Geekbrains

**Использование свободного программного обеспечения с открытым исходным кодом для построения ETL процесса и задач бизнес аналитики**

Программа: Разработчик

Специализация: Data Engineer

Пакулин А.В.

Екатеринбург

2024

**Содержание**

[**Введение 3**](#_heading=h.1fob9te)

[**Глава 1. Свободное программное обеспечение с открытым исходным кодом 5**](#_heading=h.es7xkzgar42k)

[**Глава 2. Основные этапы ETL процесса и построения дашборда 9**](#_heading=h.4tdonkxsmvm3)

[**Глава 3. Свободное программное обеспечение с открытым исходным кодом, используемое для реализации ETL процессов и построения дашбордов 13**](#_heading=h.q90gvz2t80u7)

[**Глава 4. Определение целей ETL процесса и построения дашборда, реализованного с помощью программного обеспечения с открытым исходным кодом 19**](#_heading=h.xkmyyjqwqxzn)

[**Глава 5. Действия, выполненные для достижения целей ETL процесса и построения дашборда 20**](#_heading=h.k50bkinapbma)

[**Глава 6. Запуск ETL процесса и получение результатов 25**](#_heading=h.a1inz3cxths1)

[**Заключение 29**](#_heading=h.dhwmmf3wq1kk)

[**Список используемой литературы 30**](#_heading=h.m8jvbwhv7ujg)

[**Приложения 34**](#_heading=h.s6k7t2lwcp0k)

# Введение

**Тема проекта:** Исследование возможности построения ETL процесса и реализации дашборда с использованием свободного программного обеспечения с открытым исходным кодом.

**Цель:** Изучить процесс построения ETL процесса и реализации дашборда исключительно с использованием свободного программного обеспечения и определить какое свободное программное обеспечение с открытым исходным кодом может быть использовано для этого.

**Какую проблему решает:** В условиях, когда приобретение и получение поддержки проприетарного программного обеспечения становится невозможным или непредсказуемым с точки зрения непрерывности, использование свободного программного обеспечения с открытым исходным кодом остается одной из возможных альтернатив обеспечить непрерывность процессов в информационных системах и ETL процессов в частности.

**Задачи:**

1. Изучить литературу, касающуюся темы исследования.
2. Рассмотреть основные этапы ETL процесса.
3. Рассмотреть основные этапы реализации дашборда.
4. Определить свободное программное обеспечение с открытым исходным кодом, которое может быть использовано для использовании в ETL процессе и реализации дашбордов.
5. Реализовать ETL процесс и построить дашборд с использованием свободного программного обеспечения с открытым исходным кодом.
6. Сделать выводы о возможности реализации ETL процесса и построения дашборда с использованием свободного программного обеспечения с открытым исходным кодом.

**Инструменты:** Apache Airflow, Python, Hadoop HDFS, Hadoop Hive, Hadoop HUE, Apache Superset, Docker, Linux, VS Code

**Состав команды**: Пакулин Александр Владимирович (Разработчик)

# Глава 1. Свободное программное обеспечение с открытым исходным кодом

Программное обеспечение (ПО) — программа или множество программ, используемых для управления компьютером .

Программное обеспечение является одним из видов обеспечения вычислительной системы, наряду с техническим (аппаратным), математическим, информационным, лингвистическим, организационным, методическим и правовым обеспечением[5]

Программное обеспечение можно разделить на две большие категории по типу лицензии под которой распространяется данное программное обеспечение:

* свободное программное обеспечение;
* несвободное программное обеспечение.

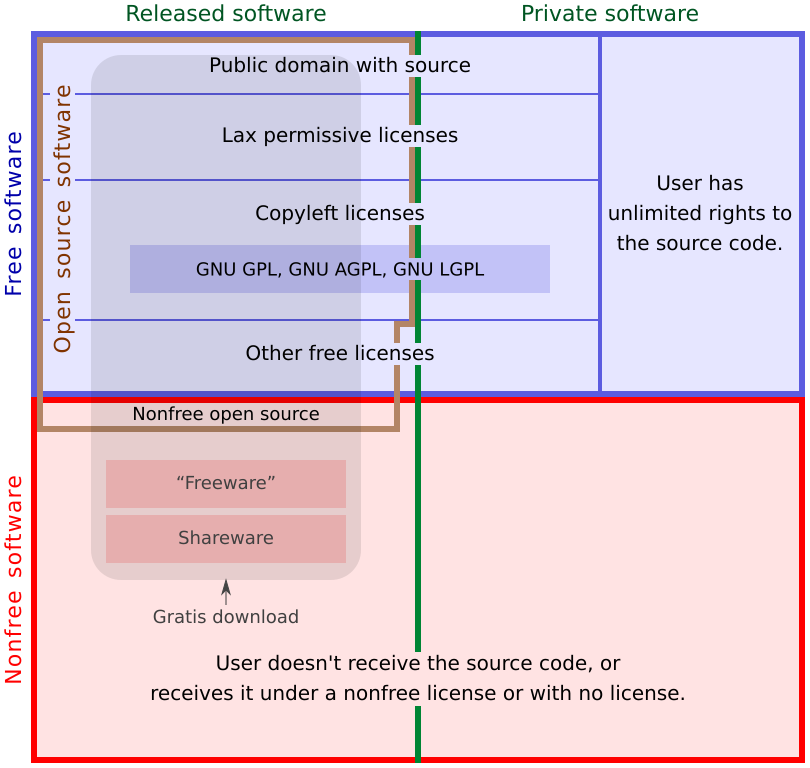


Рисунок 1. Классификация программного обеспечения по типам лицензий. [12]

К свободному программному обеспечению относится программное обеспечение, лицензия на которое позволяет пользователю:

* запускать программу в любых целях;
* изучать работу программы и адаптировать её к своим нуждам (условие - доступ к исходному коду);
* создавать и распространять копии программы;
* улучшать программу и публиковать улучшения.

По существующему законодательству, программный продукт и его исходный код по умолчанию охраняется авторским правом, дающее правообладателю, которым чаще всего является разработчик или его работодатель, полную власть над распространением и изменением программы, даже в случае, когда исходный код общедоступен. Чтобы ПО стало «свободным», правообладатель должен выпустить исходный код программного обеспечения под одной из особого рода лицензий, называемых свободными лицензиями, наделющего пользователя вышеуказанными правами.

Среди свободных лицензий выделяют:

* пермиссивные (BSD, MIT, Apache);
* копилефтные строгие (GNU GPL);
* копилефтные слабые (LGPL, MPL);
* лицензии на передачу кода в общественное достояние (the Unlicense)[ 17].

С более подробной информацией о разновидностях лицензий можно ознакомиться здесь [ 14]. Информация, приведенная в данном источнике будет в дальнейшем использована для отнесения программного обеспечения к категории свободного программного обеспечения.

В данной работе применяется формулировка “Свободное программное обеспечение с открытым исходным кодом”. Разберемся почему почему формулировки содержащей только “Свободное программное обеспечение” недостаточно.

В дополнение к классификации по типу лицензии, программное обеспечение можно разделить на две большие категории по степени открытости исходного кода:

* программного обеспечение с открытым исходным кодом;
* программное обеспечение с закрытым исходным кодом.

Почти все программное обеспечение с открытым исходным кодом является свободным программным обеспечением, но есть исключения.

Во-первых, некоторые лицензии с открытым исходным кодом слишком ограничительны, поэтому они не квалифицируются как свободные лицензии.

Во-вторых, требования к товарному знаку, добавленные поверх лицензии на авторское право кода, могут сделать программу несвободной [ 28]

Свободное и несвободное программное обеспечение имеет свои преимущества и недостатки и именно поэтому эти категории сосуществуют уже длительное время параллельно.

Однако с момента зарождения свободного программного обеспечения в университетской и научной среде, свободное программное обеспечение проникло к корпоративный сегмент и медленно и уверенно его завоевывает. Согласно данным приведенным в [ 29], в 2022 г. доля свободного программного обеспечения с открытым исходным кодом составила 29% с прогнозом увеличения за два следующих года до 34%.

Согласно тому же источнику причины по которым корпоративные пользователи выбирают свободное программное обеспечение с открытым исходным кодом (в порядке убывания важности):

* Лучшая безопасность;
* Выше качество;
* Возможность безопасно использовать технологии с открытым исходным кодом;
* Более пригодны для работы в облаке, с использованием облачных технологий.

Как можем наблюдать, стоимость не входит в список критериев по которым выбирается открытое ПО.

В России, дополнительно к вышеуказанным факторам и мировым трендам, добавилось достаточно специфическое обстоятельство в виде одностороннего отказа крупных и уникальных корпоративных поставщиков импортного программного обеспечения от выполнения договорных обязательств и отказ от продления лицензий и в поддержке на уже приобретенное и внедренное проприетарное программное обеспечение.

Таким образом, применительно к России, нет смысла сравнивать достоинства и недостатки разных категорий программного обеспечения. Фактически единственным способом обеспечить непрерывность процессов в информационных системах остается не просто продолжать использовать свободное программное обеспечение с открытым исходным кодом, потому, что его использование целесообразно само по себе, но заменить им, там где это необходимо проприетарное программное обеспечение.

# Глава 2. Основные этапы ETL процесса и построения дашборда

ETL процесс (ETL) - это процесс извлечения данных из нескольких источников, их преобразования для обеспечения согласованности и, наконец, загрузки в целевую систему для различных инициатив, основанных на данных[ 8].

Ценность ETL для организации - в том, этот процесс предоставляет организациям единый источник истины (SSOT) необходимый для точного анализа данных. Имея надежные данные, организация может более уверенно предпринимать стратегические шаги, будь то оптимизация цепочек поставок, адаптация маркетинговых усилий или улучшение качества обслуживания клиентов.

ETL процесс реализуется посредством ETL системы, которую можно рассматривать как одну из составляющих информационной системы предприятия.

Дашборд - это информационный ресурс (элемент интерфейса программного обеспечения, сайт, электронный документ), который получает данные из других систем и отображает их в лаконичном и понятном для пользователя дашборда виде. На дашбордах используют текст, графики, диаграммы и другие средства визуализации. Дашборд получает данные и обновляется автоматически с заданным интервалом или даже в режиме реального времени[ 7].

Типовой ETL процесс состоит из этапов:

1. Этап загрузки;
2. Этап валидации;
3. Этап мэппинга;
4. Этап агрегации;
5. Выгрузка.

**Шаг загрузки**. Данные попадают в ETL-систему. В ее основе лежит процесс, называемый Extract, на который принято смотреть “изнутри” системы, т.к. для ETL процесса важно как происходит сама загрузка, а не извлечение данных из исходной системы. Данные, которые попадают в систему не обработаны и не проверены. Необработанные и непроверенные данные называются сырыми. Качество данных может быть произвольным, при наличии возможности их только сверяют по количеству строк. Расхождение в количестве данных или отсутствие загруженных данных может свидетельствовать о наличии сбоя.

**Этап валидации.** ETL система проводит проверку полученных данных. В ходе валидации информация по очереди проверяется и фильтруется в соответствии с настроенными правилами. ETL система анализирует полноту данных, проверяет их корректность и наличие ошибок. В конце валидации, как правило формируется отчеты, содержащий информацию об успешной валидации, если валидация выполнена без ошибок, либо содержащий информацию обо всех найденных ошибках, если в ходе валидации были выявлены ошибки. В зависимости от принятой стратегии работы с ошибками и их критичности ETL процесс может быть остановлен или продолжить работу с учетом выявленных ошибок. Под отчетом может подразумеваться, как отправка сообщения электронной почты или в чат, так и формирование отчетов, содержащих подробную информацию о ходе валидации. Степень детализации зависит от принятой в компании стратегии информационного аудита.

Примером валидации может являться проверка номера телефона на соответствие регулярному выражению “7\d{10}”.

**Этап мэппинга.** Этап относится к части ETL процесса, называемого Transform и призван преобразовать загруженные в ETL систему данные в нужный формат и обогатить их (данные). Валидированные данные, как правило, представляются в виде табличных структур, к которым добавляются нужные столбцы и строки, загруженные из других источников или содержащихся в мастер системах, либо в справочниках ETL системы. Мэппинг может происходить с использованием различных алгоритмов в зависимости от использованного ETL-инструмента.

Примером мэппинга может быть:

* присвоение идентификатора клиента, хранящегося в мастер системе для хранения информации о клиентах, загруженной строке с данными, содержащей только Фамилию, Имя, Отчество, Номер телефона;
* преобразование номера телефона из различных форматов в формат, например, “7\d{10}”.

**Этап агрегации.** Этап относится к части ETL процесса, называемого Transform, необходим для того, чтобы преобразованные данные можно было перенести в новое хранилище. За счет изменения связей между данными, информация преобразуется в формат соответствующий формату хранения данных в хранилище.

**Выгрузка.** Этот этап реализует часть ETL процесса, называемого Load, когда проверенные, преобразованные, очищенные, агрегированные данные выгружаются из ETL-системы и отправляются в хранилище. Для этого используются коннекторы и различные части интерфейса ETL-системы и хранилища.

Типовой процесс построения дашборда состоит из этапов[ 23]:

1. Анализ целей и задач;
2. Анализ UX составляющей;
3. Построение дашборда.

**Анализ целей и задач.** На этом этапе определяют:

* кто будет целевой аудиторией дашборда? Сколько человек и какие полномочия будет даны на доступ к данным дашборда?
* какие цели и задачи будут решать пользователь или компания с помощью дашборда?
* какие ключевые метрики и показатели понадобятся? Сколько их будет?

Для достижения максимальной эффективности дашборда нужна общая цель. Например, степень достижения KPI, текущий уровень SLA, количество или объем транзакций. Для выявления общей цели дашборда используют различные методики, одна из которых называется “Метрика полярной звезды”. Обзор методик выходит за рамки настоящего проекта.

**Анализ UX составляющей.** Для создания эффективного UX - дизайна проводят два типа анализа:

* иерархический анализ, подразумевающий что расположение метрик на дашборде или на дашбордах разного уровня будет выполнено таким образом, что метрики более низкого уровня будут пояснять метрики более высокого уровня.

Например иерархия может быть представлена уровнями (от общего к детальному): Бизнес-метрики, Экономические метрики, Продуктовые метрики, специальные метрики;

* когнитивный анализ, подразумевающий составление карты поведения пользователей, чтобы выявить ожидания пользователей и определить куда они интуитивно точно должны зайти и какую кнопку кликнуть, чтобы гарантировано получить доступ к необходимой информации.

Подробный разбор анализа UX составляющей и формирование макета дашборда выходит за рамки настоящего проекта.

