Geekbrains

Развертывание, внедрение и эксплуатация системы планирования и мониторинга рабочих процессов на основе Apache Airflow в ООО РКЦ

Программа:
Разработчик — Data Engineer

Выполнил:
Морданов Дмитрий Анатольевич

Киров
2024

Содержание

Введение

Описание задачи: Существует организация ООО РКЦ в г. Киров. Организация занимается выполнением расчетов начисления платы за жилищно коммунальные услуги населению г. Кирова. Организация использует несколько расчетных комплексов, основывающихся на SQL - серверах различных типов (так сложилось исторически, и собрать все расчеты в одном комплексе не представляется возможным). Расчетные комплексы обеспечивают непосредственно расчеты, но существуют задачи, связанные с получением нестандартных (а порой и одноразовых) отчетов по существующим данных одного или нескольких расчетных комплексов сразу.

Проблема, требующая решения: Выполнение задач, описанных выше осуществлялось специалистами ИТ-отдела, с помощью инструментов доступа к данным соответствующего SQL - сервера, написание SQL - запроса, его выполнение, передачу полученных данных в MS Excel, выполнение дополнительных манипуляций в MS Excel (наложение фильтра, корректировка значений). Иногда приходилось выполнять слияние с данными, полученными из другой расчетной системы для построения объединенного отчета.

Требовалось внедрение какой - либо системы, которая смогла бы снять большую часть временных затрат со специалистов ИТ - отдела компании.

Тема проекта: Развертывание, внедрение и эксплуатация системы планирования и мониторинга рабочих процессов на основе Apache Airflow в ООО РКЦ

Цель: Получить работоспособную систему планирования и мониторинга рабочих ETL процессов, требуемых для функционирования организации. Процессы должны выполняться в автоматическом режиме по наступлению события времени, или при запуске вручную. Запуск вручную может производиться неквалифицированным специалистом организации.

Какую проблему решает: В ООО РКЦ повышает эффективность работы с информацией, более оперативная работа с данными, освобождение ИТ-отдела от рутинных задач.

План работы:

1. Изучить литературу и ресурсы интернета, касающихся развертывания и внедрения Apache Airflow.
2. Рассмотреть основные методы обработки данных в организации рабочих процессов ООО РКЦ.

3. Ознакомиться с основными принципами построения рабочих задач, и внедрить их на практике.
4. Развернуть среду выполнения Apache Airflow на ресурсах компании ООО РКЦ
5. Добиться корректной работы Apache Airflow
6. Разработать несколько рабочих процессов в виде файлов DAG, на основе которых возможно дальнейшее развитие проекта.

Инструменты: Linux, Docker, Python, PostgreSQL, FireBird, Airflow, Git.

Работу выполнял самостоятельно.

Глава 1. Проблемы управления данными в организации.

В современном мире организации сталкиваются с невероятным объемом данных, который постоянно растет. Эти данные - ключевой актив, на котором строятся стратегии, принимаются решения и создаются инновации. Однако управление этими данными может стать источником сложностей и проблем, ведущих к затруднениям в работе и потере конкурентоспособности.

Одной из главных проблем управления данными является их разнообразие и гетерогенность. Данные могут храниться в различных форматах, распределены по разным источникам и быть доступными в разных местах. Это создает вызовы в синхронизации, обновлении и анализе информации.

Второй важной проблемой является безопасность данных. С увеличением количества кибератак и усовершенствованием методов хакеров, организации должны бороться за защиту конфиденциальности и целостности своих данных. Утечки информации могут привести к серьезным последствиям, включая финансовые потери и потерю репутации.

Третьей проблемой является управление жизненным циклом данных. Организациям часто трудно определить, как долго хранить данные, когда и как их архивировать или уничтожать. Неправильное управление данными может привести к избыточным расходам на хранение или нарушению требований к соответствию.

Чтобы решить эти проблемы, организации обращаются к современным технологиям и методам управления данными. Инструменты автоматизации и управления данными, такие как системы управления данными (Data Management Systems), помогают организациям эффективно организовывать, хранить и обрабатывать информацию.

Более того, технологии и подходы, такие как облачные решения, искусственный интеллект и аналитика данных, помогают улучшить качество управления данными и сделать процессы более прозрачными и предсказуемыми.

Проблемы управления данными в организации - это серьезное вызов, требующий внимания и инновационных решений. Эффективное управление данными не только повышает производительность и конкурентоспособность организации, но и обеспечивает безопасность и целостность важнейшего актива - информации.

1.1. Обзор Apache Airflow

1.1.1. Что такое Apache Airflow?

Apache Airflow - это открытая платформа для управления рабочими процессами (Workflow Management Platform), разработанная Airbnb и выпущенная в качестве проекта с открытым исходным кодом в 2015 году. С тех пор Apache Airflow стал одним из самых популярных инструментов для оркестрации рабочих процессов и автоматизации задач в области данных.

Главной особенностью Apache Airflow является его декларативный подход к описанию рабочих процессов в виде направленного ациклического графа (DAG), который представляет собой набор задач и их зависимостей. Это позволяет разработчикам легко создавать, планировать и мониторить сложные рабочие процессы, состоящие из множества задач.

Основные компоненты Apache Airflow включают:

1. **Scheduler (Планировщик)**: Отвечает за планирование и выполнение задач в соответствии с их зависимостями и расписанием.
2. **Executor (Исполнитель)**: Определяет способ выполнения задач, такой как локальное выполнение, выполнение в облаке или в кластере Apache Mesos.
3. **Web Interface (Веб-интерфейс)**: Предоставляет пользовательский интерфейс для мониторинга и управления рабочими процессами, а также для просмотра журналов выполнения задач.
4. **Metadata Database (База данных метаданных)**: Хранит информацию о рабочих процессах, задачах и их состоянии для обеспечения отслеживаемости и восстановления состояния системы.

Apache Airflow поддерживает множество различных типов задач, включая выполнение SQL запросов, запуск Python скриптов, отправку электронных писем и выполнение произвольных команд оболочки.

Одним из ключевых преимуществ Apache Airflow является его расширяемость и гибкость. С помощью кастомных операторов и сенсоров, а также возможности интеграции с различными сервисами и инструментами, разработчики могут легко настраивать Apache Airflow под свои потребности и интегрировать его в существующие системы.

Кроме того, благодаря активному сообществу разработчиков и широкому набору плагинов, Apache Airflow продолжает развиваться и расширять свои возможности, делая его незаменимым инструментом для управления рабочими процессами в области данных.

Apache Airflow представляет собой мощный инструмент для управления рабочими процессами, обладающий широкими возможностями и гибкостью. Его декларативный подход к описанию рабочих процессов и расширяемость делают его идеальным выбором для автоматизации задач и оркестрации данных в организации.

1.1.2. Основные преимущества Apache Airflow в управлении данными

Apache Airflow предоставляет ряд значительных преимуществ при управлении данными в организации, которые делают его предпочтительным выбором для автоматизации и оркестрации рабочих процессов. Перечислим основные из них:

1. **Декларативный подход к описанию рабочих процессов**: Apache Airflow позволяет разработчикам описывать рабочие процессы в виде направленного ациклического графа (DAG), что делает их легко читаемыми и понятными. Этот подход упрощает создание, планирование и мониторинг сложных рабочих процессов.
2. **Гибкость и расширяемость**: Apache Airflow предоставляет широкий набор встроенных операторов и сенсоров для выполнения различных задач, таких как выполнение SQL запросов, запуск Python скриптов и отправка электронных писем. Кроме того, благодаря активному сообществу разработчиков и возможности создания кастомных операторов и плагинов, Apache Airflow легко расширяется и настраивается под конкретные потребности организации.
3. **Отслеживаемость и мониторинг**: Apache Airflow предоставляет удобный веб-интерфейс для отслеживания выполнения рабочих процессов, мониторинга состояния задач и просмотра журналов выполнения. Это обеспечивает прозрачность и контроль над процессами обработки данных.
4. **Графическое представление зависимостей**: Apache Airflow автоматически визуализирует граф зависимостей между задачами в рабочем процессе, что позволяет разработчикам легко понимать структуру и логику выполнения процесса.
5. **Параллельное выполнение и масштабируемость**: Apache Airflow позволяет параллельно выполнять несколько задач, что ускоряет обработку данных и повышает производительность. Кроме того, благодаря возможности запуска в кластере Apache Mesos или Kubernetes, Apache Airflow легко масштабируется для работы с большими объемами данных.
6. **Интеграция с различными технологиями и сервисами**: Apache Airflow поддерживает интеграцию с различными хранилищами данных (например, Apache Hadoop, Amazon S3, Google Cloud Storage), базами данных, облачными сервисами и инструментами для мониторинга и оповещения, что делает его универсальным решением для работы с различными источниками данных и сервисами.

В целом, Apache Airflow представляет собой мощный и гибкий инструмент для управления данными, который помогает организациям эффективно организовывать рабочие процессы, автоматизировать задачи и повышать производительность работы с данными.

1.1.3. Обзор архитектуры и ключевых компонентов Apache Airflow

Apache Airflow представляет собой мощный инструмент для организации и автоматизации рабочих процессов, особенно в области обработки данных. Его гибкая архитектура и множество ключевых компонентов делают его популярным выбором для разработчиков и инженеров данных. Далее - рассмотрим его архитектуру и основные компоненты.

1. Архитектура Apache Airflow

Архитектура Apache Airflow основана на модели направленного ациклического графа (DAG), где каждый DAG представляет собой набор задач и их зависимостей. Основные компоненты архитектуры включают:

- **Планировщик (Scheduler):** Этот компонент отвечает за планирование выполнения задач в соответствии с их зависимостями и расписанием. Планировщик опрашивает метаданные для поиска DAG, готовых к выполнению, и передает их на выполнение исполнителю.
- **Исполнитель (Executor):** Исполнитель определяет способ выполнения задач, такой как локальное выполнение, выполнение в облаке или в кластере Apache Mesos. Он принимает задачи от планировщика и управляет их выполнением на соответствующих исполнителях.
- **Веб-интерфейс (Web Interface):** Веб-интерфейс предоставляет пользовательский интерфейс для мониторинга и управления рабочими процессами. С его помощью пользователи могут просматривать текущее состояние выполнения задач, просматривать журналы выполнения, а также создавать и управлять новыми рабочими процессами.
- **База данных метаданных (Metadata Database):** Этот компонент хранит метаданные о рабочих процессах, задачах и их состоянии. Он играет ключевую роль в обеспечении отслеживаемости и восстановления состояния системы.

2. Ключевые компоненты Apache Airflow

- **Задачи (Tasks):** Задачи представляют собой отдельные шаги в рабочем процессе, которые должны быть выполнены. Они могут выполнять различные действия, такие как выполнение SQL запросов, запуск Python скриптов или отправка электронных писем.
- **DAG (Directed Acyclic Graph):** DAG представляет собой граф, в котором узлы представляют собой задачи, а ребра определяют зависимости между ними. Это позволяет Apache Airflow автоматически определять порядок выполнения задач и обрабатывать ошибки и перезапуски.
- **Операторы (Operators):** Операторы являются атомарными элементами выполнения задач и определяют тип и способ выполнения каждой задачи. В Apache Airflow существует множество встроенных операторов для выполнения

различных действий, а также возможность создания кастомных операторов под конкретные потребности.

- **Сенсоры (Sensors):** Сенсоры используются для ожидания определенного состояния или события перед выполнением задачи. Они полезны, когда задача зависит от внешних условий или данных, которые могут быть недоступны в момент ее запуска.

Apache Airflow предлагает гибкую и масштабируемую архитектуру, а также широкий набор компонентов для управления рабочими процессами и автоматизации задач в области обработки данных. Его модульный подход позволяет разработчикам создавать сложные рабочие процессы, а также интегрировать его с другими технологиями и сервисами для максимальной эффективности.

1.2. Знакомство с Docker

1.2.1. Введение в Docker и контейнеризацию

Упрощение развертывания приложений

В мире информационных технологий контейнеризация стала неотъемлемой частью разработки и развертывания приложений. Среди различных инструментов контейнеризации, Docker занимает особое место благодаря своей простоте, эффективности и широкой поддержке сообщества. Рассмотрим введение в Docker и контейнеризацию, и почему это стало настолько важным для современной разработки программного обеспечения.

Что такое Docker и контейнеризация?

Docker - это платформа для разработки, доставки и выполнения приложений в контейнерах. Контейнер - это легковесная, автономная и портативная единица программного обеспечения, которая включает в себя все необходимые зависимости (библиотеки, файлы, среды выполнения) для запуска приложения. В отличие от виртуальных машин, контейнеры не включают в себя гостевую операционную систему, а используют ядро хостовой ОС, что делает их более эффективными и быстрыми.

Применение Docker и контейнеризации:

Контейнеризация нашла применение во многих областях разработки и операций:

- **Разработка приложений:** Docker обеспечивает единое окружение разработки для всей команды, что упрощает совместную работу и устраняет проблемы с различиями в окружениях.
- **Тестирование и CI/CD:** Контейнеры используются для создания изолированных тестовых сред, что упрощает процесс тестирования и автоматизации непрерывной интеграции и доставки (CI/CD).
- **Развертывание в облаке:** Docker позволяет легко развертывать приложения в облачных средах, таких как AWS, Google Cloud и Azure, обеспечивая максимальную гибкость и масштабируемость.
- **Микросервисная архитектура:** Контейнеры идеально подходят для построения микросервисных приложений, благодаря своей легковесности, изолированности и возможности масштабирования отдельных компонентов.

Docker и контейнеризация стали неотъемлемой частью современной разработки и операций благодаря своей простоте, эффективности и портабельности. Они упрощают развертывание и управление приложениями, обеспечивают безопасность и изоляцию, а также позволяют эффективно использовать ресурсы вычислительной инфраструктуры.

1.2.2. Преимущества использования Docker для развертывания приложений

Использование Docker для развертывания приложений предоставляет множество преимуществ, которые делают его предпочтительным выбором для современных разработчиков и операционных инженеров. Перечислим основные преимущества использования Docker:

1. **Портабельность**: Docker контейнеры создаются на основе образов, которые содержат все необходимые зависимости и настройки для запуска приложения. Это делает контейнеры легко переносимыми между различными средами выполнения, включая различные операционные системы и облачные платформы. Портабельность Docker контейнеров обеспечивает единое окружение разработки, тестирования и производства, что упрощает управление приложениями в различных средах.
2. **Изоляция и безопасность**: Каждый Docker контейнер работает в изолированной среде, что позволяет избежать конфликтов между зависимостями и обеспечивает безопасность приложений. Контейнеры также обеспечивают уровень изоляции между приложениями, что минимизирует риск воздействия одного приложения на другие.
3. **Эффективное использование ресурсов**: Docker контейнеры используют общее ядро хостовой операционной системы, что позволяет им работать более эффективно и использовать меньше ресурсов по сравнению с виртуальными машинами. Это уменьшает накладные расходы и позволяет максимально эффективно использовать вычислительные ресурсы.
4. **Быстрое развертывание и масштабирование**: Docker контейнеры можно создавать и запускать в считанные секунды, что упрощает и ускоряет процесс развертывания приложений. Кроме того, автоматизированные средства масштабирования позволяют легко управлять количеством экземпляров приложений в зависимости от нагрузки, обеспечивая гибкость и масштабируемость.
5. **Удобство управления зависимостями и версиями**: Docker обеспечивает удобный механизм управления зависимостями и версиями приложений с помощью Dockerfile и Docker Compose. Это позволяет легко определять окружение выполнения приложения и управлять его конфигурацией, что упрощает процесс разработки, тестирования и обновления приложений.
6. **Поддержка микросервисной архитектуры**: Docker идеально подходит для построения микросервисных приложений благодаря своей легковесности, изолированности и возможности масштабирования отдельных компонентов. Контейнеры позволяют разбить приложение на небольшие компоненты, которые могут быть легко масштабированы и обновлены независимо друг от друга.

Использование Docker для развертывания приложений обеспечивает ряд значительных преимуществ, включая портабельность, изоляцию, эффективное использование ресурсов, быстрое развертывание и масштабирование, удобство

управления зависимостями и поддержку микросервисной архитектуры. Эти преимущества делают Docker незаменимым инструментом для современной разработки и операций.

1.2.3. Основные понятия Docker: контейнеры, образы, Dockerfile и Docker Compose

Docker - это платформа для разработки, доставки и выполнения приложений в контейнерах, которая стала неотъемлемой частью современной разработки программного обеспечения. Основные понятия Docker, такие как контейнеры, образы, Dockerfile и Docker Compose, играют ключевую роль в управлении и развертывании контейнеризованных приложений. Рассмотрим каждое из этих понятий подробнее.

1. Контейнеры (Containers)

Контейнеры - это легковесные, автономные и переносимые единицы программного обеспечения, которые включают в себя все необходимые зависимости (библиотеки, файлы, среды выполнения) для запуска приложения. Контейнеры работают в изолированной среде, что обеспечивает безопасность и изоляцию приложений от внешнего окружения и других контейнеров.

2. Образы (Images)

Образы - это шаблоны, на основе которых создаются контейнеры. Образ содержит все необходимые файлы и настройки для запуска приложения, а также инструкции по его настройке. Образы можно создавать вручную или автоматизировать с помощью Dockerfile. Образы могут быть сохранены и переданы между различными средами выполнения, что обеспечивает единое окружение разработки, тестирования и производства.

3. Dockerfile

Dockerfile - это текстовый файл, содержащий инструкции для создания Docker образа. С его помощью разработчики могут определить необходимые зависимости, настройки и команды для установки и настройки приложения в контейнере. Dockerfile позволяет автоматизировать процесс создания образов и обеспечивает единый и повторяемый способ настройки окружения выполнения приложения.

Пример простого Dockerfile для создания образа для запуска простого python - приложения:

```
FROM python:3.7.2-alpine3.8
LABEL maintainer="jeffmshale@gmail.com"
ENV ADMIN="jeff"
RUN apk update && apk upgrade && apk add bash
COPY . ./app
RUN ["mkdir", "/a_directory"]
CMD ["python", "./my_script.py"]
```

4. Docker Compose

Docker Compose - это инструмент для определения и управления многоконтейнерными приложениями с помощью файла конфигурации docker-compose.yml. С его помощью разработчики могут определить структуру и зависимости между контейнерами, а также настройки сети и хранилища данных. Docker Compose позволяет легко запускать, останавливать и масштабировать многоконтейнерные приложения, обеспечивая удобное управление контейнерами.

Пример файла docker-compose.yml для запуска веб-приложения с использованием Node.js и базы данных MongoDB:

```
version: '3'

services:
  web:
    build: .
    ports:
      - "3000:3000"
    depends_on:
      - db
  db:
    image: mongo
    ports:
      - "27017:27017"
```

Основные понятия Docker, такие как контейнеры, образы, Dockerfile и Docker Compose, играют важную роль в управлении и развертывании контейнеризованных приложений. Использование Docker обеспечивает эффективное управление окружением выполнения приложений, упрощает разработку, тестирование и развертывание приложений, а также обеспечивает портабельность и изоляцию приложений от внешних факторов.

Глава 2. Внедрение Apache Airflow с использованием Docker

2.1. Подготовка среды: установка Docker с Docker-compose и настройка окружения

Устанавливать Docker будем на подготовленный и настроенный сервер с ОС Linux Debian 12. Кроме этого в сети был выделен отдельный поддомен (dwh.rkc43.ru), соответствующим образом настроены службы DNS, DHCP, почтовый сервер. Установка и настройка данной ОС а также - системного окружения выходит за рамки нашей работы, поэтому мы не будем касаться здесь этих процессов.

Устанавливаем Docker вместе с Docker-compose:

- Обновляем список пакетов
- Добавляем необходимые пакеты для добавления репозиториев через https
- Добавляем официальный ключ GPG Docker
- Обновляем список пакетов

```
$ sudo apt update
$ sudo apt install apt-transport-https ca-certificates curl
software-properties-common
$ sudo add-apt-repository "deb [arch=amd64]
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
$ sudo apt update
```

Все готово к установке. Запускаем:

```
$ apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin
docker-compose-plugin docker-compose
```

Проверяем результат установки:

```
$ docker --version
Docker version 20.10.24+dfsg1, build 297e128
$ docker-compose --version
docker-compose version 1.29.2, build unknown
$
```

Docker и docker-compose успешно установлены на наш сервер

2.2. Создание контейнеров Apache Airflow

Для запуска и настройки контейнеров взял открытый репозиторий [6].

В данной конфигурации среди драйверов базы данных, к которым может подключаться Airflow, нет драйверов для СУБД Oracle, FireBird и MS SQL Server. Так же, в в контейнере отсутствуют некоторые требуемые python-библиотеки. Внесем их в файл requirements.txt. Добавим требуемые драйвера в Dockerfile и пересоберем контейнер.

Файл Dockerfile (см. Приложение 1. Файл Dockerfile).

Для сборки контейнера - выполним команду:

```
docker build . -f Dockerfile --pull --tag my-airflow:0.0.1
```

Наш контейнер будет называться *my-airflow:0.0.1.*

На этом этап создания и сборки контейнера Apache Airflow завершен.

