.

В случае, если обратный прокси-сервер используется для обеспечения шифрования SSL, X-Forwarded-Proto может потребоваться явное определение параметра. Для веб-сервера Apache это можно установить следующим образом:

RequestHeader set X-Forwarded-Proto "https"

## Пользовательская [конфигурация](https://superset.apache.org/docs/installation/configuring-superset#custom-oauth2-configuration) OAuth2

Помимо поставщиков, поддерживаемых FAB, RT.DataVision легко подключить к другим реализациям сервера авторизации OAuth2, которые поддерживают авторизацию «кода».

Убедитесь, что пакет pip [Authlib](https://authlib.org/) установлен на веб-сервере.

Сначала настройте авторизацию в RT.DataVision ***superset\_config.py***.

AUTH\_TYPE = AUTH\_OAUTH  
OAUTH\_PROVIDERS = [  
 { 'name':'egaSSO',  
 'token\_key':'access\_token', *# Название токена access\_token\_url*  
 'icon':'fa-address-card', *# Иконка поставщика*  
 'remote\_app': {  
 'client\_id':'myClientId', *# Идентификатор клиента*  
 'client\_secret':'MySecret', *# Секрет для клиента*  
 'client\_kwargs':{  
 'scope': 'read' *# Область для авторизации*  
 },  
 'access\_token\_method':'POST', *# Метод HTTP для вызова access\_token\_url*  
 'access\_token\_params':{ *# Дополнительные параметры для вызова access\_token\_url*  
 'client\_id':'myClientId'  
 },  
 'access\_token\_headers':{ *# Дополнительные заголовки access\_token\_url*  
 'Authorization': 'Basic Base64EncodedClientIdAndSecret'  
 },  
 'api\_base\_url':'https://myAuthorizationServer/oauth2AuthorizationServer/',  
 'access\_token\_url':'https://myAuthorizationServer/oauth2AuthorizationServer/token',  
 'authorize\_url':'https://myAuthorizationServer/oauth2AuthorizationServer/authorize'  
 }  
 }  
]  
  
*# Разрешить пользователям самостоятельную регистрацию*  
AUTH\_USER\_REGISTRATION = True  
  
*# Роль по умолчанию для пользователей, регистрирующихся самостоятельно*  
AUTH\_USER\_REGISTRATION\_ROLE = "Public"

Затем создайте ***CustomSsoSecurityManager*** расширение ***SupersetSecurityManager*** и переопределение в ***auth\_user\_info***:

import logging  
from superset.security import SupersetSecurityManager  
  
class CustomSsoSecurityManager(SupersetSecurityManager):  
  
 def oauth\_user\_info(self, provider, response=None):  
 logging.debug("Oauth2 provider: {0}.".format(provider))  
 if provider == 'egaSSO':  
   
 me = self.appbuilder.sm.oauth\_remotes[provider].get('userDetails').data  
 logging.debug("user\_data: {0}".format(me))  
 return { 'name' : me['name'], 'email' : me['email'], 'id' : me['user\_name'], 'username' : me['user\_name'], 'first\_name':'', 'last\_name':''}  
 ...

Этот файл должен находиться в той же директории, ***superset\_config.py*** что и с именем ***custom\_sso\_security\_manager.py***. Наконец, добавьте следующие 2 строки в ***superset\_config.py***:

from custom\_sso\_security\_manager import CustomSsoSecurityManager

CUSTOM\_SECURITY\_MANAGER = CustomSsoSecurityManager

**Примечания:**

* URL-адрес перенаправления будет отображаться https://<superset-webserver>/oauth-authorized/<provider-name> при настройке поставщика авторизации OAuth2, если это необходимо. Например, URL-адрес перенаправления будет https://<superset-webserver>/oauth-authorized/egaSSO для вышеуказанной конфигурации.
* Если сервер авторизации OAuth2 поддерживает OpenID Connect 1.0, вы можете настроить только его URL-адрес документа конфигурации без указания ***api\_base\_url***, ***access\_token\_url*** и ***authorize\_url*** других необходимых параметров, таких как конечная точка информации о пользователе, jwks uri и т. д. Например:

OAUTH\_PROVIDERS = [  
 { 'name':'egaSSO',  
 'token\_key':'access\_token', *# Название токена access\_token\_url*  
 'icon':'fa-address-card', *# Иконка поставщика*  
 'remote\_app': {  
 'client\_id':'myClientId', *# Идентификатор клиента*  
 'client\_secret':'MySecret', *# Секрет клиента*  
 'server\_metadata\_url': 'https://myAuthorizationServer/.well-known/openid-configuration'  
 }  
 }  
]

## Настройки приложения Flask

FLASK\_APP\_MUTATOR – это функция конфигурации, которая может быть предоставлена в вашей среде, она получает объект приложения и может изменять его любым способом. Например, добавьте ***FLASK\_APP\_MUTATOR*** в ***superset\_config.py*** настройки сеанса времени истечения файла cookie – до 24 часов:

from flask import session  
from flask import Flask  
  
def make\_session\_permanent():  
 '''  
 Enable maxAge for the cookie 'session'  
 '''  
 session.permanent = True  
  
*# Установки максимальный срок жизни сессий в 24 часа*  
PERMANENT\_SESSION\_LIFETIME = timedelta(hours=24)  
def FLASK\_APP\_MUTATOR(app: Flask) -> None:  
 app.before\_request\_funcs.setdefault(None, []).append(make\_session\_permanent)

## [Флаги](https://superset.apache.org/docs/installation/configuring-superset#feature-flags) функций

Для поддержки разнообразного набора пользователей в RT.DataVision есть некоторые функции, которые не включены по умолчанию. Например, у одних пользователей более строгие ограничения безопасности, а у других нет. Таким образом, RT.DataVision позволяет пользователям включать или отключать некоторые функции с помощью конфигурации. Для этого вы можете добавить дополнительные функции в RT.DataVision, они станут доступны для пользователей.

Вы можете включить или отключить функции с флагом из ***superset\_config.py***:

FEATURE\_FLAGS = {  
 'CLIENT\_CACHE': False,  
 'ENABLE\_EXPLORE\_JSON\_CSRF\_PROTECTION': False,  
 'PRESTO\_EXPAND\_DATA': False,  
}

## Согласование временных интервалов

Функция направлена на то, чтобы временные интервалы обрабатывались согласованным и прозрачным образом как для Druid, так и для SQLAlchemy.

Ранее SQLAlchemy использовал инклюзивные конечные точки, однако они могут вести себя как монопольные для строковых столбцов (из-за лексикографического упорядочения), если форматирование не было определено и форматирование столбца не соответствует дате-времени ISO 8601).

Чтобы исправить это, вместо того, чтобы определять формат даты/времени для каждого столбца даты и времени, отличного от IS0 8601, вы можете определить сопоставление столбца по умолчанию на уровне базы данных с помощью extra параметра:

{  
 "python\_date\_format\_by\_column\_name": {  
 "ds": "%Y-%m-%d"  
 }  
}

Все новые установки Системы должны включать функцию, подключив ее в superset\_config.py:

SIP\_15\_ENABLED = True

## Дополнительные сетевые [настройки](https://superset.apache.org/docs/installation/networking-settings#additional-networking-settings)

### CORS

Чтобы настроить CORS или совместное использование ресурсов между источниками, необходимо установить следующую зависимость:

pip install apache-superset[cors]

Для настройки CORS можно указать следующие ключи в ***superset\_config.py***:

* ***ENABLE\_CORS:*** должно быть установлено ***True***, чтобы включить CORS.
* ***CORS\_OPTIONS:*** параметры, переданные в Flask-CORS ([документация](https://flask-cors.corydolphin.com/en/latest/api.html#extension)).

### [Разделение](https://superset.apache.org/docs/installation/networking-settings#domain-sharding) домена

Chrome позволяет одновременно открывать до 6 подключений к одному домену. Когда на dashboard более 6 слайдов, запросы на получение большого количества времени ставятся в очередь и ждут следующего доступного сокета. Функция добавляет сегментирование домена в RT.DataVision, и эта функция будет включена только в конфигурации (по умолчанию RT.DataVision не разрешает междоменные запросы).

Добавьте следующий параметр в свой файл **superset\_config.py**:

* SUPERSET\_WEBSERVER\_DOMAINS: список разрешенных имен хостов для функции сегментирования домена.

### [Промежуточное ПО](https://superset.apache.org/docs/installation/networking-settings#middleware)

RT.DataVision позволяет добавлять собственное промежуточное ПО. Чтобы добавить собственное промежуточное ПО, обновите ***ADDITIONAL\_MIDDLEWARE*** ключ в файле ***superset\_config.py***. ***ADDITIONAL\_MIDDLEWARE*** должен быть списком ваших дополнительных классов промежуточного программного обеспечения.

Например, чтобы использовать ***AUTH\_REMOTE\_USER*** из-за прокси-сервера, такого как nginx, вам нужно добавить простой класс промежуточного программного обеспечения, чтобы добавить значение ***HTTP\_X\_PROXY\_REMOTE\_USER*** (или любой другой пользовательский заголовок из прокси) в переменную среды Gunicorn ***REMOTE\_USER***.

## [Кэширование](https://superset.apache.org/docs/installation/cache/#caching)

RT.DataVision использует [Flask-Caching](https://flask-caching.readthedocs.io/) для кэширования. Из соображений безопасности существуют две отдельные конфигурации кэша для собственных метаданных RT.DataVision (***CACHE\_CONFIG***) и данных диаграмм, запрашиваемых из подключенных источников данных (***DATA\_CACHE\_CONFIG***). Однако результаты запросов из SQL Lab хранятся в другом бэкэнде, который называется ***RESULTS\_BACKEND***.

Настроить кэширование так же просто, как ***CACHE\_CONFIG*** и предоставить его ***DATA\_CACHE\_CONFIG*** в ***superset\_config.py*** в соответствии со [спецификациями Flask-Caching](https://flask-caching.readthedocs.io/en/latest/#configuring-flask-caching).

Flask-Caching поддерживает различные механизмы кэширования, включая Redis, Memcached, SimpleCache (в памяти) или локальную файловую систему.

* ***Memcached***: мы рекомендуем использовать клиентскую библиотеку [pylibmc](https://pypi.org/project/pylibmc/), поскольку ***python-memcached*** не поддерживает правильное хранение двоичных данных.
* ***Redis***: мы рекомендуем пакет [Redis](https://pypi.python.org/pypi/redis) Python.

Обе эти библиотеки можно установить с помощью pip.

Для данных диаграммы RT.DataVision поднимается по «timeout search path», от конфигурации среза к источнику данных, базе данных, а затем в конечном итоге возвращается к глобальному значению по умолчанию, определенному в **DATA\_CACHE\_CONFIG.**

DATA\_CACHE\_CONFIG = {

'CACHE\_TYPE': 'redis',

'CACHE\_DEFAULT\_TIMEOUT': 60 \* 60 \* 24, # 1 день по умолчанию (в сек)

'CACHE\_KEY\_PREFIX': 'superset\_results',

'CACHE\_REDIS\_URL': 'redis://localhost:6379/0',

}

Также поддерживаются настраиваемые серверные части кэша. Подробности смотрите [здесь](https://flask-caching.readthedocs.io/en/latest/#custom-cache-backends).

В RT.DataVision есть задача Celery, которая будет периодически обновлять кеш на основе разных стратегий. Чтобы использовать его, добавьте в раздел ***CELERYBEAT\_SCHEDULE*** в ***config.py*** следующее:

CELERYBEAT\_SCHEDULE = {

'cache-warmup-hourly': {

'task': 'cache-warmup',

'schedule': crontab(minute=0, hour='\*'), *# каждый час*

'kwargs': {

'strategy\_name': 'top\_n\_dashboards',

'top\_n': 5,

'since': '7 days ago',

},

},

}

При этом будут кэшироваться все диаграммы в топ-5 самых популярных информационных панелей каждый час. Другие стратегии смотрите в файле ***superset/tasks/cache.py***.

### Кэширование графических изображений

Это необязательная функция, которую можно включить, активировав соответствующий флаг в конфигурации:

FEATURE\_FLAGS = {

"THUMBNAILS": True,

"THUMBNAILS\_SQLA\_LISTENERS": True,

}

Для этой функции вам понадобится система кэширования и celery воркеры. Все изображения хранятся в кеше и обрабатываются воркерами асинхронно.

Пример конфигурации, в которой изображения хранятся на S3, может быть таким:

from flask import Flask

from s3cache.s3cache import S3Cache

...

class CeleryConfig(object):

BROKER\_URL = "redis://localhost:6379/0"

CELERY\_IMPORTS = ("superset.sql\_lab", "superset.tasks", "superset.tasks.thumbnails")

CELERY\_RESULT\_BACKEND = "redis://localhost:6379/0"

CELERYD\_PREFETCH\_MULTIPLIER = 10

CELERY\_ACKS\_LATE = True

CELERY\_CONFIG = CeleryConfig

def init\_thumbnail\_cache(app: Flask) -> S3Cache:

return S3Cache("bucket\_name", 'thumbs\_cache/')

THUMBNAIL\_CACHE\_CONFIG = init\_thumbnail\_cache

THUMBNAIL\_SELENIUM\_USER = "Admin"

Используя приведенный выше пример, ключи кеша для информационных панелей будут ***superset\_thumb\_\_dashboard\_\_{ID}***. Вы можете переопределить базовый URL-адрес для Selenium, используя:

WEBDRIVER\_BASEURL = <https://superset.company.com>

Дополнительную конфигурацию веб-привода Selenium можно установить с помощью ***WEBDRIVER\_CONFIGURATION***. Вы можете реализовать пользовательскую функцию для аутентификации Selenium. Функция по умолчанию использует ***flask-login*** файл cookie сеанса. Вот пример сигнатуры пользовательской функции:

def auth\_driver(driver: WebDriver, user: "User") -> WebDriver:

pass

Затем по конфигурации:

WEBDRIVER\_AUTH\_FUNC = auth\_driver

## [Регистрация](https://superset.apache.org/docs/installation/event-logging/" \l "event-logging" \o "Прямая ссылка на заголовок) событий

RT.DataVision по умолчанию регистрирует события специальных действий в своей внутренней базе данных. Доступ к этим журналам можно получить в пользовательском интерфейсе, выбрав «**Безопасность > Журнал действий»**. Вы можете свободно настраивать эти журналы, реализуя свой собственный класс журнала событий.

Вот пример простого класса JSON-to-stdout:

def log(self, user\_id, action, \*args, \*\*kwargs):

records = kwargs.get('records', list())

dashboard\_id = kwargs.get('dashboard\_id')

slice\_id = kwargs.get('slice\_id')

duration\_ms = kwargs.get('duration\_ms')

referrer = kwargs.get('referrer')

for record in records:

log = dict(

action=action,

json=record,

dashboard\_id=dashboard\_id,

slice\_id=slice\_id,

duration\_ms=duration\_ms,

referrer=referrer,

user\_id=user\_id

)

print(json.dumps(log))

Обновите вашу конфигурацию, чтобы подключить собственную регистрацию событий:

EVENT\_LOGGER = JSONStdOutEventLogger()

### [Ведение](https://superset.apache.org/docs/installation/event-logging/#statsd-logging) статистики

При желании RT.DataVision можно настроить для регистрации событий в StatsD. Большинство обращений к конечным точкам, а также ключевые события, такие как начало и завершение запроса, регистрируются в Лаборатории SQL.

Чтобы настроить ведение журнала StatsD, нужно настроить регистрацию событий в файле ***superset\_config.py***.

from superset.stats\_logger import StatsdStatsLogger

STATS\_LOGGER = StatsdStatsLogger(host='localhost', port=8125, prefix='superset')

Обратите внимание, что вы также можете реализовать свою собственную регистрацию событий, получив ***superset.stats\_logger.BaseStatsLogger***.

## Обновление RT.DataVision

Чтобы обновить RT.DataVision, выполните следующие команды:

pip install apache-superset –upgrade

superset db upgrade

uperset init

Хотя обновление RT.DataVision не должно удалять ваши диаграммы и информационные панели, мы рекомендуем следовать рекомендациям и делать резервную копию базы данных метаданных перед обновлением. Перед обновлением производственной среды мы рекомендуем выполнить обновление в промежуточной среде и, наконец, обновить производственную среду во время непикового использования.

## Асинхронные запросы через [Celery](https://superset.apache.org/docs/installation/async-queries-celery/#async-queries-via-celery)

### Celery

В больших аналитических базах данных обычно выполняются запросы, которые выполняются в течение нескольких минут или часов. Чтобы включить поддержку длительных запросов, которые выполняются за пределами типичного времени ожидания веб-запроса (30–60 секунд), необходимо настроить асинхронный бэкэнд для RT.DataVision, который состоит из:

* одного или несколько рабочих процессов RT.DataVision (которые реализованы как рабочие процессы Celery) и могут быть запущены с помощью ***celery worker*** командой ***run*** ***celery worker --help*** для просмотра связанных параметров;
* брокера celery (очередь сообщений), для которого мы рекомендуем использовать Redis или RabbitMQ;
* серверной части результатов, которая определяет, где работник будет сохранять результаты запроса.