**Построение дашборда.** Этап предполагает: подключение к источникам данных, настройку преобразования данных внутри дашборда, размещение визуальных элементов на дашборде. Данный этап сильно зависит от программного обеспечения выбранного для реализации дашборда. Как предполагает цель проекта в качестве программного обеспечения для реализации дашборда будет выбрано свободное программное обеспечение с открытым исходным кодом.

# Глава 3. Свободное программное обеспечение с открытым исходным кодом, используемое для реализации ETL процессов и построения дашбордов

Критерии свободного программного обеспечения с открытым исходным кодом определены ранее в Главе 1.

Определим необходимые инструменты для реализации настоящего проекта.

* Язык (языки) программирования;
* Оркестратор процесса ETL;
* Инструменты для построения хранилища;
* Инструмент построения дашбордов;
* Платформа для развертывания приложений;
* Операционная система.

**Языки программирования.** Сравним между собой наиболее популярные языки программирования, используемые при реализации процессов ETL и в Data Science[ 6, 24]:

| Наименование | Преимущества | Недостатки | Относится ли к свободному ПО с открытым исходным кодом. |
| --- | --- | --- | --- |
| Python | 1. Фактически стандарт в области в ETL и DS;  2. Большое количество специальных библиотек для работы с данными;  3. Простой синтаксис;  4. Широко распространен, много специалистов. | 1. Медленее, чем Java подобные языки.  2. Худшая, чем у Java подобных языков многопоточность;  3. Не является нативным для распространенного фреймворка Spark. | Да[ 15] |
| R | 1. Большое количество библиотек для анализа данных;  2. Более быстрый, чем Python;  3. Популярен среди специалистов; профессионально занимающихся аналитикой и статистикой. | 1. Сложный синтаксис, высокий порог входа, мало специалистов;  2. Практически не используется вне задач аналитики и статистики в академической среде;  3. Требователен к памяти при обработке больших наборов данных. | Да[ 25] |
| Java | 1. Нативный язык для многих Big Data инструментов, например стек Hadoop;  2. Много библиотек для работы с данными;  3. Высокая скорость исполнения;  4. Более эффективный, чем у Python параллелизм.  4. Большое сообщество разработчиков. | 1. Более сложный синтаксис, чем у Python;  2. Меньшее количество библиотек для обработки данных;  3. Меньше подходит для прототипирования с быстрого построения приложений, проверки гипотез. | Не все реализации можно отнести к свободному ПО с открытым исходным кодом |
| Scala | 1. Нативный язык для многих перспективных Big Data инструментов, например Spark;  2. Высокая скорость обработки больших наборов данных;  3. Более эффективная реализация параллельный вычислений чем Python и Java.  4. Набирает популярность | 1. Более сложный синтаксис, чем у Python;  2. Меньшее количество библиотек для обработки данных;  3. Меньше подходит для прототипирования с быстрого построения приложений, проверки гипотез. | Да[ 26] |

Таблица 1. Сравнение языков программирования наиболее часто применяемых при реализации процессов ETL и в Data Science

На основании информации Таблицы 1, для достижения целей настоящего проекты более подходящим выглядит выбор языка Python, как язык с простым синтаксисом, наличием библиотек для обработки данных для задачи не требующей высокой производительности.

**Оркестратор процесса ETL**. Сравним между собой наиболее популярные оркестраторы процессов ETL с открытым исходным кодом[ (SkillFactoryMedia), (MaxRokatansky), (Trencseni)]:

| Наименование | Преимущества | Недостатки | Относится ли к свободному ПО с открытым исходным кодом. |
| --- | --- | --- | --- |
| Apache Airflow | 1. Зрелый продукт, большое сообщество;  2. Наличие документации;  3. Написан на Python;  4. Наличие хорошего веб интерфейса;  5. Наличие встроенного планировщика.  5. Высокая надежность и отказоустойчивость. | 1. Сложность развертывания.  2. Ограниченная поддержка языков программирования (только Python);  3. Несовместимость с некоторыми экосистемами такими как Spark, Hadoop;  4. Ограниченные возможности отладки. | Да[ 18] |
| Luigi | 1. Проще, чем Airflow, применяется для меньших по масштабу проектов;  2. Позволяет объединять разнородные запросы Hadoop, Python, Spark;  3. Популярен среди специалистов; профессионально занимающихся аналитикой и статистикой. | 1. Отсутствие встроенного планировщика;  2. Менее удобный GUI;  3. Требователен к памяти при обработке больших наборов данных. | Да[ 13] |

Таблица 2. Сравнение инструменты для реализации процессов ETL.

На основании информации Таблицы 1, для достижения целей настоящего проекты более подходящим выглядит выбор Apache Airflow, как наиболее распространенного, задокументированного и имеющего большое сообщество пользователей.

**Инструменты для построения хранилища.** В качестве инструмента для построения хранилища будет использован стек технологий Hadoop, как наиболее распространенный, хорошо задокументированный, имеющий большое сообщество пользователей. Hadoop - набор программных утилит, библиотек и фреймворк для разработки и выполнения распределенных программ, работающих на кластерах, количество узлов в которых может достигать сотен и тысяч.

Дистрибутивы Hadoop распространяются, под коммерческими и свободными лицензиями. Для проекта будет использован дистрибутив, распространяемый под свободной лицензией.

В рамках проекта будет использоваться кластер, состоящий из узлов, реализующих, в том числе, следующие компоненты Hadoop:

* **HDFS** - распределенную файловую систему;
* **Hive** - систему управления базами данных на основе платформы Hadoop с SQL-подобным языком запросов;
* **Hue** - это веб интерфейс для анализа данных на Hadoop.

Дополнительно, в кластере присутствуют компоненты: Apache Spark, JuputerLab server. Указанные компоненты присутствуют т.к. включены автором сборки в пресобранный кластер и не будут использоваться для реализации проекта.

**Инструмент построения дашбордов**. Сравним между собой наиболее популярные инструменты построения дашбордов, относящихся к категории ПО с открытым исходным кодом[10, 11 16, 19]:

| Наименование | Преимущества | Недостатки | Относится ли к свободному ПО с открытым исходным кодом. |
| --- | --- | --- | --- |
| Apache Superset | 1. Наибольшее количество визуальных элементов;  2. Возможность дополнительной настройки визуальных элементов, посредством CSS кода;  3. Поддерживается авторизация через LDAP;  4. Поддерживается доступ на уровне строк.  5. Возможность встраивания в сторонние сервис | 1. Сложная процедура развертывания и настройки;  2. Отсутствует функционал создания модели данных;  3. Отсутствует собственный синтаксис для расчетов;  4. Менее развитый функционал интерактивного взаимодействие с дашбордом по сравнению с Metabase;  5. Поддерживается экспорт данных только в популярные форматы;  6. Нет коннекторов к популярным источникам данных “в коробке”. | Да [ 30] |
| Metabase | 1. Простая процедура развертывания;  2. Значительное количество визуальных элементов, но меньшее, чем в Superset;  3. Имеется упрощенный графический редактор SQL;  4. Интерактивное взаимодействие с дашбордом реализовано, лучше чем в Superset; 4. Поддерживается экспорт данных в популярные форматы;  5. Поддерживается авторизация через LDAP;  6. Наличие встроенных коннекторов к большинству источников данных. | 1. Отсутствует собственный синтаксис для расчетов;  2. НЕ поддерживается доступ на уровне строк в бесплатной версии;  3. НЕ поддерживается возможность встраивания в сторонние сервисы в бесплатной версии. | Да [ 21] |
| Tipboard | 1. Простая процедура развертывания;  2. Настройка и создание дашбордов с помощью YAML;  3. Предоставляет возможность передачи данных через API в формате JSON на дашборд;  4. Позволяет создать собственные визуальные элементы. | 1. Малое количество визуальных элементов “из коробки”;  2. Недостаточно развитая документация;  3. Не поддерживается Python3. | Да [ 31] |

Таблица 2. Сравнение наиболее популярные инструменты построения дашбордов, относящихся к категории свободного программного обеспечения с открытым исходным кодом.

На основании информации Таблицы 1, выберем для проекта Apache Superset, как наиболее популярный и развитый инструмент построения дашбордов среди свободного программного обеспечения с открытым исходным кодом.

**Платформа для развертывания приложений**. В качестве платформы для развертывания приложений выберем Docker, фактически, являющийся, на текущий момент, индустриальным стандартом, несмотря на наличие быстро развивающихся альтернативных решений.

Операционная система. В качестве операционной системы будет использован Linux, фактически являющийся безальтернативным решением в корпоративной среде среди свободного программного обеспечения с открытым исходным кодом.

# Глава 4. Определение целей ETL процесса и построения дашборда, реализованного с помощью программного обеспечения с открытым исходным кодом

В работе будет реализован типовой процесс включающий в себя:

1. Загрузка данных из источника - скачивание данных об измерениях температуры воздуха с часовыми интервалами с сайта <https://archive-api.open-meteo.com> по трем локациям за период в 30 дней.
2. Валидация - проверка наличия загруженных файлов;
3. Мэппинг и агрегация:
   1. удаление заголовка файла, содержащего служебную информацию;
   2. добавление для каждой строки информации о месте расположения, полученной из заголовка файла - нормализация таблицы;
   3. корректировка временной метки измерения на основании смещения относительно UTC, полученного из заголовка файла;
   4. объединение файлов в один.
4. Загрузка результирующего файла в Hadoop HDFS;
5. Загрузка результирующего файла в хранилище Hadoop Hive;
6. Возможность просмотреть загруженные данные через Hadoop HUE;
7. Отображение информации о средней, минимальной и максимальной температуре за сутки для каждой локации на дашборде Apache Airflow, состоящем из одного графика.

В случае успешной реализации процесса в целом и отображения на дашборде графика температуры, можно будет сделать вывод о возможности реализации типового процесса ETL и построения дашборда с использованием исключительно свободного программного обеспечения с открытым исходным кодом.

# Глава 5. Действия, выполненные для достижения целей ETL процесса и построения дашборда

5.1. Установлены компоненты необходимые для построения ETL процесса и построения дашборда

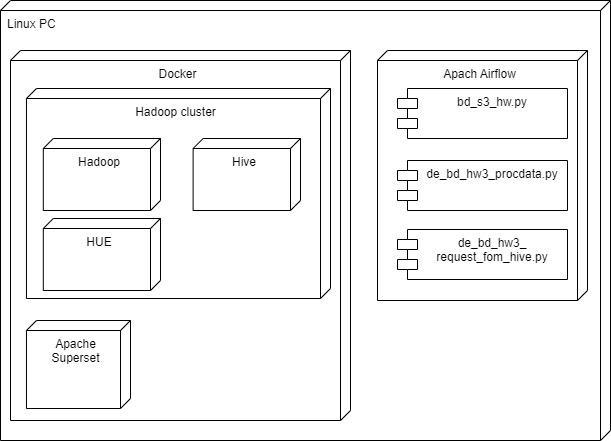


Рисунок 2. Диаграмма развертывания

5.2. В IDE VS Code разработан DAG для оркестратора Airflow

5.2.1. Граф DAG

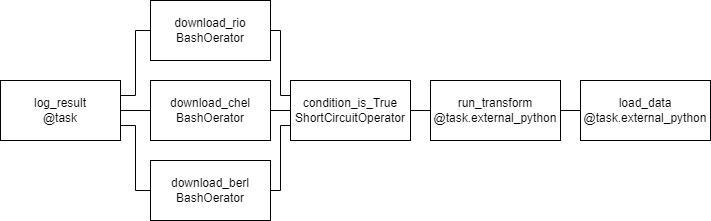


Рисунок 3. Граф DAG.

Task log\_result реализует вывод в лог информацию о сформированных именах файлов, которые будут загружены.

Task download\_\* реализует загрузку информации из источника;

Task condition\_is\_True реализует проверку наличия всех загруженных файлов;

Task run\_transform реализует вызов библиотеки исполняемой во внешнем окружении, выполняющей мэппинг и агрегацию загруженных файлов.

Task load\_data реализует копирование результирующего файла в Hadoop HDFS средствами утилит hdfs dfs и последующую загрузку в Hadoop Hive.

Листинг файла bd\_s3\_hw.py, содержащего код DAG, приведен в Приложении №

5.2.1. Загрузка библиотеки для обработки загруженных файлов выполняется во внешнем окружении в целях избежания зависимостей между окружением в котором установлен оркестратор Apache Airflow и библиотеками, используемыми для обработки загруженных файлов.

Листинг файла de\_bd\_hw3\_procdata.py, содержащего код библиотеки обработки загруженных файлов, приведен в Приложении № 2.

5.2.3. Функции взаимодействующие с файловой системой HDFS и загружающие файл в Hive выполняются во внешнем окружении в целях избежания зависимостей между окружением в котором установлен оркестратор Apache Airflow и библиотеками, используемыми для вышеуказанных операций.

Листинг файла de\_bd\_hw3\_request\_fom\_hive.py, содержащего код функций для взаимодействия с HDFS и загрузки файла в Hive, приведен в Приложении № 3.

5.2. В Hadoop Hive создана таблица для загрузки данных.

| CREATE TABLE temperature (  location CHAR(50),  date\_time TIMESTAMP,  temperature\_celsius FLOAT  )  ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'  WITH SERDEPROPERTIES  ("timestamp.formats"="yyyy-MM-dd'T'HH:mm:ss",  "separatorChar" = ",")  TBLPROPERTIES("skip.header.line.count"="2") |
| --- |

В качестве формата таблицы выбран формат csv, т.к. не предполагается удаление и обновление данных в таблице.