2.3. Конфигурация Apache Airflow через Docker Compose

После создания контейнеров Apache Airflow модифицируем файл docker-compose.yaml (см. Приложение 2. Файл docker-compose.yaml), с указанием образа нашего контейнера Airflow, с учетом работы в локальной сети и настройке необходимых переменных конфигурации

Далее - запускаем стек командой:

```
$ docker-compose up
```

В результате запуска получаем 7 контейнеров для работы:

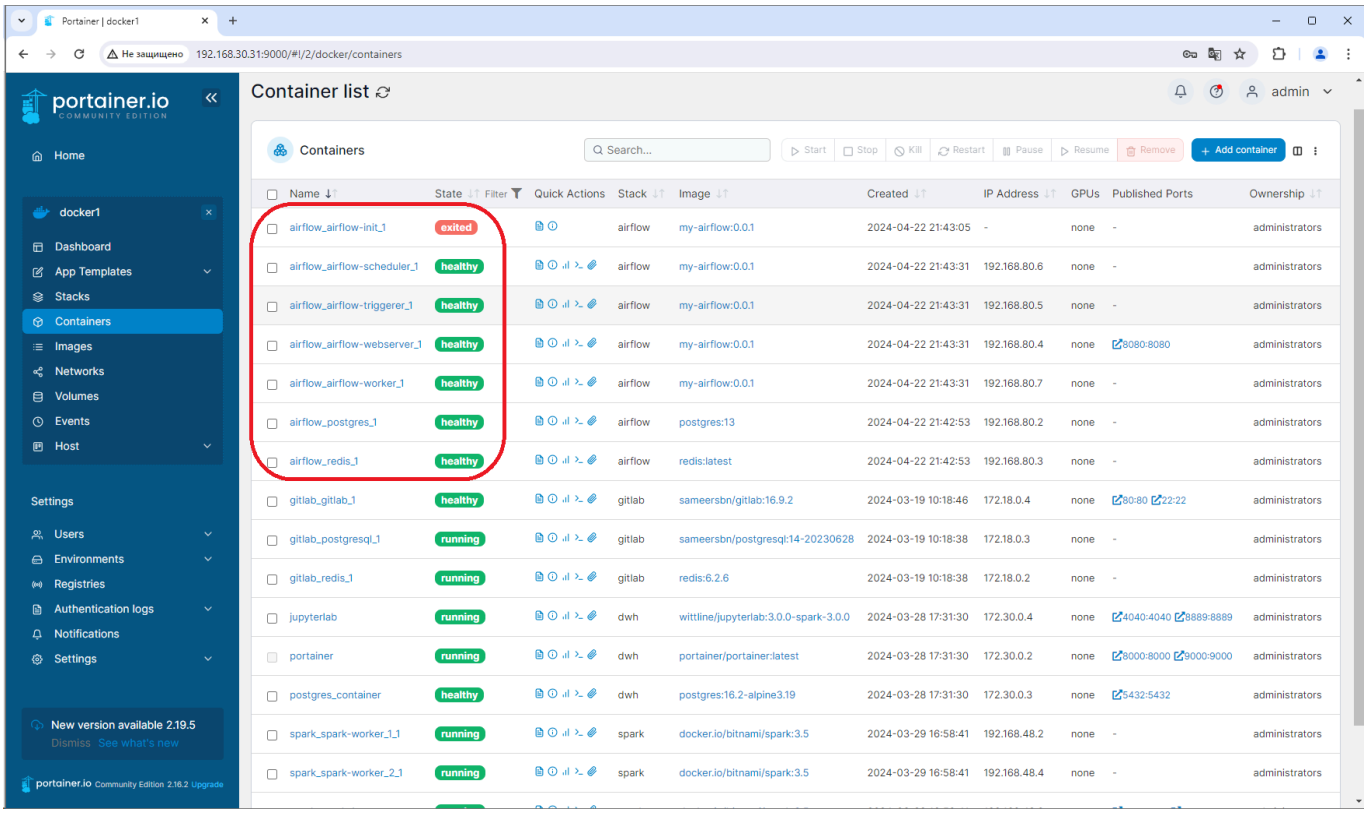


Рисунок 1. Панель Portainer

Для управления контейнерами использовал Portainer [7]. (Т.к. проект получил дополнительное развитие, на картинке можно видеть другие контейнеры, которые пока не связаны с нашей задачей).

Запускаем

В итоге - получаем 7 контейнеров для работы с Apache Airflow:

1. airflow_airflow-init_1. Контейнер предназначен для первоначального запуска всего стека, создает схему данных. В дальнейшем - в работе не участвует.
2. airflow_airflow-scheduler_1. Планировщик задач.

3. airflow_airflow-triggerer_1. Используется для запуска отложенных задач.
4. airflow_airflow-webserver_1. Веб-сервер. Позволяет управлять и мониторить задания и задачи.
5. airflow_airflow-worker_1. Непосредственно рабочий узел, на котором выполняются задачи.
6. airflow_postgres_1. База данных для хранения требуемой для работы Airflow информации. Так же - используется для передачи данных между задачами.
7. airflow_redis_1. Контейнер с Redis.

Данный стек контейнеров больше подходит для разработки или учебных целей, но может быть использован как концепт для развертывания высоконагруженного кластера, путем расположения отдельных контейнеров на отдельных физических серверах, так и путем увеличения самих серверов кластера, для добавления дополнительных worker-контейнеров.

2.4. Запуск и мониторинг Apache Airflow в Docker

После запуска стека контейнеров переходим по адресу http://airflow.dwh.rkc43.ru:8080/home

После ввода пароля - наблюдаем следующую картину:

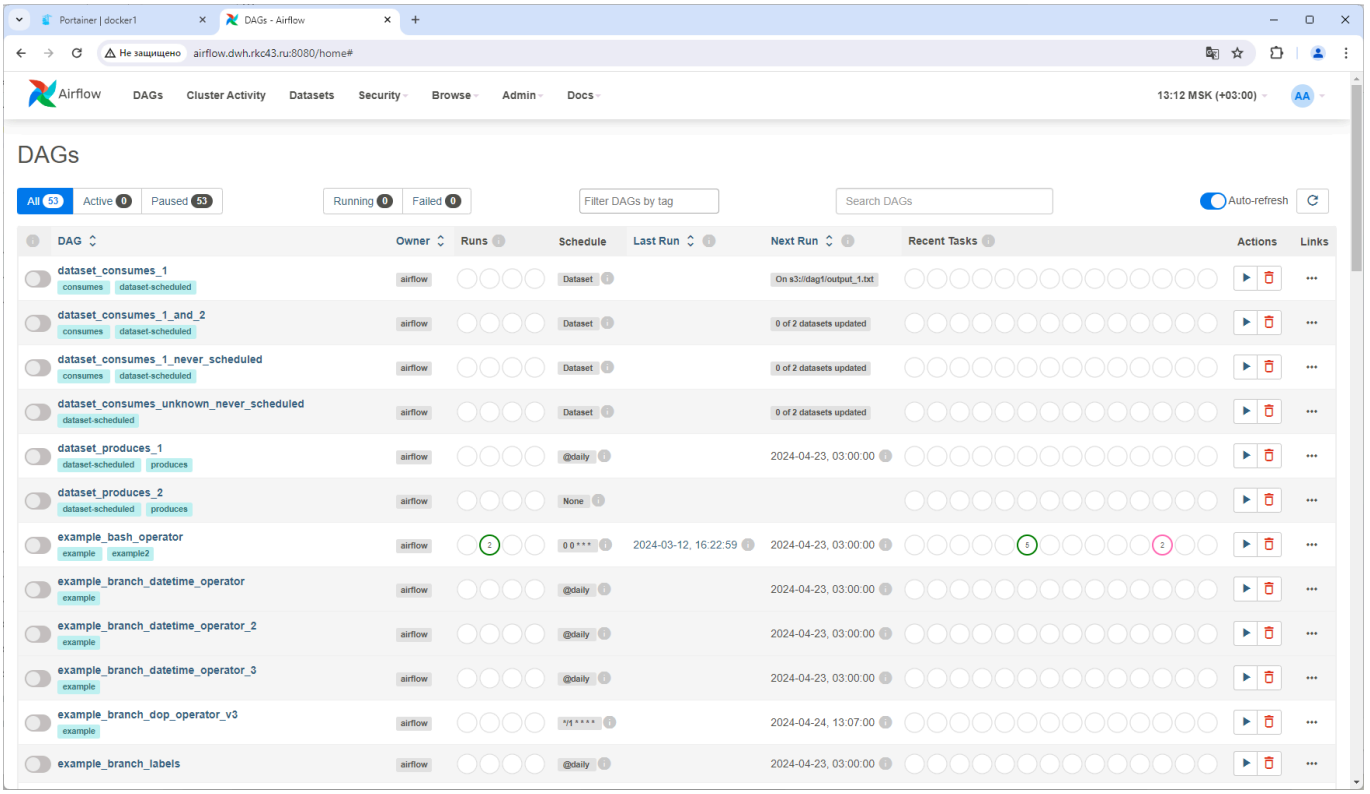


Рисунок 2. Панель Airflow по-умолчанию

Видим, что наш стек Airflow запустился и работает. В списке DAG находятся примеры DAG-файлов, которые поставляются с настройкой по умолчанию. Отключим их, изменяя переменную AIRFLOW__CORE__LOAD_EXAMPLES: ‘true’ в файле docker-compose.yaml.

Для последующей работы определим требуемые переменные (меню Admin -> Variables)

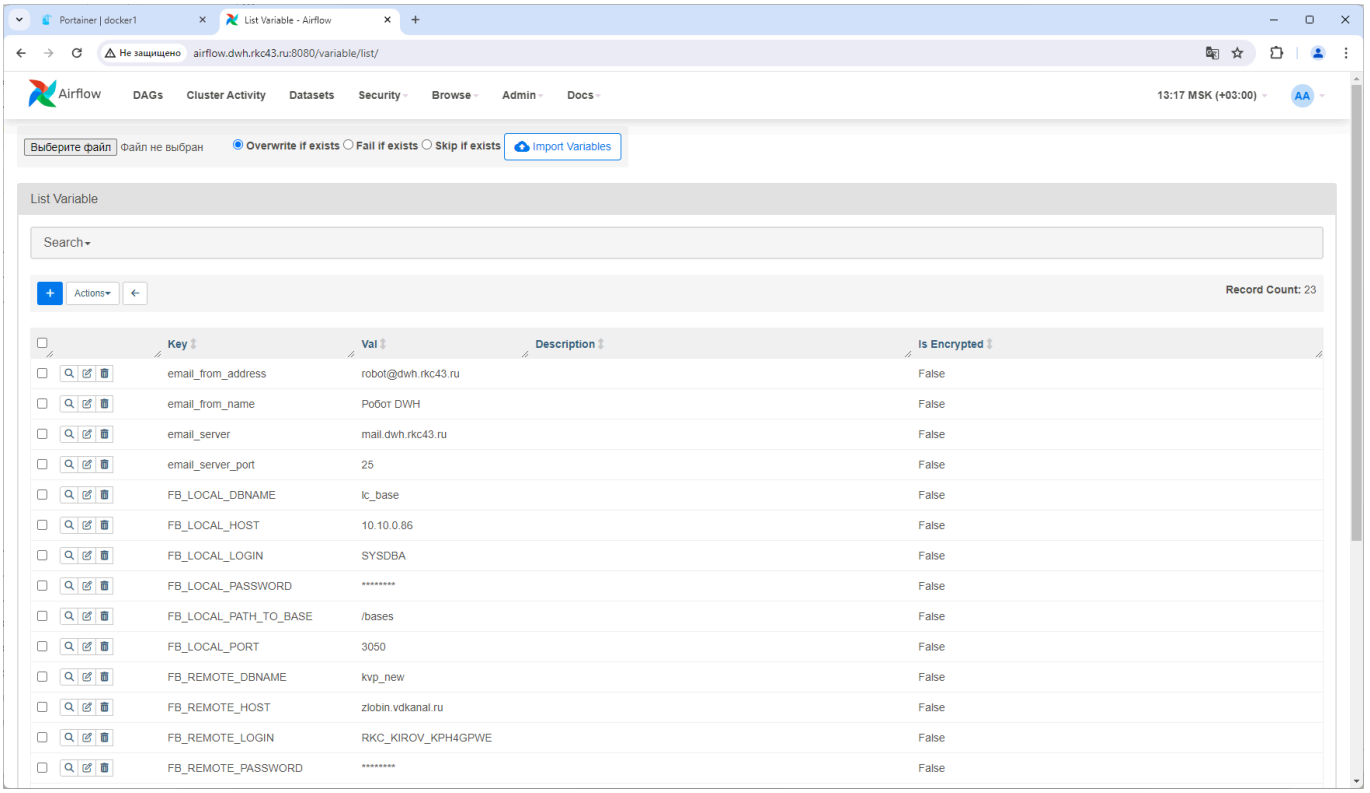


Рисунок 3. Переменные для использования внутри DAG

После настройки и помещения разработанных нами файлов DAG в определенный каталог, и перезапуска стека контейнеров, наблюдаем следующую картину в интерфейсе:

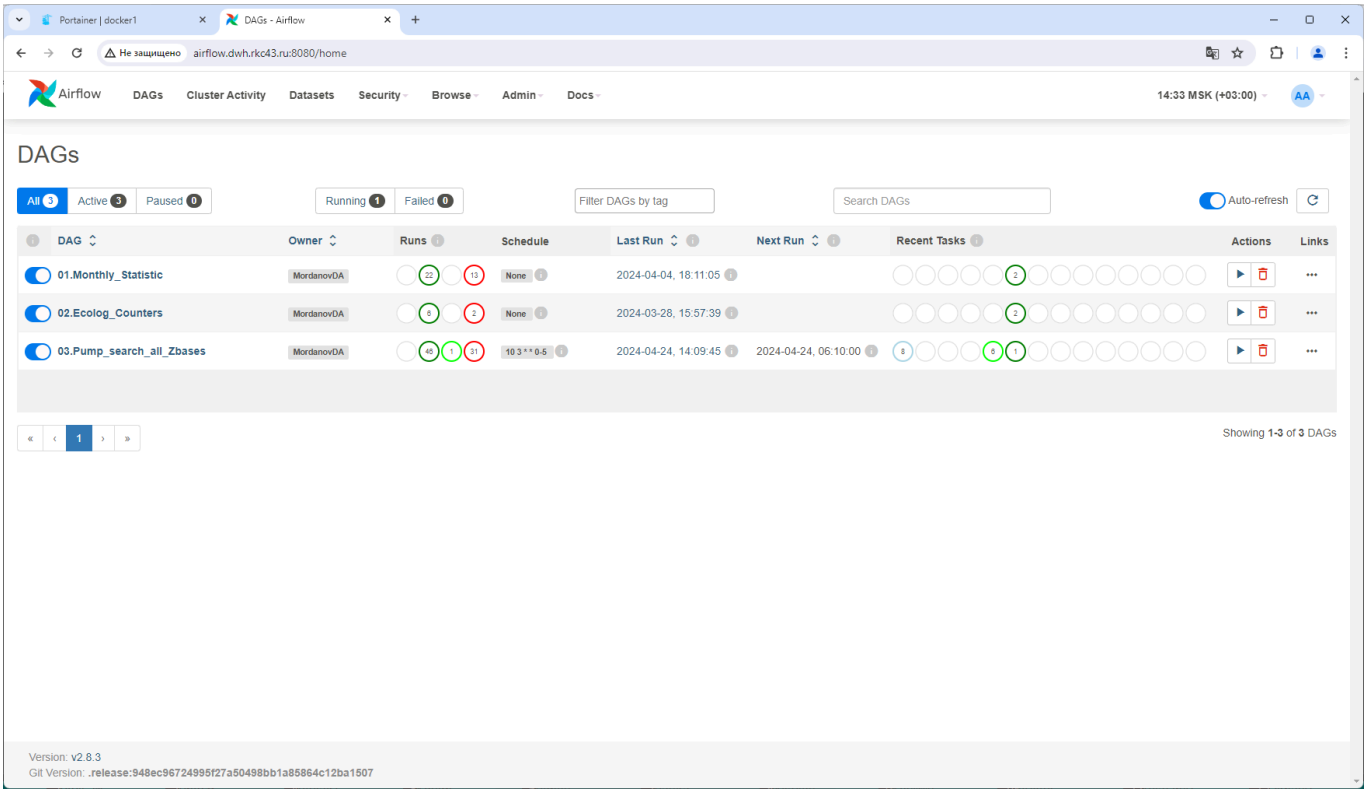


Рисунок 4. Окончательный вид панели управления Apache Airflow

Глава 3. Практическое использования Apache Airflow в среде контейнеризации Docker

3.1. Автоматизация процесса ETL

В ООО РКЦ процесс ETL (Extract, Transform, Load) является одним из ключевых этапов обработки информации, который позволяет извлекать данные из различных источников, преобразовывать их в нужный формат, а также загружать в целевую базу данных или хранилище данных. Автоматизация этого процесса играет решающую роль в ускорении времени обработки данных, снижении рисков ошибок и повышении производительности.

В организации есть процессы ETL для загрузки данных из различных источников, их трансформации и загрузки в целевую базу данных, или построения отчетов. Мы можем использовать Apache Airflow для создания DAG, который определяет последовательность задач ETL и их зависимости. Каждая задача ETL может быть реализована в виде отдельной операции (Python функции или скрипта), которая извлекает, трансформирует и загружает данные в соответствии с заданным графом выполнения.

Apache Airflow предоставляет мощные инструменты для автоматизации процесса ETL, обеспечивая гибкость, масштабируемость и отслеживаемость выполнения задач. Использование Apache Airflow позволяет эффективно управлять обработкой данных, сокращая время выполнения задач и повышая производительность в области анализа и использования данных.

3.2. Планирование и мониторинг задач обработки данных

Для эффективной работы с данными необходимо не только обеспечить их сбор, но и обработку. Планирование и мониторинг задач обработки данных стали важными этапами в рамках аналитического процесса.

Почему планирование и мониторинг задач обработки данных важны?

1. **Эффективное использование ресурсов**: Планирование задач обработки данных позволяет оптимизировать использование ресурсов, таких как вычислительная мощность и хранилище данных, что способствует повышению производительности и сокращению времени обработки данных.
2. **Обеспечение точности и своевременности**: Планирование задач обработки данных гарантирует выполнение операций в нужной последовательности и в нужное время, что обеспечивает точность и своевременность получаемых результатов анализа данных.
3. **Минимизация ошибок и проблем**: Планирование позволяет заранее обнаруживать и устранять потенциальные проблемы и ошибки в процессе обработки данных, что повышает надежность и качество получаемых данных.
4. **Легкость масштабирования**: Правильное планирование задач обработки данных обеспечивает легкость масштабирования процессов обработки данных, что позволяет эффективно работать с большими объемами данных и изменяющимися требованиями.
5. **Отслеживание выполнения задач**: Мониторинг задач обработки данных позволяет отслеживать выполнение операций в реальном времени, выявлять и решать проблемы и задержки, а также улучшать производительность и эффективность процесса обработки данных.

Все вышеперечисленное возможно выполнить с помощью построенной нами системы обработки данных.

В качестве начальной точки напишем 3 DAG, на основе которых, подобным образом, возможно продолжить описание требуемых ETL - процессов.

3.3. Разработка реальных задач

3.3.1. Создание отчета по показаниям общедомовых счетчиков

Задача: Для отдела экологов требуется создание отчета, который собирает внесенные в расчетную систему показания счетчиков определенных объектов контрагентов, за которыми установлен усиленный контроль со стороны отдела экологии.

Данные находятся в СУБД Oracle. Генерация отчета запускается вручную. Результат отправляется по электронной почте в виде файла MS Excel.

Реализация:

Расчетная система разработана сторонней организацией, и имеет обширную схему данных. Разработчиками был предоставлен SQL скрипт (см. Приложение 3. Файл 02.ecolog_counters.sql), позволяющий получать требуемые данные.

Далее - разработаем DAG файл (см. Приложение 4. Файл 02_ecolog_counters.py).

DAG состоит из 2-х задач.

- **get_data_from_oracle()** - загружает данные из СУБД Oracle в структуру pandas dataframe. Сам SQL-скрипт загрузки вынесен в отдельный файл в каталог SQL, для удобства работы. Таким же образом планируется хранить все SQL-скрипты для решения последующих заданий. После получения данных - записывается файл MS Excel во временном каталоге. Такое решение имеет недостаток - возможно, что последующая задача может быть запущена на другом узле кластера, тогда файл, записанный этой задачей будет не доступен на другом узле кластера. Можно записывать результат выполнения SQL-скрипта в разделяемую систему хранения данных, но в нашей работе такого решения, к сожалению нет.

  Имя файла передаем следующей задаче используя механизм XCom (см. Книга: Бас Харенслак, Джулиан де Руйтер. Apache Airflow и...).

- **xlsx_to_email(()** - отправляет сгенерированный отчет указанным адресатам. К СУБД Oracle обращаемся для получения наименования периода расчета (чтобы сгенерировать понятный человеку заголовок электронного письма). После - формируется необходимый объект для отправки электронного письма адресатам с вложением - результатом работы выгрузки данных. Имя и путь к файлу с отчетом берем из объекта XCom.