Для настройки Celery требуется определить ***CELERY\_CONFIG*** в вашем файле ***superset\_config.py***. И рабочие процессы, и процессы веб-сервера должны иметь одинаковую конфигурацию.

class CeleryConfig(object):  
 BROKER\_URL = 'redis://localhost:6379/0'  
 CELERY\_IMPORTS = (  
 'superset.sql\_lab',  
 'superset.tasks',  
 )  
 CELERY\_RESULT\_BACKEND = 'redis://localhost:6379/0'  
 CELERYD\_LOG\_LEVEL = 'DEBUG'  
 CELERYD\_PREFETCH\_MULTIPLIER = 10  
 CELERY\_ACKS\_LATE = True  
 CELERY\_ANNOTATIONS = {  
 'sql\_lab.get\_sql\_results': {  
 'rate\_limit': '100/s',  
 },  
 'email\_reports.send': {  
 'rate\_limit': '1/s',  
 'time\_limit': 120,  
 'soft\_time\_limit': 150,  
 'ignore\_result': True,  
 },  
 }  
 CELERYBEAT\_SCHEDULE = {  
 'email\_reports.schedule\_hourly': {  
 'task': 'email\_reports.schedule\_hourly',  
 'schedule': crontab(minute=1, hour='\*'),  
 },  
 }  
  
CELERY\_CONFIG = CeleryConfig

Чтобы запустить рабочий процесс Celery для использования конфигурации, выполните следующую команду:

celery --app=superset.tasks.celery\_app:app worker --pool=prefork -O fair -c 4

Чтобы запустить задание, которое планирует периодические фоновые задания, выполните следующую команду:

celery --app=superset.tasks.celery\_app:app beat

Чтобы настроить серверную часть результатов, вам нужно передать экземпляр производной от cachelib.base.BaseCache в ключ конфигурации RESULTS\_BACKEND в вашем superset\_config.py. Вы можете использовать Memcached, Redis, S3 (<https://pypi.python.org/pypi/s3werkzeugcache>), память или файловую систему (в конфигурации с одним сервером или для тестирования) или написать свой собственный интерфейс кэширования. Ваш ***superset\_config.py*** может выглядеть примерно так:

*# На S3*

from s3cache.s3cache import S3Cache

S3\_CACHE\_BUCKET = 'foobar-superset'

S3\_CACHE\_KEY\_PREFIX = 'sql\_lab\_result'

RESULTS\_BACKEND = S3Cache(S3\_CACHE\_BUCKET, S3\_CACHE\_KEY\_PREFIX)

*# На Redis*

from cachelib.redis import RedisCache

RESULTS\_BACKEND = RedisCache(

host='localhost', port=6379, key\_prefix='superset\_results')

Для повышения производительности [MessagePack](https://github.com/msgpack/msgpack-python) и [PyArrow](https://arrow.apache.org/docs/python/) теперь используются для сериализации результатов. Это можно отключить, установив ***RESULTS\_BACKEND\_USE\_MSGPACK = False*** в вашем ***superset\_config.py***, если возникнут какие-либо проблемы. Необходимо очистить существующее хранилище кеша результатов при обновлении существующей среды.

**Важно:**

* Важно, чтобы все рабочие узлы и веб-серверы в кластере RT.DataVision использовали общую базу данных метаданных. Это означает, что SQLite не будет работать в этом контексте, поскольку он имеет ограниченную поддержку параллелизма и обычно находится в локальной файловой системе.
* Во всей вашей установке должен быть запущен только один экземпляр celery beat. В противном случае фоновые задания могут быть запланированы несколько раз, что приведет к странному поведению, такому как двойная доставка отчетов, более высокая, чем ожидалось, нагрузка/трафик и т.д.
* Лаборатория SQL будет выполнять ваши запросы асинхронно, только если вы включите **асинхронное выполнение запросов** в настройках вашей базы данных (**Данные > Базы данных > Редактировать**).

### Celery Flower

Flower – это веб-инструмент для мониторинга кластера Celery, который вы можете установить из pip:

pip install flower

Вы можете запустить flower, используя:

celery --app=superset.tasks.celery\_app:app flower

## Оповещения и [отчеты](https://superset.apache.org/docs/installation/alerts-reports#alerts-and-reports)

Пользователи могут настраивать автоматические оповещения и отчеты для отправки информационных панелей или диаграмм получателю электронной почты или каналу Slack.

* Оповещения отправляются при достижении условия SQL.
* Отчеты отправляются по расписанию.

Оповещения и отчеты по умолчанию отключены. Чтобы их включить, нужно выполнить некоторые настройки, описанные в подразделах ниже.

### [Требования](https://superset.apache.org/docs/installation/alerts-reports#requirements)

В файле ***superset\_config.py***:

* "***ALERT\_REPORTS***" флаг функции должен быть установлен на ***True***.
* ***CELERYBEAT\_SCHEDULE*** в CeleryConfig должно быть расписание для ***reports.scheduler***.
* В зависимости от того, что вы хотите использовать, должен быть настроено по крайней мере одно из:
  + настройка электронной почты: ***SMTP\_\****;
  + Стек сообщений: ***SLACK\_API\_TOKEN.***

В ***Dockerfile***:

* Вы должны установить браузер, чтобы делать снимки экрана с диаграммами и информационными панелями. В настоящее время поддерживаются только Firefox и Chrome.

Если вы выберете Chrome, вы также должны изменить значение ***WEBDRIVER\_TYPE*** на ***"chrome"*** в файле ***superset\_config.py***.

[**Интеграция**](https://superset.apache.org/docs/installation/alerts-reports#slack-integration) **стека**

Чтобы отправлять оповещения и отчеты в каналы Slack, вам необходимо создать новое приложение Slack в своей рабочей области.

1. Подключитесь к рабочей области Slack, а затем перейдите по [ссылке https://api.slack.com/apps](https://api.slack.com/apps).
2. Создайте новое приложение.
3. Перейдите в раздел «OAuth & Permissions» и укажите следующие области для вашего приложения:
   * ***incoming-webhook,***
   * ***files:write,***
   * ***chat:write.***
4. В верхней части раздела «OAuth & Permissions» нажмите «install to workspace».
5. Выберите канал по умолчанию для вашего приложения и продолжите. (Вы можете размещать сообщения на любом канале, настроив RT.DataVision на этот канал).
6. Теперь приложение должно быть установлено в вашей рабочей области, и должен быть создан «Bot User OAuth Access Token». Скопируйте этот токен в ***SLACK\_API\_TOKEN*** переменную вашего ***superset\_config.py***.
7. Перезапустите службу (или запустите ***superset init***), чтобы получить новую конфигурацию.

Примечание. Когда вы настраиваете оповещение или отчет, список каналов Slack принимает имена каналов без начального символа «#», например, используйте ***alerts*** вместо ***#alerts***.

**Спецификация Kubernetes:**

* У вас должен быть в ***celery beat*** запущен ***pod***. Если вы используете диаграмму, включенную в репозиторий GitHub в разделе [helm/superset](https://github.com/apache/superset/tree/master/helm/superset), вам необходимо указать ***supersetCeleryBeat.enabled = true*** свои значения override.
* Более общие сведения можно найти в специальной документации по [установке Kubernetes](https://superset.apache.org/docs/installation/running-on-kubernetes).

**Спецификация Docker-compose**

Файл ***docker-compose.yaml*** должен содержать:

* брокер сообщений Redis,
* БД PostgreSQL вместо SQLlite,
* один или больше ***celery worker,***
* один ***celery beat.***

### Подробная [конфигурация](https://superset.apache.org/docs/installation/alerts-reports#detailed-config)

В файл ***superset\_config.py*** необходимо добавить представленные ниже настройки. Этот файл загружается при запуске образа, и любые конфигурации в нем переопределяют конфигурации по умолчанию, найденные в файле ***config.py***.

Вы можете найти документацию по каждому полю по умолчанию ***config.py*** в репозитории.

Вам необходимо заменить значения по умолчанию вашей пользовательской конфигурацией Redis, Slack и/или SMTP.

В ***CeleryConfig***, только ***CELERYBEAT\_SCHEDULE*** неизменна, остальные функции ***CeleryConfig*** могут быть изменены для ваших нужд.

from celery.schedules import crontab  
  
FEATURE\_FLAGS = {  
 "ALERT\_REPORTS": True  
}  
  
REDIS\_HOST = "redis-superset"  
REDIS\_PORT = "6379"  
  
class CeleryConfig:  
 BROKER\_URL = 'redis://%s:%s/0' % (REDIS\_HOST, REDIS\_PORT)  
 CELERY\_IMPORTS = ('superset.sql\_lab', "superset.tasks", "superset.tasks.thumbnails", )  
 CELERY\_RESULT\_BACKEND = 'redis://%s:%s/0' % (REDIS\_HOST, REDIS\_PORT)  
 CELERYD\_PREFETCH\_MULTIPLIER = 10  
 CELERY\_ACKS\_LATE = True  
 CELERY\_ANNOTATIONS = {  
 'sql\_lab.get\_sql\_results': {  
 'rate\_limit': '100/s',  
 },  
 'email\_reports.send': {  
 'rate\_limit': '1/s',  
 'time\_limit': 600,  
 'soft\_time\_limit': 600,  
 'ignore\_result': True,  
 },  
 }  
 CELERYBEAT\_SCHEDULE = {  
 'reports.scheduler': {  
 'task': 'reports.scheduler',  
 'schedule': crontab(minute='\*', hour='\*'),  
 },  
 'reports.prune\_log': {  
 'task': 'reports.prune\_log',  
 'schedule': crontab(minute=0, hour=0),  
 },  
 }  
CELERY\_CONFIG = CeleryConfig  
  
SCREENSHOT\_LOCATE\_WAIT = 100  
SCREENSHOT\_LOAD\_WAIT = 600  
  
*# Кинфигурация Slack*   
SLACK\_API\_TOKEN = "xoxb-"  
  
*# Конфигурация электронной почты*  
SMTP\_HOST = "smtp.sendgrid.net" *#измените на свой хост*  
SMTP\_STARTTLS = True  
SMTP\_SSL = False  
SMTP\_USER = "your\_user"  
SMTP\_PORT = 2525 *# ваш порт, например, 587*  
SMTP\_PASSWORD = "your\_password"  
SMTP\_MAIL\_FROM = "noreply@youremail.com"  
  
*# Конфигурация WebDriver*   
WEBDRIVER\_TYPE = "chrome"  
WEBDRIVER\_OPTION\_ARGS = [  
 "--force-device-scale-factor=2.0",  
 "--high-dpi-support=2.0",  
 "--headless",  
 "--disable-gpu",  
 "--disable-dev-shm-usage",  
 "--no-sandbox",  
 "--disable-setuid-sandbox",  
 "--disable-extensions",  
]  
  
*# Для внутреннего использования*   
WEBDRIVER\_BASEURL="http://superset:8088"  
*# Ссылка, отправляемая вовне*  
WEBDRIVER\_BASEURL\_USER\_FRIENDLY=<http://localhost:8088>

### Пользовательский [файл Docker](https://superset.apache.org/docs/installation/alerts-reports#custom-dockerfile)

Веб-браузер необходим для отображения диаграмм и информационных панелей, которые затем отправляются получателю. Поскольку в базовом образе RT.DataVision не установлен веб-браузер, нам необходимо объявить и установить его.

**Использование** [**Firefox**](https://superset.apache.org/docs/installation/alerts-reports#using-firefox)**:**

FROM apache/superset:1.0.1  
  
USER root  
  
RUN apt-get update && \  
 apt-get install --no-install-recommends -y firefox-esr  
  
ENV GECKODRIVER\_VERSION=0.29.0  
RUN wget -q https://github.com/mozilla/geckodriver/releases/download/v${GECKODRIVER\_VERSION}/geckodriver-v${GECKODRIVER\_VERSION}-linux64.tar.gz && \  
 tar -x geckodriver -zf geckodriver-v${GECKODRIVER\_VERSION}-linux64.tar.gz -O > /usr/bin/geckodriver && \  
 chmod 755 /usr/bin/geckodriver && \  
 rm geckodriver-v${GECKODRIVER\_VERSION}-linux64.tar.gz  
  
RUN pip install --no-cache gevent psycopg2 redis  
  
USER superset

**Использование** [**Chrome**](https://superset.apache.org/docs/installation/alerts-reports#using-chrome)

FROM apache/superset:1.0.1  
  
USER root  
  
RUN apt-get update && \  
 wget -q https://dl.google.com/linux/direct/google-chrome-stable\_current\_amd64.deb && \  
 apt-get install -y --no-install-recommends ./google-chrome-stable\_current\_amd64.deb && \  
 rm -f google-chrome-stable\_current\_amd64.deb  
  
RUN export CHROMEDRIVER\_VERSION=$(curl --silent https://chromedriver.storage.googleapis.com/LATEST\_RELEASE\_88) && \  
 wget -q https://chromedriver.storage.googleapis.com/${CHROMEDRIVER\_VERSION}/chromedriver\_linux64.zip && \  
 unzip chromedriver\_linux64.zip -d /usr/bin && \  
 chmod 755 /usr/bin/chromedriver && \  
 rm -f chromedriver\_linux64.zip  
  
RUN pip install --no-cache gevent psycopg2 redis  
  
USER superset

Не забудьте установить ***WEBDRIVER\_TYPE*** и ***WEBDRIVER\_OPTION\_ARGS*** в своей конфигурации, если вы используете Chrome.

### Краткое описание действий по включению оповещений и отчетов

Используя приведенные ниже шаблоны:

1. Создайте новый каталог и создайте Dockerfile.
2. Создайте расширенный образ с помощью Dockerfile.
3. Создайте ***docker-compose.yaml*** файл в том же каталоге.
4. Создайте новый подкаталог с именем ***config***.
5. Создайте ***superset\_config.py*** файл в ***config*** подкаталоге.
6. Запустите образ, используя ***docker-compose up*** в том же каталоге, что и файл ***docker-compose.py***.
7. В новом окне терминала обновите БД, запустив ***docker exec -it superset-1.0.1-extended superset db upgrade***.
8. Затем запустите ***docker exec -it superset-1.0.1-extended superset init***.
9. Затем, если это необходимо, настройте пользователя-администратора ***docker exec -it superset-1.0.1-extended superset fab create-admin***.
10. Наконец, перезапустите работающий экземпляр - ***CTRL-C***, затем ***docker-compose up***.

(примечание: на момент написания актуальна версия 1.0.1, вы можете изменить номер версии на последнюю версию, если доступна более новая версия)

### Компоновка Docker

В файле компоновки docker перечислены службы, которые будут использоваться при запуске образа. Конкретные службы, необходимые для оповещений и отчетов, описаны ниже.

[**Брокер**](https://superset.apache.org/docs/installation/alerts-reports#redis-message-broker) **сообщений Redis**

Чтобы переправлять запросы между celery worker и экземпляром RT.DataVision, мы используем брокер сообщений. Этот шаблон использует Redis.

**Замена SQLite на** [**Postgres**](https://superset.apache.org/docs/installation/alerts-reports#replacing-sqlite-with-postgres)

Хотя для оповещений и отчетов можно использовать SQLite, настоятельно рекомендуется использовать более совершенную БД. Базовый образ использует Postgres.

**Celery** [**worker**](https://superset.apache.org/docs/installation/alerts-reports#celery-worker)

Рабочий процесс будет обрабатывать задачи, которые необходимо выполнить при срабатывании оповещения или отчета.

**Celery beat**

Beat – это планировщик, который сообщает рабочему процессу, когда выполнять свои задачи. Это расписание определяется при создании оповещения или отчета.

**Полная** [**конфигурация**](https://superset.apache.org/docs/installation/alerts-reports#full-docker-composeyaml-configuration) ***docker-compose.yaml***

Сервисы Redis, Postgres, Celery worker и Celery beat определены в шаблоне:

Конфиг для ***docker-compose.yaml***:

version: '3.6'  
services:  
 redis:  
 image: redis:6.0.9-buster  
 restart: on-failure  
 volumes:  
 - redis:/data  
 postgres:  
 image: postgres  
 restart: on-failure  
 environment:  
 POSTGRES\_DB: superset  
 POSTGRES\_PASSWORD: superset  
 POSTGRES\_USER: superset  
 volumes:  
 - db:/var/lib/postgresql/data  
 worker:  
 image: superset-1.0.1-extended  
 restart: on-failure  
 healthcheck:  
 disable: true  
 depends\_on:  
 - superset  
 - postgres  
 - redis  
 command: "celery --app=superset.tasks.celery\_app:app worker --pool=gevent --concurrency=500"  
 volumes:  
 - ./config/:/app/pythonpath/  
 beat:  
 image: superset-1.0.1-extended  
 restart: on-failure  
 healthcheck:  
 disable: true  
 depends\_on:  
 - superset  
 - postgres  
 - redis  
 command: "celery --app=superset.tasks.celery\_app:app beat --pidfile /tmp/celerybeat.pid --schedule /tmp/celerybeat-schedule"  
 volumes:  
 - ./config/:/app/pythonpath/  
 superset:  
 image: superset-1.0.1-extended  
 restart: on-failure  
 environment:  
 - SUPERSET\_PORT=8088  
 ports:  
 - "8088:8088"  
 depends\_on:  
 - postgres  
 - redis  
 command: gunicorn --bind 0.0.0.0:8088 --access-logfile - --error-logfile - --workers 5 --worker-class gthread --threads 4 --timeout 200 --limit-request-line 4094 --limit-request-field\_size 8190 superset.app:create\_app()  
 volumes:  
 - ./config/:/app/pythonpath/  
volumes:  
 db:  
 external: true  
 redis:  
 external: false

### [Резюме](https://superset.apache.org/docs/installation/alerts-reports#summary)

После создания расширенного образа с помощью ***Dockerfile***, а затем запуска этого образа с помощью ***docker-compose.yaml***, а также необходимых конфигураций в ***superset\_config.py***, теперь у вас должны быть корректные настройки оповещений и отчетов.

* Вышеуказанные шаблоны также работают в среде Docker, вам просто нужно добавить ***Deploy:*** в службы RT.DataVision, Redis и Postgres вместе с вашими конкретными конфигурациями.