5.3. В программном обеспечении Apache Superset:

5.3.1. настроено подключение к БД Hive

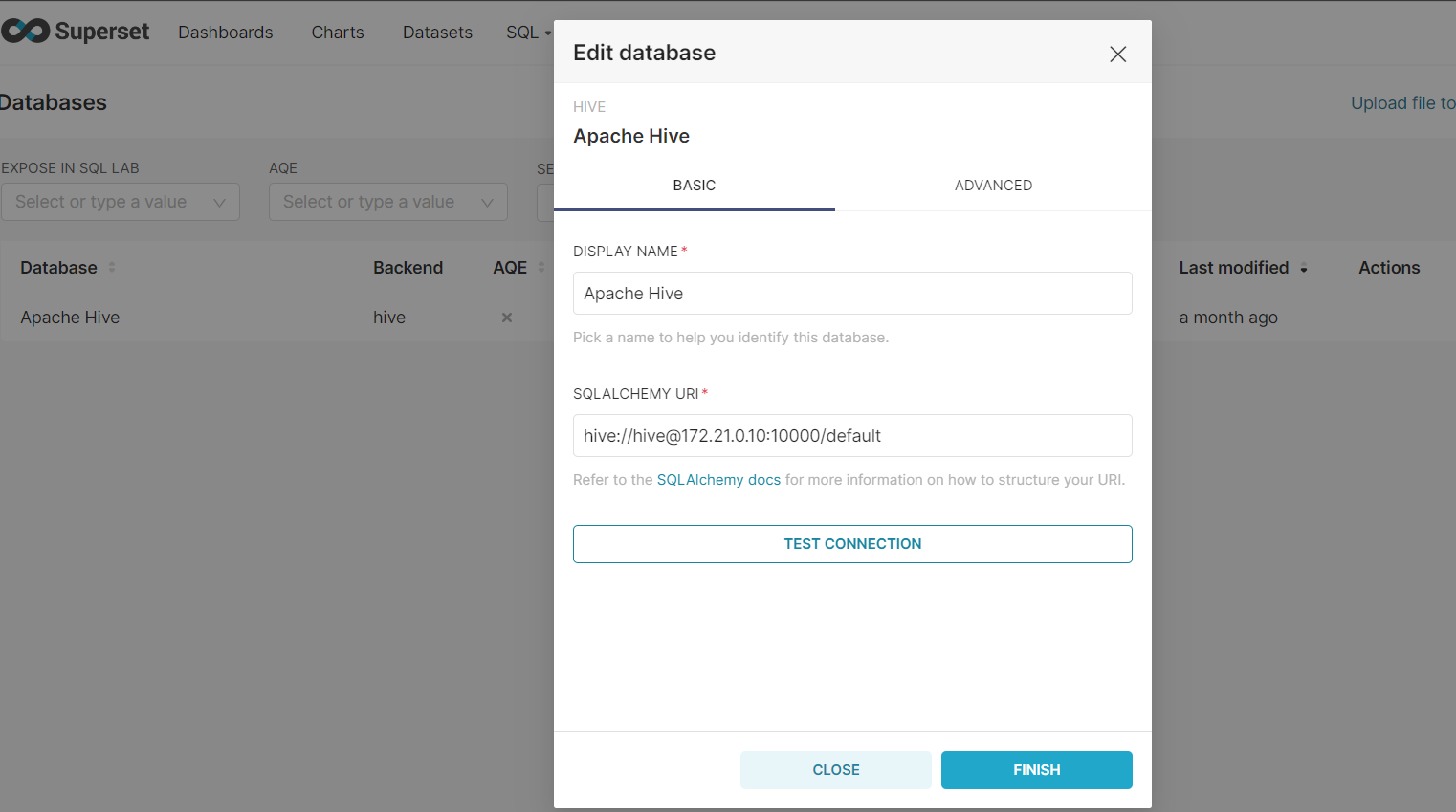


Рисунок 4. Настройка подключения к БД Hive

5.3.2. создан источник данных, использующий подключение к БД Hive

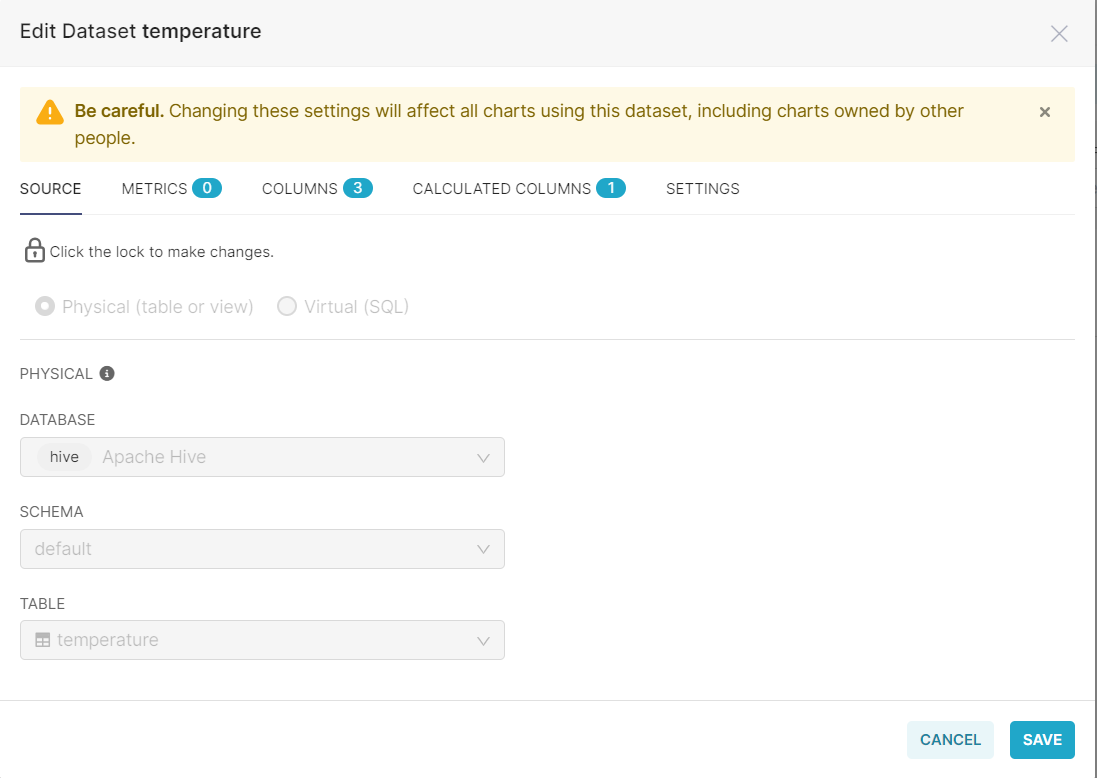


Рисунок 5. Настройка источника данных.

5.3.3. создан визуальный элемент на основании источника данных

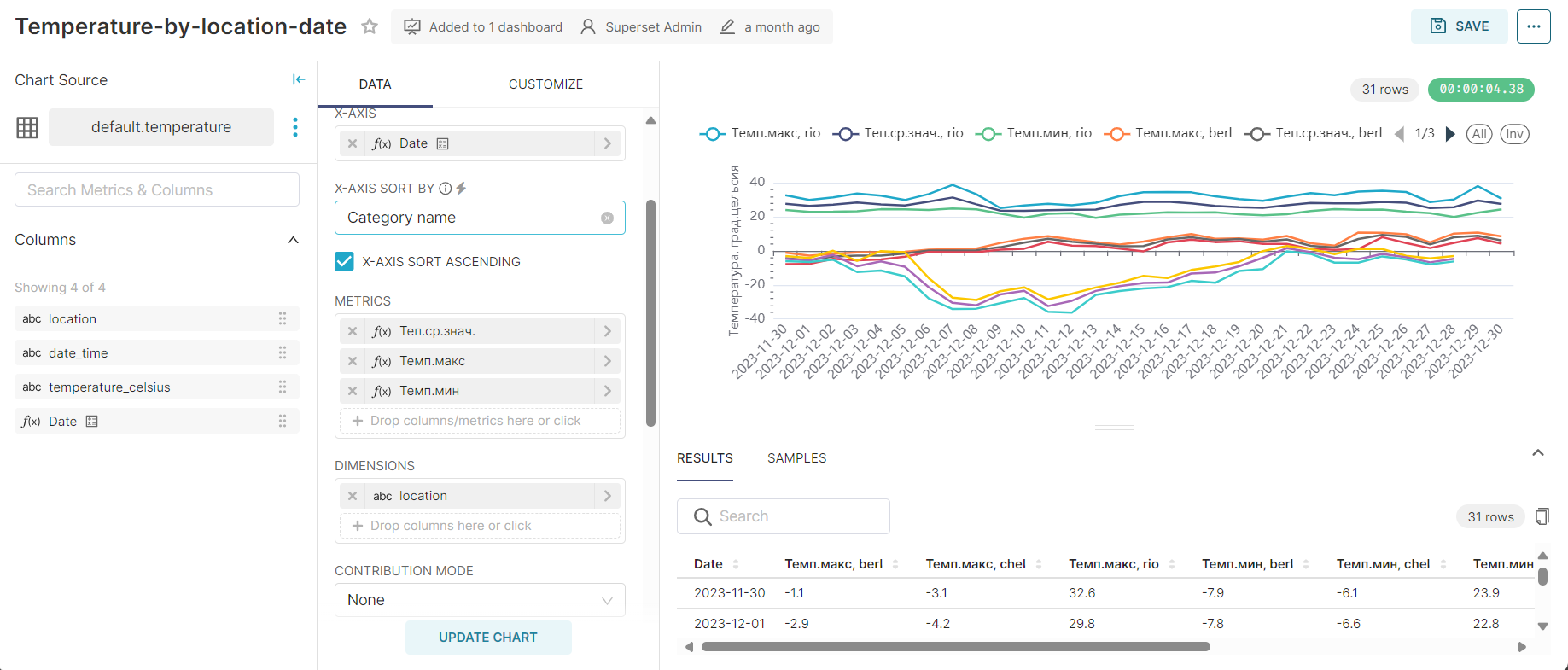


Рисунок 6. Создание визуального элемента.

