ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ «ДАТАБАЗИС»

ИНФОРМАЦИЯ ПО УСТАНОВКЕ

Листов 39

Аннотация

Документ содержит информацию о программном обеспечении «ДатаБазис» (далее – Система).

Настоящая инструкция по настройке и развертыванию является составной частью комплекта рабочей документации и содержит описание и подробные инструкции по выполнению основных этапов настройки и развертывания Системы.

Инструкция рассчитана на пользователей, знакомых с основными интернет-технологиями, соответствующей терминологией, и имеющих представление о принципах работы сайтов и мобильных приложений.

Документ разработан в соответствии с требованиями следующих документов:

- ГОСТ Р 59795-2021 «Информационные технологии. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Требования к содержанию документов»;
- ГОСТ 34.201-2020 «Информационные технологии. Комплекс стандартов на автоматизированные системы. Виды, комплектность и обозначение документов при создании автоматизированных систем».

Содержание

Термины и определения

Термин	Определение
Активация DAG	Процесс включения автоматизированного сценария обработки данных в интерфейсе Airflow.
Зеркалирование (Mirror)	Автоматическое копирование данных из одного репозитория в другой для синхронизации.
Инвентарь (Inventory)	Файл (inventory.ini), содержащий список IP-адресов и групп серверов для управления через Ansible.
Ингресс (Ingress)	Правило доступа, открывающее путь к внутренним сервисам кластера из внешней сети (Интернета или ЛВС компании).
Кластер	Группа серверов, работающих совместно как одна виртуальная машина.
Контейнер	Упакованное приложение со всеми зависимостями (библиотеками), которое запускается в изолированной среде.
Манифест	Файл в формате YAML, описывающий параметры сервиса (ресурсы, порты, адреса).
Направленный ациклический граф (DAG)	Цепочка задач в Airflow, выполняемых в строго определенном порядке.
Образ (Image)	«Замороженный» снимок контейнера, из которого создаются работающие программные среды.
Общее программное обеспечение (ОПО)	Комплект сторонних программных средств, необходимых для функционирования Системы.
Оператор (Operator)	Контроллер Kubernetes, управляющий жизненным циклом приложения через кастомные ресурсы.
Под (Pod)	Минимальная единица запуска в Kubernetes, содержащая один или несколько контейнеров.
Пространство имен (Namespace)	Логическое пространство в Kubernetes для изоляции ресурсов различных проектов или сред.
Раннер (Runner)	Агент, выполняющий команды сборки и запуска кода (например, в системе GitFlic).
Распечатывание (Unseal)	Процедура дешифрования главного ключа доступа к данным в HashiCorp Vault после старта сервиса.
Реплика (Replica)	Копия узла в группе репликации базы данных для обеспечения отказоустойчивости.
Репозиторий	Место хранения исходного кода приложений и его версий.
Секреты	Пароли, токены и ключи доступа, требующие защищенного хранения.
Сервис (Service)	Стабильная точка доступа (сетевой адрес) к набору подов в кластере.
Шард (Shard)	Логическая доля (сегмент) данных кластера ClickHouse.
Ansible	Система автоматизации конфигурации и развертывания ПО на серверах.
Apache Airflow	Платформа для оркестрации, планирования и мониторинга процессов обработки данных (ETL).

Термин	Определение
ArgoCD	Инструмент для непрерывной доставки приложений в Kubernetes, реализующий подход GitOps.
Chart (Чарт)	Пакет Helm, содержащий шаблоны манифестов и наборы параметров для установки приложения.
ClickHouse	Высокопроизводительная колоночная СУБД для аналитической обработки больших объемов данных.
ClickHouseInstallation (CHI)	Кастомный ресурс, описывающий конфигурацию кластера ClickHouse (шарды, реплики, параметры хранения).
ConfigMap	Объект Kubernetes для хранения конфигурационных файлов отдельно от кода приложения.
Containerd	Среда выполнения (движок), отвечающая за запуск и управление контейнерами на серверах.
CoreDNS	Внутренняя служба имен кластера Kubernetes, отвечающая за разрешение доменных имен в IP-адреса.
GitFlic	Российская система управления версиями кода на базе Git.
Grafana	Платформа для визуализации метрик, построения интерактивных дашбордов и настройки оповещений.
HashiCorp Vault	Система централизованного управления секретами и защиты конфиденциальных данных.
Helm	Менеджер пакетов для Kubernetes, автоматизирующий установку приложений через чарты.
Keycloak	Система управления идентификацией и доступом (IAM) с поддержкой единого входа (SSO).
Lineage (Происхождение данных)	Граф связей, отображающий путь данных от источника до итоговой таблицы или отчета.
Longhorn	Система распределенного хранения данных, обеспечивающая работу виртуальных дисков в кластере.
On-Premise	Способ размещения инфраструктуры на собственных мощностях организации (в локальном дата-центре).
OpenMetadata	Система управления метаданными, каталогизации и отслеживания происхождения данных.
OpenSearch	Инструмент для полнотекстового поиска, анализа и визуализации логов.
PersistentVolume (PV)	Объект кластера Kubernetes, представляющий реально выделенное физическое хранилище.
PostgreSQL	Реляционная система управления базами данных для хранения транзакционных данных.
StatefulSet	Контроллер Kubernetes для приложений, требующих сохранения состояния и уникальной идентификации.
StorageClass	Описание типов доступных хранилищ с правилами их динамического выделения.
Zookeeper	Сервис координации, используемый ClickHouse для синхронизации репликации и шардирования.

1 Введение

1.1 Область применения

Настоящее руководство предназначено для администраторов программного обеспечения «ДатаБазис» (ПО «ДатаБазис») (далее по тексту – Система).

Область применения настоящего документа распространяется на все подсистемы, модули и разделы ПО «ДатаБазис».

1.2 Уровень подготовки пользователя

Администратор системы должен обладать квалификацией в области информационных технологий, владеть соответствующей терминологией, и иметь представление о принципах работы Интернет-сайтов.

1.3 Перечень эксплуатационной документации, с которой необходимо ознакомиться пользователю

Администратору ПО «ДатаБазис» достаточно ознакомиться с настоящим руководством.

2 Подготовка к работе

2.1 Необходимые сторонние программы и компоненты

Общее программное обеспечение (далее – ОПО), необходимое для функционирования серверной части ПО «ДатаБазис», включает следующие компоненты:

- группа серверов Kubernetes (K8S)
  - серверная операционная система (далее – ОС) RED OS;
  - платформа для оркестрации и мониторинга ETL-процессов airflow;
  - визуализация и анализ метрик систем мониторинга Grafana;
  - управление метаданными и каталогизация данных Openmetadata;
  - российская система управления версиями кода GitFlic;
  - непрерывная доставка приложений в Kubernetes ArgoCD;
  - поиск и аналитика логов и данных Opensearch;
  - управление идентификацией и доступом (IAM) Keycloack;
  - реляционная система управления базами данных Postgresql 14.9;
- группа серверов СУБД
  - серверная операционная система (далее – ОС) RED OS;
  - колоночная СУБД для аналитической обработки ClickHouse

2.2 Назначение компонентов системы

2.2.1 Группа серверов Kubernetes (K8S)

Назначение: Оркестрация контейнеризированных приложений и сервисов

- Обеспечивает автоматическое развёртывание, масштабирование и управление контейнерами
- Предоставляет отказоустойчивую платформу для микросервисной архитектуры
- Управляет сетевым взаимодействием, балансировкой нагрузки и распределением ресурсов
- Обеспечивает механизмы для самовосстановления приложений