## [Шаблоны](https://superset.apache.org/docs/installation/sql-templating#sql-templating) SQL

### [Шаблоны](https://superset.apache.org/docs/installation/sql-templating#jinja-templates) Jinja

SQL Lab and Explore поддерживает [шаблоны Jinja](https://jinja.palletsprojects.com/en/2.11.x/) в запросах. Чтобы включить шаблоны, ***ENABLE\_TEMPLATE\_PROCESSING*** необходимо включить флаг функции в ***superset\_config.py***. Когда шаблоны включены, код Python можно встраивать в виртуальные наборы данных и в пользовательский SQL в фильтрах и элементах управления метриками в Исследовании. По умолчанию в контексте Jinja доступны следующие переменные:

* ***columns:*** столбцы, по которым следует группировать в запросе,
* ***filter:*** фильтры, примененные в запросе,
* ***from\_dttm:*** начальное ***datetime*** значение из выбранного диапазона времени (***None*** если не определено),
* ***to\_dttm:*** конечное ***datetime*** значение из выбранного диапазона времени (***None*** если не определено),
* ***groupby:*** столбцы, по которым следует группировать в запросе (устарело),
* ***metrics:*** агрегатные выражения в запросе,
* ***row\_limit:*** ограничение строки запроса,
* ***row\_offset:*** смещение строки запроса,
* ***table\_columns:*** столбцы, доступные в наборе данных,
* ***time\_column:*** временной столбец запроса (***None*** если не определен),
* ***time\_grain:*** выбранный интервал времени (***None*** если не определен).

Например, чтобы добавить диапазон времени в виртуальный набор данных, вы можете написать следующее:

SELECT \* from tbl where dttm\_col > '{{ from\_dttm }}' and dttm\_col < '{{ to\_dttm }}'

Чтобы добавить пользовательские функции в контекст Jinja, вам необходимо перегрузить контекст Jinja по умолчанию в вашей среде, определив ***JINJA\_CONTEXT\_ADDONS*** в вашей конфигурации RT.DataVision (***superset\_config.py***). Объекты, на которые есть ссылки в этом словаре, доступны для использования пользователями там, где доступен контекст Jinja.

JINJA\_CONTEXT\_ADDONS = {

'my\_crazy\_macro': lambda x: x\*2,

}

Помимо шаблонов Jinja по умолчанию, SQL Lab также поддерживает самоопределяемый обработчик шаблонов, установив ***CUSTOM\_TEMPLATE\_PROCESSORS*** в вашей конфигурации RT.DataVision. Значения в этом словаре перезаписывают обработчики шаблонов Jinja по умолчанию для указанного ядра базы данных. В приведенном ниже примере настраивается пользовательский обработчик шаблонов presto, который реализует собственную логику обработки шаблона макроса с разбором регулярных выражений. Он использует ***$макрос*** стиля вместо ***{{ }}***стиля в шаблонах Jinja.

При настройке с ***CUSTOM\_TEMPLATE\_PROCESSORS*** помощью шаблон SQL в базе данных presto обрабатывается пользовательским, а не шаблоном по умолчанию.

def DATE(  
 ts: datetime, day\_offset: SupportsInt = 0, hour\_offset: SupportsInt = 0  
) -> str:  
 """Current day as a string."""  
 day\_offset, hour\_offset = int(day\_offset), int(hour\_offset)  
 offset\_day = (ts + timedelta(days=day\_offset, hours=hour\_offset)).date()  
 return str(offset\_day)  
  
class CustomPrestoTemplateProcessor(PrestoTemplateProcessor):  
 """A custom presto template processor."""  
  
 engine = "presto"  
  
 def process\_template(self, sql: str, \*\*kwargs) -> str:  
 """Processes a sql template with $ style macro using regex."""  
 *# Добавление макроса.*  
 macros = {  
 "DATE": partial(DATE, datetime.utcnow())  
 } *# Тип: Dict[str, Any]*  
 *# Обновить макросом*  
 macros.update(self.context)  
 macros.update(kwargs)  
  
 def replacer(match):  
 """Expand $ style macros with corresponding function calls."""  
 macro\_name, args\_str = match.groups()  
 args = [a.strip() for a in args\_str.split(",")]  
 if args == [""]:  
 args = []  
 f = macros[macro\_name[1:]]  
 return f(\*args)  
  
 macro\_names = ["$" + name for name in macros.keys()]  
 pattern = r"(%s)\s\*\(([^()]\*)\)" % "|".join(map(re.escape, macro\_names))  
 return re.sub(pattern, replacer, sql)  
  
CUSTOM\_TEMPLATE\_PROCESSORS = {  
 CustomPrestoTemplateProcessor.engine: CustomPrestoTemplateProcessor  
}

Лаборатория SQL также включает функцию проверки запросов в реальном времени с подключаемыми серверными частями. Вы можете настроить, какая реализация проверки используется с каким механизмом базы данных, добавив блок, подобный следующему, в ваш файл конфигурации:

FEATURE\_FLAGS = {  
 'SQL\_VALIDATORS\_BY\_ENGINE': {  
 'presto': 'PrestoDBSQLValidator',  
 }  
}

Доступные валидаторы и имена можно найти в [sql\_validators](https://github.com/apache/superset/tree/master/superset/sql_validators).

### Доступные [макросы](https://superset.apache.org/docs/installation/sql-templating#available-macros)

В этом разделе мы рассмотрим предопределенные макросы Jinja в RT.DataVision.

**Текущее имя пользователя**

Макрос ***{{ current\_username() }}*** возвращает имя пользователя, вошедшего в систему в данный момент.

Если в вашей конфигурации RT.DataVision включено кэширование, то по умолчанию ***username*** значение будет использоваться RT.DataVision при вычислении ключа кэша. Ключ кэша – это уникальный идентификатор, который определяет, будет ли попадание в кэш в будущем, и RT.DataVision сможет получить кэшированные данные.

Вы можете отключить включение ***username*** значения в вычисление ключа кеша, добавив в свой код Jinja следующий параметр:

{{ current\_username(add\_to\_cache\_keys=False) }}

**Текущий идентификатор пользователя**

Макрос ***{{ current\_user\_id()}}*** возвращает user\_id текущего пользователя, вошедшего в систему.

Если в вашей конфигурации RT.DataVision включено кэширование, то по умолчанию ***user\_id*** значение будет использоваться RT.DataVision при вычислении ключа кэша. Ключ кэша – это уникальный идентификатор, который определяет, будет ли попадание в кэш в будущем, и RT.DataVision сможет получить кэшированные данные.

Вы можете отключить включение ***user\_id*** значения в вычисление ключа кеша, добавив в свой код Jinja следующий параметр:

{{ current\_user\_id(add\_to\_cache\_keys=False) }}

**Пользовательские параметры URL**

Макрос ***{{ url\_param('custom\_variable') }}*** позволяет вам определять произвольные параметры URL и ссылаться на них в коде SQL.

Вот конкретный пример:

* Вы пишете следующий запрос в SQL Lab:

SELECT count(\*)  
FROM ORDERS  
WHERE country\_code = '{{ url\_param('countrycode') }}'

* Вы размещаете RT.DataVision в домене [www.example.com](http://www.example.com/) и отправляете своему коллеге в России следующий URL-адрес Лаборатории SQL, [***www.example.com/superset/sqllab?countrycode=RU***](http://www.example.com/superset/sqllab?countrycode=RU), а своему коллеге в США – следующий URL-адрес Лаборатории SQL ***www.example.com/superset/sqllab?countrycode=US***
* Для вашего коллеги в России запрос Лаборатории SQL будет отображаться следующим образом:

SELECT count(\*)  
FROM ORDERS  
WHERE country\_code = 'RU'

* Для вашего коллеги в США запрос Лаборатории SQL будет отображаться следующим образом:

SELECT count(\*)  
FROM ORDERS  
WHERE country\_code = 'US'

**Явное включение значений в ключ кэша**

Функция ***{{ cache\_key\_wrapper() }}*** явно указывает RT.DataVision добавить значение в накопленный список значений, используемых при вычислении ключа кэша.

Эта функция необходима только в том случае, если вы хотите обернуть возвращаемые значения собственной пользовательской функции в ключ кэша.

Обратите внимание, что эта функция обеспечивает кэширование значений в ***user\_id*** и в ***username*** в ***current\_user\_id()*** и ***current\_username()*** (если у вас включено кэширование).

**Значения фильтра**

Вы можете получить значение для определенного фильтра в виде списка, используя ***{{ filter\_values() }}***.

Это полезно, если:

* вы хотите использовать компонент фильтра для фильтрации запроса, где имя столбца компонента фильтра не соответствует имени в операторе выбора,
* вы хотите иметь возможность фильтрации внутри основного запроса для повышения производительности.

Пример:

SELECT action, count(\*) as times  
FROM logs  
WHERE  
 action in ({{ "'" + "','".join(filter\_values('action\_type')) + "'" }})  
GROUP BY action

**Фильтры для определенного столбца**

Макрос ***{{ get\_filters() }}*** возвращает фильтры, примененные к данному столбцу. Помимо возврата значений (аналогично тому, как это делается в ***filter\_values()***), ***get\_filters()*** макрос возвращает оператор, указанный в пользовательском интерфейсе исследования.

Это полезно, если:

* вы хотите обрабатывать больше, чем оператор IN в вашем предложении SQL,
* вы хотите обрабатывать генерацию пользовательских условий SQL для фильтра,
* вы хотите иметь возможность фильтровать внутри основного запроса для ускорения.

Пример:

WITH RECURSIVE  
 superiors(employee\_id, manager\_id, full\_name, level, lineage) AS (  
 SELECT  
 employee\_id,  
 manager\_id,  
 full\_name,  
 1 as level,  
 employee\_id as lineage  
 FROM  
 employees  
 WHERE  
 1=1  
  
 {# Render a blank line #}  
 {%- for filter in get\_filters('full\_name', remove\_filter=True) -%}  
  
 {%- if filter.get('op') == 'IN' -%}  
 AND  
 full\_name IN ( {{ "'" + "', '".join(filter.get('val')) + "'" }} )  
 {%- endif -%}  
  
 {%- if filter.get('op') == 'LIKE' -%}  
 AND  
 full\_name LIKE {{ "'" + filter.get('val') + "'" }}  
 {%- endif -%}  
  
 {%- endfor -%}  
 UNION ALL  
 SELECT  
 e.employee\_id,  
 e.manager\_id,  
 e.full\_name,  
 s.level + 1 as level,  
 s.lineage  
 FROM  
 employees e,  
 superiors s  
 WHERE s.manager\_id = e.employee\_id  
 )  
  
 SELECT  
 employee\_id, manager\_id, full\_name, level, lineage  
 FROM  
 superiors  
 order by lineage, level

## Создание пользовательских плагинов Viz

Это руководство поможет вам создать плагин «Hello World». Цель состоит в том, чтобы предоставить базовую основу для создания любого вида визуализации данных, используя любую библиотеку визуализации (например, ECharts, AntV, HighCharts, VX и D3).

Вы можете создать плагин «Hello World», запустив генератор [Yeoman](https://yeoman.io/), который принимает несколько простых опций и предоставляет основу для этого плагина.

### Установите Yeoman и генератор пакетов RT.DataVision

Плагин «Hello World», который мы будем создавать, автоматически генерируется с помощью [Yeoman](https://yeoman.io/). Давайте сначала его установим, открыв терминал и установив как ***yo*** модуль, так и [генератор пакетов RT.DataVision](https://github.com/apache-superset/superset-ui/tree/master/packages/generator-superset) ***( v0.14.7***), чтобы создать новый плагин.

npm install -g yo @superset-ui/generator-superset

### Установите RT.DataVision

Инструкция по установке представлена в документе «RT.DataVision. Инструкция по установке».

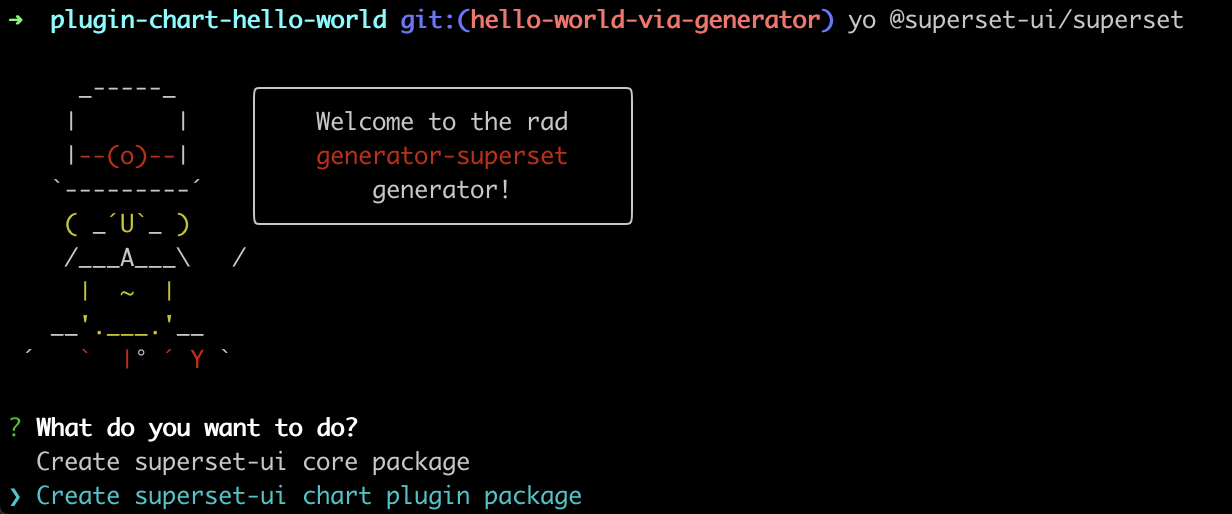
### Установить Sup RT.DataVision- [UI](https://superset.apache.org/docs/installation/building-custom-viz-plugins#install-superset-ui)

1. Клонируйте [репозиторий ***superset-ui*** на свой](https://github.com/apache-superset/superset-ui) компьютер. Он может находиться в том же родительском каталоге, что и ваш RT.DataVision репозиторий.
2. Используйте терминал, чтобы войти в каталог ***superset-ui***.
3. Запустите ***yarn install*** и дождитесь установки всех пакетов.

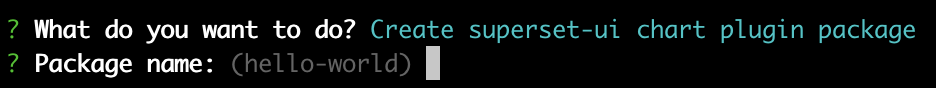
### Создайте свой «Привет, мир[»](https://superset.apache.org/docs/installation/building-custom-viz-plugins#build-your-hello-world)

**Сгенерируйте код**[**:**](https://superset.apache.org/docs/installation/building-custom-viz-plugins#write-generate-some-code)

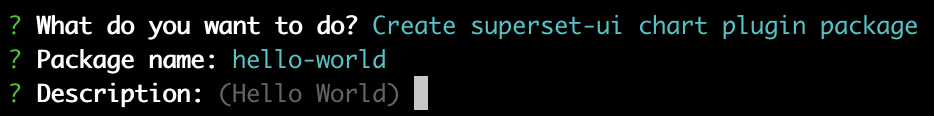
1. Используя терминал, чтобы войти в локальную папку репо в ***superset-ui***, а затем в подкаталог ***plugins***.
2. Создайте новый каталог для вашего плагина, т.е. ***mkdir plugin-chart-hello-world***. **Примечание:** мы настоятельнорекомендуем следовать ***plugin-chart-your-plugin-name*** шаблону.
3. Перейдите в созданный каталог ***cd plugin-chart-hello-world***.
4. Далее, запустите ***yo @superset-ui/superset***.
5. Выполните ***Create superset-ui chart plugin package,*** как на следующем экране:



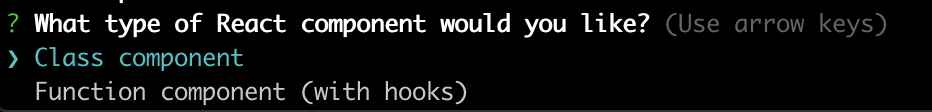
1. Дайте ему имя (в нашем случае используйте значение по умолчанию, основанное на имени папки):



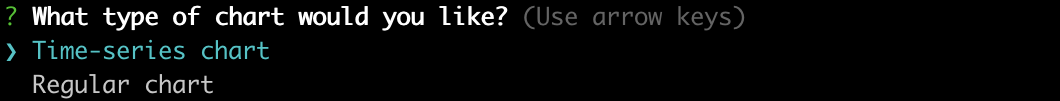
1. Дайте ему описание (опять же, по умолчанию все в порядке!)



1. Выберите тип компонента React, который вы хотите создать (компонент класса или функции).



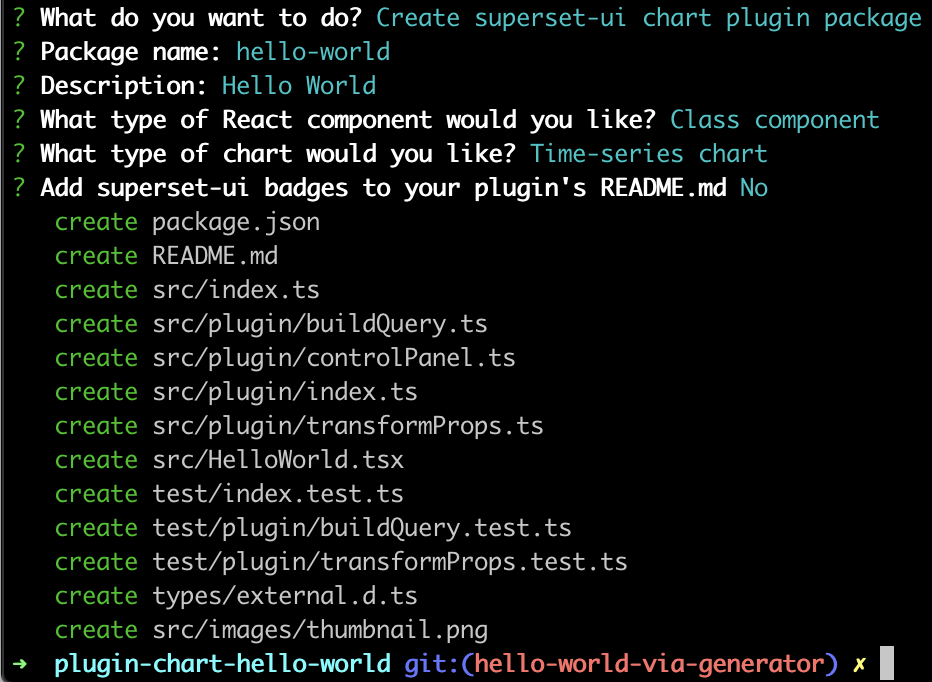
1. Выберите, хотите ли вы, чтобы ваша визуализация была основана на временных рядах или нет.