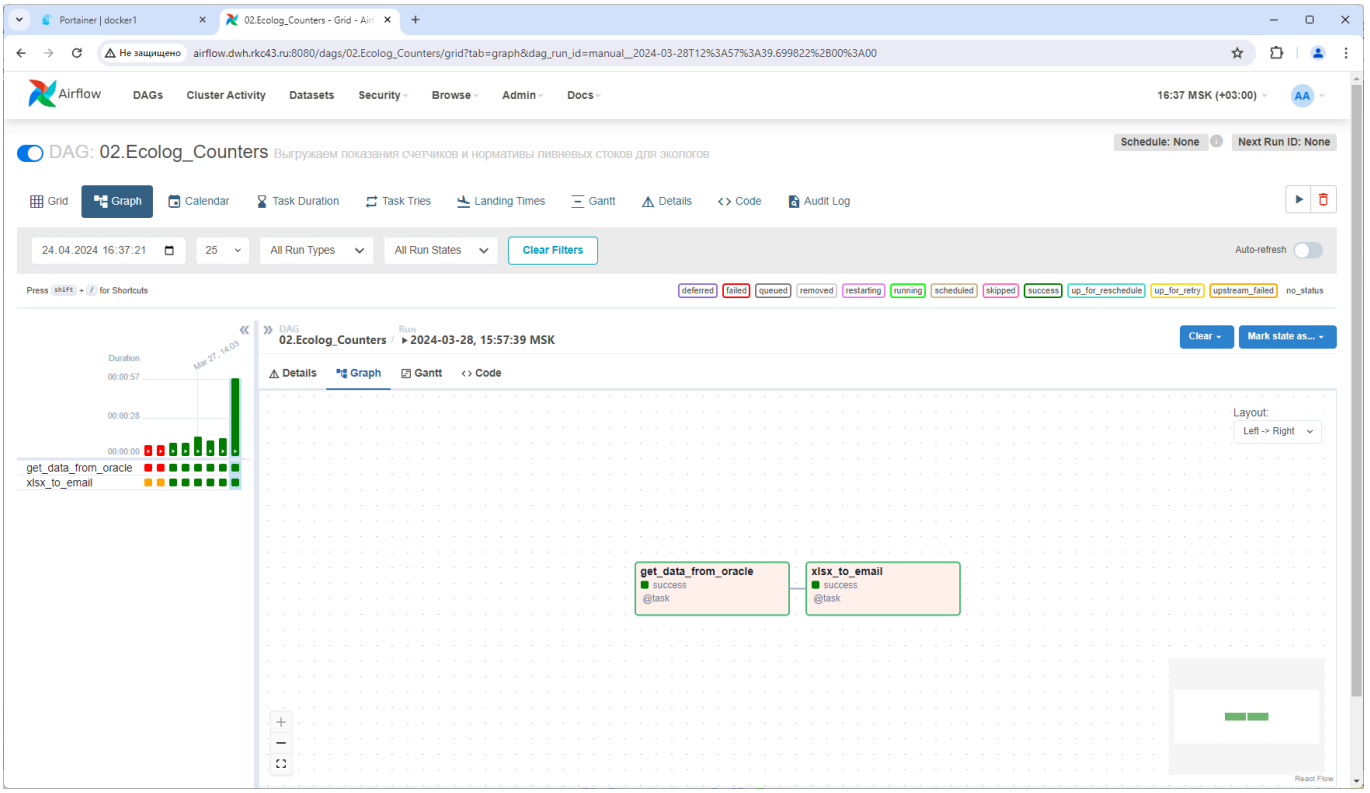


Рисунок 5. Графическое представление DAG 02_ecolog_counters.py

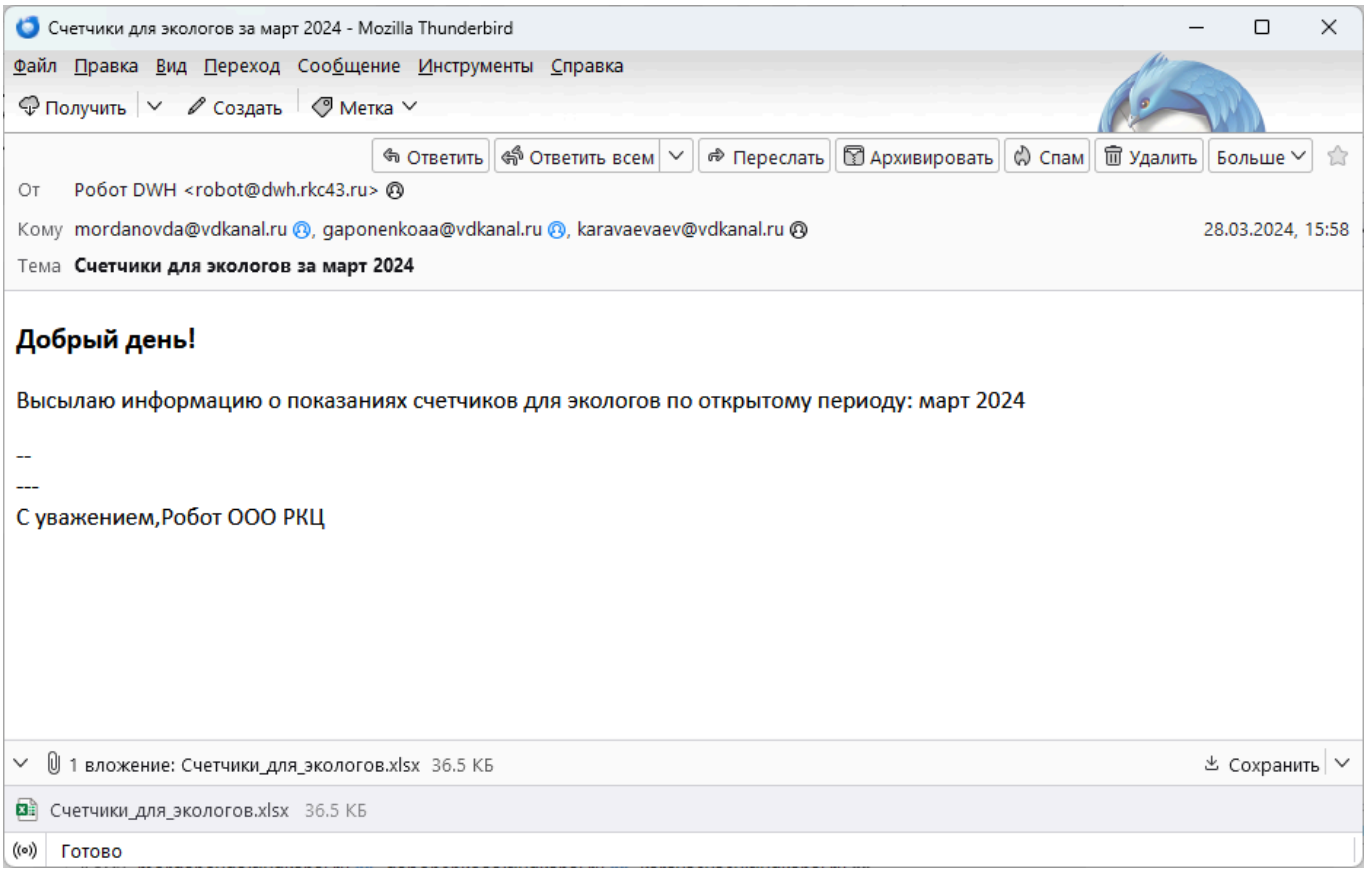
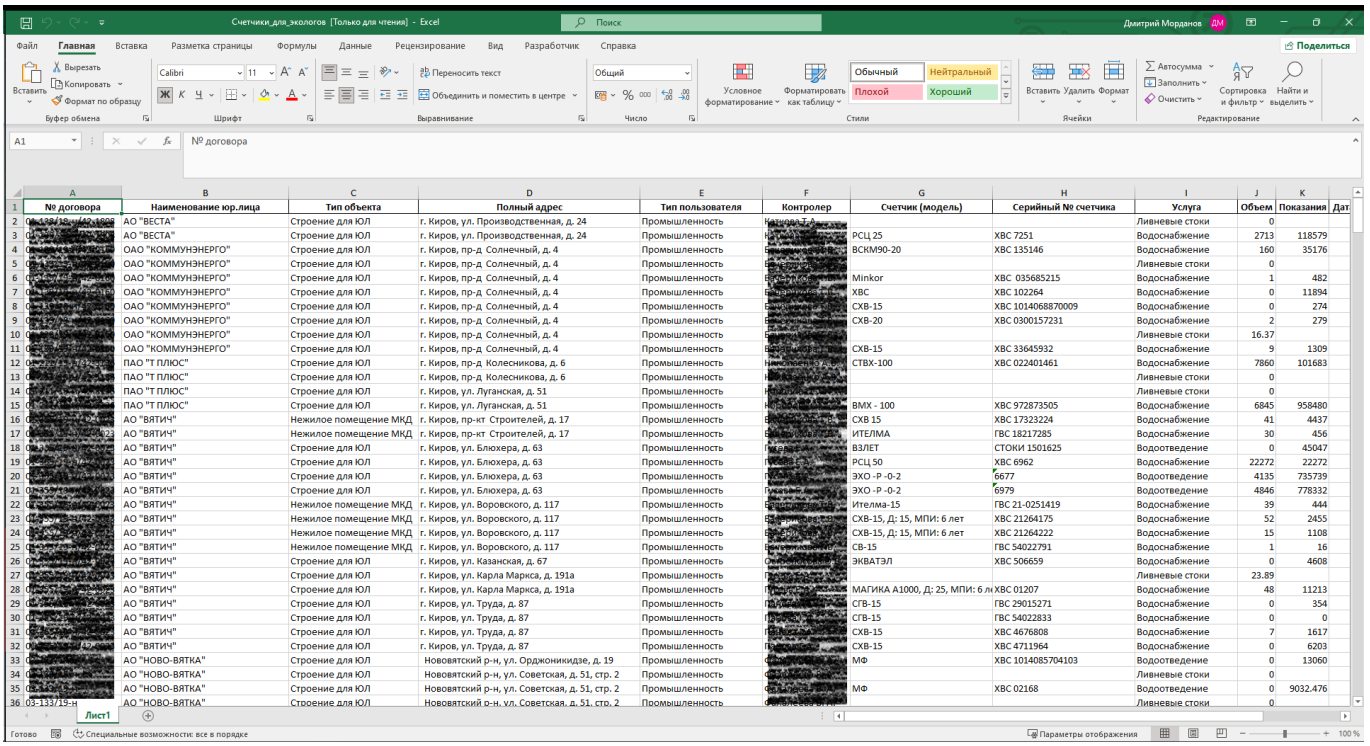


Рисунок 6. Письмо с вложением



№ договора	Наименование юр.лица	Тип объекта	Полный адрес	Тип пользователя	Контролер	Счетчик (модель)	Серийный № счетчика	Услуга	Объем	Показания	Дат
[redacted]	АО "ВЕСТА"	Строение для ЮЛ	г. Киров, ул. Производственная, д. 24	Промышленность	[redacted]			Ливневые стоки	0		
[redacted]	АО "ВЕСТА"	Строение для ЮЛ	г. Киров, ул. Производственная, д. 24	Промышленность	[redacted]	РСЦ 25	ХВС 7251	Водоснабжение	2713	118579	
[redacted]	ОАО "КОММУНЭНЕРГО"	Строение для ЮЛ	г. Киров, пр-д Солнечный, д. 4	Промышленность	[redacted]	ВСКМ90-20	ХВС 135146	Водоснабжение	160	35176	
[redacted]	ОАО "КОММУНЭНЕРГО"	Строение для ЮЛ	г. Киров, пр-д Солнечный, д. 4	Промышленность	[redacted]			Ливневые стоки	0		
[redacted]	ОАО "КОММУНЭНЕРГО"	Строение для ЮЛ	г. Киров, пр-д Солнечный, д. 4	Промышленность	[redacted]	Minkor	ХВС 035685215	Водоснабжение	1	482	
[redacted]	ОАО "КОММУНЭНЕРГО"	Строение для ЮЛ	г. Киров, пр-д Солнечный, д. 4	Промышленность	[redacted]	ХВС	ХВС 102264	Водоснабжение	0	11894	
[redacted]	ОАО "КОММУНЭНЕРГО"	Строение для ЮЛ	г. Киров, пр-д Солнечный, д. 4	Промышленность	[redacted]	СХВ-15	ХВС 1014068870009	Водоснабжение	0	274	
[redacted]	ОАО "КОММУНЭНЕРГО"	Строение для ЮЛ	г. Киров, пр-д Солнечный, д. 4	Промышленность	[redacted]	СХВ-20	ХВС 0300157231	Водоснабжение	2	279	
[redacted]	ОАО "КОММУНЭНЕРГО"	Строение для ЮЛ	г. Киров, пр-д Солнечный, д. 4	Промышленность	[redacted]			Ливневые стоки	16.37		
[redacted]	ОАО "КОММУНЭНЕРГО"	Строение для ЮЛ	г. Киров, пр-д Солнечный, д. 4	Промышленность	[redacted]	СХВ-15	ХВС 33645932	Водоснабжение	9	1309	
[redacted]	ПАО "Т ПЛЮС"	Строение для ЮЛ	г. Киров, пр-д Колесникова, д. 6	Промышленность	[redacted]	СТВХ-100	ХВС 022401461	Водоснабжение	7860	101683	
[redacted]	ПАО "Т ПЛЮС"	Строение для ЮЛ	г. Киров, пр-д Колесникова, д. 6	Промышленность	[redacted]			Ливневые стоки	0		
[redacted]	ПАО "Т ПЛЮС"	Строение для ЮЛ	г. Киров, ул. Луганская, д. 51	Промышленность	[redacted]			Ливневые стоки	0		
[redacted]	ПАО "Т ПЛЮС"	Строение для ЮЛ	г. Киров, ул. Луганская, д. 51	Промышленность	[redacted]	ВМХ - 100	ХВС 972873505	Водоснабжение	6845	958480	
[redacted]	АО "ВЯТИЧ"	Нежилое помещение МКД	г. Киров, пр-кт Строителей, д. 17	Промышленность	[redacted]	СХВ 15	ХВС 17323224	Водоснабжение	41	4437	
[redacted]	АО "ВЯТИЧ"	Нежилое помещение МКД	г. Киров, пр-кт Строителей, д. 17	Промышленность	[redacted]	ИТЕЛМА	ГВС 18217285	Водоснабжение	30	456	
[redacted]	АО "ВЯТИЧ"	Строение для ЮЛ	г. Киров, ул. Блюхера, д. 63	Промышленность	[redacted]	ВЗЛЕТ	СТОКИ 1501625	Водоотведение	0	45047	
[redacted]	АО "ВЯТИЧ"	Строение для ЮЛ	г. Киров, ул. Блюхера, д. 63	Промышленность	[redacted]	РСЦ 50	ХВС 6962	Водоснабжение	22272	22272	
[redacted]	АО "ВЯТИЧ"	Строение для ЮЛ	г. Киров, ул. Блюхера, д. 63	Промышленность	[redacted]	ЭХО -Р -0-2	6677	Водоотведение	4135	735739	
[redacted]	АО "ВЯТИЧ"	Строение для ЮЛ	г. Киров, ул. Блюхера, д. 63	Промышленность	[redacted]	ЭХО -Р -0-2	6979	Водоотведение	4846	778332	
[redacted]	АО "ВЯТИЧ"	Нежилое помещение МКД	г. Киров, ул. Воровского, д. 117	Промышленность	[redacted]	Ителма-15	ГВС 21-0251419	Водоснабжение	39	444	
[redacted]	АО "ВЯТИЧ"	Нежилое помещение МКД	г. Киров, ул. Воровского, д. 117	Промышленность	[redacted]	СХВ-15, Д: 15, МПИ: 6 лет	ХВС 21264175	Водоснабжение	52	2455	
[redacted]	АО "ВЯТИЧ"	Нежилое помещение МКД	г. Киров, ул. Воровского, д. 117	Промышленность	[redacted]	СХВ-15, Д: 15, МПИ: 6 лет	ХВС 21264222	Водоснабжение	15	1108	
[redacted]	АО "ВЯТИЧ"	Нежилое помещение МКД	г. Киров, ул. Воровского, д. 117	Промышленность	[redacted]	СВ-15	ГВС 54022791	Водоснабжение	1	16	
[redacted]	АО "ВЯТИЧ"	Строение для ЮЛ	г. Киров, ул. Казанская, д. 67	Промышленность	[redacted]	ЭКВАТЭЛ	ХВС 506659	Водоснабжение	0	4608	
[redacted]	АО "ВЯТИЧ"	Строение для ЮЛ	г. Киров, ул. Карла Маркса, д. 191а	Промышленность	[redacted]			Ливневые стоки	23.89		
[redacted]	АО "ВЯТИЧ"	Строение для ЮЛ	г. Киров, ул. Карла Маркса, д. 191а	Промышленность	[redacted]	МАГИКА А1000, Д: 25, МПИ: 6 ле	ХВС 01207	Водоснабжение	48	11213	
[redacted]	АО "ВЯТИЧ"	Строение для ЮЛ	г. Киров, ул. Труда, д. 87	Промышленность	[redacted]	СГВ-15	ГВС 29015271	Водоснабжение	0	354	
[redacted]	АО "ВЯТИЧ"	Строение для ЮЛ	г. Киров, ул. Труда, д. 87	Промышленность	[redacted]	СГВ-15	ГВС 54022833	Водоснабжение	0	0	
[redacted]	АО "ВЯТИЧ"	Строение для ЮЛ	г. Киров, ул. Труда, д. 87	Промышленность	[redacted]	СХВ-15	ХВС 4676808	Водоснабжение	7	1617	
[redacted]	АО "ВЯТИЧ"	Строение для ЮЛ	г. Киров, ул. Труда, д. 87	Промышленность	[redacted]	СХВ-15	ХВС 4711964	Водоснабжение	0	6203	
[redacted]	АО "НОВО-ВЯТКА"	Строение для ЮЛ	Нововятский р-н, ул. Орджоникидзе, д. 19	Промышленность	[redacted]	МФ	ХВС 1014085704103	Водоотведение	0	13060	
[redacted]	АО "НОВО-ВЯТКА"	Строение для ЮЛ	Нововятский р-н, ул. Советская, д. 51, стр. 2	Промышленность	[redacted]			Ливневые стоки	0		
[redacted]	АО "НОВО-ВЯТКА"	Строение для ЮЛ	Нововятский р-н, ул. Советская, д. 51, стр. 2	Промышленность	[redacted]	МФ	ХВС 02168	Водоотведение	0	9032.476	
03-133/19-Н	АО "НОВО-ВЯТКА"	Строение для ЮЛ	Нововятский р-н, ул. Советская, д. 51, стр. 2	Промышленность	[redacted]			Ливневые стоки	0		

Рисунок 7. Результат выполнения отчета по показаниям общедомовых счетчиков

Данные содержат коммерческую информацию - поэтому часть их скрыта.

3.3.2. Отчет-статистика за прошедший месяц

Задача: Для руководителей ООО РКЦ требуется создание отчета со статистикой по прошедшему периоду. Отчет требуется получать как только это станет возможным.

Данные находятся в СУБД Oracle. Генерация отчета запускается по таймеру. Результат отправляется по электронной почте в виде файла MS Excel.

Реализация:

Для выгрузки отчета был разработан соответствующий SQL скрипт (см. Приложение 5. Файл 01.monthly_stats.sql).

В связи со спецификой работы, невозможно точно установить, когда в базе данных появятся корректные данные для построения отчета. Поэтому планировщик этой задачи настроим на ежедневное выполнение с 4 по 15 числа каждого месяца. В базе данных есть признак (запись в определенной таблице), по которой можно судить о готовности выполнить отчет. Этот признак мы будем мониторить, и если он возникает - то выполняется запуск отчета. Также - мы фиксируем факт выполнения отчета в рабочей базе, чтобы не запускать его повторно.

В итоге - DAG (см. Приложение 6. Файл 01_statistic.py) состоит из следующих заданий:

- **start** - пустой оператор - используется как единая точка входа
- **check_period** - задача, выполняющая проверку возможности выполнения отчета, возвращающая результат, по которому выполняется ветвление процесса
- **check_period_true** - пустой оператор - для построения ветвления (плечо True)
- **check_period_false** - пустой оператор - для построения ветвления (плечо False)
- **get_data_from_oracle** - загружает данные из СУБД Oracle в структуру pandas dataframe. Сам SQL-скрипт загрузки вынесен в отдельный файл в каталог SQL, для удобства работы. После получения данных - записывается файл MS Excel во временном каталоге.

  Имя файла передаем следующей задаче используя механизм XCom.

- **xlsx_to_email** - отправляет сгенерированный отчет указанным адресатам. К СУБД Oracle обращаемся для получения наименования периода расчета (чтобы сгенерировать понятный человеку заголовок электронного письма). После - формируется необходимый объект для отправки электронного письма адресатам с вложением - результатом работы выгрузки данных. Путь к файлу с результатами отчета берем из объекта Xcom.
- **end** - пустой оператор, используется как единая точка выхода из процесса