2.2.2 Серверная операционная система RED OS

Назначение: Безопасная операционная среда на базе Linux

- Основана на Red Hat Enterprise Linux с российской криптографической защитой

− Соответствует требованиям регуляторов для использования в государственных информационных системах
− Обеспечивает защиту от несанкционированного доступа и кибератак
− Поддерживает отечественные средства криптографической защиты информации

2.2.3 Apache Airflow

Назначение: Платформа для оркестрации и мониторинга ETL-процессов
− Планирование и выполнение сложных рабочих процессов обработки данных
− Визуализация зависимостей между задачами в направленных ациклических графах (DAG)
− Мониторинг выполнения пайплайнов и автоматическое повторение неудачных задач
− Поддержка интеграции с различными системами хранения и обработки данных

2.2.4 Grafana

Назначение: Визуализация и анализ метрик систем мониторинга
− Создание интерактивных дашбордов для мониторинга производительности системы
− Агрегация метрик из различных источников (Prometheus, OpenSearch, БД)
− Настройка алертинга при достижении пороговых значений метрик
− Анализ трендов и производительности в реальном времени

2.2.5 OpenMetadata

Назначение: Управление метаданными и каталогизация данных
− Централизованное хранение и управление метаданными всех компонентов системы
− Обнаружение, классификация и документирование источников данных
− Отслеживание происхождения данных (data lineage)
− Обеспечение качества данных и управление политиками доступа

2.2.6 GitFlic

Назначение: Российская система управления версиями кода
− Хранение исходного кода приложений и конфигураций инфраструктуры
− Управление процессами code review и CI/CD пайплайнами

- Контроль доступа к репозиториям и ветвлениям
- Интеграция с системами непрерывной интеграции и доставки

2.2.7 ArgoCD

Назначение: Непрерывная доставка приложений в Kubernetes
- Автоматическое развёртывание приложений на основе GitOps-подхода
- Синхронизация желаемого состояния приложений с фактическим состоянием в кластере
- Визуализация различий между конфигурациями в Git и развёрнутыми версиями
- Поддержка rollback к предыдущим стабильным версиям

2.2.8 OpenSearch

Назначение: Поиск и аналитика логов и данных
- Централизованный сбор и индексирование логов всех компонентов системы
- Полнотекстовый поиск по структурированным и неструктурированным данным
- Анализ и агрегация данных в реальном времени
- Визуализация логов и создание дашбордов для анализа инцидентов

2.2.9 Keycloak

Назначение: Управление идентификацией и доступом (IAM)
- Централизованная аутентификация и авторизация пользователей
- Поддержка протоколов SSO, OAuth 2.0, OpenID Connect
- Управление ролями и правами доступа к различным компонентам системы
- Интеграция с корпоративными системами каталогов (LDAP/Active Directory)

2.2.10 PostgreSQL 14.9

Назначение: Реляционная система управления базами данных
- Хранение транзакционных данных приложений
- Обеспечение ACID-свойств для критически важных операций
- Поддержка сложных запросов и аналитических нагрузок
- Репликация данных для обеспечения отказоустойчивости

2.2.11 Группа серверов СУБД

Назначение: Кластерная инфраструктура для баз данных

- Обеспечение высокой доступности и отказоустойчивости СУБД
- Распределение нагрузки между узлами кластера
- Поддержка репликации и автоматического переключения при сбоях
- Масштабирование производительности системы хранения данных

2.2.12 ClickHouse

Назначение: Колоночная СУБД для аналитической обработки
- Высокопроизводительная обработка аналитических запросов
- Хранение и анализ больших объёмов структурированных данных
- Поддержка SQL-запросов с субсекундной задержкой на больших наборах данных
- Оптимизация для сценариев OLAP (Online Analytical Processing)

2.2.13 Взаимодействие компонентов

Данная архитектура представляет собой комплексную платформу для построения отказоустойчивых, масштабируемых систем обработки данных с соблюдением требований информационной безопасности. Все компоненты интегрированы через стандартные API и протоколы, что обеспечивает гибкость и возможность дальнейшего расширения системы.

2.3 Характеристики технических средств

Характеристики технических средств (серверов), необходимых для работы серверной части ПО «ДатаБазис», по минимальным и рекомендуемым характеристикам приведены в таблице 1.

Таблица 1 – Характеристики технических средств (серверов)

Техническое средство	**Характеристики минимальные**	**Характеристики рекомендуемые**
Серверная часть	ЦП: 120 ядер, частотой не менее 2 ГГц ОЗУ, ГБ, не менее: 196 Свободный объём дискового пространства, ГБ, не менее: 2 400 Сетевой интерфейс, Мбит/с: 100	ЦП: 320 ядер, частотой не менее 2 ГГц ОЗУ, ГБ, не менее: 588 Свободный объём SSD-диска, ГБ, не менее: 7 200 Сетевой интерфейс, Мбит/с: 100
Примечание: ЦП: центральный процессор, ОЗУ: оперативное запоминающее устройство (оперативная память)		

3 Инструкция по развертыванию (On-Premise)

Данное руководство описывает процесс развертывания платформы GitFlic, кластера Kubernetes (k8s), системы управления секретами Vault и сопутствующих сервисов (Airflow, OpenMetadata, Superset).

3.1 Требования к окружению

- **Kubernetes (версии 1.28–1.30):** Система управления контейнерами.
- **Helm (версии 3.14.1–3.17.2):** «Менеджер пакетов» для Kubernetes, позволяющий устанавливать сложные сервисы одной командой.
- **GitFlic:** Репозитории проектов.

3.2 Развертывание и запуск платформы GitFlic

На данном этапе выполняется развертывание базового сервиса для хранения исходного кода и реестра образов (**Docker Registry**). Настройка файлов конфигурации (`.env`) необходима для корректного перенаправления трафика (проброса портов) и обеспечения взаимодействия компонентов внутри изолированной среды Docker.

Для начала работы требуется импорт образа виртуальной машины GitFlic из дистрибутива. Все дальнейшие действия производятся в каталоге `/home/myfolder/docker`.

3.2.1 Конфигурация системных переменных (.env)

Редактирование файла `.env` определяет параметры доступа к системе:

- **HOST_SSH_PORT=2255** – выделенный порт для передачи кода по защищенному каналу SSH.
- **HOST_PORT=8080** – порт для доступа к веб-интерфейсу GitFlic через браузер.

3.2.1.1 Содержимое файла .env

```
#[Global]
BASE_URL=http://localhost:8080
HOST_SSH_PORT=2255
HOST_PORT=8080
CERT_VOLUME=gitflic_cert
TRANSPORT_SSH=2255
ETC_VOLUME=gitflic_etc
```

3.2.2 Настройка внешних сервисов (.external-env)

В данном файле указываются параметры интеграции с почтовым сервером для отправки уведомлений:

```
#[Global]
BASE_URL=http://localhost:8080
#[Mail]
EMAIL_SMTP_HOST=foo@bar.foo
EMAIL_SMTP_PORT=673
EMAIL_USER=foo
EMAIL_PASS=bar
SENDER_NAME=foo
SENDER_EMAIL=bar
```

3.2.3 Запуск серверных компонентов и агентов сборки

Запуск осуществляется при помощи утилиты **docker compose**, которая объединяет базу данных, серверное приложение и интерфейс в единую работающую систему.