1. Выберите, хотите ли вы включать значки в верхней части файла README (на самом деле это необходимо только в том случае, если вы собираетесь внести свой плагин в ***superset-ui*** репозиторий).

https://superset.apache.org/img/custom-plugins/plugin-6-yeoman-badges.png

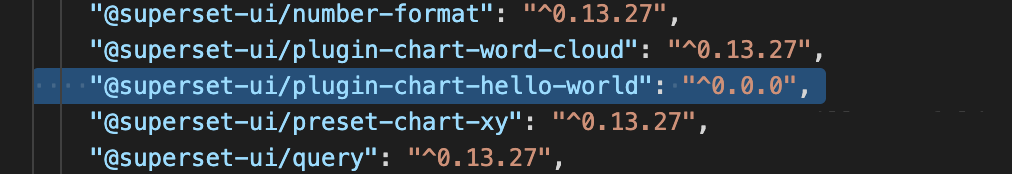
1. Посмотрите на файлы, которые генератор создал для вас. Обратите внимание, что КАЖДЫЙ из них содержит в комментариях много информации о том, для чего они нужны и как их лучше всего использовать.



### Добавьте свой плагин в RT.DataVision (с ссылкой NPM[)](https://superset.apache.org/docs/installation/building-custom-viz-plugins#add-your-plugin-to-superset-with-npm-link)

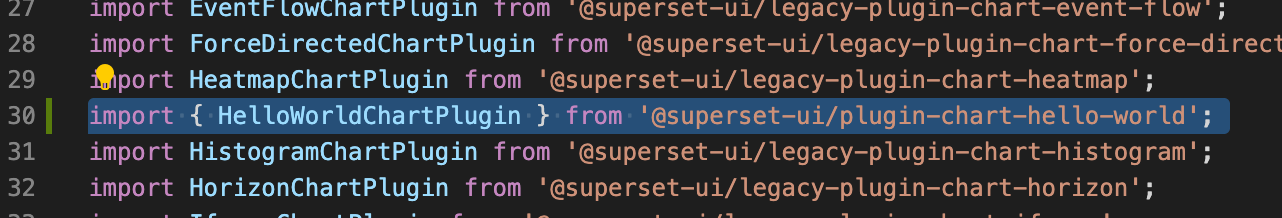
Теперь мы хотим, чтобы это работало! Для этого мы добавим ваш пакет в RT.DataVision и воспользуемся ***npm link,*** чтобы увидеть его на месте, без необходимости **создавать** плагин или открывать какие-либо PR на Github.

1. Добавьте свой пакет в ***package.json*** файл в формате ***superset/superset-frontend***.

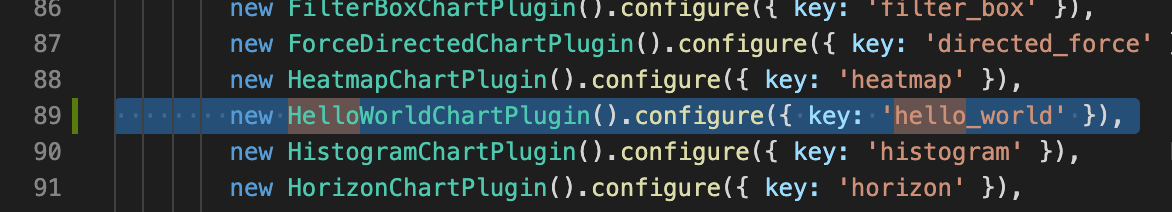


Примечание. Не запускайте ***npm install...*** объяснение ниже.

1. Добавьте свой плагин в ***MainPreset.js*** файл (расположенный в ***superset/superset-frontend/src/visualizations/presets/MainPreset.js***) в двух местах вместе с другими плагинами.



***{' '}***



1. Откройте окно терминала для ***superset/superset-frontend***. Если вы выполнили описанные выше шаги по установке RT.DataVision, у вас все еще может быть запущен веб-пакет, и вы можете просто остановить его с помощью ***ctrol-c***. Если нет, просто откройте новое окно и/или перейдите к каталогу используя команду ***cd***.
2. Используйте ***npm link*** для создания символической ссылки на плагин, используя относительный путь к ***superset-ui*** папке вашего плагина, например, ***npm link ../../superset-ui/plugins/plugin-chart-hello-world***.
3. Перезапустите сервер разработки webpack с помощью ***npm run dev-server***. Вы узнаете, что это сработало, если увидите строку с указанием ***[Superset Plugin] Use symlink source for @superset-ui/plugin-chart-hello-world @ ^0.0.0***.

**ПРИМЕЧАНИЕ.** Если/когда вы выполняете команду ***npm install***, которая стирает символическую ссылку, сгенерированную с помощью ***npm link***, вам придется повторить эти шаги.

**ПРИМЕЧАНИЕ.** Динамический импорт находится в стадии разработки. Наша цель сделать редактирование ***package.json*** и ***MainPreset.js*** ненужным, поэтому все изменения кода делаются в ОДНОМ репо.

### Отображене плагина

Теперь у вас должна быть возможность перейти в режим просмотра в локальном RT.DataVision и добавить новую диаграмму! Вы увидите свой новый плагин, когда будете выбирать тип визуализации.

Теперь вы можете загрузить некоторые данные, и вы увидите их в плагине!

Плагин также выводит в консоль вашего браузера:

* ***formData***, то есть все, что отправлено в вашу визуализацию из элементов управления,
* ***props***, как вывод из ***transformProps*** файла для использования вашим плагином,
* Фактический элемент HTML, к которому подключается ваш плагин для любых необходимых манипуляций с DOM.



## Использование плагина

Теперь вы можете свободно работать с вашим новым плагином! Вот несколько мест, где можно начать разбираться:

### Читайте комментарии и [документы](https://superset.apache.org/docs/installation/building-custom-viz-plugins#read-the-comments-and-docs)

Просмотрите полное дерево файлов плагина. Readme дает подробную информацию о работе каждого файла. Каждый из этих файлов снабжен подробными комментариями о том, для чего он предназначен, и о том, что с ним можно делать.

### Элементы управления

Плагин включает в себя пару примеров элементов управления, но вы, безусловно, можете добавлять столько, сколько вам нужно. Комментарии/документация в файле элементов управления – это только начало, но мы рекомендуем просмотреть существующие ***superset-ui*** плагины, чтобы найти дополнительные примеры того, как вы можете реализовать элементы управления для улучшения ваших запросов, работы с вашими данными и изменения отображения вашей визуализации.

### Создайте идеальный [запрос](https://superset.apache.org/docs/installation/building-custom-viz-plugins#build-the-perfect-query)

Файл, в ***buildQuery*** котором ваш плагин фактически извлекает данные из серверной части RT.DataVision. Этот файл создает запрос «контекст» для вашего плагина. Для простого плагина этот файл не должен много делать. Есть пара изменений, которые необходимо внести в плагин временных рядов, например, в генератор Yeoman.

Этот файл также позволяет вам добавлять различные операции постобработки, чтобы серверная часть RT.DataVision обрабатывала ваши данные различными способами (поворот и т. д.), но это совсем другая тема, которую мы рассмотрим отдельно в ближайшем будущем.

### Стиль с графическими значками (Emotion)

Каждый из этих методов позволяет вам добавлять пользовательские стили CSS с помощью Emotion 👩‍🎤 (подход CSS-in-JS), который имеет доступ к растущему набору переменных темы RT.DataVision, а также автоматически применяет стили к вашему плагину, чтобы они не «просачиваться» в другие области RT.DataVision.

В подключаемый модуль «Hello World» мы включили несколько примеров переменных темы (***colors***, ***gridUnits*** и типографские веса/размеры). Мы продолжим добавлять дополнительные графические значки в этот файл темы, поскольку мы продолжаем подталкивать RT.DataVision (и плагины визуализации) к стандартам редизайна RT.DataVision.

### Построение

В этом руководстве вы создали свой плагин в ***superset-ui*** репозитории. Это означает, что вы можете использовать встроенные сценарии сборки, которые предоставляет репозиторий. С выбранным вами терминалом просто ***cd*** зайдите в корневой каталог ***supeset-ui*** и запустите ***yarn build***. Это запустит сборку ВСЕХ плагинов и пакетов RT.DataVision, включая ваш.

### Тестирование

Плагин «Hello World» включает в себя несколько базовых тестов Jest, которые служат отправной точкой для добавления модульных тестов в ваш плагин. Они выполняют быструю проверку правильности загрузки плагина, а затем выполняют основные действия, чтобы убедиться, что ваши элементы управления правильно соблюдаются, путем изменения результирующих данных и/или реквизитов плагина. Запуск ***yarn test*** из корневого каталога ***superset-ui*** запустит все тесты для плагинов/пакетов, включая ваш «Hello World».

### Развертывание пользовательской визуализации в рабочей [среде](https://superset.apache.org/docs/installation/building-custom-viz-plugins#deploying-custom-visualization-to-production)

Чтобы развернуть плагины в производственной среде, у вас должен быть дополнительный код внутри RT.DataVision, который включает пакеты npm ваших плагинов, чтобы их можно было установить во внешнем интерфейсе.

Один из вариантов – создать свой Dockerfile, чтобы он содержал ваши пользовательские пакеты визуализации.

## Запуск в [Kubernetes](https://superset.apache.org/docs/installation/running-on-kubernetes#running-on-kubernetes)

Запуск в Kubernetes поддерживается предоставленной диаграммой [Helm](https://helm.sh/), которую можно найти в официальном [репозитории RT.DataVision helm](https://apache.github.io/superset/index.yaml).

[**Предпосылки**](https://superset.apache.org/docs/installation/running-on-kubernetes#prerequisites)**:**

* Кластер Kubernetes.
* Helm установлен.

**Выполнение:**

1. Добавьте в RT.DataVision репозиторий Helm

helm repo add superset https://apache.github.io/superset  
"superset" has been added to your repositories

1. Посмотрите графики в репозитории

helm search repo superset  
NAME CHART VERSION APP VERSION DESCRIPTION  
superset/superset 0.1.1 1.0 Apache Superset is a modern, enterprise-ready b...

1. Настройте переопределение настроек

Как и в любой обычной диаграмме Helm, вам нужно создать ***values.yaml*** файл, который будет определять/переопределять любые значения, представленные в [values.yaml](https://github.com/apache/superset/tree/master/helm/superset/values.yaml) по умолчанию , или из любой из зависимых диаграмм, от которых он зависит:

* [bitnami/redis](https://artifacthub.io/packages/helm/bitnami/redis),
* [bitnami/postgresql](https://artifacthub.io/packages/helm/bitnami/postgresql).

Ниже приведена дополнительная информация о некоторых важных переопределениях, которые могут вам понадобиться.

1. Установите и запустите

helm upgrade --install --values my-values.yaml superset superset/superset

Вы должны увидеть различные всплывающие поды, такие как:

kubectl get pods  
NAME READY STATUS RESTARTS AGE  
superset-celerybeat-7cdcc9575f-k6xmc 1/1 Running 0 119s  
superset-f5c9c667-dw9lp 1/1 Running 0 4m7s  
superset-f5c9c667-fk8bk 1/1 Running 0 4m11s  
superset-init-db-zlm9z 0/1 Completed 0 111s  
superset-postgresql-0 1/1 Running 0 6d20h  
superset-redis-master-0 1/1 Running 0 6d20h  
superset-worker-75b48bbcc-jmmjr 1/1 Running 0 4m8s  
superset-worker-75b48bbcc-qrq49 1/1 Running 0 4m12s

Точный список будет зависеть от некоторых ваших конкретных переопределений конфигурации, но обычно вы должны ожидать:

* N ***superset-xxxx-yyyy*** и ***superset-worker-xxxx-yyyy*** pods (в зависимости от вашего ***replicaCount*** значения),
* 1 ***superset-postgresql-0*** в зависимости от ваших настроек postgres,
* 1 ***superset-redis-master-0*** в зависимости от ваших настроек Redis,
* 1 ***superset-celerybeat-xxxx-yyyy*** модуль, если у вас есть ***supersetCeleryBeat.enabled = true*** переопределение значений.

1. Доступ к RT.DataVision

На диаграмме будут опубликованы соответствующие сервисы для отображения пользовательского интерфейса RT.DataVision внутри вашего кластера k8s. Чтобы получить к нему доступ извне, вам нужно:

* Настройте службу как ***LoadBalancerилиNodePort.***
* Настройте ***Ingress*** для него – диаграмма включает определение, но ее необходимо настроить в соответствии с вашими потребностями (имя хоста, tls, аннотации и т. д.).
* Запустите ***kubectl port-forward superset-xxxx-yyyy :8088***, чтобы напрямую туннелировать порт одного модуля на ваш локальный хост.

В зависимости от того, как вы настроили внешний доступ, URL-адрес может различаться. После того, как вы определили соответствующий URL-адрес, вы можете войти в систему с помощью:

* пользователь: ***admin,***
* пароль: ***admin.***

**Важные** [**настройки**](https://superset.apache.org/docs/installation/running-on-kubernetes#important-settings)**:**

[Настройки](https://superset.apache.org/docs/installation/running-on-kubernetes#security-settings) безопасности:

Настройки безопасности и пароли по умолчанию включены, но вам **СЛЕДУЕТ** заменить их своими собственными, в частности:

postgresql:  
 postgresqlPassword: superset

[Зависимости](https://superset.apache.org/docs/installation/running-on-kubernetes#dependencies):

Установите дополнительные пакеты и выполните любую другую настройку начальной загрузки в этом скрипте. Для рабочих кластеров рекомендуется создать собственный образ, выполнив этот шаг в CI. В следующем примере устанавливаются драйверы базы данных Big Query и Elasticsearch, чтобы вы могли подключаться к этим источникам данных в своей установке RT.DataVision.

bootstrapScript: |  
 #!/bin/bash  
 pip install psycopg2==2.8.5 \  
 redis==3.2.1 \  
 pybigquery==2.26.0 \  
 elasticsearch-dbapi==0.2.5 &&\  
 if [ ! -f ~/bootstrap ]; then echo "Running Superset with uid {{ .Values.runAsUser }}" > ~/bootstrap; fi

[superset\_config.py](https://superset.apache.org/docs/installation/running-on-kubernetes#superset_configpy)

Значение по умолчанию ***superset\_config.py*** довольно минимальное, и вам, скорее всего, потребуется его расширить. Это делается путем указания одной или нескольких записей ключ/значение в ***configOverrides***, например:

configOverrides:  
 my\_override: |  
 # This will make sure the redirect\_uri is properly computed, even with SSL offloading  
 ENABLE\_PROXY\_FIX = True  
 FEATURE\_FLAGS = {  
 "DYNAMIC\_PLUGINS": True  
 }

Они будут оцениваться как шаблоны Helm и, следовательно, смогут ссылаться на другие ***values.yaml*** переменные, например ***{{ .Values.ingress.hosts[0] }}***, будут разрешаться в ваш входной внешний домен.

Все ***superset\_config.py*** будет установлено как секретно, поэтому безопасно передавать конфиденциальные параметры напрямую... однако для этого может быть более читабельно использовать секретные переменные env.

Полные файлы Python можно получить, запустив ***helm upgrade --install --values my-values.yaml --set-file configOverrides.oauth=set\_oauth.py***

[Переменные](https://superset.apache.org/docs/installation/running-on-kubernetes#environment-variables) среды:

Они могут быть переданы как ключ/значение либо с ***extraEnv*** либо ***extraSecret*** ***Env*** если они чувствительны. Затем на них можно ссылаться, ***superset\_config.py*** используя, например, ***os.environ.get("VAR")***.

extraEnv:  
 SMTP\_HOST: smtp.gmail.com  
 SMTP\_USER: user@gmail.com  
 SMTP\_PORT: "587"  
 SMTP\_MAIL\_FROM: user@gmail.com  
  
extraSecretEnv:  
 SMTP\_PASSWORD: xxxx  
  
configOverrides:  
 smtp: |  
 import ast  
 SMTP\_HOST = os.getenv("SMTP\_HOST","localhost")  
 SMTP\_STARTTLS = ast.literal\_eval(os.getenv("SMTP\_STARTTLS", "True"))  
 SMTP\_SSL = ast.literal\_eval(os.getenv("SMTP\_SSL", "False"))  
 SMTP\_USER = os.getenv("SMTP\_USER","superset")  
 SMTP\_PORT = os.getenv("SMTP\_PORT",25)  
 SMTP\_PASSWORD = os.getenv("SMTP\_PASSWORD","superset")

Системные [пакеты](https://superset.apache.org/docs/installation/running-on-kubernetes#system-packages):

Если требуются новые системные пакеты, их можно установить перед запуском приложения, переопределив содержимое контейнера ***command***, например:

supersetWorker:  
 command:  
 - /bin/sh  
 - -c  
 - |  
 apt update  
 apt install -y somepackage  
 apt autoremove -yqq --purge  
 apt clean  
  
 *# Запуск celery-воркера*  
 . {{ .Values.configMountPath }}/superset\_bootstrap.sh; celery --app=superset.tasks.celery\_app:app worker

[Источники](https://superset.apache.org/docs/installation/running-on-kubernetes#data-sources) данных:

Определения источников данных могут быть автоматически объявлены путем предоставления определений yaml типа ключ/значение в ***extraConfigs***:

extraConfigs:  
 datasources-init.yaml: |  
 databases:  
 - allow\_file\_upload: true  
 allow\_ctas: true  
 allow\_cvas: true  
 database\_name: example-db  
 extra: "{\r\n \"metadata\_params\": {},\r\n \"engine\_params\": {},\r\n \"\  
 metadata\_cache\_timeout\": {},\r\n \"schemas\_allowed\_for\_file\_upload\": []\r\n\  
 }"  
 sqlalchemy\_uri: example://example-db.local  
 tables: []

Они также будут смонтированы как секретные и могут включать конфиденциальные параметры.