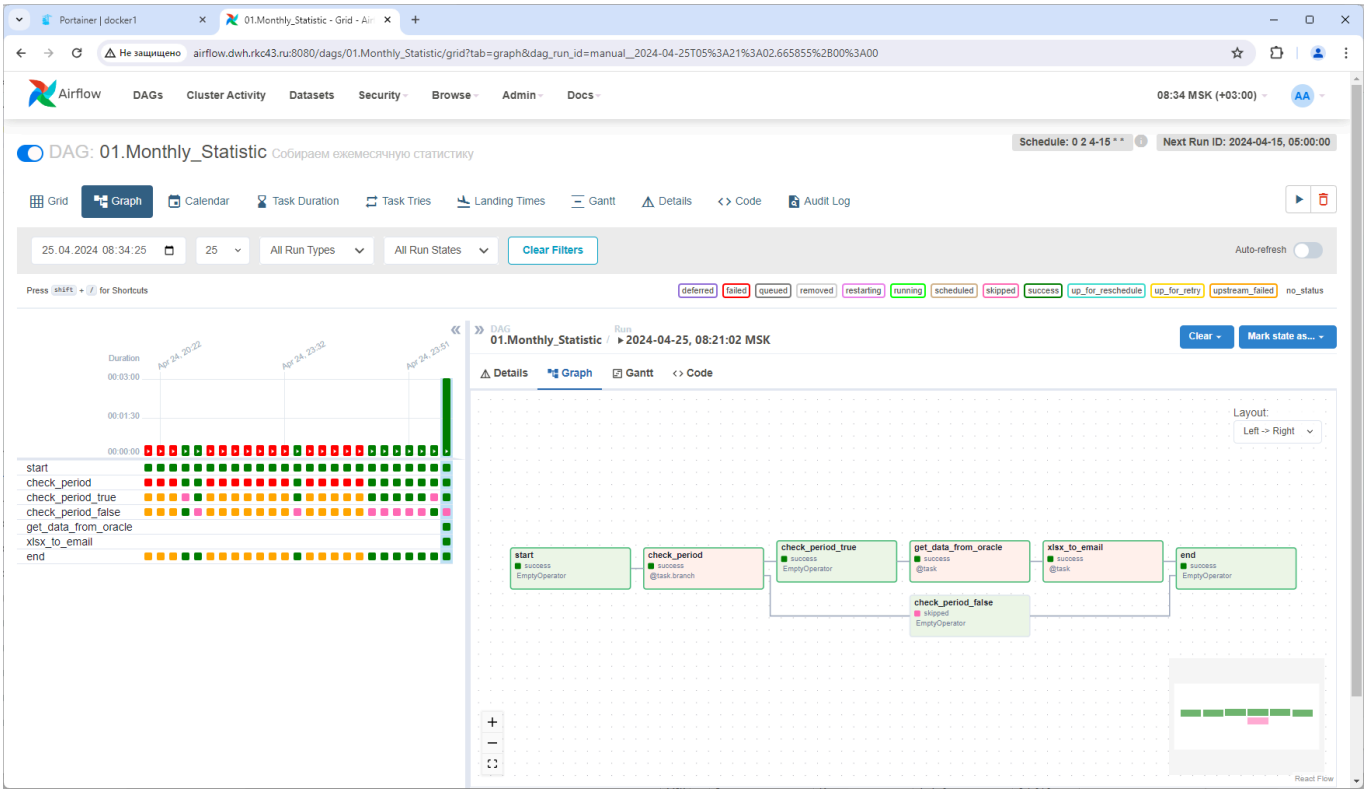


Рисунок 8. Графическое представление DAG 01_statistic.py

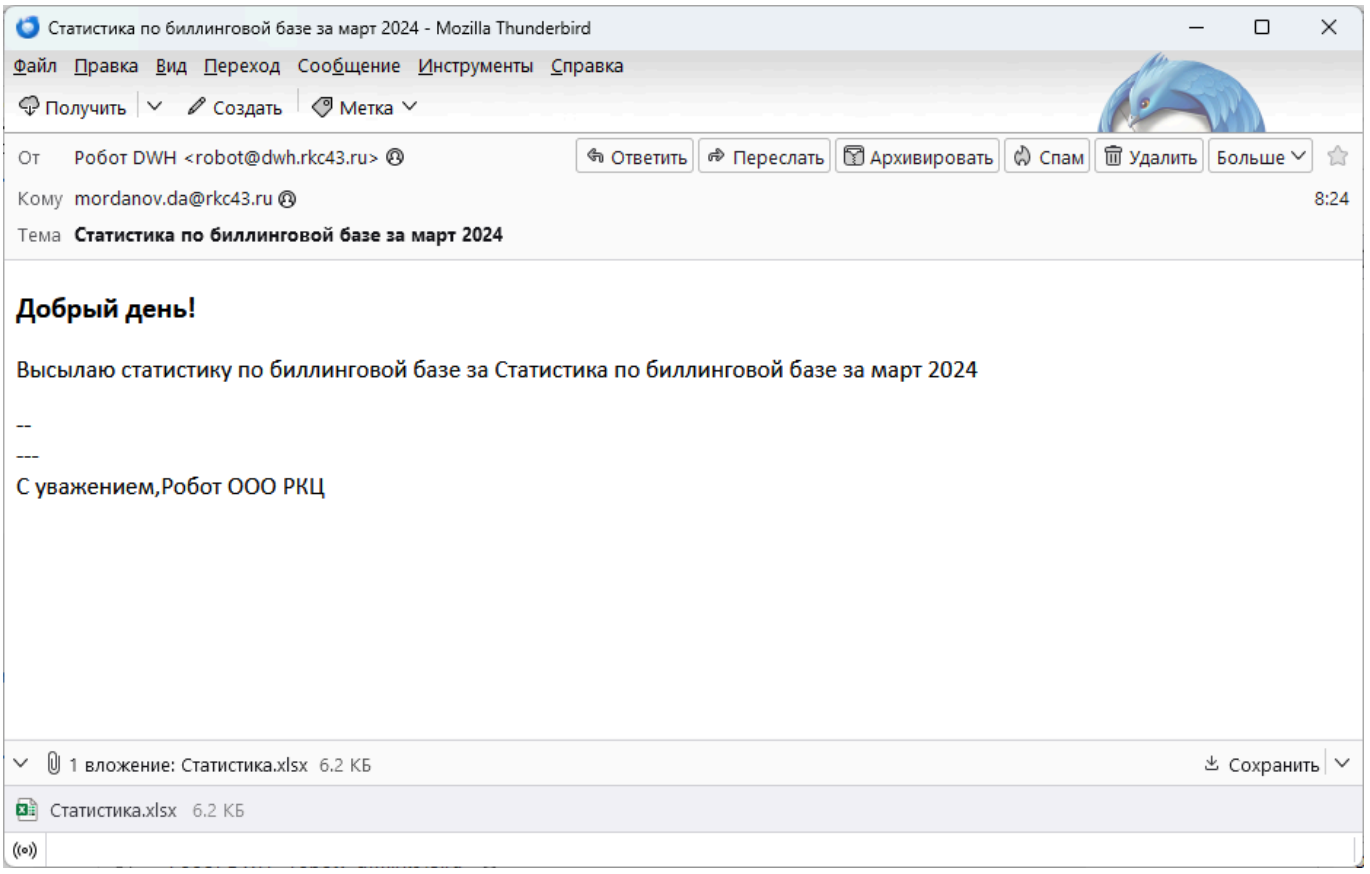
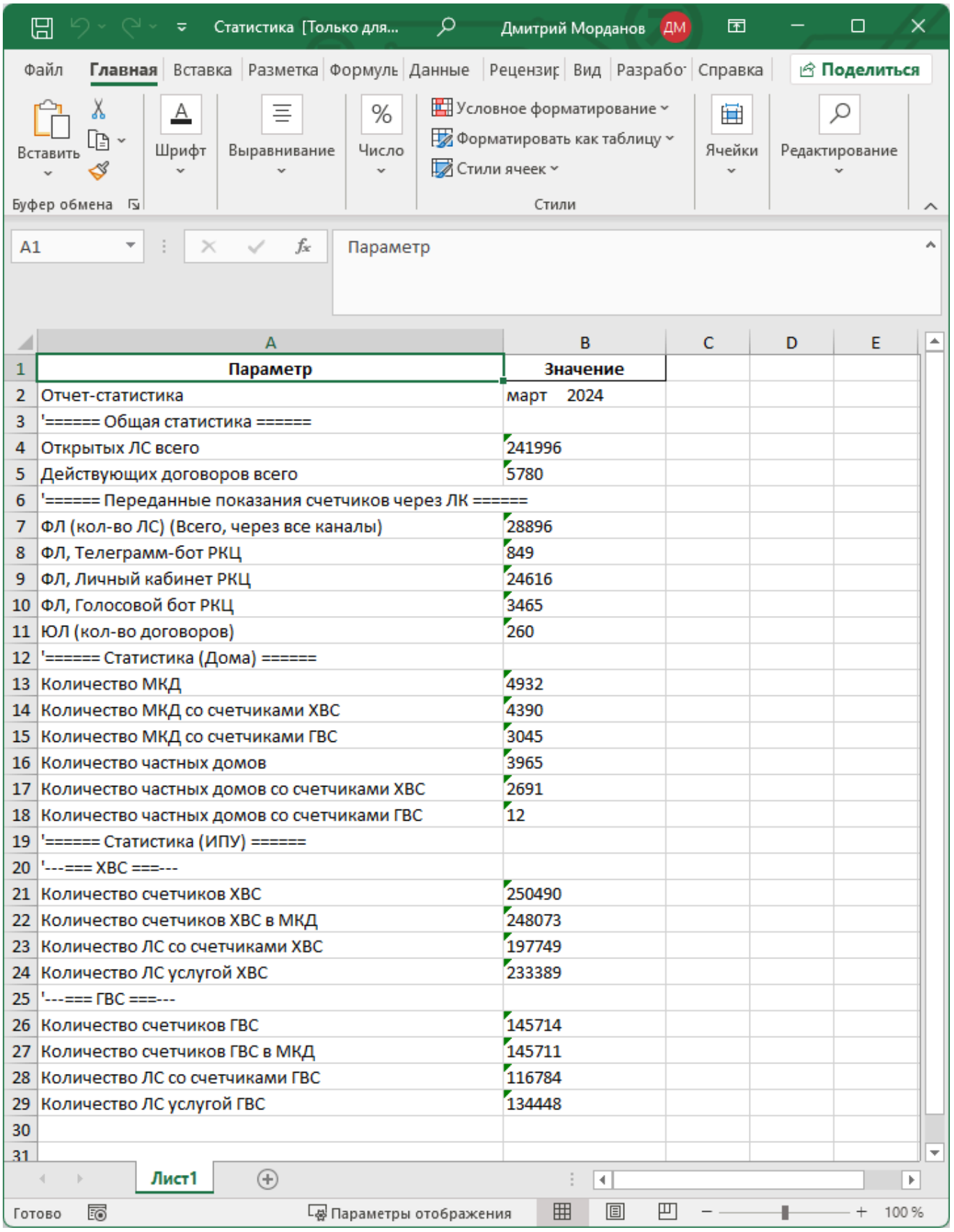


Рисунок 9. Письмо с вложением



	A	B	C	D	E
1	**Параметр**	**Значение**			
2	Отчет-статистика	март 2024			
3	'====== Общая статистика ======				
4	Открытых ЛС всего	241996			
5	Действующих договоров всего	5780			
6	'====== Переданные показания счетчиков через ЛК ======				
7	ФЛ (кол-во ЛС) (Всего, через все каналы)	28896			
8	ФЛ, Телеграмм-бот РКЦ	849			
9	ФЛ, Личный кабинет РКЦ	24616			
10	ФЛ, Голосовой бот РКЦ	3465			
11	ЮЛ (кол-во договоров)	260			
12	'====== Статистика (Дома) ======				
13	Количество МКД	4932			
14	Количество МКД со счетчиками ХВС	4390			
15	Количество МКД со счетчиками ГВС	3045			
16	Количество частных домов	3965			
17	Количество частных домов со счетчиками ХВС	2691			
18	Количество частных домов со счетчиками ГВС	12			
19	'====== Статистика (ИПУ) ======				
20	'---=== ХВС ===---				
21	Количество счетчиков ХВС	250490			
22	Количество счетчиков ХВС в МКД	248073			
23	Количество ЛС со счетчиками ХВС	197749			
24	Количество ЛС услугой ХВС	233389			
25	'---=== ГВС ===---				
26	Количество счетчиков ГВС	145714			
27	Количество счетчиков ГВС в МКД	145711			
28	Количество ЛС со счетчиками ГВС	116784			
29	Количество ЛС услугой ГВС	134448			
30					
31					

Рисунок 10. Результат выполнения отчета по статистике за прошедший месяц

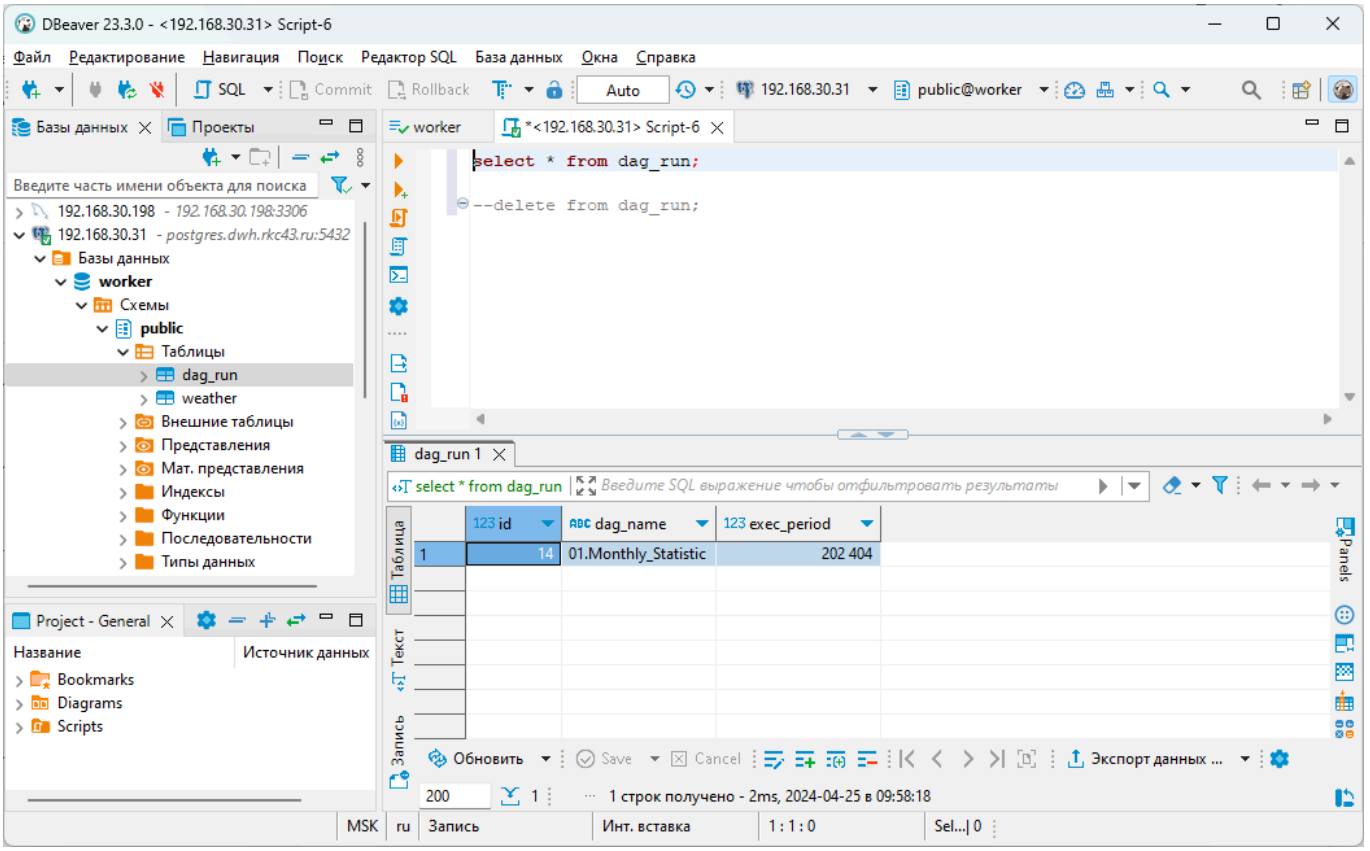


Рисунок 11. Таблица dag_run, в которой фиксируется выполнение процесса в текущем периоде.

3.3.3. ETL процесс для call-центра

Задача: В ООО РКЦ в Call-центре используется программа для поиска лицевого счета абонента в базах данных. Так сложилось, что баз данных для расчетов услуг ЖКХ несколько. В отдельной базе ведутся расчеты по одной управляющей компании. Также - товарищества жильцов имеют право менять управляющие компании. Таким образом получается, что один и тот же дом (вместе с лицевыми счетами абонентов) может находится в разных базах данных в разные периоды расчетов. В то же время, в Call-центре требуется оперативность в поиске информации по лицевому счету.

В итоге была разработана программа поиска информации по лицевым счетам, но ей требовалась подготовка данных для работы.

Данные находятся на удаленном сервере, в расчетных базах. Их требуется выгрузить из каждой базы - и сохранить требуемую информацию в одной таблице на локальном сервере. Удаленный сервер имеет ограничение на количество одновременных подключений. Это надо обязательно учитывать.

Данные обрабатываются с помощью СУБД FireBird. Программа поиска обращается также - к СУБД FireBird, но на другом сервере.

Реализация:

Задача кажется, на первый взгляд, простой, но погрузившись в проблему - начинаем понимать, что не все так просто. Выгрузка информации с удаленного сервера может вестись параллельно, в несколько потоков. После - идет очистка старой информации из таблицы, находящейся на локальном сервере. Причем нужно учитывать, что если из какой либо базы данных не удалось получить информацию, то очищать рабочую таблицу не требуется, задачу нужно завершить с ошибкой - и уже после разбираться, что же произошло, т.к. более критично иметь полный набор немного устаревшей информации, чем частичные, но новые данные.

Если все данные выгружены корректно - производится очистка старых данных и загрузка новых. Загрузка новых данных может быть выполнена параллельно, для каждой выгрузки на предыдущем шаге.

SQL скрипт выгрузки данных из одной базы данных см. Приложение 7. Файл 03_pump_search_all_zbases.sql

В итоге - был разработан DAG-файл (см. Приложение 8. Файл 03_pump_search_all_zbases.py), который состоит из следующих заданий:

- **start** - пустой оператор - используется как единая точка входа
- **get_data_zbase(i)** - Динамически генерируемые задачи выгрузки данных из удаленного сервера. Данные находятся в разных базах, потому можно выполнять выгрузки данных параллельно и независимо друг от друга, но с учетом количества возможных каналов одновременного подключения к серверу. Эту часть алгоритма делаем максимально параллельной, занимая все возможные каналы в одну

единицу времени. Также учтем, что блок параллельно-последовательных задач должен быть симметричен (строго прямоугольной формы), поэтому последний блок параллельных задач дополняем пустыми операторами для соблюдения этого условия. Результат выгрузки сохраняем в объекте XCom для передачи следующей задаче.

- **clean_kvk_base()** - Очищает рабочую таблицу, с учетом того, что все предыдущие задачи выгрузки данных завершились успешно. Если хотя бы одна загрузка данных не выполнена - задача не выполняется.
- **push_data_to_kvk(i)** - Динамически генерируемые задачи загрузки данных в рабочую таблицу. Могут выполняться параллельно для каждой выгрузки данных. Данные для загрузки берем из объекта XCom, сгенерированные предыдущими задачами.
- **end** - используется как единая точка выхода, а также - производит очистку данных в XCom (все данные, находящиеся в XCom с нашим DAG_ID просто удаляем).

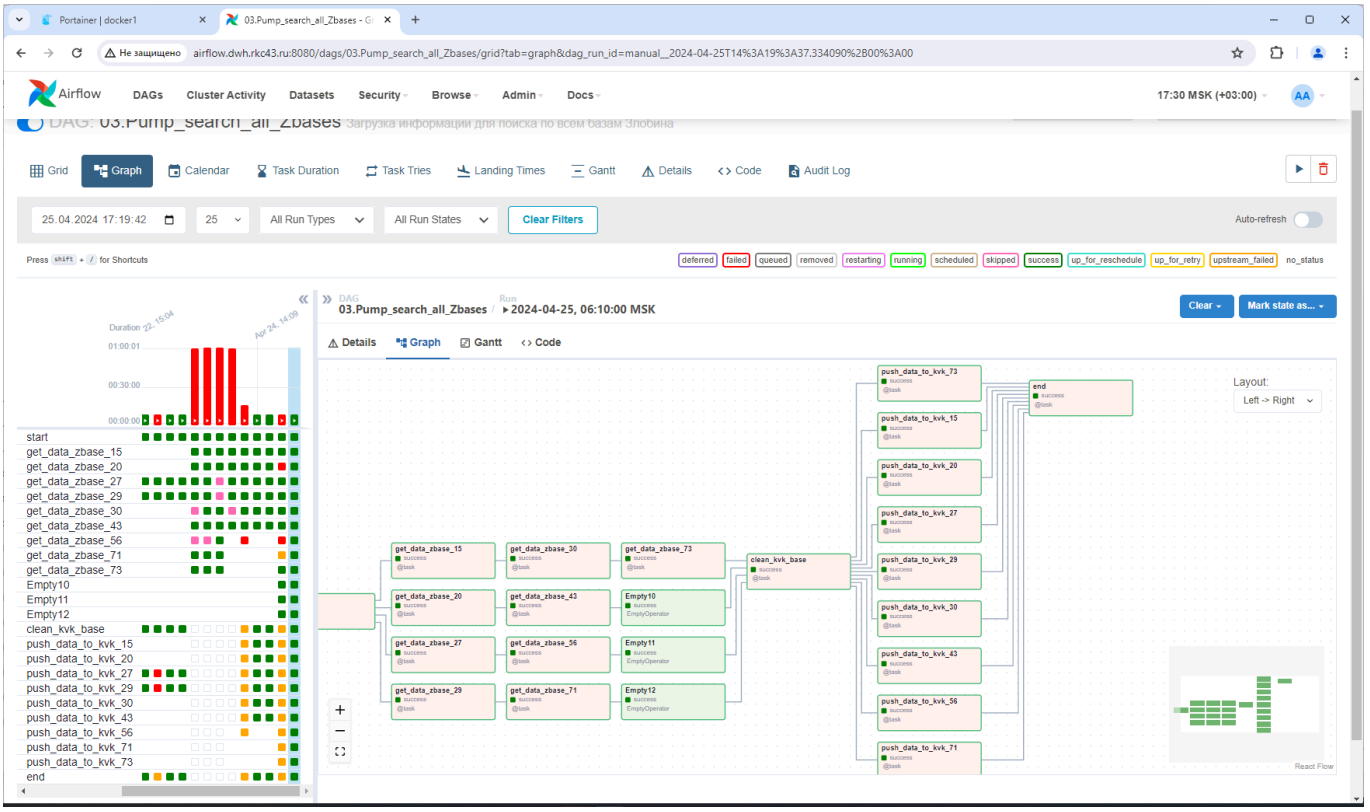


Рисунок 12. Графическое представление DAG 03_pump_search_all_zbases.py

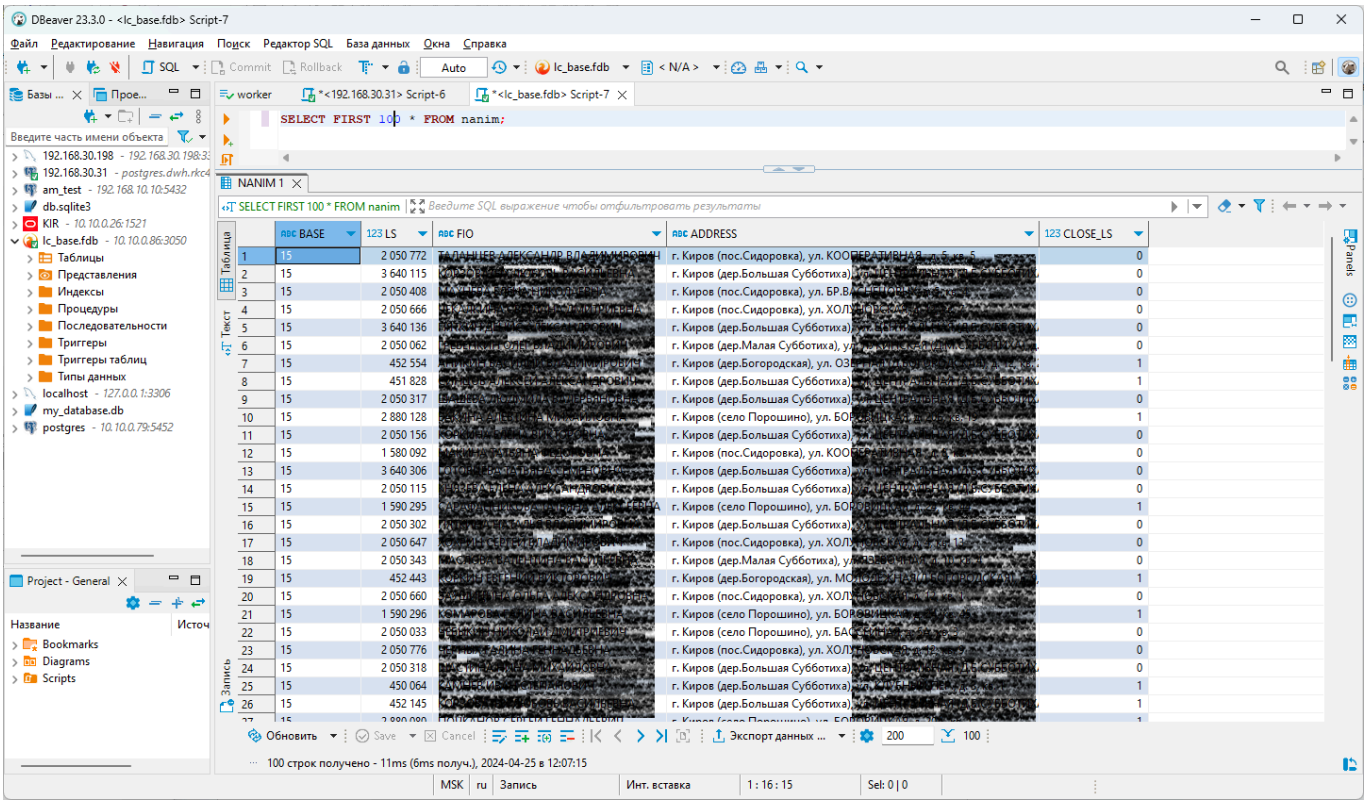


Рисунок 13. Результат выгрузки в рабочую таблицу.