Агент сборки (**раннер**) перезапускается отдельной командой. Это гарантирует корректную регистрацию и стабильное подключение агента к уже инициализированному серверу GitFlic.

```
# Запуск основного сервера GitFlic
docker compose --env-file .env up -d

# Перезапуск агента сборки (GitFlic-runner)
docker stop gitflic-runner
docker rm gitflic-runner
docker compose --env-file .env up -d
```

Верификация запуска: Статус активных агентов доступен в панели администратора по адресу: `https://<domain_name>/admin/runners`.

3.3 Настройка сетевой связности и системы имен (DNS)

Для корректного взаимодействия сервисов с платформой GitFlic по доменному имени требуется обеспечить преобразование этого имени в соответствующий IP-адрес. Настройка выполняется как на уровне операционных систем серверов, так и внутри внутренней сети кластера Kubernetes.

3.3.1 Локальное разрешение имен в ОС (Файл hosts)

Для того чтобы серверы кластера могли обращаться к GitFlic напрямую, выполняется привязка IP-адреса к доменному имени в системных настройках каждого узла (воркера).

Необходимо добавить соответствующую запись на всех серверах:

```
# Подключение к серверу и внесение записи в таблицу соответствия имен
ssh company-k3s.company-infra-name.company.domain
sudo -i
echo "10.202.3.101 gitflic.platform.local" >> /etc/hosts
```

3.3.2 Настройка внутренней службы имен Kubernetes (CoreDNS)

CoreDNS – это специализированная служба управления именами внутри кластера. Внесение правок в её конфигурацию гарантирует, что приложения, запущенные в контейнерах, смогут найти GitFlic по адресу `gitflic.platform.local`.

Для настройки требуется внести изменения в конфигурационный файл (ConfigMap) службы CoreDNS:

```
# Переход в режим редактирования конфигурации внутренней службы имен
kubectl edit configmap coredns -n kube-system
```

В открывшемся файле в раздел `data:` (блок `NodeHosts`) добавляется следующая информация:

```
data:
  NodeHosts: |
    # Указывается IP-адрес и соответствующий ему домен
    10.202.3.101 gitflic.platform.local
```

3.4 Установка Helm

Данный этап включает установку сервисов, обеспечивающих внешний доступ к приложениям и автоматизированное управление ресурсами кластера.

- **Входной шлюз (Ingress-Nginx)**: Выполняет функцию центрального диспетчера. Сервис принимает входящие запросы из сети и распределяет их между целевыми приложениями (например, запросы к адресу `airflow.local` перенаправляются в сервис Airflow).
- **Службы автоматизации (Операторы)**: Специализированные компоненты (ArgoCD, Redis и др.), обеспечивающие непрерывный мониторинг состояния кластера. В случае сбоя базы данных или приложения оператор выполняет автоматическое восстановление работоспособности без прямого вмешательства администратора.

3.4.1 Копирование и распаковка чартов

Для развертывания используются **Helm-чарты** – готовые наборы конфигураций. Перед установкой требуется передача архива с настройками на сервер и его последующая распаковка:

```
# Копирование архива на сервер через защищенный канал (scp)
scp docker.zip company-k3s.company-infra-name.company.domain:~
```

```
# Распаковка архива на удаленном сервере
ssh company-k3s.company-infra-name.company.domain "unzip ./docker.zip"
```

3.4.2 Установка Ingress и Операторов

Ниже приведены команды для установки основного шлюза:

```
# Установка шлюза Ingress-Nginx
ssh company-k3s.company-infra-name.company.domain "cd docker/helm-chart-k8s/;
unzip ingress-nginx.zip"
ssh company-k3s.company-infra-name.company.domain "helm install ingress-nginx
docker/helm-chart-k8s/ingress-nginx -n ingress-nginx --create-namespace"

# если "Error: INSTALLATION FAILED: cannot re-use a name that is still in
use", то
ssh company-k3s.company-infra-name.company.domain "helm upgrade --install
ingress-nginx docker/helm-chart-k8s/ingress-nginx -n ingress-nginx"
```

Перед установкой системы непрерывного развертывания **ArgoCD** требуется корректировка параметров сетевого доступа. Конфигурация выполняется в файле настроек `values.yaml`.

3.4.2.1 Корректировка параметров доступа ArgoCD

Необходимо открыть файл конфигурации для редактирования:

```
nano ~/docker/helm-chart-k8s/argocd/values.yaml
```

В блоке настроек сервера (`server`) следует активировать программный шлюз (Ingress) и указать актуальное доменное имя, по которому будет доступен интерфейс управления. Параметры приводятся к следующему виду:

```
server:
  ingress:
    # Активация ресурса Ingress для доступа к серверу Argo CD
    enabled: true

    # Регистрация стандартного контроллера (обычно "nginx")
    ingressClassName: "nginx"

    # Список доменных имен (хостов) для доступа
    hosts:
      # Требуется заменить на актуальное доменное имя организации
      - argocd-company-k8s.dev.company.domain
```

3.4.2.2 Развертывание операторов и системных служб

После настройки манифестов выполняется установка набора управляющих программ (операторов) через пакетный менеджер Helm. Каждая служба развертывается в собственном изолированном пространстве имен:

```
# Установка системы управления развертыванием ArgoCD
ssh company-k3s.company-infra-name.company.domain "helm install argocd
~/docker/helm-chart-k8s/argocd --namespace argocd --create-namespace"

# Установка оператора для автоматизированного получения секретов
ssh company-k3s.company-infra-name.company.domain "helm install external-
secrets-operator ~/docker/helm-chart-k8s/external-secrets-operator --
namespace external-secrets --create-namespace"

# Установка модулей управления поисковыми индексами и базами данных (Redis)
ssh company-k3s.company-infra-name.company.domain "helm install opensearch-
operator ~/docker/helm-chart-k8s/opensearch-operator --namespace opensearch-
operator --create-namespace"
ssh company-k3s.company-infra-name.company.domain "helm install redis-
operator ~/docker/helm-chart-k8s/redis-operator --namespace redis-operator --
create-namespace"
```

3.4.3 Настройка Storage (Longhorn)

Longhorn организует виртуальное распределенное пространство для хранения данных.

Технологическое требование: для обеспечения возможности подключения виртуальных дисков к подам кластера, на физических узлах (хостах) должна быть активна служба `open-iscsi`.

При необходимости использования режима доступа **ReadWriteMany** (одновременная запись данных несколькими приложениями на один диск) выполняются следующие действия:

```
# Подготовка узлов: установка и активация службы для работы с удаленными
томами
ssh company-k3s.company-infra-name.company.domain "sudo apt-get install open-
iscsi; sudo systemctl enable --now iscsid"

# Установка системы хранения Longhorn
ssh company-k3s.company-infra-name.company.domain "kubectl apply -f
~/docker/helm-chart-k8s/longhorn-1.8.1.yaml"
```

3.5 Развертывание и настройка системы управления секретами HashiCorp Vault

HashiCorp Vault выполняет роль централизованного защищенного хранилища для конфиденциальных данных (паролей, ключей доступа, токенов). Процесс «распечатывания» (**Unseal**) является обязательной процедурой дешифрования главного ключа доступа к данным после каждого запуска сервиса.