5.3.4. создан дашборд, содержащий визуальный элемент с информацией о температуре

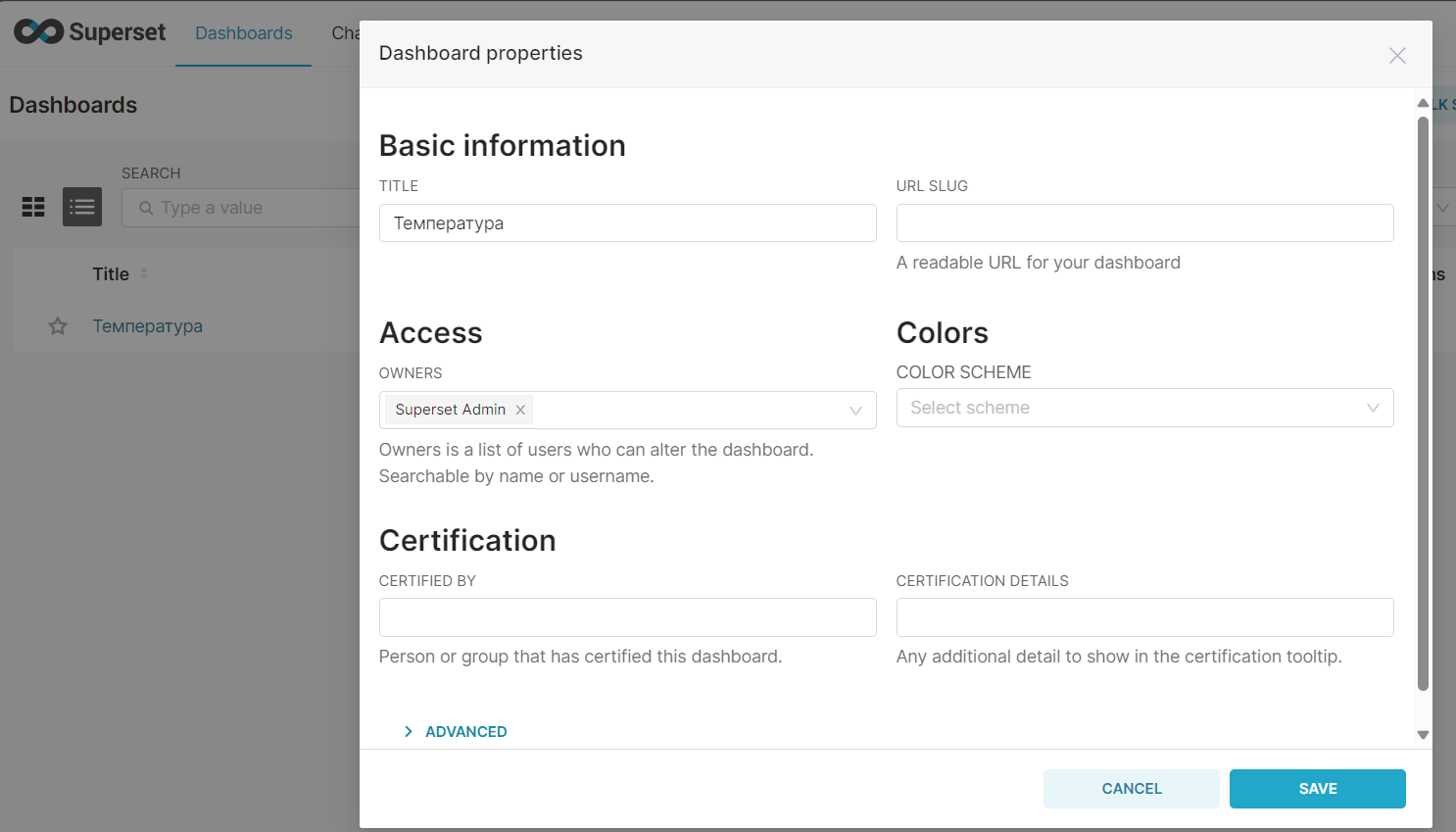


Рисунок 7. Создание дашборда

# Глава 6. Запуск ETL процесса и получение результатов

6.1. Выполним запуск ETL процесса из WEB интерфейса Apache Airflow

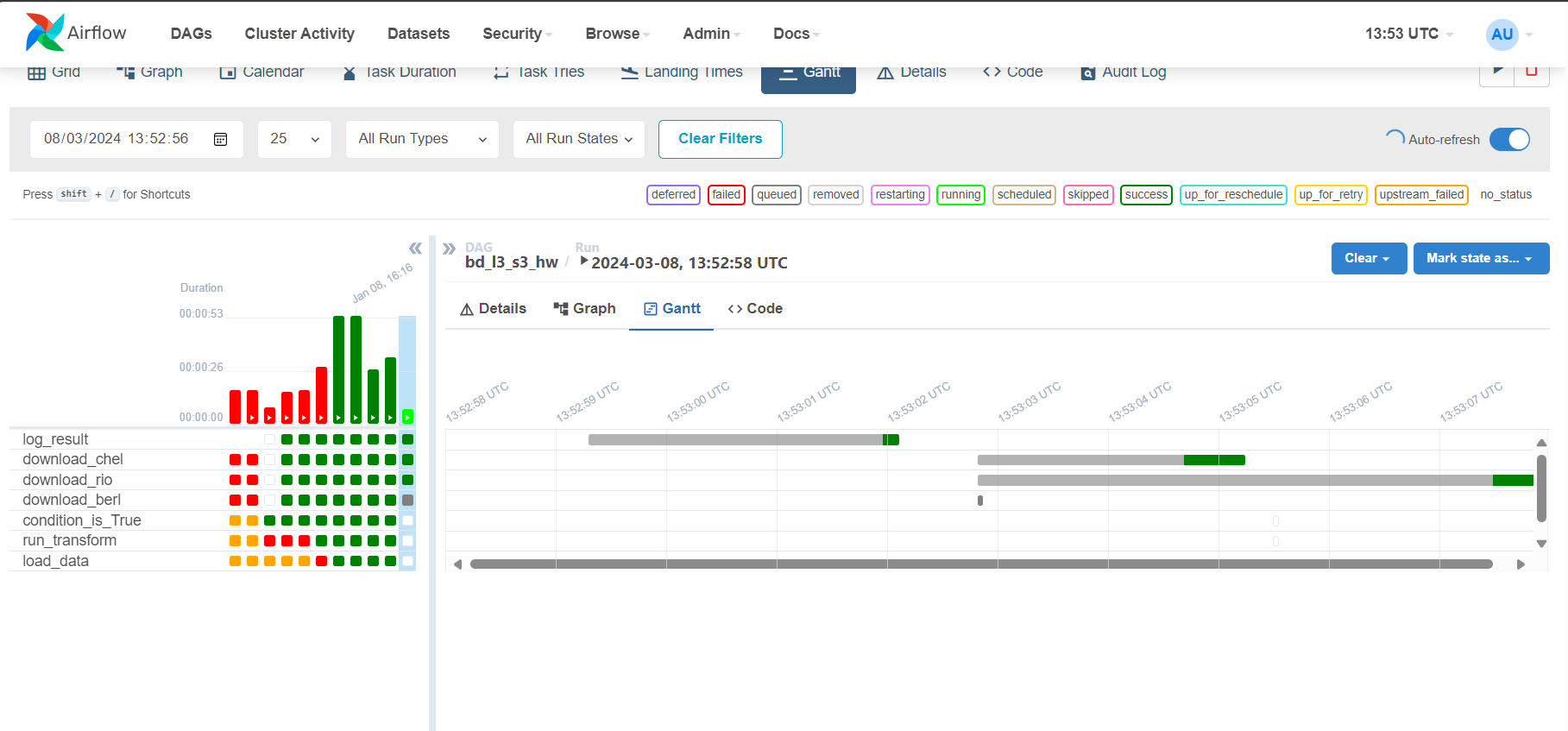


Рисунок 8. Выполняющийся DAG

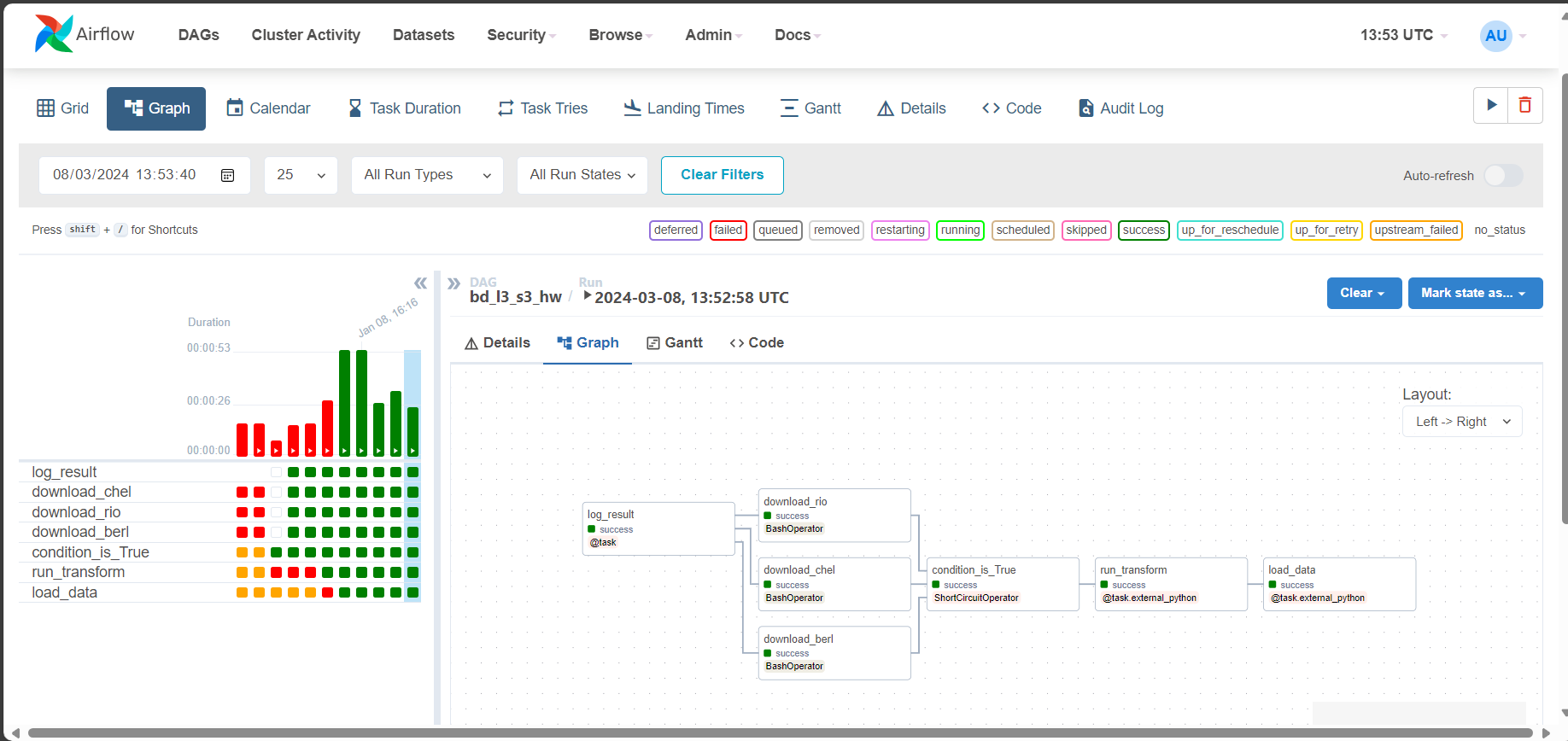


Рисунок 9. Результаты запуска DAG

Логи работы task DAG - а в Приложении № 4.

По скриншотам и логам работы DAG видим, что DAG завершил работу без ошибок.

6.2. Проверим наличие загруженных файлов в рабочем каталоге Linux

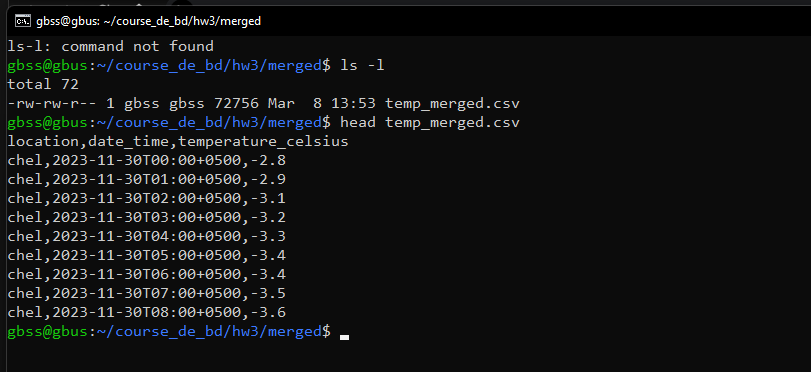


Рисунок 10. Скриншот файловой системы Linux и заголовка обработанного файла.

Файлы были загружены и обработаны.

После успешной загрузки данных в Hadoop Hive загруженный файл удаляется. Сформировать скриншот загруженного и обработанного файла в HDFS не представляется возможным.

6.3. Проверим наличие загруженной информации в интерфейсе Hadoop HUE

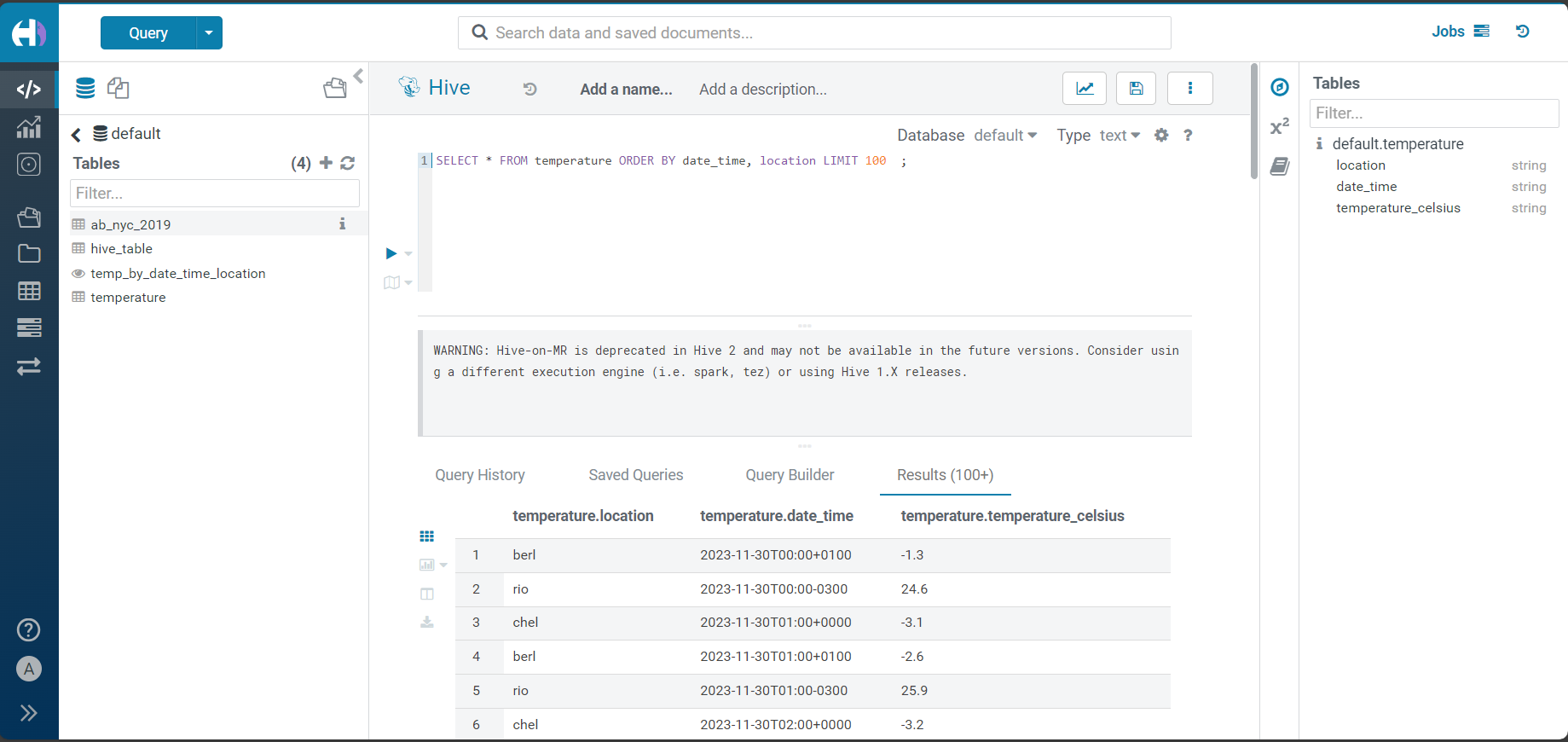


Рисунок 11. Скриншот результата запроса в Hadoop HUE.

Наблюдаем, что ранее обработанные данные загружены в Hadoop Hive

6.4. Перейдем на страницу дашборда в WEB интерфейсе Apache Superset

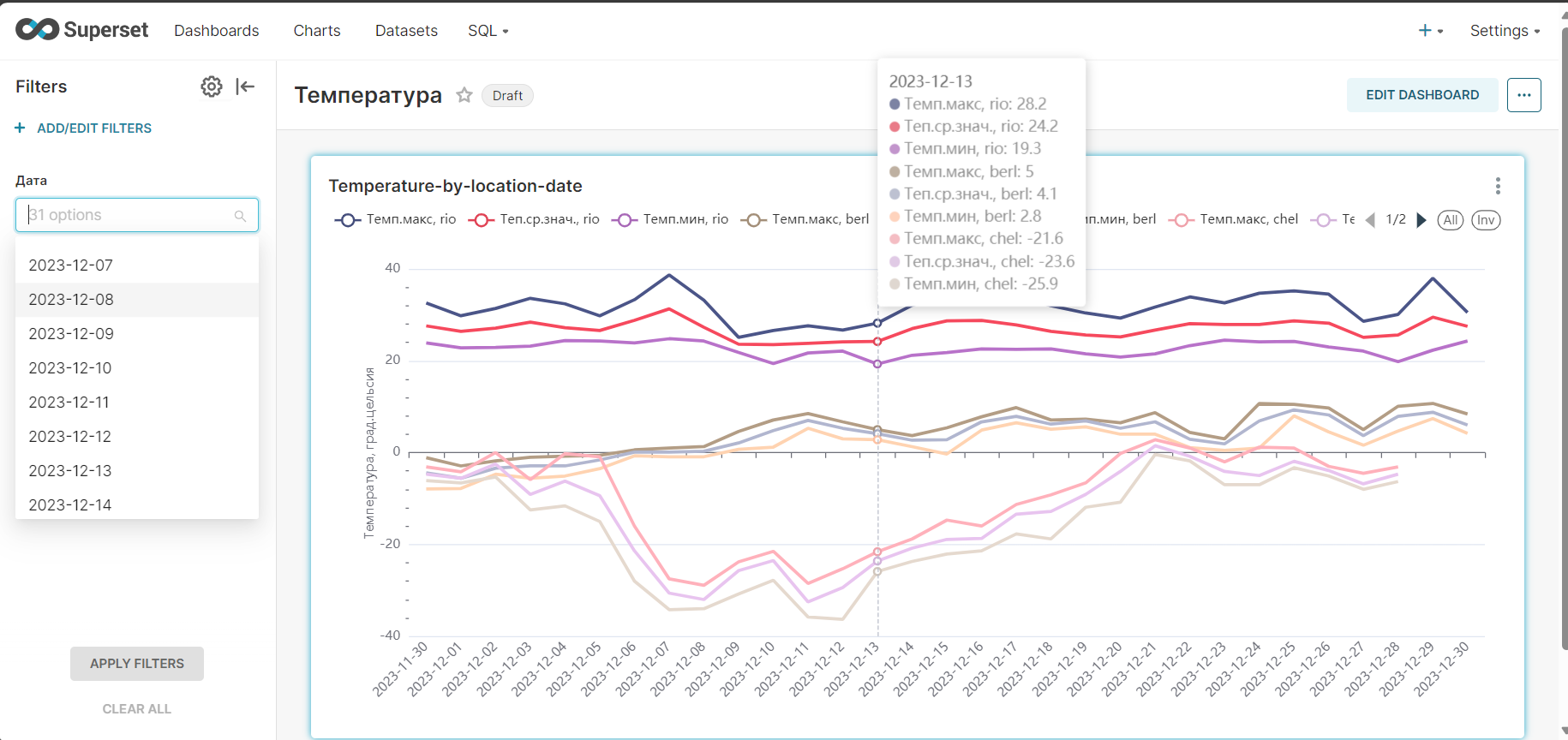


Рисунок 12. Скриншот дашборда содержащего загруженные и обработанные данные.