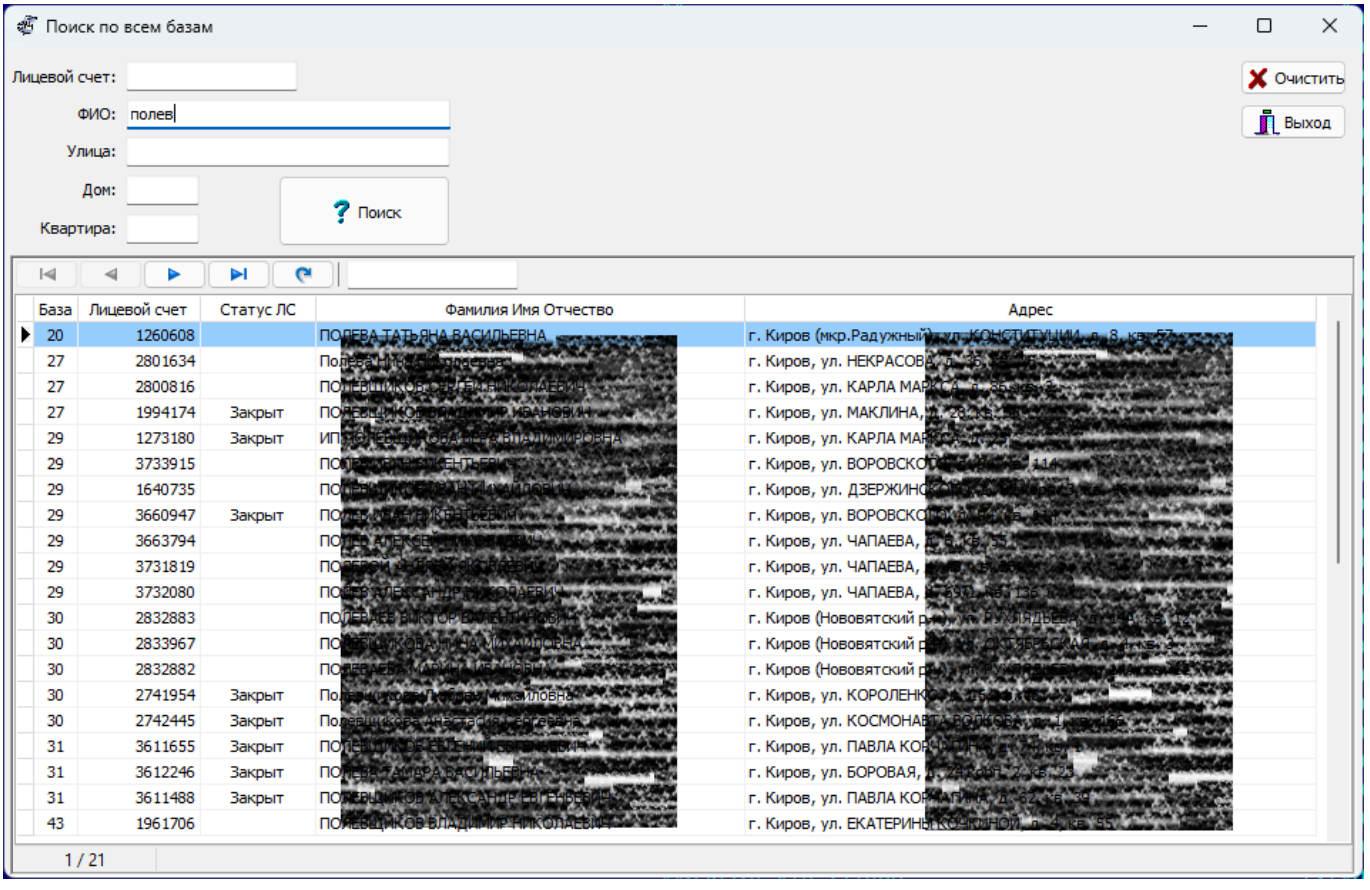


Рисунок 14. Работа программы “Поиск по всем базам”

Данные содержат персональную информацию, поэтому в этой работе их приходится скрывать.

Заключение

Внедрение Apache Airflow в организацию ООО РКЦ является стратегическим шагом, направленным на повышение эффективности работы самой организации, так и получение дополнительных преимуществ над конкурентами. Этот проект не только улучшает эффективность и надежность процессов обработки данных, но и способствует повышению производительности и качества аналитики. В заключение проекта, следует подчеркнуть несколько ключевых моментов.

1. Преимущества Apache Airflow:

Внедрение Apache Airflow предоставило множество преимуществ, таких как гибкость в планировании и масштабировании задач, интеграция с различными источниками данных, мощные инструменты мониторинга и управления выполнением процессов. Эти преимущества помогают ООО РКЦ оптимизировать работу с данными, ускоряя процесс принятия решений и повышая конкурентоспособность.

2. Эффективность и надежность:

Внедрение Apache Airflow позволило повысить эффективность и надежность процессов обработки данных за счет автоматизации и стандартизации задач, а также мониторинга и управления их выполнением. Это способствует улучшению качества данных и снижению рисков ошибок и задержек.

3. Улучшение производительности и качества аналитики:

Apache Airflow обеспечивает ускоренный и более точный анализ данных за счет оптимизации процессов обработки и предоставления оперативной информации о выполнении задач. Это помогает сократить время от идеи до результата и повысить качество принимаемых решений на основе данных.

4. Постоянное совершенствование:

Внедрение Apache Airflow - это лишь первый шаг на пути к постоянному совершенствованию процессов обработки данных в ООО РКЦ. Организация будет продолжать развивать и оптимизировать свои рабочие процессы, итеративно внедряя новые функции и улучшения на основе Apache Airflow, чтобы оставаться конкурентоспособными в своем сегменте рынка.

5. Вовлечение персонала и обучение:

Важным аспектом успешного внедрения Apache Airflow явилось обучение персонала ИТ- отдела и поддержка их в процессе адаптации к новым рабочим процессам и инструментам. Обучение сотрудников помогает повысить эффективность использования внедренного решения и обеспечивает успешное развитие проекта в дальнейшем.

В итоге, внедрение Apache Airflow в ООО РКЦ открыло новые возможности для оптимизации процессов обработки данных, улучшения качества аналитики и повышения конкурентоспособности. Этот проект явился ключевым шагом в развитии организации и обеспечивает ее готовность к вызовам современного рынка.

Список используемой литературы

1. *Книга: Бас Харенслак, Джулиан де Руйтер. Apache Airflow и конвейеры обработки данных. Москва, Издательство ДМК, 2022. 505 стр.*
2. *Книга: Иан Милл, Эйдан Хобсон Сейерс. Docker на практике. Москва, Издательство ДМК, 2020. 516 стр.*
3. *Книга: Alex Nuijten, Patrick Barel. Modern Oracle Database Programming. Apress, Oosterhout, Noord-Brabant, The Netherlands, 2023. 576 pages.*
4. *Изучаем Docker. https://habr.com/ru/companies/ruvds/articles/438796/*
5. *Airflow — инструмент, чтобы удобно и быстро разрабатывать и поддерживать batch-процессы обработки данных https://habr.com/ru/companies/vk/articles/339392/*
6. *Airflow docker-compose: https://github.com/coder2j/airflow-docker*
7. *Portainer https://www.portainer.io/*

Приложения

Приложение 1. Файл Dockerfile

```
FROM apache/airflow:2.8.3
LABEL maintainer="mordanov.da@rkc43.ru"
COPY fbclient/libfbclient.so.2.5.9 /opt/airflow/
COPY fbclient/libncurses.so.5.9 /opt/airflow/
COPY fbclient/libtinfo.so.5.9 /opt/airflow/
USER root
RUN apt-get update \
  && apt-get install -y --no-install-recommends \
      libaio1 alien wget \
  && apt-get autoremove -yqq --purge \
  && apt-get clean \
  && rm -rf /var/lib/apt/lists/* \
  && wget
https://download.oracle.com/otn_software/linux/instantclient/1922000/oracle-instant
client19.22-basiclite-19.22.0.0.0-1.x86_64.rpm \
  && alien -i --scripts
oracle-instantclient19.22-basiclite-19.22.0.0.0-1.x86_64.rpm \
  && rm -f oracle-instantclient19.22-basiclite-19.22.0.0.0-1.x86_64.rpm \
  && apt-get remove alien wget -y \
  && mv /opt/airflow/libfbclient.so.2.5.9 /usr/lib \
  && mv /opt/airflow/libncurses.so.5.9 /usr/lib \
  && mv /opt/airflow/libtinfo.so.5.9 /usr/lib \
  && chown root:root /usr/lib/libfbclient.so.2.5.9 \
  && chown root:root /usr/lib/libncurses.so.5.9 \
  && chown root:root /usr/lib/libtinfo.so.5.9 \
  && ln -s libfbclient.so.2.5.9 /usr/lib/libfbclient.so \
  && ln -s libncurses.so.5.9 /usr/lib/libncurses.so.5 \
  && ln -s libtinfo.so.5.9 /usr/lib/libtinfo.so.5
USER airflow
ADD requirements.txt .
RUN pip install apache-airflow==${AIRFLOW_VERSION} -r requirements.txt
RUN rm ./requirements.txt
```

Приложение 2. Файл docker-compose.yaml

```
$ cat /opt/airflow/docker-compose.yaml
# Licensed to the Apache Software Foundation (ASF) under one
# or more contributor license agreements.  See the NOTICE file
# distributed with this work for additional information
# regarding copyright ownership.  The ASF licenses this file
# to you under the Apache License, Version 2.0 (the
# "License"); you may not use this file except in compliance
# with the License.  You may obtain a copy of the License at
#
#   http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing,
# software distributed under the License is distributed on an
# "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
# KIND, either express or implied.  See the License for the
# specific language governing permissions and limitations
# under the License.
#

# Basic Airflow cluster configuration for CeleryExecutor with Redis and PostgreSQL.
#
# WARNING: This configuration is for local development. Do not use it in a
production deployment.
#
# This configuration supports basic configuration using environment variables or an
.env file
# The following variables are supported:
#
# AIRFLOW_IMAGE_NAME          - Docker image name used to run Airflow.
#                               Default: apache/airflow:2.8.3
# AIRFLOW_UID                 - User ID in Airflow containers
#                               Default: 50000
# AIRFLOW_PROJ_DIR            - Base path to which all the files will be volumed.
#                               Default: .
# Those configurations are useful mostly in case of standalone testing/running
Airflow in test/try-out mode
#
# _AIRFLOW_WWW_USER_USERNAME   - Username for the administrator account (if
requested).
#                               Default: airflow
# _AIRFLOW_WWW_USER_PASSWORD   - Password for the administrator account (if
requested).
#                               Default: airflow
# _PIP_ADDITIONAL_REQUIREMENTS - Additional PIP requirements to add when starting
all containers.
#                               Use this option ONLY for quick checks. Installing
requirements at container
#                               startup is done EVERY TIME the service is started.
#                               A better way is to build a custom image or extend the
official image
```

```
#                             as described in
https://airflow.apache.org/docs/docker-stack/build.html.
#                             Default: ''
#
# Feel free to modify this file to suit your needs.
---
x-airflow-common:
  &airflow-common
  # In order to add custom dependencies or upgrade provider packages you can use
your extended image.
  # Comment the image line, place your Dockerfile in the directory where you placed
the docker-compose.yaml
  # and uncomment the "build" line below, Then run `docker-compose build` to build
the images.
#  image: ${AIRFLOW_IMAGE_NAME:-apache/airflow:2.8.3}
  image: ${AIRFLOW_IMAGE_NAME:-my-airflow:0.0.1}
  # build: .
  environment:
      &airflow-common-env
      AIRFLOW__CORE__EXECUTOR: CeleryExecutor
      AIRFLOW__DATABASE__SQL_ALCHEMY_CONN:
postgresql+psycopg2://airflow:airflow@postgres/airflow
      AIRFLOW__CELERY__RESULT_BACKEND:
db+postgresql://airflow:airflow@postgres/airflow
      AIRFLOW__CELERY__BROKER_URL: redis://:@redis:6379/0
      AIRFLOW__CORE__FERNET_KEY: ''
      AIRFLOW__CORE__DAGS_ARE_PAUSED_AT_CREATION: 'true'
      AIRFLOW__CORE__LOAD_EXAMPLES: 'true'
#     AIRFLOW__CORE__LOAD_EXAMPLES: 'false'
      AIRFLOW__API__AUTH_BACKENDS:
'airflow.api.auth.backend.basic_auth,airflow.api.auth.backend.session'
      # yamllint disable rule:line-length
      # Use simple http server on scheduler for health checks
      # See
https://airflow.apache.org/docs/apache-airflow/stable/administration-and-deployment
/logging-monitoring/check-health.html#scheduler-health-check-server
      # yamllint enable rule:line-length
      AIRFLOW__SCHEDULER__ENABLE_HEALTH_CHECK: 'true'
      # WARNING: Use _PIP_ADDITIONAL_REQUIREMENTS option ONLY for a quick checks
      # for other purpose (development, test and especially production usage)
build/extend Airflow image.
      _PIP_ADDITIONAL_REQUIREMENTS: ${_PIP_ADDITIONAL_REQUIREMENTS:-}
  volumes:
      - ${AIRFLOW_PROJ_DIR:-.}/dags:/opt/airflow/dags
      - ${AIRFLOW_PROJ_DIR:-.}/logs:/opt/airflow/logs
      - ${AIRFLOW_PROJ_DIR:-.}/config:/opt/airflow/config
      - ${AIRFLOW_PROJ_DIR:-.}/plugins:/opt/airflow/plugins
      - ${AIRFLOW_PROJ_DIR:-.}/temp:/opt/airflow/temp
  user: "${AIRFLOW_UID:-50000}:0"
  depends_on:
      &airflow-common-depends-on
      redis:
      condition: service_healthy
      postgres:
```

```
      condition: service_healthy

services:
  postgres:
      image: postgres:13
      environment:
      POSTGRES_USER: airflow
      POSTGRES_PASSWORD: airflow
      POSTGRES_DB: airflow
      volumes:
      - postgres-db-volume:/var/lib/postgresql/data
      healthcheck:
      test: ["CMD", "pg_isready", "-U", "airflow"]
      interval: 10s
      retries: 5
      start_period: 5s
      restart: always

  redis:
      image: redis:latest
      expose:
      - 6379
      healthcheck:
      test: ["CMD", "redis-cli", "ping"]
      interval: 10s
      timeout: 30s
      retries: 50
      start_period: 30s
      restart: always

  airflow-webserver:
      <<: *airflow-common
      command: webserver
      ports:
      - "192.168.30.31:8080:8080"
      healthcheck:
      test: ["CMD", "curl", "--fail", "http://localhost:8080/health"]
      interval: 30s
      timeout: 10s
      retries: 5
      start_period: 30s
      restart: always
      depends_on:
      <<: *airflow-common-depends-on
      airflow-init:
      condition: service_completed_successfully

  airflow-scheduler:
      <<: *airflow-common
      command: scheduler
      healthcheck:
      test: ["CMD", "curl", "--fail", "http://localhost:8974/health"]
      interval: 30s
      timeout: 10s
```

```
      retries: 5
      start_period: 30s
      restart: always
      depends_on:
      <<: *airflow-common-depends-on
      airflow-init:
      condition: service_completed_successfully

  airflow-worker:
      <<: *airflow-common
      command: celery worker
      healthcheck:
      # yamllint disable rule:line-length
      test:
      - "CMD-SHELL"
      - 'celery --app airflow.providers.celery.executors.celery_executor.app
inspect ping -d "celery@$${HOSTNAME}" || celery --app
airflow.executors.celery_executor.app inspect ping -d "celery@$${HOSTNAME}"'
      interval: 30s
      timeout: 10s
      retries: 5
      start_period: 30s
      environment:
      <<: *airflow-common-env
      # Required to handle warm shutdown of the celery workers properly
      # See
https://airflow.apache.org/docs/docker-stack/entrypoint.html#signal-propagation
      DUMB_INIT_SETSID: "0"
      restart: always
      depends_on:
      <<: *airflow-common-depends-on
      airflow-init:
      condition: service_completed_successfully

  airflow-triggerer:
      <<: *airflow-common
      command: triggerer
      healthcheck:
      test: ["CMD-SHELL", 'airflow jobs check --job-type TriggererJob --hostname
"$${HOSTNAME}"']
      interval: 30s
      timeout: 10s
      retries: 5
      start_period: 30s
      restart: always
      depends_on:
      <<: *airflow-common-depends-on
      airflow-init:
      condition: service_completed_successfully

  airflow-init:
      <<: *airflow-common
      entrypoint: /bin/bash
      # yamllint disable rule:line-length
```

```
      command:
      - -c
      - |
      if [[ -z "${AIRFLOW_UID}" ]]; then
      echo
      echo -e "\033[1;33mWARNING!!!: AIRFLOW_UID not set!\e[0m"
      echo "If you are on Linux, you SHOULD follow the instructions below to set "
      echo "AIRFLOW_UID environment variable, otherwise files will be owned by
root."
      echo "For other operating systems you can get rid of the warning with
manually created .env file:"
      echo "See:
https://airflow.apache.org/docs/apache-airflow/stable/howto/docker-compose/index.ht
ml#setting-the-right-airflow-user"
      echo
      fi
      one_meg=1048576
      mem_available=$$(($$(getconf _PHYS_PAGES) * $$(getconf PAGE_SIZE) / one_meg))
      cpus_available=$$(grep -cE 'cpu[0-9]+' /proc/stat)
      disk_available=$$(df / | tail -1 | awk '{print $$4}')
      warning_resources="false"
      if (( mem_available < 4000 )) ; then
      echo
      echo -e "\033[1;33mWARNING!!!: Not enough memory available for Docker.\e[0m"
      echo "At least 4GB of memory required. You have $$(numfmt --to iec
$$((mem_available * one_meg)))"
      echo
      warning_resources="true"
      fi
      if (( cpus_available < 2 )); then
      echo
      echo -e "\033[1;33mWARNING!!!: Not enough CPUS available for Docker.\e[0m"
      echo "At least 2 CPUs recommended. You have $${cpus_available}"
      echo
      warning_resources="true"
      fi
      if (( disk_available < one_meg * 10 )); then
      echo
      echo -e "\033[1;33mWARNING!!!: Not enough Disk space available for
Docker.\e[0m"
      echo "At least 10 GBs recommended. You have $$(numfmt --to iec
$$((disk_available * 1024 )))"
      echo
      warning_resources="true"
      fi
      if [[ $${warning_resources} == "true" ]]; then
      echo
      echo -e "\033[1;33mWARNING!!!: You have not enough resources to run Airflow
(see above)!\e[0m"
      echo "Please follow the instructions to increase amount of resources
available:"
      echo "
https://airflow.apache.org/docs/apache-airflow/stable/howto/docker-compose/index.ht
ml#before-you-begin"
```

```
      echo
      fi
      mkdir -p /sources/logs /sources/dags /sources/plugins
      chown -R "${AIRFLOW_UID}:0" /sources/{logs,dags,plugins}
      exec /entrypoint airflow version
      # yamllint enable rule:line-length
      environment:
      <<: *airflow-common-env
      _AIRFLOW_DB_MIGRATE: 'true'
      _AIRFLOW_WWW_USER_CREATE: 'true'
      _AIRFLOW_WWW_USER_USERNAME: ${_AIRFLOW_WWW_USER_USERNAME:-airflow}
      _AIRFLOW_WWW_USER_PASSWORD: ${_AIRFLOW_WWW_USER_PASSWORD:-airflow}
      _PIP_ADDITIONAL_REQUIREMENTS: ''
      user: "0:0"
      volumes:
      - ${AIRFLOW_PROJ_DIR:-.}:/sources

  airflow-cli:
      <<: *airflow-common
      profiles:
      - debug
      environment:
      <<: *airflow-common-env
      CONNECTION_CHECK_MAX_COUNT: "0"
      # Workaround for entrypoint issue. See:
https://github.com/apache/airflow/issues/16252
      command:
      - bash
      - -c
      - airflow

  # You can enable flower by adding "--profile flower" option e.g. docker-compose
--profile flower up
  # or by explicitly targeted on the command line e.g. docker-compose up flower.
  # See: https://docs.docker.com/compose/profiles/
  flower:
      <<: *airflow-common
      command: celery flower
      profiles:
      - flower
      ports:
      - "192.168.30.31:5555:5555"
      healthcheck:
      test: ["CMD", "curl", "--fail", "http://localhost:5555/"]
      interval: 30s
      timeout: 10s
      retries: 5
      start_period: 30s
      restart: always
      depends_on:
      <<: *airflow-common-depends-on
      airflow-init:
      condition: service_completed_successfully
```

```
volumes:
  postgres-db-volume:
$
```

