Система интеллектуального анализа документов

Инструкция по развертыванию и установке

Всего листов: 82

Аннотация

Настоящий документ представляет собой руководство для развертывания и установки Системы интеллектуального анализа документов (СИАД). Руководство содержит подробное описание подготовки инфраструктуры (включая БЯМ, реестр документов, графовую БД, объектное хранилище, FTP-серверы и Kubernetes), развертывания компонентов системы (Airflow, Keycloak, интерфейс ИОИ, микросервисы OCR и DWH, система мониторинга), а также инструкции по сборке исходных кодов и проверке работоспособности компонентов. Документ предназначен для администраторов, осуществляющих внедрение и настройку СИАД.

Содержание

1. Введение

1.1. Область применения системы

Областью применения Системы является деятельность по подготовке и актуализации текстов документов.

1.2. Краткое описание возможностей системы

Система предоставляет следующие возможности:

- поиск по тексту документа ссылок на нормативно-правовые акты (НПА) и ГОСТ, а также последующее определение актуальности данных ссылок;
- определение возможности подписания документа электронной подписью;
- проверка названий документов (НПА), на которые есть ссылки по тексту документа;
- проверка наличия приложений к основной части документа, которые упоминаются по тексту;
- анализ структуры документа на предмет наличия обязательных разделов;
- при наличии в документе раздела «Список ссылок», определяется полнота такого списка в соответствии с составом ссылок, упоминаемых по тексту.

1.3. Уровень подготовки пользователя

Для выполнения операций, предусмотренных настоящим Руководством, пользователю с ролью «Администратор» потребуются знания основ информационной безопасности, навыки администрирования серверов и вычислительных сетей, рекомендуется наличие навыков и опыта работы с технологиями docker, kubernetes, Keycloak, Postgresql, Arangodb, а также опыт работы с системами управления документами.

Пользователю с ролью «Инженер ИИ» необходимо обладать опытом управления системами с искусственным интеллектом (ИИ) и опытом настройки Большой языковой модели (БЯМ), что предполагает наличие навыков создания, редактирования и оптимизации инструкций для БЯМ, навыков оценки качества выходных данных БЯМ, навыков анализа текстов для составления инструкций и примеров для БЯМ. Также потребуется понимание принципов обработки естественного языка, понимание синтаксиса, семантики и языковой структуры русского и английского языков.

1.4. Перечень эксплуатационной документации для ознакомления

Перед началом работы по установке, настройке, администрированию Системы пользователю с ролью «Администратор» необходимо ознакомиться с настоящим Руководством

в полном объеме.

1.5. Глоссарий

В тексте настоящего документа представлены следующие сокращения (см. **Таблица 1-1**).

Таблица 1-1 *— Перечень сокращений*

Сокращение/аббревиатура	Значение
АРМ	Автоматизированное рабочее место пользователя Системы
БД	База данных - совокупность данных, организованных в соответствии с концептуальной структурой, описывающей характеристики этих данных и взаимоотношения между ними, которая поддерживает одну или более областей применения
БД Реестра документов	База данных Реестра документов - Реляционная БД, содержащая атрибуты, характеризующие статус, в котором находится тот или иной документ в Системе. Запись данных в БД производится по результатам различных операций обработки документа. Чтение информации из этой БД используется для определения доступных сценариев обработки документа на различных стадиях его жизненного цикла в Системе
БЯМ	Большая языковая модель - модель машинного обучения, которая способна понимать и генерировать текст на естественном языке
ГОСТ	Государственный стандарт
ИИ	Искусственный интеллект
ИС	Информационная система
ИОИ	Интеллектуальная обработка информации
НПА	Нормативный правовой акт
ОС	Операционная система
ПО	Программное обеспечение
Руководство	Руководство по развертыванию и установке Системы интеллектуального анализа документов – настоящий документ
Система	Система интеллектуального анализа документов

Сокращение/аббревиатура	Значение
СИАД	Система интеллектуального анализа документов
СУБД	Система управления базами данных - Совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных (БД).
ЦОД	Центр обработки данных
API	Application Programming Interface - Программный интерфейс, описание способов взаимодействия одной компьютерной программы с другими, используется при разработке приложений
FTP	File Transfer Protocol - Протокол, относящийся к прикладному уровню и отвечающий за передачу данных между двумя Системами
JSON	JavaScript Object Notation - Текстовый формат обмена данными, основанный на языке программирования JavaScript
LLM	Large Language Model, Большая языковая модель
TCP/IP	Transmission Control Protocol/Internet Protocol - Набор сетевых протоколов передачи данных, используемых в сетях, включая глобальную сеть Интернет
URL	Uniform Resource Locator - система унифицированных адресов электронных ресурсов, или единообразный определитель местонахождения ресурса
VM	Виртуальная машина

В тексте настоящего документа представлены следующие термины и определения (см. **Таблица 1-2**).