В дашборде отобразилась загруженная и обработанные в рамках процесса ETL информация, агрегированная по датам инструментами дашборда.

6.8. Процесс ETL выполнен без ошибок, загруженная информация отображается в дашборде.

# Заключение

В результате исследования было показано, что существует свободное программное обеспечение с открытым исходным кодом, применимое для построения ETL процесса и задач бизнес аналитики.

Таким образом, можно считать цель исследования достигнутой.

Значимость работы:

Доля программного обеспечение с открытым исходным кодом непрерывно растет как в мире в целом, так и в России.

Существуют объективные причины перехода на свободное программное обеспечение:

* Лучшая безопасность;
* Выше качество;
* Возможность безопасно использовать технологии с открытым исходным кодом;
* Более пригодны для работы в облаке, с использованием облачных технологий;
* Меньшая, чем у проприетарного программного обеспечения зависимость от политических и страновых рисков.

Настоящий проект показывает, что свободное программное обеспечение с открытым исходным кодом может как сосуществовать совместно с проприетарным обеспечением в рамках информационной системы предприятия, так и заменять его.

Развитием настоящего проекта могло бы быть исследование и сравнение скорости работы и функциональности оркестраторов ETL процессов и программного обеспечения для построения дашбородов.

# Список используемой литературы

1. Книга: Харенслак Б., де Руйтер Дж. Apache Airflow и конвейеры обработки данных / пер. с англ. Д. А. Беликова. – М.: ДМК Пресс, 2021

2. Книга: Уайт Т. Hadoop: Подробное руководство. — СПб.: Питер, 2013

3. Книга: Иан Милл, Эйдан Хобсон Сейерс Docker на практике / пер. с англ. Д. А. Беликов. – М.: ДМК Пресс, 2020

4. Проприетарное программное обеспечение [Электронный ресурс] // Википедия. URL: https://ru.wikipedia.org/wiki/Проприетарное\_программное\_обеспечение. Accessed 5 March 2024.

5. “Программное обеспечение — Википедия.” *Википедия*, https://ru.wikipedia.org/wiki/Программное\_обеспечение. Accessed 6 March 2024.

6. Тюфякова, Виктория. *Skillfactory Media*, 9 November 2017, https://blog.skillfactory.ru/9-yazykov-dlya-raboty-s-big-data. Accessed 8 March 2024.

7. Чепурнова, Дарья. “Дашборд — что это и как сделать его в системе аналитики.” *Skillbox*, 3 April 2023, https://skillbox.ru/media/management/dashbord\_chto\_eto\_i\_zachem\_nuzhno/. Accessed 5 March 2024.

8. Хайдер, Хуррам. “Что такое ЭТЛ? Определение, процесс, лучшие практики и варианты использования | Astera.” *Astera Software*, 26 January 2024, https://www.astera.com/ru/type/blog/etl/. Accessed 5 March 2024.

9. “Что такое открытый исходный код? – Описание открытого исходного кода – AWS.” *AWS*, https://aws.amazon.com/ru/what-is/open-source/. Accessed 5 March 2024.

10. bilab.ru. “7 Open Source и 2 low-code решений для визуализации BI-аналитики. Часть 1.” *BILab*, https://bilab.ru/sravnenie-7-open-source-y-2-low-code-reshenii-dlia-vizualizacii-bi-analitiki. Accessed 8 March 2024.

11. bilab.ru. “7 Open Source и 2 low-code решений для визуализации BI-аналитики. Часть 2.” *BILab*, https://bilab.ru/7-open-source-instrumentov-y-2-low-code-chast-vtoraya. Accessed 8 March 2024.

12. “Categories of Free and Nonfree Software.” *GNU.org*, https://www.gnu.org/philosophy/categories.html. Accessed 6 March 2024.

13. “github.com | Luigi.” *github.com*, 9 November 2017, https://github.com/SAP/luigi/blob/master/LICENSE. Accessed 8 March 2024.

14. “Различные лицензии и комментарии к ним - Проект GNU - Фонд свободного программного обеспечения.” *GNU.org*, https://www.gnu.org/licenses/license-list.ru.html. Accessed 6 March 2024.

15. “History and License — Python 3.12.2 documentation.” *Python Docs*, https://docs.python.org/3/license.html. Accessed 8 March 2024.

16. igor\_yakovlev. “9 продуктов для создания дашбордов / Хабр.” *Habr*, 27 October 2022, https://habr.com/ru/articles/695310/. Accessed 8 March 2024.

17. ImlawIT. “Свободные лицензии на программное обеспечение / Хабр.” *Habr*, 26 December 2023, https://habr.com/ru/articles/783298/. Accessed 6 March 2024.

18. “License — Airflow Documentation.” *Apache Airflow*, https://airflow.apache.org/docs/apache-airflow/stable/license.html. Accessed 8 March 2024.

19. Maan, Nikhil, and Vipul Gupta. “Top 8 Open Source Dashboards.” *MetricFire*, 12 October 2023, https://www.metricfire.com/blog/top-8-open-source-dashboards/. Accessed 8 March 2024.

20. MaxRokatansky. “Обзор фреймворка Luigi для построения последовательностей выполнения задач.” *Habr*, 11 October 2017, https://habr.com/ru/companies/otus/articles/339904/. Accessed 8 March 2024.

21. “Metabase licenses and terms of service.” *Metabase*, https://www.metabase.com/license/. Accessed 8 March 2024.

22. @Mexos, Mexos. “Основные виды ПО / Хабр.” *Habr*, 17 September 2008, https://habr.com/ru/articles/40094/. Accessed 6 March 2024.

23. Modus BI. “Логика и принципы построения BI-дашборда / Хабр.” *Habr*, 13 September 2023, https://habr.com/ru/companies/modusbi/articles/760780/. Accessed 7 March 2024.

23. Pickell, Devin. “The 4 Most Important Big Data Programming Languages.” *G2*, 21 June 2019, https://www.g2.com/articles/big-data-programming-languages. Accessed 8 March 2024.

24. “R:.” *R:*, https://www.r-project.org/Licenses/. Accessed 8 March 2024.

“Scala License.” *The Scala Programming Language*, https://scala-lang.org/license/. Accessed 8 March 2024.

25. SkillFactoryMedia. *SkillFactoryMedia*, 9 November 2017, https://blog.skillfactory.ru/glossary/apache-airflow. Accessed 8 March 2024.

26. Stallman, Richard. “Why Open Source Misses the Point of Free Software.” *GNU.org*, https://www.gnu.org/philosophy/open-source-misses-the-point.html. Accessed 6 March 2024.

27. “The State of Enterprise Open Source: A Red Hat report.” *Red Hat*, 22 February 2022, https://www.redhat.com/en/resources/state-of-enterprise-open-source-report-2022. Accessed 7 March 2024.

28. “superset/LICENSE.txt at master · apache/superset.” *GitHub*, https://github.com/apache/superset/blob/master/LICENSE.txt. Accessed 8 March 2024.

29. “Tipboard license.” *tipboard.readthedocs.io*, 9 November 2017, https://tipboard.readthedocs.io/en/latest/license.html. Accessed 8 March 2024.

30. Trencseni, Marton. *bytepawn.com*, 9 November 2017, https://bytepawn.com/luigi-airflow-pinball.html. Accessed 8 March 2024.

# Приложения

Приложение № 1. Листинг файла bd\_s3\_hw.py

| from airflow import DAG  from airflow.decorators import dag, task  from airflow.operators.bash import BashOperator  from airflow.operators.python import PythonOperator, ShortCircuitOperator, ExternalPythonOperator  from airflow.providers.postgres.hooks.postgres import PostgresHook  from airflow.operators.empty import EmptyOperator  from datetime import datetime  from pathlib import Path  import logging  logging.basicConfig(level=logging.INFO)  FILES\_NAME\_FOR\_DOWNLOAD = {}  # {  # '<location\_code>':{'file': 'open-meteo-{location\_code}\_{2023-11-30}\_{2023-12-30}.csv',  # 'coordinates':{'lat':<int>, 'long': <int>}},  # ...  # }  KEY\_FILE = 'file'  KEY\_COORDINATES = 'coordinates'  FILES\_PROCESSORS = []  URL\_DOWNLOAD\_FROM = 'https://archive-api.open-meteo.com/v1/archive'  DIR\_DOWNLOAD\_TO = '/home/gbss/course\_de\_bd/hw3/downloads\_tmp'  DIR\_MERGED\_TO = '/home/gbss/course\_de\_bd/hw3/merged'  FILE\_MERGED = 'temp\_merged.csv'  PATH\_TO\_PYTHON\_FOR\_LOAD\_TO\_HIVE = '/home/gbss/course\_de\_bd/venv\_hadoop\_cluster/bin/python'  PATH\_TO\_PYTHON\_FOR\_TRANSFORM = '/home/gbss/course\_de\_etl/pd/bin/python'  START\_DATE = datetime(2023, 11, 30)  END\_DATE = datetime(2023, 12, 30)  DOWNLOAD\_FILES = True  MERGE\_FILES = True  LOAD\_DATA = True  def check\_downloaded\_files(\*\*kwargs):  dir\_download\_to = kwargs['dir\_download\_to']  files\_name\_for\_download = kwargs['files\_name\_for\_download']  for file in files\_name\_for\_download:  full\_name = str(Path(Path(dir\_download\_to) / file))  is\_exists = Path(full\_name).exists()  logging.info(f'{full\_name} is exists {is\_exists}')  if not is\_exists:  return False  return True  def get\_locations\_codes():  """  Функция заглушка  Возвращает коды локаций  Предполагается, что модуль будет извлекать коды локаций по которым нужно делать запрос из БД  """  return ['chel', 'rio', 'berl']  def get\_locations\_coordinates(location\_codes: list([str]))-> dict:  """  Функция заглушка  Возвращает координаты локаций по кодам  Предполагается, что модуль будет извлекать координаты локаций по которым нужно делать запрос из БД  """  locality = {  'chel':{'lat':55.149384, 'long': 61.38},  'berl':{'lat':52.5244, 'long': 13.4105},  'rio':{'lat':-22.90278, 'long': -43.2075},  }  result = {}  for el in location\_codes:  if el in locality:  result[el] = locality[el]  return result  def get\_files\_name(coordinates:dict, start\_date:datetime, end\_date:datetime)->dict:  files\_name\_for\_download = {}  for loc in coordinates:  files\_name\_for\_download[loc] = {}  files\_name\_for\_download[loc][KEY\_COORDINATES] = coordinates[loc]  files\_name\_for\_download[loc][KEY\_FILE] = f'open-meteo-{loc}\_{start\_date.strftime("%Y-%m-%d")}\_{end\_date.strftime("%Y-%m-%d")}.csv'  return files\_name\_for\_download  location\_codes = get\_locations\_codes()  locations\_coordinates = get\_locations\_coordinates(location\_codes)  FILES\_NAME\_FOR\_DOWNLOAD = get\_files\_name(locations\_coordinates , START\_DATE, END\_DATE)  print()  @dag(  description='Homework in seminar 3 Bigdata course',  start\_date=datetime(2023, 1, 1),  # https://stackoverflow.com/questions/53191639/airflow-set-dag-to-not-be-automatically-scheduled  schedule='@once',  is\_paused\_upon\_creation=True,  catchup=False,  tags=['Homework', 'Seminar 3', 'BD'],  )  def bd\_l3\_s3\_hw():  @task  def log\_result(files\_name\_for\_download):  logging.info(f'{files\_name\_for\_download}')  return  log\_result\_instance = log\_result(FILES\_NAME\_FOR\_DOWNLOAD)  tasks\_downloads = []  if DOWNLOAD\_FILES:  for key, val in FILES\_NAME\_FOR\_DOWNLOAD.items():  downloads = BashOperator(  task\_id=f'download\_{key}',  bash\_command=f'curl -sSLo {DIR\_DOWNLOAD\_TO}/{val[KEY\_FILE]} '\  f'"{URL\_DOWNLOAD\_FROM}?latitude={val[KEY\_COORDINATES]["lat"]}&longitude={val[KEY\_COORDINATES]["long"]}&start\_date={START\_DATE.strftime("%Y-%m-%d")}&end\_date={END\_DATE.strftime("%Y-%m-%d")}&hourly=temperature\_2m&timezone=auto&format=csv"',  )  tasks\_downloads.append(downloads)  else:  downloads = EmptyOperator(task\_id="skip\_download")  tasks\_downloads.append(downloads)  files\_is\_downloaded = ShortCircuitOperator(  task\_id="condition\_is\_True",  python\_callable=check\_downloaded\_files,  op\_kwargs={'dir\_download\_to':DIR\_DOWNLOAD\_TO, \  'files\_name\_for\_download':[val[KEY\_FILE] for val in FILES\_NAME\_FOR\_DOWNLOAD.values()]}, # список наименований файлов  )  # В литературе по Airflow не рекомендуют передвать обрабатываемую информацию между задачами напрямую т.к.:  # 1. Есть ограничения на объем передаваемой информации.  # 2. Возможны ошибки связанные с исчерпанием памяти.  # 3. Это не совпадает с иделологией использования Airflow как оркестратора а не обработчика данных.  # Поэтому принято решение передавать не данные а ссылки на скачанные файлы.  # Реализуем запуск обработки в отдельном окружении, чтобы избежать зависимостей Airflow  # и зависимостей, необходимых для обработки данных  if MERGE\_FILES:  @task.external\_python(task\_id='run\_transform', python=PATH\_TO\_PYTHON\_FOR\_TRANSFORM)  def run\_transform(path\_to\_source\_files: str, source\_files: dict, path\_to\_target\_file: str, target\_file: str):  import sys  import os  sys.path.insert(0, '/home/gbss/airflow/dags')  # import os  # os.chdir('/home/gbss/airflow/dags')  from pythonscripts.procdata\_de\_bd\_hw3.de\_bd\_hw3\_procdata import transform  transform(path\_to\_source\_files, source\_files, path\_to\_target\_file, target\_file)  return  # return os.path.abspath(\_\_file\_\_), os.getcwd(), sys.path  run\_transform\_instance = run\_transform(DIR\_DOWNLOAD\_TO, FILES\_NAME\_FOR\_DOWNLOAD, DIR\_MERGED\_TO, FILE\_MERGED)  else:  run\_transform\_instance = EmptyOperator(task\_id="skip\_transform")      if LOAD\_DATA:  @task.external\_python(task\_id='load\_data', python=PATH\_TO\_PYTHON\_FOR\_LOAD\_TO\_HIVE)  def load\_data(path\_to\_target\_file: str, target\_file: str)->None:  import sys  import os  sys.path.insert(0, '/home/gbss/airflow/dags')  from pythonscripts.procdata\_de\_bd\_hw3.de\_bd\_hw3\_request\_fom\_hive import load\_file\_from\_host\_to\_container\_by\_subprocess,\  load\_file\_from\_container\_to\_hdfs\_by\_subprocess,\  load\_data\_to\_hive  load\_file\_from\_host\_to\_container\_by\_subprocess('namenode-de', path\_to\_target\_file, target\_file,  '/tmp/', target\_file)  load\_file\_from\_container\_to\_hdfs\_by\_subprocess('namenode-de', '/tmp/', target\_file,  '/tmp/', target\_file)  load\_data\_to\_hive('/tmp/' + target\_file)  return  # except Exception as e:  # return 1, f'{e}'  load\_data\_isinstance = load\_data(DIR\_MERGED\_TO, FILE\_MERGED)  else:  load\_data\_isinstance = EmptyOperator(task\_id="skip\_load")  log\_result\_instance >> tasks\_downloads >> files\_is\_downloaded >> run\_transform\_instance >> load\_data\_isinstance  bd\_l3\_s3\_hw() |
| --- |