Приложение 3. Файл 02.ecolog_counters.sql

```
-- ИПУ на договорах юрлиц
--избранные, для экологов

select
  c.CONTRACT_NMBR as "№ договора"
  ,cc8.name as "Наименование юр.лица"
  , rk_ref.name as "Тип объекта"
  ,case when coalesce(t.name,t1.name) is null then 'г. Киров' else
coalesce(tt.short_name,tt1.short_name) || ' ' || coalesce(t.name,t1.name) end
  || case when s.NAME is null then '' else ', ' end || st.SHORT_NAME || ' ' ||
s.NAME || ', д. '||
  h.HOUSE || case when h.CORPUS is not null then ', корп. '|| h.corpus end
  || case when h.building is not null then ', стр. '|| h.building end
  || case when na.flat is not null then ', кв. ' || na.flat end as "Полный адрес"

  ,ut.name as "Тип пользователя"
  ,pgCN_Char.fGet_Descr(rk.tu_rlty_kid,
            (
            select cr.CN_CHRCT_ID
            from CN_CHAR_TYPE ct
                  join CN_CHRCTRSTC_GROUP cg on cg.CHAR_TYPE_ID = ct.CHAR_TYPE_ID
                  join CN_CHARACTERISTIC_REF cr on cr.CN_CHRCTRSTC_GROUP_ID =
cg.CN_CHRCTRSTC_GROUP_ID
            where ct.NAME = 'Характеристики объекта недвижимости'
                  and cr.CODE = 'NLC846169'
                  and cr.NAME = 'Контролеры'
            ), sysdate) as "Контролер"
  ,cc1.ser_number as "Счетчик (модель)"
  ,cc1.cntr_number as "Серийный № счетчика"
  ,r.Name as "Услуга"
  , case when r.Name = 'Ливневые стоки' then
to_number(pgCN_Char.fGet_Descr(rk.tu_rlty_kid,
            (
            select cr.CN_CHRCT_ID
            from CN_CHAR_TYPE ct
                  join CN_CHRCTRSTC_GROUP cg on cg.CHAR_TYPE_ID = ct.CHAR_TYPE_ID
                  join CN_CHARACTERISTIC_REF cr on cr.CN_CHRCTRSTC_GROUP_ID =
cg.CN_CHRCTRSTC_GROUP_ID
            where ct.NAME = 'Характеристики объекта недвижимости'
                  and cr.CODE = 'LIVN'
            ), sysdate))
      else
      Fgetindvol(ck.cntr_contract_key_id, cc1.cd_cntr_kid, sk.cntr_service_key_id,
bl.bl_bill_id) end as "Объем"
  , (select value from cd_indication where
cd_ind_id=pgcd_indication.fGet_LastIndicationID(cc1.cd_cntr_kid)) as "Показания"
  , (select date_ind_take from cd_indication where
cd_ind_id=pgcd_indication.fGet_LastIndicationID(cc1.cd_cntr_kid)) as "Дата приема
показаний"
  , (select name from cd_indication_source where cd_indic_src_id=(select
cd_indic_src_id from cd_indication where
```

```
cd_ind_id=pgcd_indication.fGet_LastIndicationID(cc1.cd_cntr_kid))) as "Источник
показаний"

from cn_contract_key ck
join cn_contract c on c.cntr_contract_key_id = ck.cntr_contract_key_id and
c.is_active=1 and c.cntr_status_id=5 and sysdate between c.date_begin and
c.date_end

join Cn_realty_bunch_key cc on ck.cntr_contract_key_id=cc.cntr_contract_key_id
join TU_REALTY_KEY rk on cc.TU_RLTY_KID=rk.tu_rlty_kid
join tu_realty_species_ref rk_ref on rk.tu_rlty_spcs_id=rk_ref.tu_rlty_spcs_id

join NS_ADDRESS na on na.NS_ADR_ID = rk.NS_ADR_ID
left join NS_HOUSE h on h.NS_HOS_ID = na.NS_HOS_ID
left join NS_STREET s on s.NS_STRT_ID = h.NS_STRT_ID
left join NS_STREET_TYPE st on st.NS_STRTTYP_ID = s.NS_STRTTYP_ID

left join NS_TOWN t on t.NS_TWN_ID = s.NS_TWN_ID
left join NS_TOWN t1 on t1.NS_TWN_ID = h.NS_TWN_ID
left join ns_town_type tt on t.ns_twntyp_id=tt.ns_twntyp_id
left join ns_town_type tt1 on t1.ns_twntyp_id=tt1.ns_twntyp_id
left outer join us_flat_account_ver fr on fr.us_flat_account_kid = ck.us_account_id

join cn_service_key sk on c.cntr_contract_key_id = sk.cntr_contract_key_id and
sk.tu_rlty_kid=cc.tu_rlty_kid
join US_COUNTERAGENT q on q.US_COUNTERAGENT_ID = ck.US_COUNTERAGENT_ID
join US_CORP_CARD cc8 on cc8.US_CORP_ID = q.US_CORP_ID
join ns_user_type ut on ut.ns_user_typ_id = q.ns_user_typ_id
left join ns_svc_ref r on r.ns_svc_id = sk.ns_svc_id

left join cd_const_service_bunch b on sk.cntr_service_key_id = b.us_cntr_svc_id and
b.is_active = 1 and sysdate between b.date_begin and b.date_end
left join cd_counter cc1 on cc1.cd_cntr_kid = b.cd_cntr_kid and cc1.is_active=1
, bl_bill_periods bl

where ck.us_account_id is null
and fr.us_flat_account_kid is null
and (cc1.ser_number is not null or (r.Name='Ливневые стоки' and
pgCN_Char.fGet_Descr(rk.tu_rlty_kid,
            (
            select cr.CN_CHRCT_ID
            from CN_CHAR_TYPE ct
                  join CN_CHRCTRSTC_GROUP cg on cg.CHAR_TYPE_ID = ct.CHAR_TYPE_ID
                  join CN_CHARACTERISTIC_REF cr on cr.CN_CHRCTRSTC_GROUP_ID =
cg.CN_CHRCTRSTC_GROUP_ID
            where ct.NAME = 'Характеристики объекта недвижимости'
                  and cr.CODE = 'LIVN'
            ), sysdate) is not null))

--and c.contract_nmbr='42-0183' and h.ns_hos_id in (2599443)  ----!!!!!!!!!!!!!!!


and (
```

```
  (c.contract_nmbr='43-2204' and h.ns_hos_id in
(3644527,2607800,2607805,2608103,3935839)) or
  (c.contract_nmbr='42-0002' and h.ns_hos_id in (3636715)) or
  (c.contract_nmbr='42-5361' and h.ns_hos_id in (2605828)) or
  (c.contract_nmbr='42-0130' and h.ns_hos_id in (3644785)) or
  (c.contract_nmbr='43-1368' and h.ns_hos_id in (3637059)) or
  (c.contract_nmbr='42-6142' and h.ns_hos_id in (2612119,2610172,2612460,2615494))
or
  (c.contract_nmbr='01-138/19-н/42-1808' and h.ns_hos_id in (2612352)) or
  (c.contract_nmbr='01-355/19-н/42-0023' and h.ns_hos_id in
(2613635,3377386,2601590,2603649,3636687,2614463)) or
  (c.contract_nmbr='42-0183' and h.ns_hos_id in (2599443)) or
  (c.contract_nmbr='42-0022' and h.ns_hos_id in (3635577,2615510,3179874)) or
  (c.contract_nmbr='03-137/19-н/42-0003' and h.ns_hos_id in (3636761,2616075)) or
  (c.contract_nmbr='42-0031' and h.ns_hos_id in (2607658,3897401)) or
  (c.contract_nmbr='43-1224' and h.ns_hos_id in (3684853)) or
  (c.contract_nmbr='42-0043' and h.ns_hos_id in (2601793,3639227,3637539)) or
  (c.contract_nmbr='42-0200' and h.ns_hos_id in (3639751,3888698)) or
  (c.contract_nmbr='42-2049' and h.ns_hos_id in
(2613657,2601714,2606513,2605216,3636595,2609968,3639299,2610475,3670424,2610482,26
10382,2613655,3644957,2612471,2616036)) or
  (c.contract_nmbr='42-4788' and h.ns_hos_id in (3635931)) or
  (c.contract_nmbr='42-5031' and h.ns_hos_id in (3386307)) or
  (c.contract_nmbr='03-133/19-н' and h.ns_hos_id in (3557249,3385969)) or
  (c.contract_nmbr='59/14' and h.ns_hos_id in (3081789)) or
  (c.contract_nmbr='42-1021' and h.ns_hos_id in (2611902)) or
  (c.contract_nmbr='42-0230' and h.ns_hos_id in
(2606912,3655743,3655748,3655776,3655785,3655756,3655763,3655760,3655735,3655783,36
55695,3655693)) or
  (c.contract_nmbr='42-0054' and h.ns_hos_id in (2601793)) or
  (c.contract_nmbr='42-3974' and h.ns_hos_id in (2606574)) or
  (c.contract_nmbr='42-2553' and h.ns_hos_id in (3645653,2615049)) or
  (c.contract_nmbr='42-1087' and h.ns_hos_id in (2614531)) or
  (c.contract_nmbr='42-7474' and h.ns_hos_id in (2610593)) or
  (c.contract_nmbr='42-4525' and h.ns_hos_id in (3386307,2613917)) or
  (c.contract_nmbr='42-9103' and h.ns_hos_id in (2610970)) or
  (c.contract_nmbr='42-3075' and h.ns_hos_id in (2604201,3267910)) or
  (c.contract_nmbr='42-9959' and h.ns_hos_id in (2613880)) or
  (c.contract_nmbr='42-2052' and h.ns_hos_id in (3683715)) or
  (c.contract_nmbr='42-1361' and h.ns_hos_id in (2614367)) or
  (c.contract_nmbr='42-5588' and h.ns_hos_id in (2603978)) or
  (c.contract_nmbr='42-4137' and h.ns_hos_id in (3894176,2605823,2607582)) or
  (c.contract_nmbr='42-8050' and h.ns_hos_id in (3910519)) or
  (c.contract_nmbr='42-1702' and h.ns_hos_id in (2612890,2606769)) or
  (c.contract_nmbr='42-0505' and h.ns_hos_id in (3646648)) or
  (c.contract_nmbr='42-0262' and h.ns_hos_id in (2613725)) or
  (c.contract_nmbr='42-0518' and h.ns_hos_id in (2615489)) or
  (c.contract_nmbr='01-139/19-н/42-0160' and h.ns_hos_id in (2612452)) or
  (c.contract_nmbr='42-0082' and h.ns_hos_id in (3655598)) or
  (c.contract_nmbr='87/16' and h.ns_hos_id in (3952166)) or
  (c.contract_nmbr='42-0368' and h.ns_hos_id in (3267892,3346199)) or
  (c.contract_nmbr='42-1858' and h.ns_hos_id in (2610384,2614933)) or
  (c.contract_nmbr='42-3765' and h.ns_hos_id in (2615621)) or
  (c.contract_nmbr='42-2268' and h.ns_hos_id in (2608772)) or
```

```
  (c.contract_nmbr='42-4882' and h.ns_hos_id in (2601667)) or
  (c.contract_nmbr='42-9016' and h.ns_hos_id in (3422006,2604565)) or
  (c.contract_nmbr='42-9151' and h.ns_hos_id in
(3544343,2601261,2599631,2604245,2609541,2611499,2610338,2611112)) or
  (c.contract_nmbr='42-2891' and h.ns_hos_id in (2613362)) or
  (c.contract_nmbr='42-9938' and h.ns_hos_id in (3376928)) or
  (c.contract_nmbr='42-6112' and h.ns_hos_id in (2608955)) or
  (c.contract_nmbr='42-1352' and h.ns_hos_id in (2615956)) or
  (c.contract_nmbr='42-4829' and h.ns_hos_id in (2614427)) or
  (c.contract_nmbr='42-4915' and h.ns_hos_id in (2607895)) or
  (c.contract_nmbr='42-0150' and h.ns_hos_id in (2614147)) or
  (c.contract_nmbr='42-2430' and h.ns_hos_id in (2609836)) or
  (c.contract_nmbr='43-0910' and h.ns_hos_id in (2610536)) or
  (c.contract_nmbr='42-5223' and h.ns_hos_id in (2607281)) or
  (c.contract_nmbr='42-3686' and h.ns_hos_id in (2609813)) or
  (c.contract_nmbr='43-0586' and h.ns_hos_id in (3639807)) or
  (c.contract_nmbr='43-1352' and h.ns_hos_id in (3945439)) or
  (c.contract_nmbr='42-7371' and h.ns_hos_id in (2602585)) or
  (c.contract_nmbr='42-7800' and h.ns_hos_id in (2615641)) or
  (c.contract_nmbr='42-2536' and h.ns_hos_id in (2612926)) or
  (c.contract_nmbr='42-0180' and h.ns_hos_id in (2614367)) or
  (c.contract_nmbr='42-6074' and h.ns_hos_id in (2605247)) or
  (c.contract_nmbr='42-1959' and h.ns_hos_id in (2610333)) or
  (c.contract_nmbr='42-0026' and h.ns_hos_id in
(2612899,2600074,2602146,2608594,2608772,3638437,3638315,2613290)) or
  (c.contract_nmbr='01-211/19-н/42-3136' and h.ns_hos_id in (3644241,3638139)) or
  (c.contract_nmbr='42-2596' and h.ns_hos_id in (3644981)) or
  (c.contract_nmbr='43-0536' and h.ns_hos_id in (3268266)) or
  (c.contract_nmbr='42-5892' and h.ns_hos_id in (3645605)) or
  (c.contract_nmbr='42-3006' and h.ns_hos_id in (2612784)) or
  (c.contract_nmbr='42-0176' and h.ns_hos_id in
(2614410,2614412,2614575,2609022,2601510,2604634,2604708,2605367,2607359,2607363,26
07365,2605007,2606543,3655632,3655729,3655660,2605285,3638385,2609149,2609294,26092
94,2609301,2611813,2611813,2610952,2611095,2614674,2614708,3635731,2612850,3655725)
) or
  (c.contract_nmbr='42-0332' and h.ns_hos_id in
(2604154,2604274,2604274,2604274,2605652,2615508,2610647)) or
  (c.contract_nmbr='42-0403' and h.ns_hos_id in
(2612877,2607629,3669158,2606863,2608287,2608289,2611520,2612388)) or
  (c.contract_nmbr='42-1058' and h.ns_hos_id in (2612574)) or
  (c.contract_nmbr='42-0404' and h.ns_hos_id in (3668877)) or
  (c.contract_nmbr='42-6192' and h.ns_hos_id in (3266136))
)

-- and bl.bl_bill_id = 202401 ------ ПАРАМЕТР ПОИСКА !!!!!!!!!!!!!!!
and bl.bl_bill_id = (select max(bl_bill_id) from bl_bill_periods where
period_status='O')  -- period_status='C' - период закрыт; period_status='O' -
период еще не закрыт

order by c.CONTRACT_NMBR,ck.cntr_contract_key_id,"Полный адрес"
```

Приложение 4. Файл 02_ecolog_counters.py

```
import datetime
import os
import requests
import pendulum
from airflow.decorators import dag, task
from airflow.operators.python import PythonOperator
from airflow.models import Variable
from sqlalchemy import create_engine
import pandas as pd
from sqlalchemy import create_engine
import cx_Oracle

from email.mime.multipart import MIMEMultipart
from email.utils import formataddr
from email.mime.base import MIMEBase
from email.mime.text import MIMEText
from email import encoders
import smtplib

receivers = ['mordanovda@vdkanal.ru'
, 'gaponenkoaa@vdkanal.ru'
, 'karavaevaev@vdkanal.ru'
]


default_args = {
      'owner': 'MordanovDA',
      'depends_on_past': False,
      'start_date': pendulum.datetime(year=2023, month=11,
day=24).in_timezone('Europe/Moscow'),
      'email': ['mordanov.da@rkc43.ru'],
      'email_on_failure': True,
      'email_on_retry': False,
      'retries': 0,
      'retry_delay': datetime.timedelta(minutes=5)
}

@dag(
      dag_id="02.Ecolog_Counters",
#     schedule="0 5 10 * *",
      schedule=None,
      start_date=pendulum.datetime(2024, 3, 26, tz="UTC"),
      catchup=False,
      dagrun_timeout=datetime.timedelta(minutes=60),
      description="Выгружаем показания счетчиков и нормативы ливневых стоков для
экологов",
      default_args=default_args,
)

def ecolog_counters():
```

```
      @task(task_id='get_data_from_oracle')
      def get_data_from_oracle(**kwargs):
      ti = kwargs['ti']
      query = ""
      # Загружаем SQL скрипт из файла
      with open(f'{Variable.get("sql_script_path")}{os.sep}02.ecolog_counters.sql',
'r') as file_sql:
            query = file_sql.read()
      oracle_connection_string =
f"oracle+cx_oracle://{Variable.get('ORA_LOGIN')}:{Variable.get('ORA_PASSWD')}" \

f"@{Variable.get('ORA_HOST')}:{Variable.get('ORA_PORT')}/{Variable.get('ORA_DATABAS
E')}"
      engine = create_engine(oracle_connection_string)
      # Выполнение скрипта
      df = pd.read_sql(query, engine)
      tmp_file = f"{Variable.get('temp_path')}{os.sep}Счетчики_для_экологов.xlsx"

      with pd.ExcelWriter(tmp_file, engine='xlsxwriter') as wb:
            df.to_excel(wb, sheet_name='Лист1', index=False)
            # Настраиваем ширину полей в Excel-файле
            sheet = wb.sheets['Лист1']
            sheet.set_column('A:A',19)
            sheet.set_column('B:B',33)
            sheet.set_column('C:C',25.43)
            sheet.set_column('D:D',44)
            sheet.set_column('E:E',24)
            sheet.set_column('F:F',17)
            sheet.set_column('G:H',28)
            sheet.set_column('I:I',20)
            sheet.set_column('J:J',7)
            sheet.set_column('K:K',10)
            sheet.set_column('L:L',22.7)
            sheet.set_column('M:M',19.71)

      # Передача данных между задачами
      ti.xcom_push(key='xlsx_file', value=tmp_file)

      @task(task_id='xlsx_to_email')
      def xlsx_to_email(**kwargs):
      # Получаем данные от предыдущей задачи
      tmp_file =
str(kwargs['ti'].xcom_pull(task_ids=['get_data_from_oracle'],key='xlsx_file')[0])
      oracle_connection_string =
f"oracle+cx_oracle://{Variable.get('ORA_LOGIN')}:{Variable.get('ORA_PASSWD')}" \

f"@{Variable.get('ORA_HOST')}:{Variable.get('ORA_PORT')}/{Variable.get('ORA_DATABAS
E')}"
      engine = create_engine(oracle_connection_string)
      df = pd.read_sql("select bill_name from bl_bill_periods where bl_bill_id=" \
                        "(select max(bl_bill_id) from bl_bill_periods where
period_status='O')", engine)
      period = df.to_string(index=False, header=False)
      # Готовим данные для электронного письма
```

```
      subject = 'Счетчики для экологов за ' + period
      email_text = f'<html>' \
                   f'<h3>' \
                   f'Добрый день!' \
                   f'</h3>' \
                   f'<p>Высылаю информацию о показаниях счетчиков для экологов по
открытому периоду: {period}' \
                   f'</p>' \
                   f'<p></p>' \
                   f'<p>--<br>' \
                   f'---<br>' \
                   f'С уважением,' \
                   f'Робот ООО РКЦ' \
                   f'</p>' \
                   f'</html>'
      msg = MIMEMultipart()
      msg['Subject'] = subject
      msg['From'] = formataddr((Variable.get('email_from_name'),
Variable.get('email_from_address')))
      msg['To'] = ", ".join(receivers)

      with open(tmp_file, 'rb') as file:  # вкладываем вложение
            part = MIMEBase('application', 'octet-stream')
            part.set_payload(file.read())
            encoders.encode_base64(part)
            part.add_header('Content-disposition', 'attachment', filename=('utf-8',
'', os.path.basename(tmp_file)))
            msg.attach(part)
      msg.attach(MIMEText(email_text.encode('utf-8'), 'html', 'UTF-8'))
      # Отправка письма
      with smtplib.SMTP(Variable.get('email_server'),
Variable.get('email_server_port')) as server:
            server.sendmail(Variable.get('email_from_address'), receivers,
msg.as_string())
      # Чистим за собой ненужные данные
      os.remove(tmp_file)

      get_data_from_oracle() >> xlsx_to_email()

dag = ecolog_counters()
```