Таблица 1-2 *— Перечень терминов и определений*

Термин	Определение
Документ, электронный документ	В целях настоящего документа термин используется в следующих значениях: вне Системы – текстовый документ в формате *.pdf, *.doc, *.docx, *.rtf, *.txt, *.csv, *.odt, *.xls, *.xlsx, содержащий в том числе

	нормативные ссылки на другие документы. Носителем документа является файл (файл с документом); в Системе – сущность. объединяющая все элементы и структуры хранения, которые содержат информацию (в том числе обработанную с использованием искусственного интеллекта) из файла с документом
Заказчик	Физическое или юридическое лицо, которое инициирует, заказывает и финансирует разработку, модификацию или поддержку программного обеспечения Системы. После приемки услуг по разработке Системы Заказчик является Владельцем Системы
Компонент (программного обеспечения)	Составная часть программного обеспечения, выполняющая определенную функцию
Нод (Node) (kubernetes)	Ноды или узлы, виртуальные или физические машины, на которых разворачивают и запускают контейнеры. Совокупность нод образует кластер kubernetes
Под (Pod) (в kubernetes)	Группа контейнеров с общими разделами, которые запускаются как одно приложение
Пользователь	Работник, наделенный правами доступа к Системе интеллектуального анализа документов
Распознавание содержания электронных документов	Преобразование текста и изображений, содержащихся в загруженных в Систему файлах документов, в машиночитаемые текстовые данные
Ролевая модель	Распределение прав доступа и обязанностей между ролями. Определяет, какие действия может выполнять каждая роль в рамках работы с Системой, включая создание и редактирование правил, проведение проверок и управление
Роль	Набор полномочий, который необходим пользователю для выполнения определённых рабочих задач. Каждый сотрудник может иметь одну или несколько ролей, а каждая роль может содержать от одного до множества полномочий. Роли пользователей в Системе: «Пользователь»; «Администратор»;

	«Инженер ИИ»
Роль «Администратор»	Главная роль Системы, предоставляющая доступ к настройкам Системы
Роль «Инженер ИИ»	Роль для лица, осуществляющего настройку правил по верификации документов
Роль «Пользователь»	Роль для лица, использующего функциональность Системы
Файл	Цифровой носитель информации, содержащий текстовую и графическую информацию
Docsplit	API-сервис, принимающий запрос от обработчика реестра событий и возвращающий разбитый по чанкам текст
Docsplit_consumer	Компонент, читающий таблицу событий РКС и отправляющий запрос в API-сервис Docsplit
Helm	Пакетный менеджер для kubernetes, который используется для управления приложениями в кластере
Kubectl	Командная строка для управления Kubernetes; необходим для взаимодействия с кластером Kubernetes
Kubernetes	Платформа для автоматизации развёртывания, масштабирования и управления контейнеризированными приложениями. Поддерживает основные технологии контейнеризации (docker, rocket) и аппаратную виртуализацию
PostgreSQL	Объектно-реляционная СУБД с открытым исходным кодом, которая использует и расширяет язык SQL в сочетании со многими функциями, которые позволяют безопасно хранить и масштабировать самые сложные рабочие нагрузки с данными
sshpass	Утилита для автоматизации ввода пароля при использовании SSH, что требуется для настройки доступа к серверам
System_check	API-сервис, принимающий запрос от обработчика реестра событий и возвращающий извлеченные атрибуты Для
System_check_ consumer	Компонент, читающий таблицу событий РКС и отправляющий запрос в API-сервис System_check

2. Общее описание системы «Система интеллектуального анализа документов»

2.1. Назначение системы

Система предназначена для решения следующих взаимосвязанных задач:

- распознавание содержания документов следующих форматов: *.pdf, *.doc, *.docx, *.rtf, *.txt, *.csv, *.odt, *.xls, *.xlsx;
- обработка текста документа, в том числе с применением технологий ИИ;
- обеспечение систематизированного хранения документов и связей между ними.

Обобщенным назначением Системы является информационная поддержка пользователя в принятии решений о необходимости редактирования текста документа.

2.2. Условия применения системы

2.2.1. Общие условия

Режим эксплуатации Системы: 5x8 (5 дней в неделю по 8 часов) в часы работы Заказчика.