Приложние № 2. Листинг файла de\_bd\_hw3\_procdata.py

| from datetime import datetime, timezone, timedelta  import csv  import logging  logging.basicConfig(level=logging.INFO)  DATA\_HEADER = ['location', 'date\_time', 'temperature\_celsius']  def transform\_file(location:str, source\_file:str, target\_file:str)->None:  # logging.info(f'{location=}, {source\_file=}, {target\_file=}')  with open(source\_file, 'r') as f:  metadata\_header = f.readline()[:-1].split(',')  print(metadata\_header)  metadata\_data = f.readline()[:-1].split(',')  print(metadata\_data)  metadata = {el:metadata\_data[i] for i, el in enumerate(metadata\_header)}  print(metadata['timezone'])  f.readline() # skip blank line  f.readline() # skip header line  # data\_header = [] #f.readline()[:-1].split(',')  # print(data\_header)  with open(target\_file, 'w', encoding='utf-8') as f\_w\_tz:  csv\_writer = csv.DictWriter(f\_w\_tz, DATA\_HEADER)  csv\_writer.writeheader()  for n, line in enumerate(f):  row\_dict = {}  row = line[:-1].split(',')  # print(row)  # timestamp = datetime.strptime(row[0], "%Y-%m-%dT%H:%M").replace(tzinfo=pytz.timezone(metadata['timezone']))  timestamp = datetime.strptime(row[0], "%Y-%m-%dT%H:%M").replace(tzinfo=timezone(timedelta(days=0, seconds=int(metadata['utc\_offset\_seconds']))))  row\_dict[DATA\_HEADER[0]] = location  row\_dict[DATA\_HEADER[1]] = datetime.strftime(timestamp, "%Y-%m-%dT%H:%M%z")  row\_dict[DATA\_HEADER[2]] = row[1]  csv\_writer.writerow(row\_dict)  return    def merge\_files(path\_to\_source\_files, files\_to\_process, path\_to\_target\_file, target\_file):  logging.info(f'merge\_files. {path\_to\_source\_files=}, {files\_to\_process=}, {path\_to\_target\_file=}, {target\_file=} ')  with open(path\_to\_target\_file + '/' + target\_file, 'w', newline='') as tf:  logging.info(f'merge\_files. open{path\_to\_target\_file + "/" + target\_file}')  csv\_writer = csv.DictWriter(tf, DATA\_HEADER)  csv\_writer.writeheader()  for file in files\_to\_process:  with open(path\_to\_source\_files +'/' + file, 'r', newline='') as sf:  csv\_reader = csv.DictReader(sf)  for row in csv\_reader:  csv\_writer.writerow(row)  logging.info(f'{path\_to\_target\_file}/{target\_file} done.')  return  def transform(path\_to\_source\_files, source\_files, path\_to\_target\_file, target\_file):  logging.info(f'transform. {path\_to\_source\_files=}, {source\_files=}, {path\_to\_target\_file=}, {target\_file=} ')  for loc in source\_files:  transform\_file(loc, path\_to\_source\_files+'/'+source\_files[loc]['file'],  path\_to\_source\_files+'/'+source\_files[loc]['file']+'.tmp')  files\_list = [source\_files[loc]['file']+'.tmp' for loc in source\_files]  merge\_files(path\_to\_source\_files, files\_list, path\_to\_target\_file, target\_file)  return  def main():  print()  FILES\_NAME\_FOR\_DOWNLOAD = {  'chel':{'file': 'open-meteo-chel\_2023-11-30\_2023-12-30.csv',  'coordinates':{'lat':10, 'long': 15}},  'rio':{'file': 'open-meteo-rio\_2023-11-30\_2023-12-30.csv',  'coordinates':{'lat':10, 'long': 15}},  'berl':{'file': 'open-meteo-berl\_2023-11-30\_2023-12-30.csv',  'coordinates':{'lat':10, 'long': 15}},  }  transform('/home/gbss/course\_de\_bd/hw3/downloads\_tmp',FILES\_NAME\_FOR\_DOWNLOAD,\  '/home/gbss/course\_de\_bd/hw3/merged', 'temp\_merged.csv')  if \_\_name\_\_ == '\_\_main\_\_':  main() |
| --- |

Приложение № 3. Листинг файла de\_bd\_hw3\_request\_fom\_hive.py

| """  Модуль для работы с Hive и hdfs  Выполняется в отдельном окружении python  """  from thrift.transport import TSocket  from thrift.transport import TTransport  from thrift.protocol import TBinaryProtocol  from thrift.protocol import TProtocol  from thrift.transport.TTransport import TBufferedTransport  from thrift\_sasl import TSaslClientTransport  # from pyhive\_service import ThriftHive ## не нашел как установить модуль pyhive\_service  from pyhive import hive  # from pyarrow import fs  import subprocess  import logging  logging.basicConfig(level=logging.INFO)  def load\_data\_to\_hive(file\_to\_load\_full\_name):  conn = hive.connect(host='localhost', port=10000, username='admin')  cursor=conn.cursor()  cursor.execute("LOAD DATA INPATH %s OVERWRITE INTO TABLE temperature", (file\_to\_load\_full\_name,))  cursor.close()  conn.close()  return  def run\_cmd(args\_list):  """  run linux commands  отсюда:  https://community.cloudera.com/t5/Community-Articles/Interacting-with-Hadoop-HDFS-using-Python-codes/ta-p/245163  """  # import subprocess  logging.info('Running system command: {0}'.format(' '.join(args\_list)))  proc = subprocess.Popen(args\_list, stdout=subprocess.PIPE, stderr=subprocess.PIPE)  s\_output, s\_err = proc.communicate()  s\_return = proc.returncode  return s\_return, s\_output, s\_err  def load\_file\_from\_container\_to\_hdfs\_by\_subprocess(container\_name, path\_in\_container, file\_name\_in\_container,  path\_in\_hdfs, file\_name\_in\_hdfs):  print()  # hdfs dfs -put /tmp/temp\_with\_tz.csv /tmp/temp\_with\_tz\_.csv  args = ['docker', 'exec', container\_name, 'hdfs', 'dfs', '-put', f'{path\_in\_container}{file\_name\_in\_container}',  f'{path\_in\_hdfs}{file\_name\_in\_hdfs}']  (ret, out, err) = run\_cmd(args)  logging.info(f'{ret=}, {out=}, {err=}')  def load\_file\_from\_host\_to\_container\_by\_subprocess(container\_name, path\_in\_host, file\_name\_in\_host,  path\_in\_container, file\_name\_in\_container):  # docker cp /home/gbss/course\_de\_bd/docker/workspace/temp\_with\_tz.csv namenode-de:/tmp/temp\_with\_tz\_.csv  print()  args = ['docker', 'cp', f'{path\_in\_host}{file\_name\_in\_host}',  f'{container\_name}:{path\_in\_container}{file\_name\_in\_container}']  (ret, out, err) = run\_cmd(args)  logging.info(f'{ret=}, {out=}, {err=}')  def get\_file\_list\_from\_hdfs\_by\_container\_and\_subprocess(container\_name, path\_in\_hdfs, file\_name\_mask=''):  """  Если файлы не найдены: ret=1 out = b'' len(out) == 0, err Содержит строку "No such file or directory"  """  # docker exec namenode-de hdfs dfs -ls /tmp/\*  print()  args = ['docker', 'exec', container\_name, 'hdfs', 'dfs', '-ls', f'{path\_in\_hdfs}{file\_name\_mask}']  (ret, out, err) = run\_cmd(args)  logging.info(f'{ret=}, {out=}, {err=}')  def main():  print()  load\_file\_from\_host\_to\_container\_by\_subprocess('namenode-de', '/home/gbss/course\_de\_bd/hw3/merged/', 'temp\_merged.csv',  '/tmp/', 'temp\_merged.csv')  load\_file\_from\_container\_to\_hdfs\_by\_subprocess('namenode-de', '/tmp/', 'temp\_merged.csv',  '/tmp/', 'temp\_merged.csv')  # get\_file\_list\_from\_hdfs\_by\_container\_and\_subprocess('namenode-de', '/tmp/', 'open-meteo-55.15N61.38E223m\_temp.cs')  # loc = get\_locations\_codes()  # logging.info(loc)  # coordinates = get\_locations\_coordinates(loc)  # logging.info(f'{coordinates=}')  load\_data\_to\_hive('/tmp/'+'temp\_merged.csv')  if \_\_name\_\_ == '\_\_main\_\_':  main() |
| --- |

Приложение № 4. Логи работы DAG

log\_result

| gbus  \*\*\* Found local files:  \*\*\* \* /home/gbss/airflow/logs/dag\_id=bd\_l3\_s3\_hw/run\_id=manual\_\_2024-03-08T13:52:58.803733+00:00/task\_id=log\_result/attempt=1.log  [2024-03-08, 13:53:02 UTC] {taskinstance.py:1159} INFO - Dependencies all met for dep\_context=non-requeueable deps ti=<TaskInstance: bd\_l3\_s3\_hw.log\_result manual\_\_2024-03-08T13:52:58.803733+00:00 [queued]>  [2024-03-08, 13:53:02 UTC] {taskinstance.py:1159} INFO - Dependencies all met for dep\_context=requeueable deps ti=<TaskInstance: bd\_l3\_s3\_hw.log\_result manual\_\_2024-03-08T13:52:58.803733+00:00 [queued]>  [2024-03-08, 13:53:02 UTC] {taskinstance.py:1361} INFO - Starting attempt 1 of 1  [2024-03-08, 13:53:02 UTC] {taskinstance.py:1382} INFO - Executing <Task(\_PythonDecoratedOperator): log\_result> on 2024-03-08 13:52:58.803733+00:00  [2024-03-08, 13:53:02 UTC] {standard\_task\_runner.py:57} INFO - Started process 67298 to run task  [2024-03-08, 13:53:02 UTC] {standard\_task\_runner.py:84} INFO - Running: ['airflow', 'tasks', 'run', 'bd\_l3\_s3\_hw', 'log\_result', 'manual\_\_2024-03-08T13:52:58.803733+00:00', '--job-id', '369', '--raw', '--subdir', 'DAGS\_FOLDER/bd\_s3\_hw.py', '--cfg-path', '/tmp/tmp55noqm5s']  [2024-03-08, 13:53:02 UTC] {standard\_task\_runner.py:85} INFO - Job 369: Subtask log\_result  [2024-03-08, 13:53:02 UTC] {task\_command.py:416} INFO - Running <TaskInstance: bd\_l3\_s3\_hw.log\_result manual\_\_2024-03-08T13:52:58.803733+00:00 [running]> on host gbus  [2024-03-08, 13:53:02 UTC] {taskinstance.py:1662} INFO - Exporting env vars: AIRFLOW\_CTX\_DAG\_OWNER='airflow' AIRFLOW\_CTX\_DAG\_ID='bd\_l3\_s3\_hw' AIRFLOW\_CTX\_TASK\_ID='log\_result' AIRFLOW\_CTX\_EXECUTION\_DATE='2024-03-08T13:52:58.803733+00:00' AIRFLOW\_CTX\_TRY\_NUMBER='1' AIRFLOW\_CTX\_DAG\_RUN\_ID='manual\_\_2024-03-08T13:52:58.803733+00:00'  [2024-03-08, 13:53:02 UTC] {bd\_s3\_hw.py:112} INFO - {'chel': {'coordinates': {'lat': 55.149384, 'long': 61.38}, 'file': 'open-meteo-chel\_2023-11-30\_2023-12-30.csv'}, 'rio': {'coordinates': {'lat': -22.90278, 'long': -43.2075}, 'file': 'open-meteo-rio\_2023-11-30\_2023-12-30.csv'}, 'berl': {'coordinates': {'lat': 52.5244, 'long': 13.4105}, 'file': 'open-meteo-berl\_2023-11-30\_2023-12-30.csv'}}  [2024-03-08, 13:53:02 UTC] {python.py:194} INFO - Done. Returned value was: None  [2024-03-08, 13:53:03 UTC] {taskinstance.py:1400} INFO - Marking task as SUCCESS. dag\_id=bd\_l3\_s3\_hw, task\_id=log\_result, execution\_date=20240308T135258, start\_date=20240308T135302, end\_date=20240308T135302  [2024-03-08, 13:53:03 UTC] {local\_task\_job\_runner.py:228} INFO - Task exited with return code 0  [2024-03-08, 13:53:03 UTC] {taskinstance.py:2778} INFO - 3 downstream tasks scheduled from follow-on schedule check |
| --- |