Приложение 5. Файл 01.monthly_stats.sql

```
-- Ежемесячная статистика с учетом нулевого тарифа

select 'Отчет-статистика' as "Параметр", (select bill_name from bl_bill_periods
where bl_bill_id=(select max(bl_bill_id) from bl_bill_periods where
period_status='O')) as "Значение" from dual
union all

select '''====== Общая статистика ======' as "1", '' as "2" from dual
union all
select 'Открытых ЛС всего',
to_char(count(*))
from US_FLAT_ACCOUNT_ver
where is_active=1
and us_baseclose_doc_id is null
and (select bill_date from bl_bill_periods where bl_bill_id=(select max(bl_bill_id)
from bl_bill_periods where period_status='O')) between date_begin and date_end

union all

select 'Действующих договоров всего',
to_char(count(*))
from CN_CONTRACT_KEY ck
join cn_contract c on c.cntr_contract_key_id = ck.cntr_contract_key_id
join US_COUNTERAGENT q on q.US_COUNTERAGENT_ID = ck.US_COUNTERAGENT_ID
join ns_user_type ut on ut.ns_user_typ_id = q.ns_user_typ_id

where c.IS_ACTIVE = 1
and c.cntr_status_id=5
and (select bill_date from bl_bill_periods where bl_bill_id=(select max(bl_bill_id)
from bl_bill_periods where period_status='O')) between c.date_begin and c.date_end
and ut.name <> 'Физическое лицо'

union all
select '''====== Переданные показания счетчиков через ЛК ======' as "1", '' as "2"
from dual
union all
select 'ФЛ (кол-во ЛС) (Всего, через все каналы)', to_char(count(num_account))
num_accounts
from (
  select distinct
  bp.bl_bill_id,
  fa.NUM_ACCOUNT

from US_FLAT_ACCOUNT_ver fa
  join bl_bill_periods bp on bp.bill_date between fa.date_begin and fa.date_end
  left join cn_contract_key ck on fa.us_flat_account_kid = ck.us_account_id
      left join cn_service_key sk on ck.cntr_contract_key_id =
sk.cntr_contract_key_id
  join cd_const_service_bunch b on sk.cntr_service_key_id = b.us_cntr_svc_id and
b.is_active = 1 and bp.bill_date between b.date_begin and b.date_end
  join (
```

```
  select ci1.cd_cntr_kid, ci1.bl_bill_id,
ci1.date_ind_take,ci1.cd_indic_src_id,ci1.value
  from cd_indication ci1
  join (select bl_bill_id,cd_cntr_kid,max(date_ind_take) as
date_ind_take,max(value) as value from cd_indication group by
bl_bill_id,cd_cntr_kid) a14 on ci1.cd_cntr_kid=a14.cd_cntr_kid and
ci1.bl_bill_id=a14.bl_bill_id and a14.date_ind_take=ci1.date_ind_take and
a14.value=ci1.value
  ) ci on b.cd_cntr_kid=ci.cd_cntr_kid and ci.bl_bill_id=bp.bl_bill_id

  join cd_indication_source src on src.cd_indic_src_id = ci.cd_indic_src_id

  where fa.IS_ACTIVE = '1'
      and fa.US_BASECLOSE_DOC_ID is NULL
      and src.name in ('Личный кабинет РКЦ','Заведено через ЛК','Телеграмм-бот
РКЦ','Голосовой бот РКЦ')
)
where bl_bill_id in (select bl_bill_id from bl_bill_periods where
bl_bill_id=(select max(bl_bill_id) from bl_bill_periods where period_status='0'))

union all

select 'ФЛ, '||name,to_char(count(NUM_ACCOUNT)) from (
  select distinct
  src.name,
  fa.NUM_ACCOUNT

from US_FLAT_ACCOUNT_ver fa
  join bl_bill_periods bp on bp.bill_date between fa.date_begin and fa.date_end
  left join cn_contract_key ck on fa.us_flat_account_kid = ck.us_account_id
      left join cn_service_key sk on ck.cntr_contract_key_id =
sk.cntr_contract_key_id
  join cd_const_service_bunch b on sk.cntr_service_key_id = b.us_cntr_svc_id and
b.is_active = 1 and bp.bill_date between b.date_begin and b.date_end
  join (
  select ci1.cd_cntr_kid, ci1.bl_bill_id,
ci1.date_ind_take,ci1.cd_indic_src_id,ci1.value
  from cd_indication ci1
  join (select bl_bill_id,cd_cntr_kid,max(date_ind_take) as
date_ind_take,max(value) as value from cd_indication group by
bl_bill_id,cd_cntr_kid) a14 on ci1.cd_cntr_kid=a14.cd_cntr_kid and
ci1.bl_bill_id=a14.bl_bill_id and a14.date_ind_take=ci1.date_ind_take and
a14.value=ci1.value
  ) ci on b.cd_cntr_kid=ci.cd_cntr_kid and ci.bl_bill_id=bp.bl_bill_id

  join cd_indication_source src on src.cd_indic_src_id = ci.cd_indic_src_id

  where fa.IS_ACTIVE = '1'
      and fa.US_BASECLOSE_DOC_ID is NULL
      and src.name in ('Личный кабинет РКЦ','Заведено через ЛК','Телеграмм-бот
РКЦ','Голосовой бот РКЦ')
      and bp.bl_bill_id in (select bl_bill_id from bl_bill_periods where
bl_bill_id=(select max(bl_bill_id) from bl_bill_periods where period_status='0'))
) group by name
```

```
union all

select 'ЮЛ (кол-во договоров)',
to_char(count(CONTRACT_NMBR))
from (
  select distinct
  bp.bl_bill_id
  ,c.CONTRACT_NMBR
  from cn_contract_key ck
  join cn_contract c on c.cntr_contract_key_id = ck.cntr_contract_key_id
  join bl_bill_periods bp on bp.bill_date between c.date_begin and c.date_end
  join Cn_realty_bunch_key cc on ck.cntr_contract_key_id=cc.cntr_contract_key_id
  join TU_REALTY_KEY rk on cc.TU_RLTY_KID=rk.tu_rlty_kid
  join TU_REALTY tr on rk.tu_rlty_kid = tr.tu_rlty_kid and tr.is_active=1 and
bp.bill_date between tr.date_begin and tr.date_end
  join cn_service_key sk on c.cntr_contract_key_id = sk.cntr_contract_key_id and
sk.tu_rlty_kid=cc.tu_rlty_kid
  join cd_const_service_bunch b on sk.cntr_service_key_id = b.us_cntr_svc_id and
b.is_active = 1 and bp.bill_date between b.date_begin and b.date_end
  join (
  select ci1.cd_cntr_kid, ci1.bl_bill_id,
ci1.date_ind_take,ci1.cd_indic_src_id,ci1.value
  from cd_indication ci1
  join (select bl_bill_id,cd_cntr_kid,max(date_ind_take) as
date_ind_take,max(value) as value from cd_indication group by
bl_bill_id,cd_cntr_kid) a14 on ci1.cd_cntr_kid=a14.cd_cntr_kid and
ci1.bl_bill_id=a14.bl_bill_id and a14.date_ind_take=ci1.date_ind_take and
a14.value=ci1.value
  ) ci on b.cd_cntr_kid=ci.cd_cntr_kid and ci.bl_bill_id=bp.bl_bill_id
  join cd_indication_source src on src.cd_indic_src_id = ci.cd_indic_src_id
  where ck.us_account_id is null
      and c.is_active= 1
      and c.cntr_status_id = 5
      and src.name in ('Личный кабинет РКЦ')
)
where bl_bill_id in (select bl_bill_id from bl_bill_periods where
bl_bill_id=(select max(bl_bill_id) from bl_bill_periods where period_status='O'))

union all
select '''====== Статистика (Дома) ======' as "1", '' as "2" from dual
union all

select 'Количество МКД', to_char(count(*)) from (
  select distinct na.NS_HOS_ID
  from US_FLAT_ACCOUNT_ver fa
      join TU_REALTY_KEY rk on rk.TU_RLTY_KID = fa.TU_RLTY_KID
      join tu_realty_species_ref rk_ref on
rk.tu_rlty_spcs_id=rk_ref.tu_rlty_spcs_id
      join NS_ADDRESS na on na.NS_ADR_ID = rk.NS_ADR_ID

  where fa.IS_ACTIVE = '1'
```

```
      and (select bill_date from bl_bill_periods where bl_bill_id=(select
max(bl_bill_id) from bl_bill_periods where period_status='O')) between
fa.DATE_BEGIN and fa.DATE_END
      and fa.US_BASECLOSE_DOC_ID is NULL
  --  and fa.num_account in ('3081700', '3340983')
      and rk_ref.name in ('Квартира в МКД', 'Комната', 'Комната в общежитии',
'МКД')
  )

union all

select 'Количество МКД со счетчиками ХВС', to_char(count(*)) from (
  select distinct na.NS_HOS_ID

  from US_FLAT_ACCOUNT_ver fa
      join TU_REALTY_KEY rk on rk.TU_RLTY_KID = fa.TU_RLTY_KID
      join tu_realty_species_ref rk_ref on
rk.tu_rlty_spcs_id=rk_ref.tu_rlty_spcs_id
      join NS_ADDRESS na on na.NS_ADR_ID = rk.NS_ADR_ID
      join cn_contract_key ck on fa.us_flat_account_kid = ck.us_account_id
      join cn_service_key sk on ck.cntr_contract_key_id = sk.cntr_contract_key_id
      join ns_svc_ref r on r.ns_svc_id = sk.ns_svc_id
      join cd_const_service_bunch b on sk.cntr_service_key_id = b.us_cntr_svc_id
and b.is_active = 1 and (select bill_date from bl_bill_periods where
bl_bill_id=(select max(bl_bill_id) from bl_bill_periods where period_status='O'))
between b.date_begin and b.date_end
  where fa.IS_ACTIVE = '1'
      and (select bill_date from bl_bill_periods where bl_bill_id=(select
max(bl_bill_id) from bl_bill_periods where period_status='O')) between
fa.DATE_BEGIN and fa.DATE_END
      and fa.US_BASECLOSE_DOC_ID is NULL
      and rk_ref.name in ('Квартира в МКД', 'Комната', 'Комната в общежитии',
'МКД')
      and (r.name like 'Водоснабжение%')
)

union all

select 'Количество МКД со счетчиками ГВС', to_char(count(*)) from (
  select distinct na.NS_HOS_ID

  from US_FLAT_ACCOUNT_ver fa
      join TU_REALTY_KEY rk on rk.TU_RLTY_KID = fa.TU_RLTY_KID
      join tu_realty_species_ref rk_ref on
rk.tu_rlty_spcs_id=rk_ref.tu_rlty_spcs_id
      join NS_ADDRESS na on na.NS_ADR_ID = rk.NS_ADR_ID

      join cn_contract_key ck on fa.us_flat_account_kid = ck.us_account_id
      join cn_service_key sk on ck.cntr_contract_key_id = sk.cntr_contract_key_id
      join ns_svc_ref r on r.ns_svc_id = sk.ns_svc_id

      join cd_const_service_bunch b on sk.cntr_service_key_id = b.us_cntr_svc_id
and b.is_active = 1 and (select bill_date from bl_bill_periods where
```

```
bl_bill_id=(select max(bl_bill_id) from bl_bill_periods where period_status='O'))
between b.date_begin and b.date_end


  where fa.IS_ACTIVE = '1'
      and (select bill_date from bl_bill_periods where bl_bill_id=(select
max(bl_bill_id) from bl_bill_periods where period_status='O')) between
fa.DATE_BEGIN and fa.DATE_END
      and fa.US_BASECLOSE_DOC_ID is NULL
      and rk_ref.name in ('Квартира в МКД', 'Комната', 'Комната в общежитии',
'МКД')
      and (r.name like 'ГВС%' or r.name like 'Горячее водоснабжение%' or r.name
like 'Холодная вода для ГВС%' or r.name like 'Холодная вода для приготовления
горячей%')

)

union all

select 'Количество частных домов', to_char(count(*)) from (
  select distinct na.NS_HOS_ID
  from US_FLAT_ACCOUNT_ver fa
      join TU_REALTY_KEY rk on rk.TU_RLTY_KID = fa.TU_RLTY_KID
      join tu_realty_species_ref rk_ref on
rk.tu_rlty_spcs_id=rk_ref.tu_rlty_spcs_id
      join NS_ADDRESS na on na.NS_ADR_ID = rk.NS_ADR_ID

  where fa.IS_ACTIVE = '1'
      and (select bill_date from bl_bill_periods where bl_bill_id=(select
max(bl_bill_id) from bl_bill_periods where period_status='O')) between
fa.DATE_BEGIN and fa.DATE_END
      and fa.US_BASECLOSE_DOC_ID is NULL
      and rk_ref.name in ('Частный дом', 'Квартира в ЧД')
  )

union all

select 'Количество частных домов со счетчиками ХВС', to_char(count(*)) from (
  select distinct na.NS_HOS_ID

  from US_FLAT_ACCOUNT_ver fa
      join TU_REALTY_KEY rk on rk.TU_RLTY_KID = fa.TU_RLTY_KID
      join tu_realty_species_ref rk_ref on
rk.tu_rlty_spcs_id=rk_ref.tu_rlty_spcs_id
      join NS_ADDRESS na on na.NS_ADR_ID = rk.NS_ADR_ID
      join cn_contract_key ck on fa.us_flat_account_kid = ck.us_account_id
      join cn_service_key sk on ck.cntr_contract_key_id = sk.cntr_contract_key_id
      join ns_svc_ref r on r.ns_svc_id = sk.ns_svc_id
      join cd_const_service_bunch b on sk.cntr_service_key_id = b.us_cntr_svc_id
and b.is_active = 1 and (select bill_date from bl_bill_periods where
bl_bill_id=(select max(bl_bill_id) from bl_bill_periods where period_status='O'))
between b.date_begin and b.date_end
  where fa.IS_ACTIVE = '1'
```

```
      and (select bill_date from bl_bill_periods where bl_bill_id=(select
max(bl_bill_id) from bl_bill_periods where period_status='O')) between
fa.DATE_BEGIN and fa.DATE_END
      and fa.US_BASECLOSE_DOC_ID is NULL
      and rk_ref.name in ('Частный дом', 'Квартира в ЧД')
      and (r.name like 'Водоснабжение%')
)

union all

select 'Количество частных домов со счетчиками ГВС', to_char(count(*)) from (
  select distinct na.NS_HOS_ID

  from US_FLAT_ACCOUNT_ver fa
      join TU_REALTY_KEY rk on rk.TU_RLTY_KID = fa.TU_RLTY_KID
      join tu_realty_species_ref rk_ref on
rk.tu_rlty_spcs_id=rk_ref.tu_rlty_spcs_id
      join NS_ADDRESS na on na.NS_ADR_ID = rk.NS_ADR_ID

      join cn_contract_key ck on fa.us_flat_account_kid = ck.us_account_id
      join cn_service_key sk on ck.cntr_contract_key_id = sk.cntr_contract_key_id
      join ns_svc_ref r on r.ns_svc_id = sk.ns_svc_id

      join cd_const_service_bunch b on sk.cntr_service_key_id = b.us_cntr_svc_id
and b.is_active = 1 and (select bill_date from bl_bill_periods where
bl_bill_id=(select max(bl_bill_id) from bl_bill_periods where period_status='O'))
between b.date_begin and b.date_end


  where fa.IS_ACTIVE = '1'
      and (select bill_date from bl_bill_periods where bl_bill_id=(select
max(bl_bill_id) from bl_bill_periods where period_status='O')) between
fa.DATE_BEGIN and fa.DATE_END
      and fa.US_BASECLOSE_DOC_ID is NULL
      and rk_ref.name in ('Частный дом', 'Квартира в ЧД')
      and (r.name like 'ГВС%' or r.name like 'Горячее водоснабжение%' or r.name
like 'Холодная вода для ГВС%' or r.name like 'Холодная вода для приготовления
горячей%')

)

union all

select '''====== Статистика (ИПУ) ======' as "1", '' as "2" from dual
union all
select '''--=== ХВС ======-' as "1", '' as "2" from dual
union all

select 'Количество счетчиков ХВС', to_char(count(*)) from (
  select
      fa.NUM_ACCOUNT

      from US_FLAT_ACCOUNT_ver fa
      join cn_contract_key ck on fa.us_flat_account_kid = ck.us_account_id
```

```
      join cn_service_key sk on ck.cntr_contract_key_id = sk.cntr_contract_key_id
      join ns_svc_ref r on r.ns_svc_id = sk.ns_svc_id
      join cd_const_service_bunch b on sk.cntr_service_key_id = b.us_cntr_svc_id
and b.is_active = 1 and (select bill_date from bl_bill_periods where
bl_bill_id=(select max(bl_bill_id) from bl_bill_periods where period_status='O'))
between b.date_begin and b.date_end
      inner join cd_counter cc on cc.cd_cntr_kid = b.cd_cntr_kid
      join cn_service s on ck.cntr_contract_key_id = s.cntr_contract_key_id and
s.is_active=1 and (select bill_date from bl_bill_periods where bl_bill_id=(select
max(bl_bill_id) from bl_bill_periods where period_status='O')) between s.date_begin
and s.date_end and s.cntr_service_key_id=sk.cntr_service_key_id
      join ns_tariff_plan tp on s.ns_tariff_plan_id = tp.ns_tariff_plan_id
      where fa.IS_ACTIVE = '1'
      and (select bill_date from bl_bill_periods where bl_bill_id=(select
max(bl_bill_id) from bl_bill_periods where period_status='O')) between
fa.DATE_BEGIN and fa.DATE_END
      and fa.US_BASECLOSE_DOC_ID is NULL
      and cc.is_active=1
      and (r.Name like 'Водоснабжение%')  -- только счетчики ХВС
      and tp.name not like '%улев%'
)

union all
select 'Количество счетчиков ХВС в МКД', to_char(count(*)) from (
  select
      fa.NUM_ACCOUNT
      from US_FLAT_ACCOUNT_ver fa
      join TU_REALTY_KEY rk on rk.TU_RLTY_KID = fa.TU_RLTY_KID
      join tu_realty_species_ref rk_ref on
rk.tu_rlty_spcs_id=rk_ref.tu_rlty_spcs_id
      join cn_contract_key ck on fa.us_flat_account_kid = ck.us_account_id
      join cn_service_key sk on ck.cntr_contract_key_id = sk.cntr_contract_key_id
      join ns_svc_ref r on r.ns_svc_id = sk.ns_svc_id
      join cd_const_service_bunch b on sk.cntr_service_key_id = b.us_cntr_svc_id
and b.is_active = 1 and (select bill_date from bl_bill_periods where
bl_bill_id=(select max(bl_bill_id) from bl_bill_periods where period_status='O'))
between b.date_begin and b.date_end
      inner join cd_counter cc on cc.cd_cntr_kid = b.cd_cntr_kid
      join cn_service s on ck.cntr_contract_key_id = s.cntr_contract_key_id and
s.is_active=1 and (select bill_date from bl_bill_periods where bl_bill_id=(select
max(bl_bill_id) from bl_bill_periods where period_status='O')) between s.date_begin
and s.date_end and s.cntr_service_key_id=sk.cntr_service_key_id
      join ns_tariff_plan tp on s.ns_tariff_plan_id = tp.ns_tariff_plan_id
      where fa.IS_ACTIVE = '1'
      and (select bill_date from bl_bill_periods where bl_bill_id=(select
max(bl_bill_id) from bl_bill_periods where period_status='O')) between
fa.DATE_BEGIN and fa.DATE_END
      and fa.US_BASECLOSE_DOC_ID is NULL
      and cc.is_active=1
      and (r.Name like 'Водоснабжение%')  -- только счетчики ХВС
      and rk_ref.name in ('Квартира в МКД', 'Комната', 'Комната в общежитии',
'МКД')
      and tp.name not like '%улев%'
)
```

```
union all
select 'Количество ЛС со счетчиками ХВС', to_char(count(*)) from (
  select distinct
      fa.NUM_ACCOUNT, r.name
      from US_FLAT_ACCOUNT_ver fa
      join cn_contract_key ck on fa.us_flat_account_kid = ck.us_account_id
      join cn_service_key sk on ck.cntr_contract_key_id = sk.cntr_contract_key_id
      join ns_svc_ref r on r.ns_svc_id = sk.ns_svc_id
      join cn_service s on ck.cntr_contract_key_id = s.cntr_contract_key_id and
s.is_active=1 and (select bill_date from bl_bill_periods where bl_bill_id=(select
max(bl_bill_id) from bl_bill_periods where period_status='O')) between s.date_begin
and s.date_end and s.cntr_service_key_id=sk.cntr_service_key_id
      join cd_const_service_bunch b on sk.cntr_service_key_id = b.us_cntr_svc_id
and b.is_active = 1 and (select bill_date from bl_bill_periods where
bl_bill_id=(select max(bl_bill_id) from bl_bill_periods where period_status='O'))
between b.date_begin and b.date_end
      inner join cd_counter cc on cc.cd_cntr_kid = b.cd_cntr_kid
      join ns_tariff_plan tp on s.ns_tariff_plan_id = tp.ns_tariff_plan_id
      where fa.IS_ACTIVE = '1'
      and (select bill_date from bl_bill_periods where bl_bill_id=(select
max(bl_bill_id) from bl_bill_periods where period_status='O')) between
fa.DATE_BEGIN and fa.DATE_END
      and fa.US_BASECLOSE_DOC_ID is NULL
      and cc.is_active=1
      and (r.Name like 'Водоснабжение%')  -- только счетчики ХВС
      and tp.name not like '%улев%'
)
union all
select 'Количество ЛС услугой ХВС', to_char(count(*)) from (
  select distinct
      fa.NUM_ACCOUNT
      from US_FLAT_ACCOUNT_ver fa
      join cn_contract_key ck on fa.us_flat_account_kid = ck.us_account_id
      join cn_service_key sk on ck.cntr_contract_key_id = sk.cntr_contract_key_id
      join ns_svc_ref r on r.ns_svc_id = sk.ns_svc_id
      join cn_service s on ck.cntr_contract_key_id = s.cntr_contract_key_id and
s.is_active=1 and (select bill_date from bl_bill_periods where bl_bill_id=(select
max(bl_bill_id) from bl_bill_periods where period_status='O')) between s.date_begin
and s.date_end and s.cntr_service_key_id=sk.cntr_service_key_id
      join ns_tariff_plan tp on s.ns_tariff_plan_id = tp.ns_tariff_plan_id

      where fa.IS_ACTIVE = '1'
      and (select bill_date from bl_bill_periods where bl_bill_id=(select
max(bl_bill_id) from bl_bill_periods where period_status='O')) between
fa.DATE_BEGIN and fa.DATE_END
      and fa.US_BASECLOSE_DOC_ID is NULL
      and (r.Name like 'Водоснабжение%')  -- только ХВС
      and tp.name not like '%улев%'
)
union all
select '''------ ГВС ======' as "1", '' as "2" from dual
union all
```

```
select 'Количество счетчиков ГВС', to_char(count(*)) from (
  select
      fa.NUM_ACCOUNT
      from US_FLAT_ACCOUNT_ver fa
      join cn_contract_key ck on fa.us_flat_account_kid = ck.us_account_id
      join cn_service_key sk on ck.cntr_contract_key_id = sk.cntr_contract_key_id
      join ns_svc_ref r on r.ns_svc_id = sk.ns_svc_id
      join cd_const_service_bunch b on sk.cntr_service_key_id = b.us_cntr_svc_id
and b.is_active = 1 and (select bill_date from bl_bill_periods where
bl_bill_id=(select max(bl_bill_id) from bl_bill_periods where period_status='O'))
between b.date_begin and b.date_end
      inner join cd_counter cc on cc.cd_cntr_kid = b.cd_cntr_kid
      join cn_service s on ck.cntr_contract_key_id = s.cntr_contract_key_id and
s.is_active=1 and (select bill_date from bl_bill_periods where bl_bill_id=(select
max(bl_bill_id) from bl_bill_periods where period_status='O')) between s.date_begin
and s.date_end and s.cntr_service_key_id=sk.cntr_service_key_id
      join ns_tariff_plan tp on s.ns_tariff_plan_id = tp.ns_tariff_plan_id
      where fa.IS_ACTIVE = '1'
      and (select bill_date from bl_bill_periods where bl_bill_id=(select
max(bl_bill_id) from bl_bill_periods where period_status='O')) between
fa.DATE_BEGIN and fa.DATE_END
      and fa.US_BASECLOSE_DOC_ID is NULL
      and cc.is_active=1
      and (r.Name like '%ГВС%' or r.Name like '%Горяч%')  -- только счетчики ГВС
      and tp.name not like '%улев%'
)
union all
select 'Количество счетчиков ГВС в МКД', to_char(count(*)) from (
  select
      fa.NUM_ACCOUNT
      from US_FLAT_ACCOUNT_ver fa
      join TU_REALTY_KEY rk on rk.TU_RLTY_KID = fa.TU_RLTY_KID
      join tu_realty_species_ref rk_ref on
rk.tu_rlty_spcs_id=rk_ref.tu_rlty_spcs_id
      join cn_contract_key ck on fa.us_flat_account_kid = ck.us_account_id
      join cn_service_key sk on ck.cntr_contract_key_id = sk.cntr_contract_key_id
      join ns_svc_ref r on r.ns_svc_id = sk.ns_svc_id
      join cd_const_service_bunch b on sk.cntr_service_key_id = b.us_cntr_svc_id
and b.is_active = 1 and (select bill_date from bl_bill_periods where
bl_bill_id=(select max(bl_bill_id) from bl_bill_periods where period_status='O'))
between b.date_begin and b.date_end
      inner join cd_counter cc on cc.cd_cntr_kid = b.cd_cntr_kid
      join cn_service s on ck.cntr_contract_key_id = s.cntr_contract_key_id and
s.is_active=1 and (select bill_date from bl_bill_periods where bl_bill_id=(select
max(bl_bill_id) from bl_bill_periods where period_status='O')) between s.date_begin
and s.date_end and s.cntr_service_key_id=sk.cntr_service_key_id
      join ns_tariff_plan tp on s.ns_tariff_plan_id = tp.ns_tariff_plan_id
      where fa.IS_ACTIVE = '1'
      and (select bill_date from bl_bill_periods where bl_bill_id=(select
max(bl_bill_id) from bl_bill_periods where period_status='O')) between
fa.DATE_BEGIN and fa.DATE_END
      and fa.US_BASECLOSE_DOC_ID is NULL
      and cc.is_active=1
```

```
      and (r.Name like '%ГВС%' or r.Name like '%Горяч%')  -- только счетчики ГВС
      and rk_ref.name in ('Квартира в МКД', 'Комната', 'Комната в общежитии',
'МКД')
      and tp.name not like '%улев%'
)

union all
select 'Количество ЛС со счетчиками ГВС', to_char(count(*)) from (
  select distinct
      fa.NUM_ACCOUNT, r.name
      from US_FLAT_ACCOUNT_ver fa
      join cn_contract_key ck on fa.us_flat_account_kid = ck.us_account_id
      join cn_service_key sk on ck.cntr_contract_key_id = sk.cntr_contract_key_id
      join ns_svc_ref r on r.ns_svc_id = sk.ns_svc_id
      join cd_const_service_bunch b on sk.cntr_service_key_id = b.us_cntr_svc_id
and b.is_active = 1 and (select bill_date from bl_bill_periods where
bl_bill_id=(select max(bl_bill_id) from bl_bill_periods where period_status='O'))
between b.date_begin and b.date_end
      join cn_service s on ck.cntr_contract_key_id = s.cntr_contract_key_id and
s.is_active=1 and (select bill_date from bl_bill_periods where bl_bill_id=(select
max(bl_bill_id) from bl_bill_periods where period_status='O')) between s.date_begin
and s.date_end and s.cntr_service_key_id=sk.cntr_service_key_id
      join ns_tariff_plan tp on s.ns_tariff_plan_id = tp.ns_tariff_plan_id
      inner join cd_counter cc on cc.cd_cntr_kid = b.cd_cntr_kid
      where fa.IS_ACTIVE = '1'
      and (select bill_date from bl_bill_periods where bl_bill_id=(select
max(bl_bill_id) from bl_bill_periods where period_status='O')) between
fa.DATE_BEGIN and fa.DATE_END
      and fa.US_BASECLOSE_DOC_ID is NULL
      and cc.is_active=1
      and (r.Name like '%ГВС%' or r.Name like '%Горяч%')  -- только счетчики ГВС
      and tp.name not like '%улев%'
)

union all
select 'Количество ЛС услугой ГВС', to_char(count(*)) from (
  select distinct
      fa.NUM_ACCOUNT
      from US_FLAT_ACCOUNT_ver fa
      join cn_contract_key ck on fa.us_flat_account_kid = ck.us_account_id
      join cn_service_key sk on ck.cntr_contract_key_id = sk.cntr_contract_key_id
      join ns_svc_ref r on r.ns_svc_id = sk.ns_svc_id
      join cn_service s on ck.cntr_contract_key_id = s.cntr_contract_key_id and
s.is_active=1 and (select bill_date from bl_bill_periods where bl_bill_id=(select
max(bl_bill_id) from bl_bill_periods where period_status='O')) between s.date_begin
and s.date_end and s.cntr_service_key_id=sk.cntr_service_key_id
      join ns_tariff_plan tp on s.ns_tariff_plan_id = tp.ns_tariff_plan_id

      where fa.IS_ACTIVE = '1'
      and (select bill_date from bl_bill_periods where bl_bill_id=(select
max(bl_bill_id) from bl_bill_periods where period_status='O')) between
fa.DATE_BEGIN and fa.DATE_END
      and fa.US_BASECLOSE_DOC_ID is NULL
      and (r.Name like '%ГВС%' or r.Name like '%Горяч%')  -- только ГВС
```

```
      and tp.name not like '%улев%'
)
```