Для возможности администрирования Системы должны выполняться условия, предусмотренные пп. 2.2.2, 2.2.3, 2.2.4 Руководства.

Для настройки журналирования событий по информационной безопасности уровень логирования рекомендуется выставлять в warning - как компромиссный уровень между объемом информации и детализацией.

2.2.2. Требования к инфраструктуре

Для работы Системы требуется 10 виртуальных машин, состав которых определен в **Таблице 2-1** настоящего Руководства.

***Таблица 2-1** – Состав виртуальных машин*

Компонент	Состав	Название сервера	ip	Примечание
Центр анализа – Большая языковая модель	code LLM, nvidia GPU drivers, docker-toolkit docker engine	vdc-rk-llm-gpu	10.10.10.10	-
СУБД	PostgreSQL	vdc-rk-postgres01	10.10.10.11	-
Master node kubernetes 01	k8s	vdc-rk-k8s01	10.10.10.12	-

Worker node kubernetes 02	k8s	vdc-rk-k8s02	10.10.10.13	-
Worker node kubernetes 03	k8s	vdc-rk-k8s03	10.10.10.14	-
Worker node kubernetes 04	k8s	vdc-rk-k8s04	10.10.10.15	-
СУБД Графовой модели данных	ArangoDB	vdc-rk-arango	10.10.10.16	-
FTP-сервер загрузки документов	vsftpd	vdc-rk-ftp01	10.10.10.17	-
FTP-сервер загрузки правил	vsftpd	vdc-rk-ftp02	10.10.10.18	-
Сервер объектного хранилище S3	minio	vdc-rk-minio01	10.10.10.19	-

2.2.3. Требования к рабочей станции для развертывания контура

Для выполнения операций по развертыванию контура Системы на рабочей станции должны быть выполнены следующие требования:

- операционная система Linux:
  - рабочая станция должна быть запущена на базе операционной системы (ОС) Linux;
  - ОС должна взаимодействовать с драйверами на GPU;
  - дополнительные требования к ОС определяются внутренними документами Заказчика;
- программное обеспечение:
  - helm;
  - kubectl;
  - sshpass;
- сетевые настройки: рабочая станция должна иметь доступ по SSH ко всем серверам, участвующим в развертывании контура Системы;

- доступ по SSH-ключам: для упрощения процесса развертывания рекомендуется настроить беспарольный доступ по SSH с рабочей станции на все серверы;
- права доступа: пользователь на рабочей станции должен иметь привилегированный доступ (sudo) для выполнения административных задач;
- настройки безопасности: SSH-ключи, используемые для доступа к серверам, должны быть защищены и безопасно храниться;
- дополнительные утилиты: утилиты `wget`, `curl`, `tar`, `unzip`, `telnet nc net-tools` должны быть подготовлены для скачивания и распаковки файлов;
- резервное копирование: рекомендуется регулярно создавать резервные копии конфигурационных файлов и ключей на рабочей станции;
- проверка и тестирование: на рабочей станции должны быть установлены инструменты для тестирования и проверки конфигураций, такие как ping, telnet, nslookup, net-tools, nc;
- документация: рабочая станция должна иметь доступ к актуальной документации о Системе и инструкциям по развертыванию.

2.2.4. Требования к программному обеспечению инфраструктуры

Для обеспечения работы Системы необходимо соблюдать следующие требования:

- наличие программного обеспечения (ПО) на виртуальных машинах (VM), составляющих инфраструктуру Системы, соответствующего требованиям, установленным в п. 2.2.4 Руководства;
- ОС: поддерживает драйверы GPU Nvidia H100 и соответствует техническим требованиям Заказчика по RHEL;
- требования к виртуальным машинам – предоставляются отдельно;
- в контуре должен быть установлен NTP-сервер;
- все машины контура должны синхронизировать время;
- на всех хостах должен быть заведён служебный пользователь с правом выполнения команд с привилегиями суперпользователя для установки и настройки компонентов Системы;
- на всех хостах должен быть установлен и настроен ssh с возможностью авторизации служебным пользователем;
- CPU – рекомендуется Xeon Gold или аналог;
- отсутствие VPN соединений между VM;

- сеть:
  - скорость не менее 1 Гбит;
  - агрегация физических линков на серверах для отказоустойчивости (не менее 2 подключений);
  - для схемы размещения в одном ЦОД все серверы должны быть в одной подсети без фильтрации трафика между ними;
- на каждой виртуальной машине должен быть установлен и настроен iptables;
- наличие Nexus: в Nexus должны быть созданы по списку репозитории для apt, helm, npm, maven, docker, pypi;
- наличие docker для запуска контейнеризованных приложений, необходимый для работы с docker-образами и контейнерами.