download\_chel

| gbus  \*\*\* Found local files:  \*\*\* \* /home/gbss/airflow/logs/dag\_id=bd\_l3\_s3\_hw/run\_id=manual\_\_2024-03-08T13:52:58.803733+00:00/task\_id=download\_chel/attempt=1.log  [2024-03-08, 13:53:05 UTC] {taskinstance.py:1159} INFO - Dependencies all met for dep\_context=non-requeueable deps ti=<TaskInstance: bd\_l3\_s3\_hw.download\_chel manual\_\_2024-03-08T13:52:58.803733+00:00 [queued]>  [2024-03-08, 13:53:05 UTC] {taskinstance.py:1159} INFO - Dependencies all met for dep\_context=requeueable deps ti=<TaskInstance: bd\_l3\_s3\_hw.download\_chel manual\_\_2024-03-08T13:52:58.803733+00:00 [queued]>  [2024-03-08, 13:53:05 UTC] {taskinstance.py:1361} INFO - Starting attempt 1 of 1  [2024-03-08, 13:53:05 UTC] {taskinstance.py:1382} INFO - Executing <Task(BashOperator): download\_chel> on 2024-03-08 13:52:58.803733+00:00  [2024-03-08, 13:53:05 UTC] {standard\_task\_runner.py:57} INFO - Started process 67302 to run task  [2024-03-08, 13:53:05 UTC] {standard\_task\_runner.py:84} INFO - Running: ['airflow', 'tasks', 'run', 'bd\_l3\_s3\_hw', 'download\_chel', 'manual\_\_2024-03-08T13:52:58.803733+00:00', '--job-id', '370', '--raw', '--subdir', 'DAGS\_FOLDER/bd\_s3\_hw.py', '--cfg-path', '/tmp/tmp390jbiva']  [2024-03-08, 13:53:05 UTC] {standard\_task\_runner.py:85} INFO - Job 370: Subtask download\_chel  [2024-03-08, 13:53:05 UTC] {task\_command.py:416} INFO - Running <TaskInstance: bd\_l3\_s3\_hw.download\_chel manual\_\_2024-03-08T13:52:58.803733+00:00 [running]> on host gbus  [2024-03-08, 13:53:05 UTC] {taskinstance.py:1662} INFO - Exporting env vars: AIRFLOW\_CTX\_DAG\_OWNER='airflow' AIRFLOW\_CTX\_DAG\_ID='bd\_l3\_s3\_hw' AIRFLOW\_CTX\_TASK\_ID='download\_chel' AIRFLOW\_CTX\_EXECUTION\_DATE='2024-03-08T13:52:58.803733+00:00' AIRFLOW\_CTX\_TRY\_NUMBER='1' AIRFLOW\_CTX\_DAG\_RUN\_ID='manual\_\_2024-03-08T13:52:58.803733+00:00'  [2024-03-08, 13:53:05 UTC] {subprocess.py:63} INFO - Tmp dir root location: /tmp  [2024-03-08, 13:53:05 UTC] {subprocess.py:75} INFO - Running command: ['/usr/bin/bash', '-c', 'curl -sSLo /home/gbss/course\_de\_bd/hw3/downloads\_tmp/open-meteo-chel\_2023-11-30\_2023-12-30.csv "https://archive-api.open-meteo.com/v1/archive?latitude=55.149384&longitude=61.38&start\_date=2023-11-30&end\_date=2023-12-30&hourly=temperature\_2m&timezone=auto&format=csv"']  [2024-03-08, 13:53:05 UTC] {subprocess.py:86} INFO - Output:  [2024-03-08, 13:53:06 UTC] {subprocess.py:97} INFO - Command exited with return code 0  [2024-03-08, 13:53:06 UTC] {taskinstance.py:1400} INFO - Marking task as SUCCESS. dag\_id=bd\_l3\_s3\_hw, task\_id=download\_chel, execution\_date=20240308T135258, start\_date=20240308T135305, end\_date=20240308T135306  [2024-03-08, 13:53:06 UTC] {local\_task\_job\_runner.py:228} INFO - Task exited with return code 0  [2024-03-08, 13:53:06 UTC] {taskinstance.py:2778} INFO - 0 downstream tasks scheduled from follow-on schedule check |
| --- |

download\_rio

| gbus  \*\*\* Found local files:  \*\*\* \* /home/gbss/airflow/logs/dag\_id=bd\_l3\_s3\_hw/run\_id=manual\_\_2024-03-08T13:52:58.803733+00:00/task\_id=download\_rio/attempt=1.log  [2024-03-08, 13:53:08 UTC] {taskinstance.py:1159} INFO - Dependencies all met for dep\_context=non-requeueable deps ti=<TaskInstance: bd\_l3\_s3\_hw.download\_rio manual\_\_2024-03-08T13:52:58.803733+00:00 [queued]>  [2024-03-08, 13:53:08 UTC] {taskinstance.py:1159} INFO - Dependencies all met for dep\_context=requeueable deps ti=<TaskInstance: bd\_l3\_s3\_hw.download\_rio manual\_\_2024-03-08T13:52:58.803733+00:00 [queued]>  [2024-03-08, 13:53:08 UTC] {taskinstance.py:1361} INFO - Starting attempt 1 of 1  [2024-03-08, 13:53:08 UTC] {taskinstance.py:1382} INFO - Executing <Task(BashOperator): download\_rio> on 2024-03-08 13:52:58.803733+00:00  [2024-03-08, 13:53:08 UTC] {standard\_task\_runner.py:57} INFO - Started process 67307 to run task  [2024-03-08, 13:53:08 UTC] {standard\_task\_runner.py:84} INFO - Running: ['airflow', 'tasks', 'run', 'bd\_l3\_s3\_hw', 'download\_rio', 'manual\_\_2024-03-08T13:52:58.803733+00:00', '--job-id', '371', '--raw', '--subdir', 'DAGS\_FOLDER/bd\_s3\_hw.py', '--cfg-path', '/tmp/tmpt3zvp5f8']  [2024-03-08, 13:53:08 UTC] {standard\_task\_runner.py:85} INFO - Job 371: Subtask download\_rio  [2024-03-08, 13:53:08 UTC] {task\_command.py:416} INFO - Running <TaskInstance: bd\_l3\_s3\_hw.download\_rio manual\_\_2024-03-08T13:52:58.803733+00:00 [running]> on host gbus  [2024-03-08, 13:53:08 UTC] {taskinstance.py:1662} INFO - Exporting env vars: AIRFLOW\_CTX\_DAG\_OWNER='airflow' AIRFLOW\_CTX\_DAG\_ID='bd\_l3\_s3\_hw' AIRFLOW\_CTX\_TASK\_ID='download\_rio' AIRFLOW\_CTX\_EXECUTION\_DATE='2024-03-08T13:52:58.803733+00:00' AIRFLOW\_CTX\_TRY\_NUMBER='1' AIRFLOW\_CTX\_DAG\_RUN\_ID='manual\_\_2024-03-08T13:52:58.803733+00:00'  [2024-03-08, 13:53:08 UTC] {subprocess.py:63} INFO - Tmp dir root location: /tmp  [2024-03-08, 13:53:08 UTC] {subprocess.py:75} INFO - Running command: ['/usr/bin/bash', '-c', 'curl -sSLo /home/gbss/course\_de\_bd/hw3/downloads\_tmp/open-meteo-rio\_2023-11-30\_2023-12-30.csv "https://archive-api.open-meteo.com/v1/archive?latitude=-22.90278&longitude=-43.2075&start\_date=2023-11-30&end\_date=2023-12-30&hourly=temperature\_2m&timezone=auto&format=csv"']  [2024-03-08, 13:53:08 UTC] {subprocess.py:86} INFO - Output:  [2024-03-08, 13:53:08 UTC] {subprocess.py:97} INFO - Command exited with return code 0  [2024-03-08, 13:53:08 UTC] {taskinstance.py:1400} INFO - Marking task as SUCCESS. dag\_id=bd\_l3\_s3\_hw, task\_id=download\_rio, execution\_date=20240308T135258, start\_date=20240308T135308, end\_date=20240308T135308  [2024-03-08, 13:53:09 UTC] {local\_task\_job\_runner.py:228} INFO - Task exited with return code 0  [2024-03-08, 13:53:09 UTC] {taskinstance.py:2778} INFO - 0 downstream tasks scheduled from follow-on schedule check |
| --- |

download\_berl

| gbus  \*\*\* Found local files:  \*\*\* \* /home/gbss/airflow/logs/dag\_id=bd\_l3\_s3\_hw/run\_id=manual\_\_2024-03-08T13:52:58.803733+00:00/task\_id=download\_berl/attempt=1.log  [2024-03-08, 13:53:11 UTC] {taskinstance.py:1159} INFO - Dependencies all met for dep\_context=non-requeueable deps ti=<TaskInstance: bd\_l3\_s3\_hw.download\_berl manual\_\_2024-03-08T13:52:58.803733+00:00 [queued]>  [2024-03-08, 13:53:11 UTC] {taskinstance.py:1159} INFO - Dependencies all met for dep\_context=requeueable deps ti=<TaskInstance: bd\_l3\_s3\_hw.download\_berl manual\_\_2024-03-08T13:52:58.803733+00:00 [queued]>  [2024-03-08, 13:53:11 UTC] {taskinstance.py:1361} INFO - Starting attempt 1 of 1  [2024-03-08, 13:53:11 UTC] {taskinstance.py:1382} INFO - Executing <Task(BashOperator): download\_berl> on 2024-03-08 13:52:58.803733+00:00  [2024-03-08, 13:53:11 UTC] {standard\_task\_runner.py:57} INFO - Started process 67312 to run task  [2024-03-08, 13:53:11 UTC] {standard\_task\_runner.py:84} INFO - Running: ['airflow', 'tasks', 'run', 'bd\_l3\_s3\_hw', 'download\_berl', 'manual\_\_2024-03-08T13:52:58.803733+00:00', '--job-id', '372', '--raw', '--subdir', 'DAGS\_FOLDER/bd\_s3\_hw.py', '--cfg-path', '/tmp/tmpla06kioo']  [2024-03-08, 13:53:11 UTC] {standard\_task\_runner.py:85} INFO - Job 372: Subtask download\_berl  [2024-03-08, 13:53:11 UTC] {task\_command.py:416} INFO - Running <TaskInstance: bd\_l3\_s3\_hw.download\_berl manual\_\_2024-03-08T13:52:58.803733+00:00 [running]> on host gbus  [2024-03-08, 13:53:11 UTC] {taskinstance.py:1662} INFO - Exporting env vars: AIRFLOW\_CTX\_DAG\_OWNER='airflow' AIRFLOW\_CTX\_DAG\_ID='bd\_l3\_s3\_hw' AIRFLOW\_CTX\_TASK\_ID='download\_berl' AIRFLOW\_CTX\_EXECUTION\_DATE='2024-03-08T13:52:58.803733+00:00' AIRFLOW\_CTX\_TRY\_NUMBER='1' AIRFLOW\_CTX\_DAG\_RUN\_ID='manual\_\_2024-03-08T13:52:58.803733+00:00'  [2024-03-08, 13:53:11 UTC] {subprocess.py:63} INFO - Tmp dir root location: /tmp  [2024-03-08, 13:53:11 UTC] {subprocess.py:75} INFO - Running command: ['/usr/bin/bash', '-c', 'curl -sSLo /home/gbss/course\_de\_bd/hw3/downloads\_tmp/open-meteo-berl\_2023-11-30\_2023-12-30.csv "https://archive-api.open-meteo.com/v1/archive?latitude=52.5244&longitude=13.4105&start\_date=2023-11-30&end\_date=2023-12-30&hourly=temperature\_2m&timezone=auto&format=csv"']  [2024-03-08, 13:53:11 UTC] {subprocess.py:86} INFO - Output:  [2024-03-08, 13:53:11 UTC] {subprocess.py:97} INFO - Command exited with return code 0  [2024-03-08, 13:53:11 UTC] {taskinstance.py:1400} INFO - Marking task as SUCCESS. dag\_id=bd\_l3\_s3\_hw, task\_id=download\_berl, execution\_date=20240308T135258, start\_date=20240308T135311, end\_date=20240308T135311  [2024-03-08, 13:53:11 UTC] {local\_task\_job\_runner.py:228} INFO - Task exited with return code 0  [2024-03-08, 13:53:11 UTC] {taskinstance.py:2778} INFO - 1 downstream tasks scheduled from follow-on schedule check |
| --- |

condition\_is\_True

| gbus  \*\*\* Found local files:  \*\*\* \* /home/gbss/airflow/logs/dag\_id=bd\_l3\_s3\_hw/run\_id=manual\_\_2024-03-08T13:52:58.803733+00:00/task\_id=condition\_is\_True/attempt=1.log  [2024-03-08, 13:53:14 UTC] {taskinstance.py:1159} INFO - Dependencies all met for dep\_context=non-requeueable deps ti=<TaskInstance: bd\_l3\_s3\_hw.condition\_is\_True manual\_\_2024-03-08T13:52:58.803733+00:00 [queued]>  [2024-03-08, 13:53:14 UTC] {taskinstance.py:1159} INFO - Dependencies all met for dep\_context=requeueable deps ti=<TaskInstance: bd\_l3\_s3\_hw.condition\_is\_True manual\_\_2024-03-08T13:52:58.803733+00:00 [queued]>  [2024-03-08, 13:53:14 UTC] {taskinstance.py:1361} INFO - Starting attempt 1 of 1  [2024-03-08, 13:53:14 UTC] {taskinstance.py:1382} INFO - Executing <Task(ShortCircuitOperator): condition\_is\_True> on 2024-03-08 13:52:58.803733+00:00  [2024-03-08, 13:53:14 UTC] {standard\_task\_runner.py:57} INFO - Started process 67338 to run task  [2024-03-08, 13:53:14 UTC] {standard\_task\_runner.py:84} INFO - Running: ['airflow', 'tasks', 'run', 'bd\_l3\_s3\_hw', 'condition\_is\_True', 'manual\_\_2024-03-08T13:52:58.803733+00:00', '--job-id', '373', '--raw', '--subdir', 'DAGS\_FOLDER/bd\_s3\_hw.py', '--cfg-path', '/tmp/tmppot0pfje']  [2024-03-08, 13:53:14 UTC] {standard\_task\_runner.py:85} INFO - Job 373: Subtask condition\_is\_True  [2024-03-08, 13:53:14 UTC] {task\_command.py:416} INFO - Running <TaskInstance: bd\_l3\_s3\_hw.condition\_is\_True manual\_\_2024-03-08T13:52:58.803733+00:00 [running]> on host gbus  [2024-03-08, 13:53:14 UTC] {taskinstance.py:1662} INFO - Exporting env vars: AIRFLOW\_CTX\_DAG\_OWNER='airflow' AIRFLOW\_CTX\_DAG\_ID='bd\_l3\_s3\_hw' AIRFLOW\_CTX\_TASK\_ID='condition\_is\_True' AIRFLOW\_CTX\_EXECUTION\_DATE='2024-03-08T13:52:58.803733+00:00' AIRFLOW\_CTX\_TRY\_NUMBER='1' AIRFLOW\_CTX\_DAG\_RUN\_ID='manual\_\_2024-03-08T13:52:58.803733+00:00'  [2024-03-08, 13:53:14 UTC] {bd\_s3\_hw.py:50} INFO - /home/gbss/course\_de\_bd/hw3/downloads\_tmp/open-meteo-chel\_2023-11-30\_2023-12-30.csv is exists True  [2024-03-08, 13:53:14 UTC] {bd\_s3\_hw.py:50} INFO - /home/gbss/course\_de\_bd/hw3/downloads\_tmp/open-meteo-rio\_2023-11-30\_2023-12-30.csv is exists True  [2024-03-08, 13:53:14 UTC] {bd\_s3\_hw.py:50} INFO - /home/gbss/course\_de\_bd/hw3/downloads\_tmp/open-meteo-berl\_2023-11-30\_2023-12-30.csv is exists True  [2024-03-08, 13:53:14 UTC] {python.py:194} INFO - Done. Returned value was: True  [2024-03-08, 13:53:14 UTC] {python.py:264} INFO - Condition result is True  [2024-03-08, 13:53:14 UTC] {python.py:267} INFO - Proceeding with downstream tasks...  [2024-03-08, 13:53:14 UTC] {taskinstance.py:1400} INFO - Marking task as SUCCESS. dag\_id=bd\_l3\_s3\_hw, task\_id=condition\_is\_True, execution\_date=20240308T135258, start\_date=20240308T135314, end\_date=20240308T135314  [2024-03-08, 13:53:14 UTC] {local\_task\_job\_runner.py:228} INFO - Task exited with return code 0  [2024-03-08, 13:53:14 UTC] {taskinstance.py:2778} INFO - 1 downstream tasks scheduled from follow-on schedule check |
| --- |