3.5.1 Установка и первичная инициализация

Перед установкой требуется корректировка параметра `ingress.hosts` в файле конфигурации `vault/values.yaml` для обеспечения корректного сетевого доступа.

```
global:
.....
  ingress:
    enabled: true # Включаем ингресс
    labels: {}
      # traffic: external
    annotations: {}
      # |
      # kubernetes.io/ingress.class: nginx
      # kubernetes.io/tls-acme: "true"
      #   or
      # kubernetes.io/ingress.class: nginx
      # kubernetes.io/tls-acme: "true"

    # Optionally use ingressClassName instead of deprecated annotation.
    # See: https://kubernetes.io/docs/concepts/services-
networking/ingress/#deprecated-annotation
    ingressClassName: "nginx" # Прописываем класс, обычно "nginx"

    # As of Kubernetes 1.19, all Ingress Paths must have a pathType
configured. The default value below should be sufficient in most cases.
    # See: https://kubernetes.io/docs/concepts/services-
networking/ingress/#path-types for other possible values.
    pathType: Prefix

    # When HA mode is enabled and K8s service registration is being used,
    # configure the ingress to point to the Vault active service.
    activeService: true
    hosts:
      - host: vault-company-k8s.dev.company.domain # Указывавем хост по
которому будет доступен волт
        paths: ["/"]
.......
```

```
# Установка Helm-чарта Vault в кластер Kubernetes
ssh company-k3s.company-infra-name.company.domain "helm install vault
~/docker/helm-chart-k8s/vault --namespace vault --create-namespace"

# Инициализация хранилища (выполняется один раз)
kubectl exec vault-0 -n vault -- vault operator init
```

Важно: В результате выполнения команды будут созданы 5 ключей распечатывания (**Unseal Keys**) и один мастер-токен (**Root Token, например** hvs.ft95OiABgM3v7HB8KMkxIJ5Z). Данные сведения необходимо сохранить в защищенном месте.

3.5.2 Процедура распечатывания (Unseal)

Для приведения хранилища в рабочее состояние требуется ввести любые 3 ключа из 5 полученных при инициализации. Процедуру можно выполнить через веб-интерфейс по настроенному адресу или через командную строку:

```
# Вход в интерактивный режим контейнера Vault
kubectl -n vault exec -it vault-0 -- sh

# Выполнение команды распечатывания (повторить минимум 3 раза с разными
ключами)
/ $ vault operator unseal
```

3.5.3 Настройка механизмов хранения и правил доступа

Для организации хранения данных используется движок версии **KV2** (хранилище «ключ-значение» с поддержкой версионности). Правила доступа (политики) ограничивают возможности приложений, разрешая только чтение данных из определенного пути.

```
# Авторизация в системе и создание пути хранения для GitFlic
/ $ vault login <ROOT_TOKEN>
/ $ vault secrets enable -path=gitflic kv-v2

# Создание файла политики с правами на просмотр и чтение (list, read)
/ $ vi /tmp/gitflic_policy.hcl
path "gitflic/*" {
        capabilities = ["list","read"]
}

# Регистрация политики в системе под именем gitflic_ro
/ $ vault policy write gitflic_ro /tmp/gitflic_policy.hcl
```

3.5.4 Настройка методов авторизации

3.5.5 Настройка методов авторизации

В системе настраиваются два независимых способа проверки подлинности:

1. Для внешних процессов (CI/CD): Использование пары логин/пароль.

```
/ $ vault auth enable userpass
/ $ vault write auth/userpass/users/ci_user_vault_ro
password=password4ci_user_vault_ro policies=gitflic_ro
```

2. Для внутренних ресурсов Kubernetes: Автоматическая авторизация через сервисные учетные записи кластера.

```
# Получение сертификата безопасности кластера (выполнить на сервере k3s)
kubectl config view --raw --minify --flatten -o
jsonpath='{.clusters[].cluster.certificate-authority-data}' | base64 --decode

# Настройка связи Vault с API Kubernetes (указать IP сервера и сертификат)
/ $ vault auth enable kubernetes
/ $ vault write auth/kubernetes/config
kubernetes_host="https://10.202.3.110:6443"
kubernetes_ca_cert="[ДАННЫЕ_СЕРТИФИКАТА]"

# Создание роли для привязки ServiceAccount кластера к политике Vault
/ $ vault write auth/kubernetes/role/gitflic-k8s \
        alias_name_source=serviceaccount_uid \
        bound_service_account_names=k8s-sa \
        bound_service_account_namespaces=* \
        policies=gitflic_ro
```

3.5.6 Интеграция Vault с кластером (External Secrets)

Для автоматической передачи секретов из Vault в приложения Kubernetes используется оператор External Secrets. Для его работы необходимо создать системную учетную запись и настроить точки подключения.

```
# Создание системной учетной записи в пространстве имен vault
kubectl apply -f ./docker/helm-chart-k8s/service_account.yaml

# Настройка глобальных точек подключения к Vault
kubectl apply -f ./docker/helm-chart-k8s/secretstore.yaml
```

3.5.7 Наполнение структуры секретов

Для автоматизации создания дерева папок и записи начальных значений используется вспомогательный инструмент импорта.

```
# Запуск импорта (выполняется на сервере GitFlic)
# Параметры: адрес Vault, Root Token и целевой путь
docker run -e DEST_URL="https://vault-company-k8s.dev.company.domain" \
           -e DEST_TOKEN="[ВАШ_ROOT_TOKEN]" \
```

```
          -e DEST_KV_PATH="gitflic" \
          -e DEST_SECRET_PATH="" \
          --name vault-import vault-import:0.1
```

После завершения импорта требуется через веб-интерфейс Vault вручную актуализировать ключевые переменные (адреса сервисов Airflow, ClickHouse, Keycloak и др.) в соответствии с параметрами текущей среды развертывания.

3.5.8 Конфигурация ClusterSecretStore

Для обеспечения взаимодействия оператора внешних секретов (**External Secrets Operator**) с хранилищем Vault применяется следующий манифест (Файл `secretstore.yaml`). Он описывает способ подключения и авторизации для всех пространств имен в кластере:

```
# Файл: secretstore.yaml
# Описание: Конфигурация глобальных хранилищ секретов для интеграции
Kubernetes с HashiCorp Vault

apiVersion: external-secrets.io/v1beta1 # Версия API оператора External
Secrets
kind: ClusterSecretStore                # Тип ресурса: общекластерное
хранилище секретов
metadata:
  name: vault-backend                   # Уникальное имя хранилища для сервиса
GitFlic
  namespace: vault                      # Пространство имен, в котором
развернут оператор
spec:
  provider:                             # Определение поставщика секретов
    vault:                              # Использование HashiCorp Vault в
качестве провайдера
      server: "http://vault.vault:8200" # Внутренний сетевой адрес сервера
Vault в кластере
      path: "gitflic"                   # Путь (драйвер) хранения секретов в
Vault
      version: "v2"                     # Использование движка KV (Key-Value)
версии 2
      auth:                             # Настройки механизмов аутентификации
        kubernetes:                     # Использование метода проверки
подлинности через Kubernetes
          mountPath: "platform-k8s"     # Путь активации метода аутентификации
в настройках Vault
          role: "gitflic-k8s"           # Роль в Vault, определяющая права
доступа для этого хранилища
          serviceAccountRef:            # Ссылка на системную учетную запись
для авторизации
```

```
            name: "k8s-sa"              # Имя ServiceAccount, от которого
выполняются запросы к Vault
---
apiVersion: external-secrets.io/v1beta1
kind: ClusterSecretStore
metadata:
  name: vault-backend-gitlab            # Хранилище, выделенное для интеграции
с внешними Git-системами
  namespace: vault
spec:
  provider:
    vault:
      server: "http://vault.vault:8200"
      path: "gitflic"
      version: "v2"
      auth:
        kubernetes:
          mountPath: "platform-k8s"
          role: "gitflic-k8s"
          serviceAccountRef:
            name: "k8s-sa"
---
apiVersion: external-secrets.io/v1beta1
kind: ClusterSecretStore
metadata:
  name: vault-backend-k8s               # Универсальное хранилище для
системных секретов кластера
  namespace: vault
spec:
  provider:
    vault:
      server: "http://vault.vault:8200"
      path: "gitflic"
      version: "v2"
      auth:
        kubernetes:
          mountPath: "platform-k8s"
          role: "gitflic-k8s"
          serviceAccountRef:
            name: "k8s-sa"
```

3.5.9 Наполнение хранилища секретов и управление ключами доступа

Для первичного заполнения системы **HashiCorp Vault** данными и автоматизации создания структуры папок применяется вспомогательный контейнер `vault-import`.