Приложение 6. Файл 01_statistic.py

```
import datetime
import os
import requests
import pendulum
from airflow.decorators import dag, task
from airflow.operators.empty import EmptyOperator
from airflow.models import Variable
from airflow.utils.trigger_rule import TriggerRule
from sqlalchemy import create_engine, MetaData, insert, Table, Column, Integer,
String
import pandas as pd
from sqlalchemy import create_engine
import cx_Oracle
import xlsxwriter

from email.mime.multipart import MIMEMultipart
from email.utils import formataddr
from email.mime.base import MIMEBase
from email.mime.text import MIMEText
from email import encoders
import smtplib

DAG_ID = "01.Monthly_Statistic"

receivers = ['mordanov.da@rkc43.ru'
, 'dudnikov.an@rkc43.ru'
, 'mikrukova.ts@rkc43.ru'
, 'hodyrevrv@vdkanal.ru'
]


default_args = {
      'owner': 'MordanovDA',
      'depends_on_past': False,
      'start_date': pendulum.datetime(year=2024, month=4,
day=25).in_timezone('Europe/Moscow'),
      'email': ['mordanov.da@rkc43.ru'],
      'email_on_failure': True,
      'email_on_retry': False,
      'retries': 0,
      'retry_delay': datetime.timedelta(minutes=5)
}



@dag(
      dag_id=DAG_ID,
      schedule="0 2 4-15 * *",
#     schedule=None,
      start_date=pendulum.datetime(2024, 3, 26, tz="UTC"),
      catchup=False,
```

```
      dagrun_timeout=datetime.timedelta(minutes=60),
      description="Собираем ежемесячную статистику",
      default_args=default_args,
)

def Monthly_Statistic():
      operator_start = EmptyOperator(task_id='start')

      @task.branch(task_id='check_period')
      def check_period():
      # Проверка, когда запускался DAG (сравниваемм обработанные периоды)
      oracle_connection_string =
f"oracle+cx_oracle://{Variable.get('ORA_LOGIN')}:{Variable.get('ORA_PASSWD')}" \

f"@{Variable.get('ORA_HOST')}:{Variable.get('ORA_PORT')}/{Variable.get('ORA_DATABAS
E')}"
      engine_ora = create_engine(oracle_connection_string)
      df_ora = pd.read_sql("select bl_bill_id from bl_bill_periods where
bl_bill_id=" \
                           "(select max(bl_bill_id) from bl_bill_periods where
period_status='O')", engine_ora)
      period_ora = int(df_ora.to_string(index=False, header=False))

      pg_connection_string =
f"postgresql://{Variable.get('PG_LOGIN')}:{Variable.get('PG_PASSWD')}" \

f"@{Variable.get('PG_HOST')}:{Variable.get('PG_PORT')}/{Variable.get('PG_DB')}"
      engine_pg = create_engine(pg_connection_string)
      df_pg = pd.read_sql(f"select coalesce(max(exec_period),0) as exec_period from
dag_run where dag_name='{DAG_ID}'", engine_pg)
      period_pg = int(df_pg.to_string(index=False, header=False))

      if period_ora != period_pg:
            # в этом периоде обработки еще не было
            metadata_obj = MetaData()
            metadata_obj.reflect(engine_pg)
            with engine_pg.connect() as connection:
            metadata_obj.reflect(connection)
            t = Table('dag_run', metadata_obj,
                  Column('id', Integer),
                  Column('dag_name', String),
                  Column('exec_period', Integer),
                  extend_existing=True
            )
            with connection.begin():
                  # фиксируем событие обработки
                  result = connection.execute(insert(t), {'dag_name':DAG_ID,
'exec_period':period_ora})
            return 'check_period_true'
      else:
            return 'check_period_false'

      check_period = check_period.override(task_id='check_period')()
```

```
      check_period_true = EmptyOperator(task_id='check_period_true')   # фейковые
операторы - для "маршрута" ветвления
      check_period_false = EmptyOperator(task_id='check_period_false')

      @task(task_id='get_data_from_oracle')
      def get_data_from_oracle(**kwargs):
      # достаем информацию из Oracle
      ti = kwargs['ti']
      query = ""
      # Скрипт - достаем из общей папки
      with open(f'{Variable.get("sql_script_path")}{os.sep}01.monthly_stats.sql',
'r') as file_sql:
            query = file_sql.read()
      oracle_connection_string =
f"oracle+cx_oracle://{Variable.get('ORA_LOGIN')}:{Variable.get('ORA_PASSWD')}" \

f"@{Variable.get('ORA_HOST')}:{Variable.get('ORA_PORT')}/{Variable.get('ORA_DATABAS
E')}"
      engine = create_engine(oracle_connection_string)
      # Выполняем SQL
      df = pd.read_sql(query, engine)
      tmp_file = f"{Variable.get('temp_path')}{os.sep}Статистика.xlsx"

      # Пишем результат в xlsx файл
      with pd.ExcelWriter(tmp_file, engine='xlsxwriter') as wb:
            df.to_excel(wb, sheet_name='Лист1', index=False)
            sheet = wb.sheets['Лист1']
            sheet.set_column('A:A',50)  # ширина полей
            sheet.set_column('B:B',17)

      # передаем имя файла следующей задаче
      ti.xcom_push(key='xlsx_file', value=tmp_file)

      @task(task_id='xlsx_to_email')
      def xlsx_to_email(**kwargs):
      # готовим электронное письмо и отправляем его получателям
      tmp_file =
str(kwargs['ti'].xcom_pull(task_ids=['get_data_from_oracle'],key='xlsx_file')[0])
      oracle_connection_string =
f"oracle+cx_oracle://{Variable.get('ORA_LOGIN')}:{Variable.get('ORA_PASSWD')}" \

f"@{Variable.get('ORA_HOST')}:{Variable.get('ORA_PORT')}/{Variable.get('ORA_DATABAS
E')}"
      engine = create_engine(oracle_connection_string)
      df = pd.read_sql("select bill_name from bl_bill_periods where bl_bill_id=" \
                       "(select max(bl_bill_id) from bl_bill_periods where
period_status='C')", engine)
      period = df.to_string(index=False, header=False)
      subject = 'Статистика по биллинговой базе за ' + period
      email_text = f'<html>' \
                  f'<h3>' \
                  f'Добрый день!' \
                  f'</h3>' \
                  f'<p>Высылаю статистику по биллинговой базе за {subject}' \
```

```
                  f'</p>' \
                  f'<p></p>' \
                  f'<p>--<br>' \
                  f'---<br>' \
                  f'С уважением,' \
                  f'Робот ООО РКЦ' \
                  f'</p>' \
                  f'</html>'
      msg = MIMEMultipart()
      msg['Subject'] = subject
      msg['From'] = formataddr((Variable.get('email_from_name'),
Variable.get('email_from_address')))
      msg['To'] = ", ".join(receivers)

      with open(tmp_file, 'rb') as file:  # вкладываем вложение
            part = MIMEBase('application', 'octet-stream')
            part.set_payload(file.read())
            encoders.encode_base64(part)
            part.add_header('Content-disposition', 'attachment', filename=('utf-8',
'', os.path.basename(tmp_file)))
            msg.attach(part)
      msg.attach(MIMEText(email_text.encode('utf-8'), 'html', 'UTF-8'))

      with smtplib.SMTP(Variable.get('email_server'),
Variable.get('email_server_port')) as server:
            server.sendmail(Variable.get('email_from_address'), receivers,
msg.as_string())

      os.remove(tmp_file)

      operator_end = EmptyOperator(
      task_id='end',
      trigger_rule=TriggerRule.NONE_FAILED_MIN_ONE_SUCCESS,
      )

      # "Маршрут" задачи
      operator_start >> check_period
      check_period >> check_period_true >> get_data_from_oracle() >>
xlsx_to_email() >> operator_end
      check_period >> check_period_false >> operator_end

dag = Monthly_Statistic()
```

Приложение 7. Файл 03_pump_search_all_zbases.sql

```
select
  lic as ls, fio,close_ls
  ,Replace(IIF(d1.name='', 'Киров г', 'Киров г ('||d1.name||')') || ', ул. ' ||
                        a1.name || ', д. ' || b1.dom ||
iif(char_length(b1.dom2)>0,b1.dom2,'') ||
                        iif(char_length(b1.korp)>0,' корп. ' || b1.korp,'')
  || IIF(a.kv=0,'',', кв. '||a.kv) || IIF(a.kv2 is null, '',a.kv2)
  ,'Киров г','г. Киров') as address
from nanim as a
  left join dom as b1 on b1.id=a.adres_id
  left join ulicy as a1 on a1.id=b1.ul_id
  left join city as d1 on a1.city_id=d1.id
where data_my=(select max(data_my) from nanim);
```

Приложение 8. Файл 03_pump_search_all_zbases.py

```
import datetime
import os
import requests
import pendulum
from airflow.decorators import dag, task
from airflow.operators.python import PythonOperator
from airflow.operators.empty import EmptyOperator
from airflow.models import Variable
from airflow.utils.trigger_rule import TriggerRule
from sqlalchemy import create_engine, MetaData, text, delete, insert, Table,
Column, Integer, String
from airflow.utils.db import provide_session
from airflow.models import XCom
from airflow import settings
from airflow.models.baseoperator import chain

# Список баз данных, которые требуется обработать
BASES = [15, 20, 27, 29, 30, 43, 56, 71, 73]

# Ограничение на количество одновременных подключений к удаленному серверу
CHANNELS = 4

LOCAL_RECEIVE_TABLE = 'NANIM'
DAG_ID = "03.Pump_search_all_Zbases"

columns_in = ('LS', 'FIO', 'CLOSE_LS', 'ADDRESS')
columns_out = ('BASE', 'LS', 'FIO', 'ADDRESS', 'CLOSE_LS')

default_args = {
      'owner': 'MordanovDA',
      'depends_on_past': False,
      'start_date': pendulum.datetime(year=2024, month=4,
day=18).in_timezone('Europe/Moscow'),
      'email': ['mordanov.da@rkc43.ru'],
      'email_on_failure': True,
      'email_on_retry': False,
      'retries': 0,
      'retry_delay': datetime.timedelta(minutes=5)
}

@dag(
      dag_id=DAG_ID,
      schedule="10 3 * * 0-5",
#     schedule=none,
      start_date=pendulum.datetime(2024, 3, 26, tz="UTC"),
      catchup=False,
      dagrun_timeout=datetime.timedelta(minutes=60),
      description="Загрузка информации для поиска по всем базам Злобина",
      default_args=default_args,
)
```

```
def Pump_all_Zbases():
      tasks = []
# ============================================================
      @task(task_id='start')
      def start_task(**kwargs):
      # Фейковая задача - просто единая точка входа
      print('start')
      tasks.append(start_task())
# ============================================================
      i = 1
      tasks.append([])
      for n in BASES:
      # Динамически генерируем параллельные задачи выгрузки для каждой базы с
учетом количества каналов доступа
      @task(task_id=f'get_data_zbase_{n}')
      def get_data_zbase(i, **kwargs):
            # Выгрузка данных по одной управляющей компании с удаленного сервера
            ti = kwargs['ti']
            query = ""
            with
open(f'{Variable.get("sql_script_path")}{os.sep}03_pump_search_all_zbases.sql',
'r') as file_sql:
            # Читаем SQL-скрипт из файла
            query = file_sql.read()
            fb_remote_connection_string =
f"firebird+fdb://{Variable.get('FB_REMOTE_LOGIN')}:{Variable.get('FB_REMOTE_PASSWOR
D')}" \

f"@{Variable.get('FB_REMOTE_HOST')}:{Variable.get('FB_REMOTE_PORT')}//{Variable.get
('FB_REMOTE_PATH_TO_BASE')}/" \

f"{Variable.get('FB_REMOTE_DBNAME')}{i}.fdb?charset=WIN1251&fb_library_name=/usr/li
b/libfbclient.so"
            textual_sql = text(query)
            engine = create_engine(fb_remote_connection_string)
            metadata_obj = MetaData()
            metadata_obj.reflect(engine)
            with engine.connect() as connection:
                  metadata_obj.reflect(connection)
                  with connection.begin():
                        # Выполняем скрипт
                        data = connection.execute(textual_sql)
                        # помещаем данные в объект XCom для передачи на следующий
этап обработки
                        ti.xcom_push(key=f'zbase_{i}',
value=[dict(zip(columns_in,row)) for row in data])

      if i % CHANNELS == 0:  # Раскидываем задачи паралельно, занимая все возможные
каналы
            tasks[-1].append(get_data_zbase(n))
            if n != BASES[-1]:
            tasks.append([])
      else:
            tasks[-1].append(get_data_zbase(n))
```

```
      i += 1
      while len(tasks[-1]) < CHANNELS:  # Дописываем последний блок параллельных
зачад пустыми операторами
      tasks[-1].append(EmptyOperator(task_id=f'Empty{i}'))   # Параллельные блоки
должны иметь одинаковое
      i += 1                                                 # количество задач
# ============================================================
      @task(task_id='clean_kvk_base', trigger_rule=TriggerRule.ALL_SUCCESS)
      def clean_kvk_base(**kwargs):
      # Очищаем рабочую таблицу, при условии, что все предыдущие задачи завершились
успешно
      fb_local_connection_string =
f"firebird+fdb://{Variable.get('FB_LOCAL_LOGIN')}:{Variable.get('FB_LOCAL_PASSWORD'
)}" \

f"@{Variable.get('FB_LOCAL_HOST')}:{Variable.get('FB_LOCAL_PORT')}//{Variable.get('
FB_LOCAL_PATH_TO_BASE')}/" \

f"{Variable.get('FB_LOCAL_DBNAME')}.fdb?charset=WIN1251&fb_library_name=/usr/lib/li
bfbclient.so"
      engine = create_engine(fb_local_connection_string)
      metadata_obj = MetaData()
      metadata_obj.reflect(engine)

      with engine.connect() as connection:
            metadata_obj.reflect(connection)
            t = Table(LOCAL_RECEIVE_TABLE, metadata_obj)
            # Очистка данных
            with connection.begin():
            result = connection.execute(delete(t))
      tasks.append(clean_kvk_base())
# ============================================================
      tasks.append([])
      i = 1
      # Динамически генерируем параллельные задачи загрузки для каждой базы
      for n in BASES:
      @task(task_id=f'push_data_to_kvk_{n}')
      def push_data_to_kvk(i, **kwargs):
            # Загружаем информацию по одной базе
            fb_local_connection_string =
f"firebird+fdb://{Variable.get('FB_LOCAL_LOGIN')}:{Variable.get('FB_LOCAL_PASSWORD'
)}" \

f"@{Variable.get('FB_LOCAL_HOST')}:{Variable.get('FB_LOCAL_PORT')}//{Variable.get('
FB_LOCAL_PATH_TO_BASE')}/" \

f"{Variable.get('FB_LOCAL_DBNAME')}.fdb?charset=WIN1251&fb_library_name=/usr/lib/li
bfbclient.so"
            engine = create_engine(fb_local_connection_string)
            metadata_obj = MetaData()
            metadata_obj.reflect(engine)
            with engine.connect() as connection:
            metadata_obj.reflect(connection)
            t = Table(LOCAL_RECEIVE_TABLE, metadata_obj,
```

```
                  Column('BASE', Integer),
                  Column('LS', Integer),
                  Column('FIO', String),
                  Column('ADDRESS', String),
                  Column('CLOSE_LS', Integer),
                  extend_existing=True
            )
            data =
kwargs['ti'].xcom_pull(task_ids=[f'get_data_zbase_{i}'],key=f'zbase_{i}')[0]
            # Вставка данных
            with connection.begin():
                  result = connection.execute(insert(t), [dict(row, **{'BASE': i})
for row in data], )
      tasks[-1].append(push_data_to_kvk(n))
# ====================================================================
      @task(task_id='end', trigger_rule=TriggerRule.ALL_SUCCESS)
      @provide_session
      def end_task(session=None, **context):
      # Завершение работы + единая точка выхода
      print('end')
      # Чистим данные, которые были в XCom
      with settings.Session() as session:
            res = session.query(XCom).filter(XCom.dag_id == DAG_ID).delete()
            session.commit()
      tasks.append(end_task())
      # "Маршрут" процесса
#     start_task() >> tasks1 >> clean_kvk_base() >> tasks2 >> end_task()
      chain(*tasks)

dag = Pump_all_Zbases()
```