[**Примеры**](https://superset.apache.org/docs/installation/running-on-kubernetes#configuration-examples) **конфигурации:**

Настройка [OAuth](https://superset.apache.org/docs/installation/running-on-kubernetes#setting-up-oauth):

extraEnv:  
 AUTH\_DOMAIN: example.com  
  
extraSecretEnv:  
 GOOGLE\_KEY: xxxxxxxxxxxx-xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx.apps.googleusercontent.com  
 GOOGLE\_SECRET: xxxxxxxxxxxxxxxxxxxxxxxx  
  
configOverrides:  
 enable\_oauth: |  
 ENABLE\_PROXY\_FIX = True  
  
 from flask\_appbuilder.security.manager import (AUTH\_OAUTH, AUTH\_DB)  
 AUTH\_TYPE = AUTH\_OAUTH  
 OAUTH\_PROVIDERS = [  
 {  
 "name": "google",  
 "icon": "fa-google",  
 "token\_key": "access\_token",  
 "remote\_app": {  
 "client\_id": os.getenv("GOOGLE\_KEY"),  
 "client\_secret": os.getenv("GOOGLE\_SECRET"),  
 "api\_base\_url": "https://www.googleapis.com/oauth2/v2/",  
 "client\_kwargs": {"scope": "email profile"},  
 "request\_token\_url": None,  
 "access\_token\_url": "https://accounts.google.com/o/oauth2/token",  
 "authorize\_url": "https://accounts.google.com/o/oauth2/auth",  
 "authorize\_params": {"hd": os.getenv("AUTH\_DOMAIN", "")}  
 },  
 }  
 ]  
  
 *# Map Authlib roles to superset roles*  
 AUTH\_ROLE\_ADMIN = 'Admin'  
 AUTH\_ROLE\_PUBLIC = 'Public'  
  
 AUTH\_USER\_REGISTRATION = True  
  
 AUTH\_USER\_REGISTRATION\_ROLE = "Admin"

Включение оповещений и [отчетов](https://superset.apache.org/docs/installation/running-on-kubernetes#enable-alerts-and-reports)

Для этого, п. 3.12, вам необходимо:

Установите поддерживаемый веб-браузер как [рабочий](https://superset.apache.org/docs/installation/running-on-kubernetes#install-a-supported-webdriver-in-the-celery-worker).

Это делается либо с помощью пользовательского образа с предустановленным веб-драйвером, либо путем установки во время запуска путем переопределения файла ***command***. Вот рабочий пример для ***chromedriver***:

supersetWorker:  
 command:  
 - /bin/sh  
 - -c  
 - |  
 apt update  
 wget https://dl.google.com/linux/direct/google-chrome-stable\_current\_amd64.deb  
 apt install -y --no-install-recommends ./google-chrome-stable\_current\_amd64.deb  
 wget https://chromedriver.storage.googleapis.com/88.0.4324.96/chromedriver\_linux64.zip  
 unzip chromedriver\_linux64.zip  
 chmod +x chromedriver  
 mv chromedriver /usr/bin  
 apt autoremove -yqq --purge  
 apt clean  
 rm -f google-chrome-stable\_current\_amd64.deb chromedriver\_linux64.zip  
  
 *# Запуск*  
 . {{ .Values.configMountPath }}/superset\_bootstrap.sh; celery --app=superset.tasks.celery\_app:app worker

Выполните Celery beat:

Этот модуль будет запускать запланированные задачи, настроенные в разделе пользовательского интерфейса оповещений и отчетов:

supersetCeleryBeat:  
 enabled: true

Настройте соответствующие задания Celery и [параметры](https://superset.apache.org/docs/installation/running-on-kubernetes#configure-the-appropriate-celery-jobs-and-smtpslack-settings):

extraEnv:  
 SMTP\_HOST: smtp.gmail.com  
 SMTP\_USER: user@gmail.com  
 SMTP\_PORT: "587"  
 SMTP\_MAIL\_FROM: user@gmail.com  
  
extraSecretEnv:  
 SLACK\_API\_TOKEN: xoxb-xxxx-yyyy  
 SMTP\_PASSWORD: xxxx-yyyy  
  
configOverrides:  
 feature\_flags: |  
 import ast  
  
 FEATURE\_FLAGS = {  
 "ALERT\_REPORTS": True  
 }  
  
 SMTP\_HOST = os.getenv("SMTP\_HOST","localhost")  
 SMTP\_STARTTLS = ast.literal\_eval(os.getenv("SMTP\_STARTTLS", "True"))  
 SMTP\_SSL = ast.literal\_eval(os.getenv("SMTP\_SSL", "False"))  
 SMTP\_USER = os.getenv("SMTP\_USER","superset")  
 SMTP\_PORT = os.getenv("SMTP\_PORT",25)  
 SMTP\_PASSWORD = os.getenv("SMTP\_PASSWORD","superset")  
 SMTP\_MAIL\_FROM = os.getenv("SMTP\_MAIL\_FROM","superset@superset.com")  
  
 SLACK\_API\_TOKEN = os.getenv("SLACK\_API\_TOKEN",None)  
 celery\_conf: |  
 from celery.schedules import crontab  
  
 class CeleryConfig(object):  
 BROKER\_URL = f"redis://{env('REDIS\_HOST')}:{env('REDIS\_PORT')}/0"  
 CELERY\_IMPORTS = ('superset.sql\_lab', )  
 CELERY\_RESULT\_BACKEND = f"redis://{env('REDIS\_HOST')}:{env('REDIS\_PORT')}/0"  
 CELERY\_ANNOTATIONS = {'tasks.add': {'rate\_limit': '10/s'}}  
 CELERY\_IMPORTS = ('superset.sql\_lab', "superset.tasks", "superset.tasks.thumbnails", )  
 CELERY\_ANNOTATIONS = {  
 'sql\_lab.get\_sql\_results': {  
 'rate\_limit': '100/s',  
 },  
 'email\_reports.send': {  
 'rate\_limit': '1/s',  
 'time\_limit': 600,  
 'soft\_time\_limit': 600,  
 'ignore\_result': True,  
 },  
 }  
 CELERYBEAT\_SCHEDULE = {  
 'reports.scheduler': {  
 'task': 'reports.scheduler',  
 'schedule': crontab(minute='\*', hour='\*'),  
 },  
 'reports.prune\_log': {  
 'task': 'reports.prune\_log',  
 'schedule': crontab(minute=0, hour=0),  
 },  
 'cache-warmup-hourly': {  
 'task': 'cache-warmup',  
 'schedule': crontab(minute='\*/30', hour='\*'),  
 'kwargs': {  
 'strategy\_name': 'top\_n\_dashboards',  
 'top\_n': 10,  
 'since': '7 days ago',  
 },  
 }  
 }  
  
 CELERY\_CONFIG = CeleryConfig  
 reports: |  
 EMAIL\_PAGE\_RENDER\_WAIT = 60  
 WEBDRIVER\_BASEURL = "http://{{ template "superset.fullname" . }}:{{ .Values.service.port }}/"  
 WEBDRIVER\_BASEURL\_USER\_FRIENDLY = "https://www.example.com/"  
 WEBDRIVER\_TYPE= "chrome"  
 WEBDRIVER\_OPTION\_ARGS = [  
 "--force-device-scale-factor=2.0",  
 "--high-dpi-support=2.0",  
 "--headless",  
 "--disable-gpu",  
 "--disable-dev-shm-usage",  
 # This is required because our process runs as root (in order to install pip packages)  
 "--no-sandbox",  
 "--disable-setuid-sandbox",  
 "--disable-extensions",  
 ]

# Подключение к базам данных

Для подключения к базам данных необходимо установить драйверы баз данных.

## Установка [драйверов](https://superset.apache.org/docs/connecting-to-databases/installing-database-drivers#install-database-drivers)

RT.DataVision требует установки драйвера базы данных Python DB-API и диалекта SQLAlchemy для каждого хранилища данных, к которому вы хотите подключиться.

### Поддерживаемые базы данных и [зависимости](https://superset.apache.org/docs/connecting-to-databases/installing-database-drivers#supported-databases-and-dependencies)

RT.DataVision не поставляется в комплекте с возможностью подключения к базам данных, за исключением SQLite, который является частью стандартной библиотеки Python. Вам потребуется установить необходимые пакеты для базы данных, которую вы хотите использовать в качестве базы данных метаданных, а также пакеты, необходимые для подключения к базам данных, к которым вы хотите получить доступ через RT.DataVision.

Список некоторых рекомендуемых пакетов подключения RT.DataVision к базам данных представлен в *Приложении 2*.

Обратите внимание, что поддерживаются многие другие базы данных, основными критериями которых являются наличие функционального диалекта SQLAlchemy и драйвера Python. Поиск по ключевому слову «sqlalchemy + (имя базы данных)» должен помочь вам получить данные по подключению нужной базы данных.

## Добавление новых драйверов базы данных в [Docker](https://superset.apache.org/docs/connecting-to-databases/docker-add-drivers#adding-new-database-drivers-in-docker)

RT.DataVision требует установки драйвера базы данных Python для каждого дополнительного типа базы данных, к которому вы хотите подключиться. При локальной настройке RT.DataVision через ***docker-compose***, драйверы и пакеты, содержащиеся в файле [requirements.txt](https://github.com/apache/superset/blob/master/requirements.txt) и [requirements-dev.txt](https://github.com/apache/superset/blob/master/requirements-dev.txt), будут установлены автоматически.

В этом разделе мы рассмотрим, как установить библиотеку коннекторов MySQL. Процесс установки библиотеки соединителя одинаков для всех дополнительных библиотек, и мы закончим этот раздел рекомендуемой библиотекой соединителя для каждой базы данных.

### Определите драйвер, который вам [нужен](https://superset.apache.org/docs/connecting-to-databases/docker-add-drivers#1-determine-the-driver-you-need)

Чтобы выяснить, как установить [драйвер базы данных](https://superset.apache.org/docs/connecting-to-databases/installing-database-drivers) по вашему выбору.

В этом примере мы рассмотрим процесс установки драйвера MySQL в RT.DataVision.

### Установите [драйвер](https://superset.apache.org/docs/connecting-to-databases/docker-add-drivers#2-install-mysql-driver)

Поскольку в настоящее время мы работаем внутри контейнера Docker через ***docker compose***, мы не можем просто работать ***pip install mysqlclient*** в нашей локальной оболочке и ожидать, что драйверы будут установлены в контейнерах Docker для RT.DataVision.

Чтобы решить эту проблему, установка RT.DataVision ***docker compose*** поставляется с механизмом для локальной установки пакетов, которые будут игнорироваться Git для целей локальной разработки. Пожалуйста, выполните следующие действия:

Создавать requirements-local.txt

# From the repo root...  
touch ./docker/requirements-local.txt

Добавьте драйвер, выбранный на шаге выше:

echo "mysqlclient" >> ./docker/requirements-local.txt

Восстановите свой локальный образ с новым драйвером:

docker-compose build --force-rm

После завершения перестроения образов Docker (что занимает несколько минут) вы можете перезапустить его с помощью следующей команды:

docker-compose up

Другой вариант – запустить RT.DataVision через Docker Compose, используя рецепт из ***docker-compose-non-dev.yml***, который будет использовать предварительно созданные интерфейсные ресурсы и пропустить сборку интерфейсных ресурсов:

docker-compose -f docker-compose-non-dev.yml pull  
docker-compose -f docker-compose-non-dev.yml up

### Подключиться к [MySQL](https://superset.apache.org/docs/connecting-to-databases/docker-add-drivers#3-connect-to-mysql)

Теперь, когда у вас локально установлен драйвер MySQL, вы сможете протестировать его.

Теперь мы можем создать источник данных в RT.DataVision, который можно использовать для подключения к экземпляру MySQL. Предполагая, что ваш экземпляр MySQL работает локально и к нему можно получить доступ через локальный хост, используйте следующую строку подключения в «SQL Alchemy URI», перейдя в «Данные > Базы данных > + База данных» (чтобы добавить новый источник данных) в RT.DataVision.

Для Docker, работающего в Linux:

mysql://mysqluser:mysqluserpassword@localhost/example?charset=utf8

Затем нажмите «Тестирование подключения», после чего должно появиться сообщение «ОК». Если нет, посмотрите в своем терминале сообщения об ошибках и обратитесь за помощью.

Вы можете повторить этот процесс для каждой базы данных, к которой вы хотите подключить надмножество.

## Использование пользовательского интерфейса подключения к базе данных

Вот документация о том, как использовать новый пользовательский интерфейс подключения к БД. Это даст администраторам возможность улучшить UX для пользователей, которые хотят подключаться к новым базам данных.

Теперь при подключении к базе данных в новом пользовательском интерфейсе есть 3 шага:

Шаг 1: Сначала администратор должен сообщить RT.DataVision, к какому движку они хотят подключиться. Эта страница поддерживается ***/available*** конечной точкой, которая использует механизмы, установленные в настоящее время в вашей среде, поэтому отображаются только поддерживаемые базы данных.

Шаг 2: Далее администратору будет предложено ввести параметры базы данных. В зависимости от того, доступна ли динамическая форма для этого конкретного механизма, администратор увидит либо новую пользовательскую форму, либо устаревшую форму SQLAlchemy. В настоящее время мы создали динамические формы для (Redshift, MySQL, Postgres и BigQuery). Новая форма запрашивает у пользователя параметры, необходимые для подключения (например, имя пользователя, пароль, хост, порт и т.д.), и обеспечивает немедленную обратную связь в случае возникновения ошибок.

Шаг 3: Наконец, после того, как администратор подключится к своей БД с помощью динамической формы, у него будет возможность обновить любые дополнительные настройки.

Мы надеемся, что эта функция поможет устранить огромные препятствия для пользователей, которые не могут войти в приложение и начать создавать наборы данных.

### Как настроить предпочтительные параметры базы данных и [изображения](https://superset.apache.org/docs/connecting-to-databases/db-connection-ui#how-to-setup-up-preferred-database-options-and-images)

Мы добавили новый параметр конфигурации, в котором администратор может определить свои предпочтительные базы данных, например:

*# Список предпочтительных БД*  
PREFERRED\_DATABASES: List[str] = [  
 "PostgreSQL",  
 "Presto",  
 "MySQL",  
 "SQLite",  
]

По причинам авторского права логотипы для каждой базы данных не распространяются вместе с RT.DataVision.

### Установка [изображений](https://superset.apache.org/docs/connecting-to-databases/db-connection-ui#setting-images)

Чтобы установить графические изображения предпочитаемых вами базы данных, администраторы должны создать сопоставление в ***superset\_text.yml*** файле с движком и расположением образа. Файлы с изображениями могут быть размещены локально внутри вашего статического/файлового каталога или онлайн (например, S3).

DB\_IMAGES:  
 postgresql: "path/to/image/postgres.jpg"  
 bigquery: "path/to/s3bucket/bigquery.jpg"  
 snowflake: "path/to/image/snowflake.jpg"

### Как сделать доступными новые механизмы баз данных

В настоящее время актуальная версия поддерживает следующие базы данных:

* Postgres,
* Redshift,
* MySQL,
* BigQuery.

Когда пользователь выбирает базу данных, которой нет в этом списке, он увидит старый диалог, запрашивающий URI SQLAlchemy. Новые базы данных можно добавлять постепенно в новый поток. Для поддержки расширенной конфигурации спецификация ядра БД должна иметь следующие атрибуты:

1. ***parameters\_schema:*** схема Marshmallow, определяющая параметры, необходимые для настройки базы данных. Для Postgres это включает имя пользователя, пароль, хост, порт и т. д.
2. ***default\_driver:*** имя рекомендуемого драйвера для спецификации механизма БД. Многие диалекты SQLAlchemy поддерживают несколько драйверов, но обычно один из них является официальной рекомендацией. Для Postgres мы используем «psycopg2».
3. ***sqlalchemy\_uri\_placeholder:*** строка, которая помогает пользователю в случае, если он хочет ввести URI напрямую.
4. ***encryption\_parameters:*** параметры, используемые для создания URI, когда пользователь выбирает зашифрованное соединение. Для Постгреса это ***{"sslmode": "require"}***.

Кроме того, спецификация механизма БД должна реализовывать следующие методы класса:

* ***build\_sqlalchemy\_uri(cls, parameters, encrypted\_extra):*** этот метод получает отдельные параметры и строит из них URI.
* ***get\_parameters\_from\_uri(cls, uri, encrypted\_extra):*** этот метод делает обратное, извлекая параметры из заданного URI.
* ***validate\_parameters(cls, parameters):*** этот метод используется для ***onBlur*** проверки формы. Он должен вернуть список с ***SupersetError*** указанием, какие параметры отсутствуют, а какие параметры явно неверны.