Данный инструмент переносит заранее подготовленный шаблон структуры «ключ-значение» в итоговое хранилище.

3.5.10 Автоматизированный импорт структуры данных

Запуск процесса импорта осуществляется через Docker-контейнер. В параметрах команды указываются:

- --dns=8.8.8.8 – ДНС, который может резолвить домен vault (`DEST_URL`)
- адрес сервера назначения (`DEST_URL`),
- токен доступа с правами записи (`DEST_TOKEN`) (*значение в блоке кода приведено справочно*)
- целевой путь в хранилище (`DEST_KV_PATH`)
- путь до каталога (`DEST_SECRET_PATH`), куда будем разворачивать структуру секретов, оставляем пустым.

```
# Запуск контейнера для автоматического наполнения Vault
docker run --dns=*your_dns_server* \
-e DEST_URL="https://vault-company-k8s.dev.company.domain" \
-e DEST_TOKEN="hvs.ft95OiABgM3v7HB8KMkxIJ5Z" \
-e DEST_KV_PATH="gitflic" \
-e DEST_SECRET_PATH="" \
--name vault-import vault-import:0.1
```

После внедрения структуры необходимо внести изменения вручную в данные переменные через `web ui vault`:

```
https://vault-company-k8s.dev.company.domain
# входим по токену hvs.ft95OiABgM3v7HB8KMkxIJ5Z и заполняем указанные ниже
карманы

company/dev/airflow-dags/variables/AIRFLOW_HOSTPORT
      "value": "https://airflow-company-dev.k8s-dev.company.domain"

company/dev/airflow-dags/variables/CLICKHOUSE_HOSTPORT
      "value": "10.223.0.200:10004"

company/dev/airflow-dags/variables/OPENMETADATA_HOSTPORT
      "value": "https://datacatalog-company-dev.k8s-dev.company.domain/api"

company/dev/airflow-dags/variables/SFTP_HOSTPORT
      "value": "sftp://sftp@10.223.0.33:32022"

company/dev/airflow-dags/variables/SUPERSET_DB_HOSTPORT
      "value": "superset-db-rw.company.infra-name:5432"

company/dev/airflow-dags/variables/SUPERSET_HOSTPORT
```

```
      "value": "http://superset-company-dev.k8s-dev.company.domain:8088"

company/dev/airflow/env
      "connection": "postgresql://airflow:empty@10.221.0.122:5432/airflow"

company/dev/common/imagepullsecret-company
      "dockerconfigjson":
"{\"auths\":{\"registry.company.domain\":{\"username\":\"baseimage-
imagepullsecret\",\"password\":\"empty\"}}}"

company/dev/data-mart
      "airflow_address": "https://airflow-company-dev.k8s-dev.company.domain"
      "argocd_address": "https://argocd-company.dev.company.domain"
      "clickhouse_host": "10.223.0.200"
      "clickhouse_host_01": "company-clickhouse-01-dev.infra-
name.company.domain"
      "clickhouse_host_02": "company-clickhouse-02-dev.infra-
name.company.domain"
      "connection": "postgresql://airflow:empty@airflow-db-rw.companyplatform-
company:5432/airflow"
      "gitflic_address": "https://gitflic.dev.company.domain"
      "opensearch_dashboards_address": "https://dashboard-company-dev.k8s-
dev.company.domain"
      "sftp_address": "sftp://....company.domain:22"

company/dev/keycloak/env
      "airflow_client_redirect_uris": "https://airflow-company-dev.k8s-
dev.company.domain/*"
      "argocd_client_redirect_uris": "https://argocd.k8s-pub-
common.company.domain/*"
      "clickhouse_client_redirect_uris": "https://grafana-company-dev.k8s-
dev.company.domain/*"
      "gitflic_client_redirect_uris": "https://gitflic-
02.dev.company.domain/*"
      "grafana_client_redirect_uris": "https://grafana-company-dev.k8s-
dev.company.domain/*"
      "keycloak_address": "https://dataplatform-company-keycloak.k8s-
dev.company.domain/auth/"
      "opendashboards_client_redirect_uris": "https://dashboard-company-
dev.k8s-dev.company.domain/*"
      "openmetadata_client_redirect_uris": "https://datacatalog-company-
dev.k8s-dev.company.domain/*"
      "superset_client_redirect_uris": "https://superset-company-dev.k8s-
dev.company.domain/*"

company/dev/omd/env
      "openmetadata_address": "https://datacatalog-company-dev.k8s-
dev.company.domain"
```

```
company/dev/sql_exporter/env
      "CLICKHOUSE_CONNECTION": "clickhouse+http://admin:empty@company-
clickhouse-01-dev.infra-name.company.domain:8123"

company/dev/superset/env
      "superset_address": "https://superset-company-dev.k8s-
dev.company.domain"
```

3.6 Конфигурация среды запуска контейнеров и параметров приложений

Перед инициализацией автоматического развертывания через ArgoCD требуется выполнить настройку сетевых реестров на узлах кластера и подготовить конфигурационные файлы приложений.

3.6.1 Настройка локальных реестров (Mirroring)

Для обеспечения возможности загрузки образов из локального хранилища GitFlic необходимо настроить перенаправление запросов (зеркалирование) в параметрах среды выполнения контейнеров.

Для кластеров на базе k3s: Требуется создать или отредактировать файл `/etc/rancher/k3s/registries.yaml`, добавив в него адреса локального реестра и правила перезаписи для внешних источников (`docker.io`, `ghcr.io`):

```
mirrors:
  "gitflic.platform.local":
endpoint:
  - "http://gitflic.platform.local:8080"
  "ghcr.io":
endpoint:
  - "http://gitflic.platform.local:8080"
rewrite:
  "(.*)": "company/company/$1"
  "docker.io":
endpoint:
  - "http://gitflic.platform.local:8080"
rewrite:
  "(.*)": "company/company/$1"
configs:
  "*":
tls:
  insecure_skip_verify: true
```

Для стандартных кластеров Kubernetes (containerd): Если используется стандартный Kubernetes, аналогичные правки вносятся в файл `/etc/containerd/config.toml` в секцию конфигурации реестров:

```
[[plugins."io.containerd.grpc.v1.cri".registry]
  [plugins."io.containerd.grpc.v1.cri".registry.mirrors]

[plugins."io.containerd.grpc.v1.cri".registry.mirrors."gitflic.platform.local
"]
      endpoint = ["http://gitflic.platform.local:8080"]
  [plugins."io.containerd.grpc.v1.cri".registry.mirrors."ghcr.io"]
     endpoint = ["http://gitflic.platform.local:8080"]
[plugins."io.containerd.grpc.v1.cri".registry.mirrors."docker.io"]
     endpoint = ["http://gitflic.platform.local:8080"]
  [plugins."io.containerd.grpc.v1.cri".registry.configs]
    [plugins."io.containerd.grpc.v1.cri".registry.configs."*".tls]
       insecure_skip_verify = true
```

После внесения изменений требуется обязательная перезагрузка системной службы:

```
systemctl restart containerd
```

3.6.2 Настройка манифестов приложений (values.yaml)

В репозитории управления основной конфигурацией (`devops/gitops.git`) необходимо зайти в файлы `values.yaml` для каждого сервиса и актуализировать параметры сетевого доступа (**Ingress**) и хранения данных (**StorageClass**).