run\_transform

| gbus  \*\*\* Found local files:  \*\*\* \* /home/gbss/airflow/logs/dag\_id=bd\_l3\_s3\_hw/run\_id=manual\_\_2024-03-08T13:52:58.803733+00:00/task\_id=run\_transform/attempt=1.log  [2024-03-08, 13:53:17 UTC] {taskinstance.py:1159} INFO - Dependencies all met for dep\_context=non-requeueable deps ti=<TaskInstance: bd\_l3\_s3\_hw.run\_transform manual\_\_2024-03-08T13:52:58.803733+00:00 [queued]>  [2024-03-08, 13:53:17 UTC] {taskinstance.py:1159} INFO - Dependencies all met for dep\_context=requeueable deps ti=<TaskInstance: bd\_l3\_s3\_hw.run\_transform manual\_\_2024-03-08T13:52:58.803733+00:00 [queued]>  [2024-03-08, 13:53:17 UTC] {taskinstance.py:1361} INFO - Starting attempt 1 of 1  [2024-03-08, 13:53:17 UTC] {taskinstance.py:1382} INFO - Executing <Task(\_PythonExternalDecoratedOperator): run\_transform> on 2024-03-08 13:52:58.803733+00:00  [2024-03-08, 13:53:17 UTC] {standard\_task\_runner.py:57} INFO - Started process 67361 to run task  [2024-03-08, 13:53:17 UTC] {standard\_task\_runner.py:84} INFO - Running: ['airflow', 'tasks', 'run', 'bd\_l3\_s3\_hw', 'run\_transform', 'manual\_\_2024-03-08T13:52:58.803733+00:00', '--job-id', '374', '--raw', '--subdir', 'DAGS\_FOLDER/bd\_s3\_hw.py', '--cfg-path', '/tmp/tmp\_n6pe\_6y']  [2024-03-08, 13:53:17 UTC] {standard\_task\_runner.py:85} INFO - Job 374: Subtask run\_transform  [2024-03-08, 13:53:17 UTC] {task\_command.py:416} INFO - Running <TaskInstance: bd\_l3\_s3\_hw.run\_transform manual\_\_2024-03-08T13:52:58.803733+00:00 [running]> on host gbus  [2024-03-08, 13:53:17 UTC] {taskinstance.py:1662} INFO - Exporting env vars: AIRFLOW\_CTX\_DAG\_OWNER='airflow' AIRFLOW\_CTX\_DAG\_ID='bd\_l3\_s3\_hw' AIRFLOW\_CTX\_TASK\_ID='run\_transform' AIRFLOW\_CTX\_EXECUTION\_DATE='2024-03-08T13:52:58.803733+00:00' AIRFLOW\_CTX\_TRY\_NUMBER='1' AIRFLOW\_CTX\_DAG\_RUN\_ID='manual\_\_2024-03-08T13:52:58.803733+00:00'  [2024-03-08, 13:53:17 UTC] {python.py:779} WARNING - When checking for Airflow installed in venv got Command '['/home/gbss/course\_de\_etl/pd/bin/python', '-c', 'from airflow import \_\_version\_\_; print(\_\_version\_\_)']' returned non-zero exit status 1.  [2024-03-08, 13:53:17 UTC] {python.py:780} WARNING - This means that Airflow is not properly installed by /home/gbss/course\_de\_etl/pd/bin/python. Airflow context keys will not be available. Please Install Airflow 2.7.3 in your environment to access them.  [2024-03-08, 13:53:17 UTC] {process\_utils.py:182} INFO - Executing cmd: /home/gbss/course\_de\_etl/pd/bin/python /tmp/tmd7cou5hpn/script.py /tmp/tmd7cou5hpn/script.in /tmp/tmd7cou5hpn/script.out /tmp/tmd7cou5hpn/string\_args.txt /tmp/tmd7cou5hpn/termination.log  [2024-03-08, 13:53:17 UTC] {process\_utils.py:186} INFO - Output:  [2024-03-08, 13:53:17 UTC] {process\_utils.py:190} INFO - INFO:root:transform. path\_to\_source\_files='/home/gbss/course\_de\_bd/hw3/downloads\_tmp', source\_files={'chel': {'coordinates': {'lat': 55.149384, 'long': 61.38}, 'file': 'open-meteo-chel\_2023-11-30\_2023-12-30.csv'}, 'rio': {'coordinates': {'lat': -22.90278, 'long': -43.2075}, 'file': 'open-meteo-rio\_2023-11-30\_2023-12-30.csv'}, 'berl': {'coordinates': {'lat': 52.5244, 'long': 13.4105}, 'file': 'open-meteo-berl\_2023-11-30\_2023-12-30.csv'}}, path\_to\_target\_file='/home/gbss/course\_de\_bd/hw3/merged', target\_file='temp\_merged.csv'  [2024-03-08, 13:53:17 UTC] {process\_utils.py:190} INFO - INFO:root:merge\_files. path\_to\_source\_files='/home/gbss/course\_de\_bd/hw3/downloads\_tmp', files\_to\_process=['open-meteo-chel\_2023-11-30\_2023-12-30.csv.tmp', 'open-meteo-rio\_2023-11-30\_2023-12-30.csv.tmp', 'open-meteo-berl\_2023-11-30\_2023-12-30.csv.tmp'], path\_to\_target\_file='/home/gbss/course\_de\_bd/hw3/merged', target\_file='temp\_merged.csv'  [2024-03-08, 13:53:17 UTC] {process\_utils.py:190} INFO - INFO:root:merge\_files. open/home/gbss/course\_de\_bd/hw3/merged/temp\_merged.csv  [2024-03-08, 13:53:17 UTC] {process\_utils.py:190} INFO - INFO:root:/home/gbss/course\_de\_bd/hw3/merged/temp\_merged.csv done.  [2024-03-08, 13:53:17 UTC] {process\_utils.py:190} INFO - ['latitude', 'longitude', 'elevation', 'utc\_offset\_seconds', 'timezone', 'timezone\_abbreviation']  [2024-03-08, 13:53:17 UTC] {process\_utils.py:190} INFO - ['55.149384', '61.38', '259.0', '18000', 'Asia/Yekaterinburg', '+05']  [2024-03-08, 13:53:17 UTC] {process\_utils.py:190} INFO - Asia/Yekaterinburg  [2024-03-08, 13:53:17 UTC] {process\_utils.py:190} INFO - ['latitude', 'longitude', 'elevation', 'utc\_offset\_seconds', 'timezone', 'timezone\_abbreviation']  [2024-03-08, 13:53:17 UTC] {process\_utils.py:190} INFO - ['-22.88225', '-43.07611', '3.0', '-10800', 'America/Sao\_Paulo', '-03']  [2024-03-08, 13:53:17 UTC] {process\_utils.py:190} INFO - America/Sao\_Paulo  [2024-03-08, 13:53:17 UTC] {process\_utils.py:190} INFO - ['latitude', 'longitude', 'elevation', 'utc\_offset\_seconds', 'timezone', 'timezone\_abbreviation']  [2024-03-08, 13:53:17 UTC] {process\_utils.py:190} INFO - ['52.54833', '13.407822', '46.0', '3600', 'Europe/Berlin', 'CET']  [2024-03-08, 13:53:17 UTC] {process\_utils.py:190} INFO - Europe/Berlin  [2024-03-08, 13:53:17 UTC] {python.py:194} INFO - Done. Returned value was: None  [2024-03-08, 13:53:17 UTC] {taskinstance.py:1400} INFO - Marking task as SUCCESS. dag\_id=bd\_l3\_s3\_hw, task\_id=run\_transform, execution\_date=20240308T135258, start\_date=20240308T135317, end\_date=20240308T135317  [2024-03-08, 13:53:17 UTC] {local\_task\_job\_runner.py:228} INFO - Task exited with return code 0  [2024-03-08, 13:53:17 UTC] {taskinstance.py:2778} INFO - 1 downstream tasks scheduled from follow-on schedule check |
| --- |

load\_data

| gbus  \*\*\* Found local files:  \*\*\* \* /home/gbss/airflow/logs/dag\_id=bd\_l3\_s3\_hw/run\_id=manual\_\_2024-03-08T13:52:58.803733+00:00/task\_id=load\_data/attempt=1.log  [2024-03-08, 13:53:20 UTC] {taskinstance.py:1159} INFO - Dependencies all met for dep\_context=non-requeueable deps ti=<TaskInstance: bd\_l3\_s3\_hw.load\_data manual\_\_2024-03-08T13:52:58.803733+00:00 [queued]>  [2024-03-08, 13:53:20 UTC] {taskinstance.py:1159} INFO - Dependencies all met for dep\_context=requeueable deps ti=<TaskInstance: bd\_l3\_s3\_hw.load\_data manual\_\_2024-03-08T13:52:58.803733+00:00 [queued]>  [2024-03-08, 13:53:20 UTC] {taskinstance.py:1361} INFO - Starting attempt 1 of 1  [2024-03-08, 13:53:20 UTC] {taskinstance.py:1382} INFO - Executing <Task(\_PythonExternalDecoratedOperator): load\_data> on 2024-03-08 13:52:58.803733+00:00  [2024-03-08, 13:53:20 UTC] {standard\_task\_runner.py:57} INFO - Started process 67408 to run task  [2024-03-08, 13:53:20 UTC] {standard\_task\_runner.py:84} INFO - Running: ['airflow', 'tasks', 'run', 'bd\_l3\_s3\_hw', 'load\_data', 'manual\_\_2024-03-08T13:52:58.803733+00:00', '--job-id', '375', '--raw', '--subdir', 'DAGS\_FOLDER/bd\_s3\_hw.py', '--cfg-path', '/tmp/tmpmvigeb7v']  [2024-03-08, 13:53:20 UTC] {standard\_task\_runner.py:85} INFO - Job 375: Subtask load\_data  [2024-03-08, 13:53:20 UTC] {task\_command.py:416} INFO - Running <TaskInstance: bd\_l3\_s3\_hw.load\_data manual\_\_2024-03-08T13:52:58.803733+00:00 [running]> on host gbus  [2024-03-08, 13:53:20 UTC] {taskinstance.py:1662} INFO - Exporting env vars: AIRFLOW\_CTX\_DAG\_OWNER='airflow' AIRFLOW\_CTX\_DAG\_ID='bd\_l3\_s3\_hw' AIRFLOW\_CTX\_TASK\_ID='load\_data' AIRFLOW\_CTX\_EXECUTION\_DATE='2024-03-08T13:52:58.803733+00:00' AIRFLOW\_CTX\_TRY\_NUMBER='1' AIRFLOW\_CTX\_DAG\_RUN\_ID='manual\_\_2024-03-08T13:52:58.803733+00:00'  [2024-03-08, 13:53:20 UTC] {python.py:779} WARNING - When checking for Airflow installed in venv got Command '['/home/gbss/course\_de\_bd/venv\_hadoop\_cluster/bin/python', '-c', 'from airflow import \_\_version\_\_; print(\_\_version\_\_)']' returned non-zero exit status 1.  [2024-03-08, 13:53:20 UTC] {python.py:780} WARNING - This means that Airflow is not properly installed by /home/gbss/course\_de\_bd/venv\_hadoop\_cluster/bin/python. Airflow context keys will not be available. Please Install Airflow 2.7.3 in your environment to access them.  [2024-03-08, 13:53:20 UTC] {process\_utils.py:182} INFO - Executing cmd: /home/gbss/course\_de\_bd/venv\_hadoop\_cluster/bin/python /tmp/tmd2tdqgmn8/script.py /tmp/tmd2tdqgmn8/script.in /tmp/tmd2tdqgmn8/script.out /tmp/tmd2tdqgmn8/string\_args.txt /tmp/tmd2tdqgmn8/termination.log  [2024-03-08, 13:53:20 UTC] {process\_utils.py:186} INFO - Output:  [2024-03-08, 13:53:20 UTC] {process\_utils.py:190} INFO - INFO:root:Running system command: docker cp /home/gbss/course\_de\_bd/hw3/mergedtemp\_merged.csv namenode-de:/tmp/temp\_merged.csv  [2024-03-08, 13:53:20 UTC] {process\_utils.py:190} INFO - INFO:root:ret=1, out=b'', err=b'lstat /home/gbss/course\_de\_bd/hw3/mergedtemp\_merged.csv: no such file or directory\n'  [2024-03-08, 13:53:20 UTC] {process\_utils.py:190} INFO - INFO:root:Running system command: docker exec namenode-de hdfs dfs -put /tmp/temp\_merged.csv /tmp/temp\_merged.csv  [2024-03-08, 13:53:23 UTC] {process\_utils.py:190} INFO - INFO:root:ret=0, out=b'', err=b''  [2024-03-08, 13:53:23 UTC] {process\_utils.py:190} INFO - INFO:pyhive.hive:USE `default`  [2024-03-08, 13:53:23 UTC] {process\_utils.py:190} INFO - INFO:pyhive.hive:LOAD DATA INPATH '/tmp/temp\_merged.csv' OVERWRITE INTO TABLE temperature  [2024-03-08, 13:53:24 UTC] {process\_utils.py:190} INFO -  [2024-03-08, 13:53:24 UTC] {python.py:194} INFO - Done. Returned value was: None  [2024-03-08, 13:53:24 UTC] {taskinstance.py:1400} INFO - Marking task as SUCCESS. dag\_id=bd\_l3\_s3\_hw, task\_id=load\_data, execution\_date=20240308T135258, start\_date=20240308T135320, end\_date=20240308T135324  [2024-03-08, 13:53:24 UTC] {local\_task\_job\_runner.py:228} INFO - Task exited with return code 0  [2024-03-08, 13:53:24 UTC] {taskinstance.py:2778} INFO - 0 downstream tasks scheduled from follow-on schedule check |
| --- |