Для баз данных, таких как MySQL и Postgres, которые используют стандартный формат, ***engine+driver://user:password@host:port/dbname*** все, что вам нужно сделать, это добавить в ***BasicParametersMixin*** спецификацию механизма БД, а затем определить параметры 2-4 (***parameters\_schema*** уже присутствует в mixin).

Для других баз данных вам необходимо реализовать эти методы самостоятельно. Спецификация движка BigQuery DB – хороший пример того, как это сделать.

В следующих ниже подразделах представлено подключение к наиболее популярным БД.

### [MySQL](https://superset.apache.org/docs/connecting-to-databases/mysql#mysql)

Рекомендуемая библиотека коннекторов для MySQL: ***[mysqlclient](https://pypi.org/project/mysqlclient/***).

Строка подключения:

mysql://{username}:{password}@{host}/{database}

Host:

* Для Localhost или Docker под управлением Linux: ***localhostили127.0.0.1***.
* Для локальной сети: IP-адрес или имя хоста.
* Для Docker, работающего в OSX: ***docker.for.mac.host.internal*** Порт: ***3306*** – по умолчанию.

Одна из проблем ***mysqlclient*** заключается в том, что он не сможет подключиться к более новым базам данных MySQL, используя ***caching\_sha2\_password*** для аутентификации, поскольку плагин не включен в клиент. В этом случае вы должны использовать ***[mysql-connector-python](https://pypi.org/project/mysql-connector-python/***) вместо этого:

mysql+mysqlconnector://{username}:{password}@{host}/{database}

### Oracle

Рекомендуемая библиотека коннекторов – [cx\_Oracle](https://cx-oracle.readthedocs.io/en/latest/user_guide/installation.html).

Строка подключения имеет следующий формат:

oracle://<username>:<password>@<hostname>:<port>

### Postgres

Обратите внимание, что если вы используете docker-compose, библиотека коннектора Postgres [psycopg2](https://www.psycopg.org/docs/) поставляется вместе с RT.DataVision.

Пример параметров подключения Postgres:

* **User Name**: Имя пользователя.
* **Password**: DBPassword.
* **Database Host**:
  + Для Localhost: localhost или 127.0.0.1.
  + For On Prem: IP-адрес или имя хоста.
  + Для конечной точки AWS.
* **Database Name**: Имя базы данных.
* **Port**: по умолчанию 5432.

Строка подключения:

postgresql://{username}:{password}@{host}:{port}/{database}

Вы можете потребовать SSL, добавив ***?sslmode=require*** в конце:

postgresql://{username}:{password}@{host}:{port}/{database}?sslmode=require

Вы можете прочитать о других режимах SSL, которые поддерживает Postgres, в  [этой документации](https://www.postgresql.org/docs/9.1/libpq-ssl.html).

Дополнительную информацию о параметрах подключения к PostgreSQL можно найти в документации по [SQLAlchemy и документации](https://docs.sqlalchemy.org/en/13/dialects/postgresql.html#module-sqlalchemy.dialects.postgresql.psycopg2) по [PostgreSQL](https://www.postgresql.org/docs/9.1/libpq-connect.html#LIBPQ-PQCONNECTDBPARAMS).

### SQL- [сервер](https://superset.apache.org/docs/connecting-to-databases/sql-server#sql-server)

Рекомендуемая библиотека коннекторов для SQL Server – [pymssql](https://github.com/pymssql/pymssql).

Строка подключения для SQL Server:

mssql+pymssql://<Username>:<Password>@<Host>:<Port-default:1433>/<Database Name>/?Encrypt=yes

## Дополнительные [настройки](https://superset.apache.org/docs/connecting-to-databases/extra-settings#extra-database-settings) БД

### Более глубокая [интеграция](https://superset.apache.org/docs/connecting-to-databases/extra-settings#deeper-sqlalchemy-integration)

Можно настроить информацию о соединении с базой данных, используя параметры, предоставляемые SQLAlchemy. В представлении ***Редактировать базу данных*** вы можете редактировать **Экстра** поле как большой двоичный объект JSON.

Эта строка JSON содержит дополнительные элементы конфигурации. Объект ***engine\_params*** распаковывается в ***sqlalchemy.create\_engine***, а ***metadata\_params*** распаковывается в ***sqlalchemy.MetaData***. Дополнительные сведения см. в документации по SQLAlchemy.

### [Схемы](https://superset.apache.org/docs/connecting-to-databases/extra-settings#schemas)

Такие базы данных, как Postgres и Redshift, используют ***схему*** в качестве логической сущности поверх ***базы данных***. Чтобы RT.DataVision подключался к определенной схеме, можно задать параметр **схема** в форме «**Редактировать таблицы**» (Данные > Датасеты > Редактировать).

### Внешнее хранилище паролей для [соединений](https://superset.apache.org/docs/connecting-to-databases/extra-settings#external-password-store-for-sqlalchemy-connections)

RT.DataVision можно настроить для использования внешнего хранилища паролей базы данных. Это полезно, если вы используете собственную структуру хранения секретных данных и не хотите хранить их в метабазе данных RT.DataVision.

Пример. Напишите функцию, которая принимает один аргумент типа ***sqla.engine.url*** и возвращает пароль для данной строки подключения. Затем установите ***SQLALCHEMY\_CUSTOM\_PASSWORD\_STORE*** в файле конфигурации, чтобы он указывал на эту функцию.

def example\_lookup\_password(url):  
 secret = <<get password from external framework>>  
 return 'secret'  
  
SQLALCHEMY\_CUSTOM\_PASSWORD\_STORE = example\_lookup\_password

Распространенным шаблоном является использование переменных среды, чтобы сделать секретные данные доступными. Также может быть использован для этой цели ***SQLALCHEMY\_CUSTOM\_PASSWORD\_STORE***.

def example\_password\_as\_env\_var(url):  
*# assuming the uri looks like*  
*# mysql://localhost?superset\_user:{SUPERSET\_PASSWORD}*  
return url.password.format(os.environ)  
  
SQLALCHEMY\_CUSTOM\_PASSWORD\_STORE = example\_password\_as\_env\_var

### SSL-доступ к [базам данных](https://superset.apache.org/docs/connecting-to-databases/extra-settings#ssl-access-to-databases)

Вы можете использовать ***Экстра*** поле в форме «**Редактировать базу данных**» для настройки SSL:

{  
 "metadata\_params": {},  
 "engine\_params": {  
 "connect\_args":{  
 "sslmode":"require",  
 "sslrootcert": "/path/to/my/pem"  
 }  
 }  
}

# Безопасность

## [Роли](https://superset.apache.org/docs/security/#roles)

Безопасность в RT.DataVision обеспечивается Flask AppBuilder (FAB), платформой разработки приложений, созданной поверх Flask. FAB обеспечивает аутентификацию, управление пользователями, разрешения и роли. Пожалуйста, прочтите его [документацию по безопасности](https://flask-appbuilder.readthedocs.io/en/latest/security.html).

## Предоставленные [роли](https://superset.apache.org/docs/security/#provided-roles)

RT.DataVision поставляется с набором ролей, которые обрабатываются самим RT.DataVision. Вы можете предположить, что эти роли будут обновляться по мере развития RT.DataVision (и по мере того, как вы будете обновлять версии RT.DataVision).

Несмотря на то, что у пользователей с правами ***Admin*** есть такая возможность, мы не рекомендуем изменять разрешения, связанные с каждой ролью (например, удаляя или добавляя к ним разрешения). Разрешения, связанные с каждой ролью, будут повторно синхронизированы с исходными значениями при выполнении команды **superset init** (часто выполняется между версиями надмножества).

## Роль Admin

Администраторы имеют все возможные права, включая предоставление или отзыв прав других пользователей и изменение чужих фрагментов и информационных панелей.

## Роль Alpha

Альфа-пользователи имеют доступ ко всем источникам данных, но не могут предоставлять или отзывать доступ другим пользователям. Они также ограничены в изменении объектов, которыми они владеют. Пользователи Alpha могут добавлять и изменять источники данных.

## Роль Gamma

Пользователи Gamma имеют ограниченный доступ. Они могут потреблять только данные, поступающие из источников данных, к которым им предоставлен доступ через другую дополнительную роль. У них есть доступ только для просмотра срезов и информационных панелей, созданных из источников данных, к которым у них есть доступ. В настоящее время пользователи Gamma не могут изменять или добавлять источники данных. Мы предполагаем, что они в основном потребители контента, хотя могут создавать срезы и информационные панели.

Также обратите внимание, что, когда пользователи Gamma просматривают панели мониторинга и список срезов, они увидят только те объекты, к которым у них есть доступ.

## Роль [sql\_lab](https://superset.apache.org/docs/security/#sql_lab)

Роль **sql\_lab** предоставляет доступ к Лаборатории SQL. Обратите внимание, что, хотя пользователи с **Admin** по умолчанию имеют доступ ко всем базам данных, пользователям **Alpha** и **Gamma** необходимо предоставить доступ для каждой базы данных.

## Public

Чтобы разрешить пользователям, выходящим из системы, доступ к некоторым функциям RT.DataVision, вы можете использовать параметр ***PUBLIC\_ROLE\_LIKE*** конфигурации и назначить его другой роли, разрешения которой вы хотите передать этой роли.

Например, установив ***PUBLIC\_ROLE\_LIKE = Gamma*** в своем ***superset\_config.py*** файле, вы предоставляете роли **public** тот же набор разрешений, что и для роли **Gamma**. Это полезно, если вы хотите разрешить анонимным пользователям просматривать информационные панели. Явное предоставление для определенных наборов данных по-прежнему требуется, а это означает, что вам нужно отредактировать роль **Public** и добавить общедоступные источники данных в роль вручную.

## Управление доступом к источникам данных для Gamma [-ролей](https://superset.apache.org/docs/security/#managing-data-source-access-for-gamma-roles)

Вот как предоставить пользователям доступ только к определенным наборам данных. Сначала убедитесь, что пользователям с ограниченным доступом назначена [только] роль **Gamma**. Во-вторых, создайте новую роль (**Настройки -> Список ролей**) и нажмите знак «**+**».

Это новое окно позволяет вам дать этой новой роли имя, присвоить ее пользователям и выбрать таблицы в раскрывающемся списке «**Разрешения**». Чтобы выбрать источники данных, которые вы хотите связать с этой ролью, просто щелкните раскрывающийся список и используйте ввод для поиска имен таблиц.

Затем вы можете подтвердить с помощью пользователей, которым назначена роль **Gamma**, что они видят объекты (панели мониторинга и срезы), связанные с таблицами, которые вы только что расширили.

## Настройка [разрешений](https://superset.apache.org/docs/security/#customizing-permissions)

Разрешения, предоставляемые FAB, очень детализированы и обеспечивают высокий уровень настройки. FAB автоматически создает множество разрешений для каждой создаваемой модели (can\_add, can\_delete, can\_show, can\_edit, …), а также для каждого представления. Кроме того, RT.DataVision может предоставлять более детальные разрешения, такие как ***all\_datasource\_access***.

**Мы не рекомендуем изменять 3 базовые роли, так как существует набор предположений, на которых построен RT.DataVision**. Однако вы можете создавать свои собственные роли и объединять их с существующими.

## [Разрешения](https://superset.apache.org/docs/security/#permissions)

Роли состоят из набора разрешений, а RT.DataVision имеет много категорий разрешений. Вот различные категории разрешений:

* *Модель и действие:* модели – это объекты, такие как панель инструментов, фрагмент или пользователь. Каждая модель имеет фиксированный набор разрешений, примеры: ***can\_edit****,* ***can\_show****,* ***can\_delete****,* ***can\_list****,* ***can\_add*** и так далее. Например, вы можете разрешить пользователю удалять информационные панели, добавив ***can\_delete*** на сущность информационной панели к роли и предоставив этому пользователю эту роль.
* *Представления:* представления – это отдельные веб-страницы, такие как представление «Исследование» или представление «Лаборатория SQL». Предоставленный пользователю, он увидит это представление в пунктах меню и сможет загрузить эту страницу.
* *Источник данных:* для каждого источника данных создается разрешение. Если у пользователя нет ***all\_datasource\_access permission*** разрешения, он сможет только просматривать срезы или исследовать источники данных, которые ему предоставлены.
* *База данных:* Предоставление доступа к базе данных позволяет пользователю получить доступ ко всем источникам данных в этой базе данных и позволит пользователю запрашивать эту базу данных в Лаборатории SQL при условии, что пользователю предоставлено специальное разрешение Лаборатории SQL.

## Ограничение доступа к подмножеству [источников](https://superset.apache.org/docs/security/#restricting-access-to-a-subset-of-data-sources)

Мы рекомендуем предоставить пользователю роль **Gamma**, а также любые другие роли, которые добавят доступ к определенным источникам данных. Мы рекомендуем создавать отдельные роли для каждого профиля доступа. Например, пользователи финансовой группы могут иметь доступ к набору баз данных и источников данных; эти разрешения могут быть объединены в одной роли. Затем пользователям с этим профилем необходимо назначить роль «**Gamma**» в качестве основы для моделей и представлений, к которым они могут получить доступ, и роль «Финансы», которая представляет собой набор разрешений для объектов данных.

С пользователем может быть связано несколько ролей. Например, руководителю группы «Финансы» могут быть предоставлены роли «**Gamma**», «**Finance**» и «**Executive».** Роль **Executive** может предоставлять доступ к набору источников данных и информационных панелей, доступных только для руководителей. В представлении **Дашборды** пользователь может видеть только те, к которым у него есть доступ, в зависимости от назначенных ролей и разрешений.

## [Безопасность](https://superset.apache.org/docs/security/#row-level-security) на уровне строк

Используя фильтры безопасности на уровне строк (в меню «**Настройки**»), вы можете создавать фильтры, которые назначаются конкретной таблице, а также набору ролей. Если вы хотите, чтобы члены финансового отдела имели доступ только к строкам, где ***department = "finance"***, вы можете:

* Создайте фильтр безопасности на уровне строк с этим предложением (***department = "finance"***)
* Затем назначьте предложение роли «**Finance**» и таблице, к которой оно применяется.

Затем поле **clause**, которое может содержать произвольный текст, добавляется к предложению WHERE сгенерированного оператора SQL. Таким образом, вы даже можете сделать что-то вроде создания фильтра за последние 30 дней и применить его к определенной роли с таким пунктом, как ***date\_field > DATE\_SUB(NOW(), INTERVAL 30 DAY)***. Он также может поддерживать несколько условий: ***client\_id = 6*** и ***advertiser="foo"*** и т. д.

Все соответствующие фильтры безопасности на уровне строк будут объединены вместе (внутри различные предложения SQL объединяются с помощью операторов AND). Это означает, что можно создать ситуацию, когда две роли конфликтуют таким образом, что подмножество таблиц может быть пустым.

Например, фильтры ***client\_id=4*** и ***client\_id=5***, примененные к роли, приведут к тому, что пользователи этой роли будут добавлять ***client\_id=4*** и ***client\_id=5*** к своему запросу, что никогда не может быть правдой.

# Разное

## [Визуализация](https://superset.apache.org/docs/miscellaneous/country-map-tools#the-country-map-visualization) карты страны

Визуализация «Карта стран» позволяет вам строить облегченные картограммы стран по провинциям, штатам или другим типам объединений. Она не зависит от каких-либо сторонних картографических сервисов, но потребует от вас предоставить коды [ISO-3166-2](https://en.wikipedia.org/wiki/ISO_3166-2) подразделений верхнего уровня вашей страны. По сравнению с полными названиями провинций или штатов код ISO менее неоднозначен и уникален для всех регионов мира.

### Включенные [карты](https://superset.apache.org/docs/miscellaneous/country-map-tools#included-maps)

Визуализация «Карты стран» уже поставляется с картами следующих стран:

* Бельгия,
* Бразилия,
* Болгария,
* Канада,
* Китай,
* Египет,
* Франция,
* Германия,
* Индия,
* Иран,
* Италия,
* Япония,
* Корея,
* Лихтенштейн,
* Марокко,
* Мьянма,
* Нидерланды,
* Португалия,
* Россия,
* Сингапур,
* Испания,
* Швейцария,
* Сирия,
* Таиланд,
* Тиморлеште,
* Соединенное Королевство,
* Украина,
* Уругвай,
* США,
* Замбия.

### Добавление новой [страны](https://superset.apache.org/docs/miscellaneous/country-map-tools#adding-a-new-country)

Чтобы добавить новую страну в список, вам нужно отредактировать файлы в [@superset-ui/legacy-plugin-chart-country-map](https://github.com/apache-superset/superset-ui/tree/master/plugins/legacy-plugin-chart-country-map).

1. Создайте новый файл GeoJSON для страны, следуя инструкциям в [этом блокноте Jupyter](https://github.com/apache-superset/superset-ui/blob/master/plugins/legacy-plugin-chart-country-map/scripts/Country%20Map%20GeoJSON%20Generator.ipynb).
2. Отредактируйте список стран в [legacy-plugin-chart-country-map/src/countries.js](https://github.com/apache-superset/superset-ui/blob/master/plugins/legacy-plugin-chart-country-map/src/countries.js).
3. Отправьте эхо-запрос одному из коммиттеров RT.DataVision, чтобы ***@superset-ui/legacy-plugin-chart-country-map*** опубликовать пакет, или опубликуйте его под другим именем самостоятельно.
4. Обновите зависимости ***npm*** ***superset-frontend/package.json***, чтобы установить обновленный пакет плагинов.