Настройка сетевых адресов (Ingress): В блоке `ingress.hosts` соответствующих файлов необходимо указать актуальные доменные имена для доступа к сервисам через браузер:

- **Airflow:** `onpremise/20_data-apps/airflow/values.yaml`
- **OpenMetadata (Dependencies):** `onpremise/20_data-apps/omd/values/omd-dependencies.yaml`
- **OpenMetadata (Core):** `onpremise/20_data-apps/omd/values/omd.yaml`
- **Superset:** `onpremise/20_data-apps/superset/values.yaml`

Настройка классов хранения данных (StorageClass): Для сервисов, требующих одновременного доступа нескольких приложений к одним и тем же данным (режим **ReadWriteMany**), необходимо указать класс хранения `longhorn` в следующих параметрах:

- **Airflow:** блоки `dags.storageClassName` и `logs.storageClassName`.
- **OpenMetadata Dependencies:** параметры `opensearch.persistence.storageClass`, `airflow.dags.storageClass` и `airflow.logs.storageClass`.

Если используемый в кластере класс хранения по умолчанию отличается от стандартного, его название также требуется зафиксировать в указанных конфигурационных файлах перед запуском синхронизации в ArgoCD.

3.6.3 Создание и регистрация ключей доступа (SSH)

Для обеспечения защищенного взаимодействия между системой непрерывного развертывания (**ArgoCD**) и хранилищем исходного кода (**GitFlic**) используется авторизация по протоколу SSH с использованием ключевых пар.

3.6.3.1 Создание пары ключей

Генерация новой ключевой пары на базе алгоритма Ed25519 выполняется следующими командами:

```
# Создание директории для хранения сертификатов
mkdir ./certs

# Генерация ключей без кодовой фразы
ssh-keygen -t ed25519 -N "" -q -f ./certs/public_key.pem
```

3.6.3.2 Регистрация приватного ключа в Vault

Содержимое файла приватного ключа (`./certs/public_key.pem`) необходимо сохранить в системе управления секретами Vault для его последующего использования сервисом ArgoCD.

1. В интерфейсе Vault следует перейти по пути: `gitflic/kv/commonSecrets/argocd/repositories/git`.
2. В параметрах секретов (**env**) требуется создать запись с ключом `SSH_PRIVATE_KEY` и вставить в качестве значения содержимое сгенерированного приватного ключа.

3.6.3.3 Регистрация публичного ключа в GitFlic

Для идентификации системы развертывания публичный ключ необходимо добавить в профиль администратора платформы GitFlic:

1. В веб-интерфейсе GitFlic следует перейти в раздел: **Настройки** (**Settings**) -> **Ключи SSH** (**Keys**).
2. Нажать кнопку «Добавить SSH ключ».

3. В поле содержимого вставить данные из файла `./certs/public_key.pem.pub`.

4. В поле названия указать идентификатор: `argocd-token-01`.

5. Поле даты окончания действия оставить пустым и подтвердить добавление.

3.6.3.4 Настройка доверия в интерфейсе ArgoCD

Для подтверждения подлинности сервера GitFlic при подключении из ArgoCD требуется добавить открытый ключ в список известных хостов (**Known Hosts**):

1. В интерфейсе ArgoCD перейти в раздел: Settings -> Repository certificates and known hosts.

2. Нажать кнопку ADD SSH KNOWN HOSTS.

3. Внести запись, содержащую адрес сервера, порт и публичный ключ в следующем формате:

```
[gitflic.platform.local]:2255 ssh-ed25519
AAAAC3NzaC1lZDI1NTE5AAAAIDISMXQpoHbLaG4ZrwelQFYz5PiRyU5SqAQv4HHVvMNh
```

4. Сохранить изменения для применения настроек безопасности.

3.7 Развертывание систем управления базами данных (СУБД)

Установка аналитической системы **ClickHouse** и отказоустойчивого кластера **PostgreSQL** (с использованием технологии **Patroni**) выполняется через систему автоматизации **Ansible**. Данный этап обеспечивает настройку серверных мощностей вне контура Kubernetes с соблюдением требований к производительности дисковых подсистем и сетевых интерфейсов.

3.7.1 Конфигурация сетевых узлов (Inventory)

Для определения целевых серверов требуется внести изменения в файл `inventory.ini`, распределив хосты по функциональным группам:

```
[clickhouse_onpremise]
# Узлы для размещения базы данных ClickHouse
company-clickhouse-01.company-infra-name.company.domain
ansible_host=10.202.3.105
company-clickhouse-02.company-infra-name.company.domain
ansible_host=10.202.3.106

[patroni_onpremise]
# Узлы кластера PostgreSQL (Patroni)
company-pg-db-01 ansible_host=10.202.3.107
company-pg-db-02 ansible_host=10.202.3.108
company-pg-db-03 ansible_host=10.202.3.109
```

При необходимости настройки специфических параметров (память, пути к данным) выполняется редактирование соответствующих файлов переменных в директориях `group_vars` и `host_vars`.

3.7.2 Развертывание ClickHouse и проверка доступа

Установка выполняется путем запуска соответствующего сценария. Для доступа к зашифрованным паролям в параметрах указывается файл с ключом доступа (`--vault-password-file`).

```
# Запуск процесса установки ClickHouse
ansible-playbook -i ./inventory/inventory.ini -l clickhouse_onpremise
./playbooks/clickhouse.yml --vault-password-file ./pwd-file
```

Верификация установки: Для проверки доступности базы данных используется команда запроса списка доступных баз. Пароль администратора, хранящийся в секретах Ansible, предварительно дешифруется:

```
# Дешифровка пароля из конфигурации
echo '$ANSIBLE_VAULT;1.1;AES256...[полный код]...' | ansible-vault decrypt --
vault-password-file pwd-file

# Выполнение тестового запроса к серверу
clickhouse-client -u admin --password [ПОЛУЧЕННЫЙ_ПАРОЛЬ] --query "SHOW
DATABASES"
```

3.7.3 Установка вспомогательного шлюза (CHProxy)

Прокси-сервер для оптимизации запросов к ClickHouse развертывается отдельным сценарием. Перед запуском рекомендуется убедиться в наличии актуальных коллекций для работы с Docker-платформой:

```
# Обновление необходимых библиотек Ansible
ansible-galaxy collection install community.docker --force

# Установка шлюза CHProxy
ansible-playbook -i ./inventory/inventory.ini -l clickhouse_onpremise --
limit=company-clickhouse-01.company-infra-name.company.domain
./playbooks/chproxy.yml
```

3.7.4 Развертывание кластера высокой доступности PostgreSQL

Отказоустойчивость базы данных обеспечивается совместной работой компонентов **ETCD** (координация) и **Patroni** (автоматическое переключение при сбоях). Установка производится последовательно:

```
# Этап 1: Развертывание службы координации ETCD
ansible-playbook -i ./inventory/inventory.ini -l patroni_onpremise
./playbooks/etcd.yml
```

```
# Этап 2: Развертывание системы управления Patroni
ansible-playbook -i ./inventory/inventory.ini -l patroni_onpremise
./playbooks/patroni.yml --vault-password-file ./pwd-file
```

3.8 Развертывание системы управления идентификацией Keycloak

Для обеспечения централизованной аутентификации и управления доступом пользователей развертывается сервис **Keycloak**. Процесс автоматизирован с помощью конвейеров сборки (CI/CD) на платформе GitFlic.

3.8.1 Проверка параметров среды (Переменные CI/CD)

Перед запуском процесса установки необходимо убедиться в актуальности учетных данных и адресов сервисов. Проверка выполняется в веб-интерфейсе платформы GitFlic по пути: `Администрирование` -> `Переменные CI/CD`.

Необходимые параметры:

- **VAULT_ADDR:** Сетевой адрес хранилища секретов (например, `http://vault-company-k8s.dev.company.domain`).
- **VAULT_USER / VAULT_USER_PASS:** Учетные данные пользователя Vault с правами доступа «только чтение» (`gitflic_ro`).
- **CI_KUBECONFIG:** Конфигурационный файл доступа к кластеру Kubernetes в формате Base64.
- **CI_RG_USER / CI_RG_PASSWORD:** Имя пользователя и транспортный токен для авторизации в реестре образов GitFlic.

3.8.2 Запуск процесса сборки и установки

Развертывание выполняется через запуск автоматизированного конвейера в проекте `devops/company-auth`.

Порядок действий:

1. Перейти в раздел **Конвейеры** (**CI/CD**) указанного проекта.
2. Нажать кнопку **«Создать конвейер»**.
3. Выбрать целевую ветку `onpremise` и подтвердить запуск.

При первичной инсталляции системы требуется последовательный ручной запуск следующих этапов (задач) в рамках одного и того же конвейера:

- **build-image:** Сборка и публикация программного образа Keycloak в локальный реестр.
- **On-Premise:** Формирование и запуск ресурсов сервиса в кластере Kubernetes.

– **01_plan_dev:** Предварительный анализ изменений в конфигурации инфраструктуры (инструмент Terraform).

– **01_apply_dev:** Применение настроек и финальная конфигурация окружения Keycloak.

Важное технологическое требование: Все указанные задачи должны выполняться строго внутри одной сессии конвейера. Это гарантирует передачу идентификатора (тега) собранного образа на последующие этапы развертывания. Подтверждением успешного завершения является переход всех компонентов Keycloak в состояние «Запущено» в кластере.

3.9 Настройка ArgoCD и запуск автоматического развертывания приложений

После установки базовых компонентов кластера требуется вход в систему управления развертыванием **ArgoCD** для подключения репозиториев и запуска основного управляющего приложения.

3.9.1 Первый вход в интерфейс и получение учетных данных

Доступ к системе осуществляется через веб-интерфейс по ранее настроенному доменному имени (например, `argocd-company-k8s.dev.company.domain`).

– Логин по умолчанию: `admin`.

– **Получение пароля:** Изначальный пароль администратора генерируется автоматически при установке и хранится в защищенном объекте кластера Kubernetes. Для его извлечения и расшифровки выполняется следующая команда:

```
# Извлечение и декодирование пароля администратора из секретов кластера
ssh company-k3s.company-infra-name.company.domain "kubectl -n argocd get
secret argocd-initial-admin-secret -o jsonpath='{.data.password}' | base64 -
d"
```

3.9.2 Подключение репозитория конфигураций

Для того чтобы ArgoCD мог считывать настройки приложений из GitFlic, необходимо добавить репозиторий, используя сгенерированную ранее пару SSH-ключей.

1. В интерфейсе ArgoCD перейти в раздел: **Settings** -> **Repositories** -> **CONNECT REPO**.
2. Выбрать метод подключения: **VIA SSH**.
3. Заполнить следующие поля:
   – **Name:** `gitops-base`.

– **Repository URL:** `ssh://git@gitflic.platform.local:2255/devops/gitops.git`.

– **SSH private key data:** Вставить содержимое сгенерированного ранее приватного ключа.

3.9.3 Создание главного управляющего приложения (Init App)

Развертывание всех прикладных сервисов платформы выполняется через создание единого объекта типа **Application**, который автоматически синхронизирует состояние кластера с конфигурацией в Git-репозитории.

1. В разделе **Applications** нажать кнопку **NEW APP**.
2. Выбрать режим редактирования **EDIT AS YAML** и вставить следующий манифест:

```
kind: Application
apiVersion: argoproj.io/v1alpha1
metadata:
  name: init-company
  namespace: argocd
spec:
  project: default
  source:
    # Ссылка на репозиторий с конфигурациями всех сервисов
    repoURL: ssh://git@gitflic.platform.local:2255/devops/gitops.git
    targetRevision: main
    path: ./init/onpremise/
  destination:
    # Указание внутреннего адреса API кластера
    server: "https://kubernetes.default.svc"
    # Базовое пространство имен для прикладных сервисов
    namespace: companyplatform-company
  syncPolicy:
    syncOptions:
      # Автоматическое создание целевого пространства имен, если оно
отсутствует
      - CreateNamespace=true
    automated:
      # Автоматическое исправление отклонений (Self-Heal)
      selfHeal: true
      # Удаление устаревших ресурсов (Prune)
      prune: true
```

После сохранения манифеста в интерфейсе ArgoCD начнется процесс последовательной установки всех сервисов платформы в автоматическом режиме. Контроль завершения развертывания осуществляется через мониторинг статусов приложений в веб-панели.

3.10 Инициализация хранилищ данных и запуск аналитических процессов

После успешного развертывания программных модулей выполняется настройка связей между компонентами и запуск процессов обработки данных. На данном этапе осуществляется создание структур таблиц (**DDL**) в базе данных ClickHouse и активация сценариев автоматизации в Airflow.

3.10.1 Настройка подключений в Airflow

Для взаимодействия системы управления задачами (**Airflow**) с внешними ресурсами необходимо настроить параметры авторизации в разделе «Подключения» (**Connections**). Данные для заполнения полей должны быть извлечены из соответствующих разделов в **Vault**:

- **SFTP-сервер:** Путь к учетным данным – `company/dev/airflow-dags/connections/SFTP_COMPANY`. Используется для обмена файлами через внутренний адрес кластера: `sftp.platform-company:32022`.
- **База данных ClickHouse:** Путь к учетным данным – `secrets/gitflic/company/dev/airflow-dags/connections/CLICKHOUSE_COMPANY`.

3.10.2 Создание объектов в ClickHouse (Data-Mart)

Процесс автоматического создания таблиц и витрин данных запускается через конвейер в проекте `data-mart`:

1. Открыть проект в GitFlic и перейти в раздел **Конвейеры**.
2. Создать и запустить новый конвейер из ветки `main`.
3. По завершении требуется проверить протоколы выполнения (логи) на наличие ошибок, даже при получении системой общего статуса «Успешно».

3.10.3 Развертывание и активация сценариев (DAGs)

Сценарии автоматизации (**DAGs**) синхронизируются из основного репозитория в GitFlic с периодичностью 30 минут. При необходимости внесения изменений или отладки конвейеров используется следующий регламент:

Порядок внесения изменений в сценарии:

1. Выполнить доработку кода в репозитории `company-airflow-dags`.
2. Если изменения затрагивают вложенные модули, актуализировать зависимости командой:

```
git submodule update --recursive
```

3. Убедиться в успешном прохождении проверок пайплайна.