## Импорт и экспорт [источников данных](https://superset.apache.org/docs/miscellaneous/importing-exporting-datasources#importing-and-exporting-datasources)

Расширенный ***cli*** позволяет импортировать и экспортировать источники данных из YAML и в него. Источники данных включают как базы данных, так и кластеры ***druid***. Предполагается, что данные будут организованы в следующей иерархии:

├──databases  
| ├──database\_1  
| | ├──table\_1  
| | | ├──columns  
| | | | ├──column\_1  
| | | | ├──column\_2  
| | | | └──... (more columns)  
| | | └──metrics  
| | | ├──metric\_1  
| | | ├──metric\_2  
| | | └──... (more metrics)  
| | └── ... (more tables)  
| └── ... (more databases)  
└──druid\_clusters  
 ├──cluster\_1  
 | ├──datasource\_1  
 | | ├──columns  
 | | | ├──column\_1  
 | | | ├──column\_2  
 | | | └──... (more columns)  
 | | └──metrics  
 | | ├──metric\_1  
 | | ├──metric\_2  
 | | └──... (more metrics)  
 | └── ... (more datasources)  
 └── ... (more clusters)

### Экспорт источников данных в [YAML](https://superset.apache.org/docs/miscellaneous/importing-exporting-datasources#exporting-datasources-to-yaml)

Вы можете вывести свои текущие источники данных на стандартный вывод, запустив:

superset export\_datasources

Чтобы сохранить источники данных в файл, выполните:

superset export\_datasources -f <filename>

По умолчанию значения по умолчанию (нулевые) будут опущены. Используйте флаг ***-d***, чтобы включить их. Если вы хотите, чтобы были включены обратные ссылки (например, столбец для включения идентификатора таблицы, к которой он принадлежит), используйте флаг ***-b***.

Кроме того, вы можете экспортировать источники данных с помощью пользовательского интерфейса:

1. Откройте «**Данные -> Базы данных»** для экспорта всех таблиц, связанных с одной или несколькими базами данных. («**Таблицы»** для одной или нескольких таблиц, «**Druid кластеры»** для кластеров, «**Druid источники данных»** для источников данных)
2. Выберите элементы, которые вы хотите экспортировать.
3. Нажмите «**Действия -> Экспорт» **.
4. Если вы хотите импортировать элемент, который вы экспортировали через пользовательский интерфейс, вам нужно будет вложить его в его родительский элемент, например, база данных должна быть вложена в базы данных, а таблица должна быть вложена в элемент базы данных.

Чтобы получить «**exhaustive list of all fields**», которые вы можете импортировать, используя запуск импорта YAML:

superset export\_datasource\_schema

Напоминаем, что вы можете использовать «–b» флаг для включения обратных ссылок.

### Импорт источников данных из [YAML](https://superset.apache.org/docs/miscellaneous/importing-exporting-datasources#importing-datasources-from-yaml)

Чтобы импортировать источники данных из файлов YAML, запустите:

superset import\_datasources -p <path or filename>

Если вы укажете путь, все файлы, заканчивающиеся на «**yaml»** или «**yml»**, будут проанализированы. Вы можете применить дополнительные флаги (например, для рекурсивного поиска по указанному пути):

superset import\_datasources -p <path> -r

Флаг синхронизации «***–s***» принимает параметры для синхронизации предоставленных элементов с вашим файлом. Будьте осторожны, это может удалить содержимое вашей метабазы данных. Пример:

superset import\_datasources -p <path / filename> -s columns,metrics

Это синхронизирует все показатели и столбцы для всех источников данных, найденных в ***<path /filename>*** метабазе данных RT.DataVision. Это означает, что столбцы и показатели, не указанные в YAML, будут удалены. Если вы добавите таблицы в столбцы, показатели также будут синхронизированы.

Если вы не укажете флаг синхронизации ***(-s***), импорт будет только добавлять и обновлять (переопределять) поля. Например, вы можете добавить «**verbose\_name**» в столбец «**ds в таблице random\_time\_series**» из примеров наборов данных, сохранив следующий YAML в файл и затем выполнив команду «**import\_datasources»**.

databases:  
- database\_name: main  
 tables:  
 - table\_name: random\_time\_series  
 columns:  
 - column\_name: ds  
 verbose\_name: datetime

# Подключение клиентских приложений к Dremio

Dremio предоставляет интерфейсы ODBC и JDBC, поддерживающие широкий спектр клиентских приложений.

1. Драйверы ODBC/JDBC.
2. Клиенты (в рамках данного ПО клиентом выступает RT.DataVision).

## Клиенты

Помимо RT.DataVision Dremio поддерживает широкий спектр клиентов, включая следующие приложения:

1. Alteryx Designer.
2. DataGrip.
3. DBeaver.
4. Dbt.
5. DbVisualizer.
6. Domo.
7. IBM Cognos Analytics.
8. Looker.
9. Microsoft Excel.
10. Microsoft Excel PowerPivot.
11. Microsoft Power BI Desktop, Microsoft Power BI Gateway, and Microsoft Power BI Service.
12. Microstrategy Workstation.
13. Python.
14. Qlik Sense Desktop.
15. SAP Business Objects.
16. Tableau.

### RT.DataVision

Используйте RT.DataVision для запроса и визуализации данных.

### Поддерживаемые версии​

1. RT.DataVision 1.5.3 и новее.
2. Коннектор Dremio SQLAlchemy 3.0.2 и более поздних версий.

### Поддерживаемые методы аутентификации​

1. Используйте имя пользователя и пароль учетной записи в вашем кластере Dremio.
2. Используйте имя пользователя учетной записи в вашем кластере Dremio и токен личного доступа (PAT), созданный в Dremio. После получения PAT рекомендуется закодировать его по URL-адресу. Чтобы закодировать его локально в вашей системе, вы можете выполнить следующие шаги:

* В окне браузера щелкните правой кнопкой мыши пустую область страницы и выберите **Просмотреть код**.
* В верхней панели панели проверки нажмите **Консоль**.
* Введите **encodeURIComponent("<PAT>")**, где **<PAT>** — это токен личного доступа. PAT в кодировке URL-адреса отображается красным цветом в следующей строке. Вы можете выделить его и скопировать в буфер обмена.

### Предварительные условия​

Если вы установили RT.DataVision согласно инструкции по установке с нуля, установите Dremio SQLAlchemy Connector в систему или на виртуальную машину, где работает RT.DataVision.

### Создание соединения​

1. Если вы используете версию RT.DataVision более раннюю, чем 2.1.0, выполните следующие действия:

* Выберите **Данные** > **Базы данных** в строке меню вверху экрана.
* Нажмите кнопку **База данных** в правом верхнем углу экрана.

1. Если вы используете версию 2.1.0 или более позднюю версию Superset, выполните следующие действия:

* Нажмите **Датасеты** в строке меню вверху экрана.
* Нажмите значок плюса (**+**) в правом верхнем углу.
* Выберите **Данные** > **Подключение базы данных**.

1. В диалоговом окне **Подключение базы данных** выполните следующие действия:

* Выберите **Другие** в поле **Поддерживаемые базы данных**.
* В поле **Отображение имени** укажите имя нового соединения.
* Если вы хотите пройти аутентификацию с использованием имени пользователя и пароля, укажите в поле **SQLAlchemy URI** в следующем формате URI:

dremio+flight://<имя пользователя>:<пароль>@<хост>:<порт>/<схема>[?option1=value[&,...]]

* **<имя пользователя>** ⎯ имя пользователя учетной записи Dremio, которую нужно использовать.
* **<пароль>** ⎯ пароль используемой учетной записи Dremio.
* **<хост>** ⎯ имя хоста или IP-адрес узла-координатора кластера Dremio.
* **<порт>** ⎯ порт для подключения на узле-координаторе. Если явно не изменено на узле, порт — 32010.
* **<схема>** ⎯ имя схемы базы данных, которая будет использоваться по умолчанию, если схема не указана в запросе. Предоставление схемы не является обязательным. Указание схемы не препятствует выполнению запросов для других схем; такие запросы должны явно включать схему.
* **[?option1=value[&,...]]** ⎯ одно или несколько необязательных свойств, разделенных амперсандами (**&**).

Пример URI с аутентификацией по имени пользователя и паролю

dremio+flight://myUserID:myPassword@myHost:32010/Samples?UseEncryption=false

* Если вы хотите пройти аутентификацию с использованием токена личного доступа, укажите в поле **SQLAlchemy URI** в следующем формате URI:

dremio+flight://<имя пользователя>:<PAT>@<хост>:<порт>/<схема>[?option1=value[&,...]]

* **<имя пользователя>** ⎯ имя пользователя учетной записи Dremio, которую нужно использовать.
* **<PAT>** ⎯ токен личного доступа в URL-адресе, полученный вами из Dremio Cloud.
* **<хост>** ⎯ имя хоста или IP-адрес узла-координатора кластера Dremio.
* **<порт>** ⎯ порт для подключения на узле-координаторе. Если явно не изменено на узле, порт — 32010.
* **<схема>** ⎯ имя схемы базы данных, которая будет использоваться по умолчанию, если схема не указана в запросе. Предоставление схемы не является обязательным. Указание схемы не препятствует выполнению запросов для других схем; такие запросы должны явно включать схему.
* **[?option=value[;...]]** ⎯ одно или несколько необязательных свойств, разделенных точкой с запятой.

Пример URI с аутентификацией PAT

dremio+flight://myUserID:myPAT@myHost:32010/Samples?UseEncryption=false

* Проверьте соединение. Если тест не пройден, проверьте синтаксис и значения в URI соединения.
* Нажмите **Подключить**.

### Свойства SSL-соединения

Используйте следующие свойства для настройки методов SSL-шифрования и проверки:

| **Имя** | **Тип** | **Описание** | **Значение по умолчанию** |
| --- | --- | --- | --- |
| **UseEncryption** | **integer** | Целое число **UseEncryption** заставляет клиента использовать соединение с шифрованием SSL для связи с Dremio. Принятые значения включают в себя:   * **True** ⎯ клиент взаимодействует с Dremio, используя SSL-шифрование. * **False** ⎯ клиент не взаимодействует с Dremio с помощью SSL-шифрования. | **true** |
| **disableCertificateVerification** | **integer** | **displayCertificateVerification** целое число. Указывает, следует ли проверять сертификат узла на соответствие хранилищу доверенных сертификатов. Принятые значения:   * **False** ⎯ проверяет сертификат по хранилищу доверенных сертификатов. * **True** ⎯ не проверяет сертификат по хранилищу доверенных сертификатов. | **false** |
| **trustedCerts** | **string** | **trustCerts** строка Полный путь к файлу .**pem**, содержащему сертификаты, которым доверяет центр сертификации, с целью проверки сервера. Если этот параметр не установлен, по умолчанию используется файл **.pem** сертификатов доверенного центра сертификации.  Соединение TLS завершается неудачно, если вы не укажете значение, если **UseEncryption** имеет значение **true**, а значение **DisableCertificateVerification** — **false**. | N**/A** |

### Расширенные свойства​

| **Имя** | **Тип** | **Описание** | **Значение по умолчанию** |
| --- | --- | --- | --- |
| **routing\_queue** | **string** | Указывает очередь, в которую будут маршрутизироваться запросы во время сеанса.  Прямая маршрутизация используется для указания точной очереди и кластера выполнения для выполнения запросов для данного сеанса. При прямой маршрутизации правила управления рабочей нагрузкой (WLM) не учитываются; вместо этого запросы направляются непосредственно в указанную очередь. | **N/A** |
| **routing\_tag** | **string** | Если это свойство установлено, указанный тег связан со всеми запросами, выполняемыми в рамках сеанса. Правила могут проверять наличие тега с помощью функции «tag()». | **N/A** |

## Драйвера

Dremio предоставляет драйвер JDBC и драйвер ODBC для Arrow Flight SQL:

1. ODBC.
2. JDBC.

### Драйвер JDBC для Arrow Flight SQL

Драйвер JDBC для Arrow Flight SQL — это драйвер с открытым исходным кодом, основанный на спецификациях API подключения к базе данных Java (JDBC). Однако драйвер JDBC Flight использует Apache Arrow, поэтому он может перемещать большие объемы данных быстрее, отчасти потому, что ему не нужно сериализовать, а затем десериализовать данные.

Этот драйвер решает проблему, которая является общей для многих инструментов бизнес-аналитики, обращающихся к базам данных через JDBC. Эти инструменты включают в себя разные драйверы JDBC для каждого типа поддерживаемых ими баз данных, поскольку каждая из этих баз данных имеет свой собственный драйвер. Объединение нескольких драйверов JDBC для нескольких баз данных может оказаться затруднительным в обслуживании, а решение проблем поддержки нескольких драйверов может оказаться дорогостоящим. Теперь, если в базе данных включена конечная точка Apache Arrow Flight SQL, драйвер JDBC для Arrow Flight SQL может подключиться к ней.

**Примечание**. Планирование запроса выполняется на указанном узле. Чтобы распределить планирование запросов для соединений JDBC, настройте вторичные узлы координатора для вашего развертывания.

### Предварительные условия для использования драйвера JDBC для Arrow Flight SQL​

1. Java 1.8 или новее.
2. Одна из следующих операционных систем: Windows, MacOS или Linux.

### Поддерживаемые методы аутентификации​

1. Используйте имя пользователя и пароль учетной записи Dremio, к которой вы хотите подключиться.
2. Используйте имя пользователя и токен личного доступа (PAT).

### Загрузка драйвера JDBC для Arrow Flight SQL​

Скачать драйвер можно [здесь](https://www.dremio.com/drivers/jdbc/).

### Интеграция драйвера JDBC для Arrow Flight SQL​

Чтобы интегрировать драйвер в вашу среду разработки, добавьте его в свой путь к классам.

### Название класса​

Имя класса — **org.apache.arrow.driver.jdbc.ArrowFlightJdbcDriver**.

### Подключение к базам данных​

1. Используйте этот шаблон для создания прямого подключения к базе данных, в которой включена конечная точка Apache Arrow Flight SQL:

Создать прямое соединение с базой данных

jdbc:arrow-flight-sql://<hostname-or-IP-address>:<port-number>/?useEncryption=false[&schema=<optional\_schema>][&<properties>]

* **<optional\_schema>** — имя схемы (источник данных или пространство, включая дочерние пути, например myDatasource.folder1 и mySpace.folder1.folder2), которое будет использоваться по умолчанию, если схема не указана в запросе.
* **<properties>** — список свойств JDBC. Значения должны быть закодированы в URI.

1. Используйте этот шаблон для создания прямого подключения к узлу координатора Dremio:

Создайте прямое соединение с узлом координатора Dremio.

jdbc:arrow-flight-sql://<Dremio\_coordinator>:32010[/?schema=<optional\_schema>][&<properties>]

* **<Dremio\_coordinator>** — имя хоста или IP-адрес узла-координатора в вашем кластере Dremio.
* **<optional\_schema>** — имя схемы (источник данных или пространство, включая дочерние пути, например myDatasource.folder1 и mySpace.folder1.folder2), которое будет использоваться по умолчанию, если схема не указана в запросе.
* **<properties>** — список свойств JDBC. Значения должны быть закодированы в URL.

Для аутентификации в Dremio используйте любой из этих методов:

1. Передайте имя пользователя и пароль со свойствами пользователя и пароля.
2. Передайте имя пользователя и токен личного доступа (PAT) со свойствами пользователя и пароля. Используйте PAT в качестве значения свойства пароля. Вы должны URL-кодировать PAT, которые вы включаете в URL-адреса JDBC.

### Драйвер ODBC для Arrow Flight SQL

Начиная с Dremio v22.0, вы можете использовать драйвер ODBC для Arrow Flight SQL для подключения к Dremio из клиентских приложений ODBC. Этот драйвер распространяется под лицензией GNU Library General Public License, версия 2.

### Поддерживаемые операционные системы​

Вы можете использовать драйвер в системах, работающих под управлением следующих 64-разрядных операционных систем:

1. Linux: RedHat/CentOS.
2. Windows 10 и более поздние версии.
3. macOS.

**Примечание**. macOS на архитектуре M1 не поддерживается изначально. Чтобы запустить приложение, установленное в macOS для M1 и настроенное на использование этого драйвера, запустите эту команду в окне терминала, где <приложение> — это путь к приложению:

arch -x86\_64 <application>

### Поддерживаемые методы аутентификации​

1. Используйте имя пользователя и пароль учетной записи Dremio, к которой вы хотите подключиться.
2. Используйте имя пользователя и токен личного доступа (PAT).

### Загрузка и установка в Windows​

**Примечание**. Драйвер ODBC Arrow Flight SQL недоступен для 32-разрядных версий Windows.

**Примечание**. Если вы планируете использовать Microsoft Power BI Desktop от апреля 2022 г. или более поздней версии для подключения к Dremio, вам не нужно использовать этот драйвер. Power BI Desktop, апрель 2022 г. и более поздние версии, включает коннектор, который можно использовать для подключения к Dremio.

Чтобы загрузить и установить драйвер ODBC для Arrow Flight SQL:

1. Загрузите 64-разрядную версию драйвера для Windows со [страницы загрузки драйвера ODBC](https://www.dremio.com/drivers/odbc/).
2. Запустите установщик.
3. (Необязательно) На странице **Управление аккаунтом пользователя** нажмите **Да**. Эта страница отображается только в том случае, если на вашем компьютере с Windows настроен контроль учетных записей пользователей.
4. На странице **Добро пожаловать в Dremio** нажмите **Далее**.
5. Нажмите **Установить**.
6. На странице **Установка завершена** нажмите **Далее**.
7. На странице **Завершение установки драйвера ODBC Arrow Flight SQL** нажмите **Закончить**.