3.10.4 Верификационная проверка автономности GitFlic

Для подтверждения независимости среды от внешних ресурсов (например, GitLab) рекомендуется провести тестовый цикл внесения изменений:

1. Временно отключить режим зеркалирования для проекта `company-airflow-dags`.
2. Создать тестовую ветку и внести в комментарий кода (файл `dag_sftp2datamart_main_runner.py`) контрольную метку (например, `companyCOMPANY-334`).
3. Оформить и подтвердить запрос на слияние (**Merge Request**).
4. Проверить автоматическое применение изменений в интерфейсе Airflow (раздел кода сценария).

По завершении проверки тестовый проект следует удалить и повторно настроить исходное зеркалирование репозитория.

3.11 Настройка системы визуализации данных Apache Superset

Для формирования аналитических отчетов и графиков выполняется подключение системы **Superset** к базе данных ClickHouse. Идентификация в системе базы данных осуществляется через специально выделенную учетную запись с правами доступа на чтение.

3.11.1 Получение учетных данных доступа

Пароль для пользователя `superset_reader` хранится в защищенном виде в конфигурационных файлах проекта Ansible. Для его извлечения необходимо выполнить процедуру дешифрования:

```
# Выполнение команды дешифровки в директории проекта Ansible
echo '$ANSIBLE_VAULT;1.1;AES256...[полный код из инструкции]...' | ansible-vault decrypt --vault-password-file pwd-file

# Примечание: В результате будет получен пароль (например: Pa93142w0rd)
```

3.11.2 Настройка подключения к базе данных

Регистрация источника данных производится в веб-интерфейсе Superset.

Порядок действий:

1. Перейти в раздел **Settings -> Database Connections**.
2. Нажать кнопку добавления нового подключения и выбрать тип **ClickHouse Connect**.
3. В параметрах подключения указать следующие данные:
   - **Имя базы данных (Display Name)**: `company-clickhouse`.
   - **Сетевой адрес и порт:** `10.202.3.105:9090` (через шлюз CHProxy).
   - **Логин:** `superset_reader`.
   - **Пароль:** Полученный на этапе дешифровки пароль.
   - **Название базы данных для** подключения**:** `superset_reader`.

Важное примечание: При создании подключения интерфейс может выдать предупреждение об ошибке связи. В большинстве случаев это не влияет на работоспособность, и соединение устанавливается корректно.

3.11.3 Верификация работы

Для подтверждения успешной интеграции рекомендуется выполнить тестовый запрос или построить произвольный график на основе доступных таблиц ClickHouse в разделе **Chart**. Появление данных в предварительном просмотре графика подтверждает правильность настройки всех компонентов аналитического контура.

3.12 Инициализация каталога метаданных OpenMetadata

Для формирования единого реестра данных, отслеживания их происхождения (**Lineage**) и контроля качества используется система **OpenMetadata**. Процесс первичного наполнения каталога автоматизирован через сценарии в **Airflow**.

3.12.1 Верификация параметров подключения

Перед запуском процессов импорта метаданных необходимо подтвердить актуальность сетевого адреса сервиса в хранилище секретов. Проверка выполняется в веб-интерфейсе **Vault** по следующему пути: `vault/secrets/gitflic/kv/company/dev/airflow-dags/variables/OPENMETADATA_HOSTPORT`

3.12.2 Регламентированная последовательность запуска сценариев

Первичное наполнение системы выполняется путем поочередного запуска сценариев (**DAG**). Для корректного построения связей между объектами переход к следующему этапу допускается только после успешного завершения предыдущего.

Порядок выполнения в интерфейсе Airflow:

1. **data_catalog_service** – регистрация базовых сервисов и структур данных из источников.
2. **data_catalog__govern** – привязка бизнес-терминов, метрик и тегов из глоссария.
3. **data_catalog_data_quality** – запуск первичных тестов на соответствие критериям качества.
4. **data_catalog_data_quality_alert** – настройка оповещений о выявленных отклонениях.
5. **data_catalog_monitoring** – активация общих механизмов мониторинга состояния данных.

Указанные действия выполняются однократно для инициализации системы. Дальнейшая актуализация метаданных и мониторинг происходят автоматически в интерфейсе OpenMetadata согласно установленным расписаниям.

3.12.3 Проверка результатов инициализации

По завершении всех этапов требуется убедиться в корректности наполнения разделов системы OpenMetadata:

- **Каталог –** содержит перечень всех подключенных сервисов и баз данных.
- **Происхождение (Lineage) –** на вкладке таблиц визуализированы связи и зависимости между объектами.
- **Домен –** зафиксирована принадлежность данных к соответствующим бизнес-направлениям.
- **Глоссарий –** импортированы термины, описание метрик и системные теги.
- **Качество данных –** в соответствующем разделе отображаются результаты выполненных проверок.

3.13 Настройка экспорта технических метрик ClickHouse

Для мониторинга активности базы данных и сбора статистических показателей используется компонент **sql_exporter**. Он обеспечивает передачу данных о производительности запросов и объемах обрабатываемой информации в систему мониторинга.

3.13.1 Развертывание конфигурации экспортера

Активация процесса мониторинга выполняется через запуск автоматизированного конвейера в проекте `sql_exporter`:

1. В интерфейсе GitFlic перейти в раздел **CI/CD** проекта `sql_exporter`.
2. Инициировать создание и выполнение нового конвейера для применения актуальных настроек сбора метрик.

3.13.2 Верификация поступления данных

Для подтверждения корректности настройки необходимо инициировать любую операцию записи в базу данных и убедиться, что система мониторинга фиксирует соответствующее событие.

Порядок проверки:

1. В интерфейсе Airflow запустить сценарий `sftp2datamart_main_runner` для выполнения типовой загрузки данных в ClickHouse.
2. После завершения задачи перейти в систему мониторинга (Grafana или консоль Prometheus).
3. Выполнить запрос к метрике: `company-clickhouse-written-rows-from-system-query-log`.

Наличие актуальных числовых значений в данной метрике подтверждает успешную настройку канала передачи данных между ClickHouse и системой мониторинга инфраструктуры.

3.14 Финальная проверка работоспособности системы мониторинга

Заключительным этапом ввода платформы в эксплуатацию является проверка доступности аналитических панелей и корректности сбора журналов событий (логов). Это подтверждает, что все компоненты инфраструктуры находятся под наблюдением и передают данные в единый центр мониторинга.

3.14.1 Верификация аналитических панелей в Grafana

Доступ к системе визуализации метрик осуществляется по адресу: `grafana-company-k8s.dev.company.domain`. Необходимо убедиться в наличии актуальных данных на следующих контрольных панелях:

– **COMPANY (Бизнес-мониторинг):** отображение графиков на основе данных из ClickHouse. Наличие значений в метрике `company-clickhouse-written-rows-from-`

`system-query-log` (ранее проверенной в рамках настройки экспортера) подтверждает связность базы данных и системы мониторинга.

– **Node Exporter Full:** визуализация технических параметров серверов (нагрузка на процессоры, использование оперативной памяти и дискового пространства).

3.14.2 Проверка журналов регистрации событий (логов)

Для контроля состояния приложений кластера необходимо удостовериться в поступлении записей в централизованное хранилище логов.

Порядок проверки:

1. Перейти в раздел **Explore** интерфейса Grafana.
2. Выбрать соответствующий источник данных (Victoria Metrics или ClickHouse).
3. Выполнить тестовый запрос к поисковому индексу: `kubernetes-logs-company-*`.

Наличие актуальных записей в результатах поиска подтверждает корректную настройку системы сбора логов из всех сервисных пространств имен платформы.

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

Изм.	Номера листов (страниц)				Всего листов (страниц) в документе	№ документа	Входящий № сопроводительного документа	Подпись	Дата
	измененных	замененных	новых	аннулированных					