### Загрузка и установка в Linux​

Чтобы загрузить и установить драйвер ODBC для Arrow Flight SQL:

1. Загрузите версию драйвера для Linux со [страницы загрузки драйвера ODBC](https://www.dremio.com/drivers/odbc/).
2. Выполните следующую команду, чтобы установить драйвер и автоматически создать имя источника данных (DSN) **Arrow Flight SQL ODBC DSN:**

sudo yum localinstall <dremio-odbc-rpm-path>

### Загрузка и установка на macOS​

Чтобы загрузить и установить драйвер ODBC для Arrow Flight SQL:

1. Загрузите версию драйвера для macOS со [страницы загрузки драйвера ODBC](https://www.dremio.com/drivers/odbc/).
2. Перейдите в папку загрузки и дважды щелкните загруженный файл .dmg.
3. Дважды щелкните файл .pkg.
4. На странице **Добро пожаловать в установку драйвера ODBC Arrow Flight SQL** нажмите **Продолжить**.
5. На странице **Стандартная установка "Macintosh HD"** нажмите **Установить**. При желании, если вы хотите изменить место установки, нажмите **Изменить место установки** и перейдите к новому местоположению.
6. В диалоговом окне **Установщик пытается установить новое ПО** укажите пароль macOS. Затем нажмите **Установить ПО**.
7. После завершения установки нажмите **Закрыть**.

# Интерфейс командной строки администратора в Dremio

В этом разделе кратко описаны команды **CLI dremio-admin**.

## Синтаксис​

dremio-admin [--config <conf-dir>] (encrypt|set-password|upgrade|recommend-reflections|delete-user-homespace|optimize-acls|export-profiles|remove-duplicate-users|clean|remove-duplicate-roles|reset-catalog-search|backup|delete-all-users|restore|repair-acls|nessie-maintenance) [args...]

## Параметры​

|  |  |
| --- | --- |
| **Параметр** | **Описание** |
| **--help, -h** | Отображает информацию об использовании команд CLI. |
| **--config <conf-dir>** | Используется, когда расположение файла конфигурации отличается от каталога по умолчанию **/opt/dremio/conf**.  Например, если **dremio.conf**, **dremio-env**, **logbook.xml** и **logbook-admin.xml** расположены в **/etc/dremio**, вам придется запускать все команды **dremio-admin** как **./dremio-admin --config /etc/dremio <command> <arguments>**. |

## Команды​

| **Команда** | **Описание** |
| --- | --- |
| **backup** | Создает резервные копии метаданных Dremio и файлов, загруженных пользователем. |
| **clean** | Очищает метаданные Dremio. |
| **delete-all-users** | Удаляет всех внутренних пользователей Dremio. |
| **delete-user-homespace** | Удаляет домашнее пространство пользователя. |
| **encrypt** | Зашифровать строку, введенную пользователем. |
| **export-profiles** | Экспортирует профили вакансий из Dremio. |
| **nessie-maintenance** | Запускает встроенные задачи обслуживания репозитория Nessie |
| **optimize-acls** | Оптимизирует списки управления доступом к источникам, пространствам и наборам данных. |
| **recommend-reflections** | Рекомендовать отражения. |
| **remove-duplicate-roles** | Удаляет повторяющиеся роли из Dremio. |
| **remove-duplicate-users** | Удаляет повторяющихся пользователей из Dremio. |
| **reset-catalog-search** | Сбрасывает индекс для восстановления поиска по каталогу. |
| **repair-acls** | Восстанавливает списки управления доступом к источникам, пространствам и наборам данных. |
| **restore** | Восстанавливает метаданные Dremio и файлы, загруженные пользователем. |
| **set-password** | Устанавливает пароли для пользователей Dremio (не LDAP). |
| **upgrade** | Обновляет версию магазина KV. Для этой команды нет доступных опций. |

## Каталог журналов​

Значение по умолчанию для **DREMIOADMIN\_LOG\_DIR** равно нулю/не установлено. Если для этого параметра не установлено значение, файлы журналов не создаются.

Чтобы установить каталог журнала, укажите путь к каталогу журнала, выполнив следующую команду:

export DREMIO\_ADMIN\_LOG\_DIR=<path>

**Примечание**. Опция экспорта должна быть установлена, и доступ должен быть доступен для пользователя, выполняющего команду dremio-admin.

## Детализация журнала​

Детализация журнала используется вместе с **DREMIO\_ADMIN\_LOG\_DIR**. В противном случае весь вывод выводится на **stdout**; нет контроля над настройкой детализации для **stdout**.

Значения уровня детализации включают в себя:

* TRACE
* DEBUG
* INFO (default)
* WARN
* ERROR

Чтобы установить уровень детализации журнала (по умолчанию: **INFO**), укажите уровень детализации, выполнив следующую команду:

export DREMIO\_ADMIN\_LOG\_VERBOSITY=<value>

# Управление нагрузками заданий

## Управление очередью

### Классы работы​

В Dremio есть 3 статических класса заданий для разделения ЦП. Вот они в порядке убывания приоритета:

1. **Near-realtime** — этот класс включает в себя все задания предварительного просмотра пользовательского интерфейса. Это сделано для обеспечения быстрого реагирования пользователя на все операции в пользовательском интерфейсе Dremio.
2. **Interactive** — этот класс предназначен для всех пользовательских заданий, включая те, которые отправляются через ODBC и JDBC, а также для полных запусков в пользовательском интерфейсе.
3. **Background** — этот класс включает в себя все задания по созданию и обслуживанию Reflection. Эти задания имеют самый низкий приоритет, чтобы обеспечить более высокую производительность заданий, отправленных пользователем.

### Управление очередью запросов

Dremio поддерживает размещение пользовательских заданий в две статические очереди заданий (большую и малую) на основе настраиваемого порога стоимости запроса.

Максимальное количество одновременных запросов и время ожидания очереди можно установить для каждой очереди в разделе **Администрирование > Кластер > Управление очередью > Управление очередью запросов**. Мы рекомендуем настроить параллелизм большой очереди запросов ниже, чем у маленькой очереди запросов. Это гарантирует, что в любой момент времени выполняется только несколько крупных заданий и позволяет выполнить несколько более мелких запросов.

Тайм-аут очереди можно настроить, по умолчанию оно составляет пять (5) минут.

### Управление очередью отражений​

Dremio поддерживает размещение заданий отражения в две статические очереди заданий (большую и малую) в разделе **Администрирование > Кластер > Управление очередью > Управление очередью отражений**.

Это гарантирует, что в любой момент времени выполняются только минимальные большие задания отражения и допускается выполнение нескольких более мелких запросов.

Тайм-аут очереди отражения можно настроить по умолчанию на один (1) день.

### Управление памятью запросов​

Максимальный объем памяти на запрос также можно настроить для каждого типа задания в разделе **Администрирование > Кластер > Управление очередью > Управление памятью запросов**.

### Порог запроса​

Порог запроса устанавливается путем анализа общей стоимости различных рабочих нагрузок и установки порогового значения на основе ожидаемых рабочих нагрузок, соглашений об уровне обслуживания и доступных ресурсов кластера. Пороговое значение по умолчанию — 30 000 000 и представляет собой большое пороговое значение запроса или стоимость плана.

Порог запроса можно настроить в разделе **Администрирование > Кластер > Управление очередью > Порог запроса.**

Стоимость запроса для конкретного задания можно найти в разделе **Задания > Профиль > Аллокация ресурса > Стоимость запроса**.

**Примечание**. Общая стоимость запроса, используемая для размещения заданий в небольших и больших очередях, вычисляется путем суммирования значений числа строк для всех операторов (каждой строки) в физическом плане для данного задания.

## История работы и сведения о работе

Новый пользовательский интерфейс: если вы используете Dremio 18.0+, вы можете заменить существующие экраны заданий обновленным и более интерактивным интерфейсом.

Переход на страницу **История заданий** позволяет вам просмотреть все ранее выполненные и текущие запросы по мере их выполнения Dremio. В таблице слева вакансии перечислены в хронологическом порядке, а столбцы содержат минимальную информацию о самой работе. Нажатие на отдельную строку развернет это задание в правой части экрана на панели **Детали заданий**. В этом разделе экрана представлена подробная разбивка запроса и его результатов.

### Новый интерфейс заданий​

**Примечание**. Требование к версии: эти новые экраны доступны только в экземплярах, использующих Dremio v18.0+. Экраны История заданий и Детали заданий заменены на более интерактивный и наглядный макет.

Страница **История заданий** действует как динамичный и удобный интерфейс, с помощью которого вы можете просматривать все ранее выполненные запросы и углубляться в детали задания по каждому запросу.

### Включение нового пользовательского интерфейса​

**Примечание**. Эта функция включена по умолчанию в Dremio v19.0+, и действия, описанные ниже, выполнять не нужно.

По умолчанию этот новый макет отключен, и его необходимо активировать вручную. Чтобы включить новый пользовательский интерфейс и заменить существующие страницы **История заданий** или **Детали заданий**, используйте ключ поддержки **dremio.jobs.new.ui**.

Это будет автоматически включено в будущей версии Dremio.

### Взаимодействие с заданием

Термин «задание» (job) используется для описания отдельного запроса, выполняемого с помощью Dremio. Таким образом, на этой странице отображается список всех ранее выполненных заданий за предыдущие XX дней. Порядок этих заданий можно регулировать с помощью данных столбца в порядке возрастания или убывания.

1. Чтобы изменить столбцы для сортировки заданий по возрастанию или убыванию, щелкните имя нужного столбца в верхней части таблицы **Просмотр запросов**. Это изменит сортировку таблицы с восходящей на нисходящую или наоборот.
2. Чтобы отобразить или скрыть отдельные столбцы в **Просмотр запросов**, нажмите кнопку **Столбцы** в правом верхнем углу экрана рядом с панелью поиска. Затем установите флажок слева от имени каждого столбца, который вы хотите просмотреть или удалить. Изменения в визуализаторе вносятся немедленно при каждом выборе.
3. Чтобы изменить порядок отдельных столбцов, нажмите кнопку **Столбцы**. Затем выберите нужную стрелку справа от имени столбца, чтобы изменить его порядок. Стрелка, направленная вверх, переместит столбец дальше влево на экране документ-камеры. Выбор стрелки, указывающей вниз, переместит столбец вправо.
4. Чтобы просмотреть подробную информацию об отдельном столбце задания, наведите указатель мыши на нужную строку и столбец. Это немедленно вызовет наведение на экран с краткими сведениями об этом задании, такими как используемый SQL-запрос, список запрошенных наборов данных или разбивка процесса выполнения запроса.
5. Чтобы открыть экран сведений о задании для отдельного запроса, щелкните нужную строку задания в таблице визуализатора запросов.

### Визуализатор плана запроса​

Визуализатор плана запроса предлагает графическое представление дерева выполнения SQL, эффективно позволяя пользователям видеть все ресурсы, используемые для отдельного запроса, включая узлы, отражения и отдельные наборы данных. Те узлы, которые считаются наиболее дорогими, то есть требующими много времени, будут выглядеть более красочными, тогда как менее используемые ресурсы будут отображаться затемненными.

Чтобы получить доступ к этому ресурсу:

1. На экране **Задания** выберите желаемое завершенное задание. Откроется страница **Детали заданий.**
2. Нажмите вкладку **SQL** в верхней части экрана.
3. Прокрутите вниз до раздела **Граф запроса**. Здесь находится визуальное представление плана запроса.

Приложение 1. Подключения к базам данных

| ***База данных*** | | ***Пакет PyPI*** | ***Строка подключения*** |
| --- | --- | --- | --- |
| [Amazon Athena](https://superset.apache.org/docs/connecting-to-databases/athena) | pip install "PyAthenaJDBC>1.0.9 , pip install "PyAthena>1.2.0 | | awsathena+rest://{aws\_access\_key\_id}:{aws\_secret\_access\_key}@athena.{region\_name}.amazonaws.com/{ |
| [Amazon Redshift](https://superset.apache.org/docs/connecting-to-databases/redshift) | pip install sqlalchemy-redshift | | redshift+psycopg2://<userName>:<DBPassword>@<AWS End Point>:5439/<Database Name> |
| [Apache Drill](https://superset.apache.org/docs/connecting-to-databases/drill) | pip install sqlalchemy-drill | | drill+sadrill:// For JDBC drill+jdbc:// |
| [Apache Druid](https://superset.apache.org/docs/connecting-to-databases/druid) | pip install pydruid | | druid://<User>:<password>@<Host>:<Port-default-9088>/druid/v2/sql |
| [Apache Hive](https://superset.apache.org/docs/connecting-to-databases/hive) | pip install pyhive | | hive://hive@{hostname}:{port}/{database} |
| [Apache Impala](https://superset.apache.org/docs/connecting-to-databases/impala) | pip install impyla | | impala://{hostname}:{port}/{database} |
| [Apache Kylin](https://superset.apache.org/docs/connecting-to-databases/kylin) | pip install kylinpy | | kylin://<username>:<password>@<hostname>:<port>/<project>?<param1>=<value1>&<param2>=<value2> |
| [Apache Pinot](https://superset.apache.org/docs/connecting-to-databases/pinot) | pip install pinotdb | | pinot://BROKER:5436/query?server=http://CONTROLLER:5983/ |
| [Apache Solr](https://superset.apache.org/docs/connecting-to-databases/solr) | pip install sqlalchemy-solr | | solr://{username}:{password}@{hostname}:{port}/{server\_path}/{collection} |
| [Apache Spark SQL](https://superset.apache.org/docs/connecting-to-databases/spark-sql) | pip install pyhive | | hive://hive@{hostname}:{port}/{database} |
| [Ascend.io](https://superset.apache.org/docs/connecting-to-databases/ascend) | pip install impyla | | ascend://{username}:{password}@{hostname}:{port}/{database}?auth\_mechanism=PLAIN;use\_ssl=true |
| [Azure MS SQL](https://superset.apache.org/docs/connecting-to-databases/sql-server) | pip install pymssql | | mssql+pymssql://UserName@presetSQL:TestPassword@presetSQL.database.windows.net:1433/TestSchema |
| [Big Query](https://superset.apache.org/docs/connecting-to-databases/bigquery) | pip install pybigquery | | bigquery://{project\_id} |
| [ClickHouse](https://superset.apache.org/docs/connecting-to-databases/clickhouse) | pip install clickhouse-driver==0.2.0 && pip install clickhouse-sqlalchemy==0.1.6 | | clickhouse+native://{username}:{password}@{hostname}:{port}/{database} |
| [CockroachDB](https://superset.apache.org/docs/connecting-to-databases/cockroachdb) | pip install cockroachdb | | cockroachdb://root@{hostname}:{port}/{database}?sslmode=disable |
| [Dremio](https://superset.apache.org/docs/connecting-to-databases/dremio) | pip install sqlalchemy\_dremio | | dremio://user:pwd@host:31010/ |
| [Elasticsearch](https://superset.apache.org/docs/connecting-to-databases/elasticsearch) | pip install elasticsearch-dbapi | | elasticsearch+http://{user}:{password}@{host}:9200/ |
| [Exasol](https://superset.apache.org/docs/connecting-to-databases/exasol) | pip install sqlalchemy-exasol | | exa+pyodbc://{username}:{password}@{hostname}:{port}/my\_schema?CONNECTIONLCALL=en\_US.UTF-8&driver=EXAODBC |
| [Google Sheets](https://superset.apache.org/docs/connecting-to-databases/google-sheets) | pip install shillelagh[gsheetsapi] | | gsheets:// |
| [Firebolt](https://superset.apache.org/docs/connecting-to-databases/firebolt) | pip install firebolt-sqlalchemy | | firebolt://{username}:{password}@{database} or firebolt://{username}:{password}@{database}/{engine\_name} |
| [Hologres](https://superset.apache.org/docs/connecting-to-databases/hologres) | pip install psycopg2 | | postgresql+psycopg2://<UserName>:<DBPassword>@<Database Host>/<Database Name> |
| [IBM Db2](https://superset.apache.org/docs/connecting-to-databases/ibm-db2) | pip install ibm\_db\_sa | | db2+ibm\_db:// |
| [IBM Netezza Performance Server](https://superset.apache.org/docs/connecting-to-databases/netezza) | pip install nzalchemy | | netezza+nzpy://<UserName>:<DBPassword>@<Database Host>/<Database Name> |
| [MySQL](https://superset.apache.org/docs/connecting-to-databases/mysql) | pip install mysqlclient | | mysql://<UserName>:<DBPassword>@<Database Host>/<Database Name> |
| [Oracle](https://superset.apache.org/docs/connecting-to-databases/oracle) | pip install cx\_Oracle | | oracle:// |
| [PostgreSQL](https://superset.apache.org/docs/connecting-to-databases/postgres) | pip install psycopg2 | | postgresql://<UserName>:<DBPassword>@<Database Host>/<Database Name> |
| [Trino](https://superset.apache.org/docs/connecting-to-databases/trino) | pip install sqlalchemy-trino | | trino://{username}:{password}@{hostname}:{port}/{catalog} |
| [Presto](https://superset.apache.org/docs/connecting-to-databases/presto) | pip install pyhive | | presto:// |
| [SAP Hana](https://superset.apache.org/docs/connecting-to-databases/hana) | pip install hdbcli sqlalchemy-hana or pip install apache-superset[hana] | | hana://{username}:{password}@{host}:{port} |
| [Snowflake](https://superset.apache.org/docs/connecting-to-databases/snowflake) | pip install snowflake-sqlalchemy | | snowflake://{user}:{password}@{account}.{region}/{database}?role={role}&warehouse={warehouse} |
| SQLite |  | | sqlite:// |
| [SQL Server](https://superset.apache.org/docs/connecting-to-databases/sql-server) | pip install pymssql | | mssql:// |
| [Teradata](https://superset.apache.org/docs/connecting-to-databases/teradata) | pip install sqlalchemy-teradata | | teradata://{user}:{password}@{host} |
| [Vertica](https://superset.apache.org/docs/connecting-to-databases/vertica) | pip install sqlalchemy-vertica-python | | vertica+vertica\_python://<UserName>:<DBPassword>@<Database Host>/<Database Name> |