3. Инструкция по развертыванию системы «Система интеллектуального анализа документов»

3.1. Подготовка инфраструктуры к развертыванию системы

3.1.1. Порядок разворачивания БЯМ (vdc-rk-gpu)

3.1.1.1. Общие сведения

Для разворачивания необходимо последовательно выполнить следующие шаги:

a) Создать машину в VMware vCloudDirector с названием vdc-rk-gpu на основе ОС Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-94-generic x86_64) и с характеристиками, приведенными в **Таблице 2-1** настоящего Руководства.

b) При загрузке ОС убедиться, что версия ядра – ОС 5.15.0-94-generic.

c) Выполнить настройку сетевых интерфейсов, указав IP-адрес 10.10.10.10.

d) Создать отдельного не привилегированного пользователя «adduser user-worker» для выполнения настроек.

e) Добавить пользователя в группу sudo с использованием команды:

```
sudo usermod -aG sudo user-worker
```

3.1.1.2. Установка драйвера для nVidia GPU

Для установки необходимо последовательно выполнить следующие шаги:

a) Проверить подключение видеокарт к серверу перед установкой драйвера с использованием команды:

```
lspci -nnk
```

Ожидаемый результат – вывод списка видеокарт, подключенных к серверу.

a) При наличии видеокарт выполнить установку драйверов.

3.1.1.3. Установка NVIDIA CUDA drivers

Установку следует осуществлять на основании официальной документации, размещенной в сети Интернет по ссылке: https://docs.nvidia.com/cuda/cuda-installation-guide-linux/.

3.1.1.4. Проверка GPU

Для проверки наличия графического процессора с поддержкой CUDA GPU необходимо в командной строке указать команду:

```
lspci | grep -i nvidia
```

Если не отображаются какие-либо параметры, то следует обновить базу данных PCI-

оборудования, которая ведется в Linux, для чего необходимо:

a) Указать «update-pciids» в командной строке (обычно находится в /sbin).

b) Повторно выполнить предыдущую команду «lspci».

Примечание – GPU поддерживает CUDA в случае, если видеокарта – NVIDIA и указана в списке, размещенном в сети Интернет по ссылке: https://developer.nvidia.com/cuda-gpus).

3.1.1.5. Проверка версии Linux

Для проверки наличия поддерживаемой версии Linux (определение установленного дистрибутива и номера релиза) необходимо указать команду в командной строке:

```
uname -m &amp;&amp; cat /etc/*release
```

Примечание – средства разработки CUDA поддерживаются только в некоторых определенных дистрибутивах Linux; перечислены в примечаниях к выпуску CUDA Toolkit.

Ожидаемый результат должен быть похож на указанный в примере ниже, и изменен для конкретной системы.

Пример

```
x86_64
Red Hat Enterprise Linux Workstation выпуск 6.0 (Santiago)
```

Примечание – строка x86_64 указывает на то, что работа выполняется на 64-битной системе. Остальная часть содержит информацию о дистрибутиве.

3.1.1.6. Установка CUDA и проверки установки

Установку следует осуществлять на основании официальной документации, размещенной в сети Интернет по ссылке: https://developer.nvidia.com/cuda-downloads?target_os =Linux&target_arch=x86_64&Distribution=Ubuntu & target_version = 22.04&target_type=deb_ local.

3.1.1.7. CUDA Toolkit Installer

Для установки необходимо последовательно выполнить следующие шаги:

```
wget
https://developer.download.nvidia.com/compute/cuda/repos/ubunt
u2204/x86_64/cuda-ubuntu2204.pin

sudo mv cuda-ubuntu2204.pin /etc/apt/preferences.d/cuda-
repository-pin-600

wget
https://developer.download.nvidia.com/compute/cuda/12.6.2/loca
l_installers/cuda-repo-ubuntu2204-12-6-local_12.6.2-560.35.03-
1_amd64.deb

sudo dpkg -i cuda-repo-ubuntu2204-12-6-local_12.6.2-560.35.03-
1_amd64.deb

sudo cp /var/cuda-repo-ubuntu2204-12-6-local/cuda-*-
keyring.gpg /usr/share/keyrings/sudo apt-get update
```

```
sudo apt-get -y install cuda-toolkit-12-6
```

3.1.1.8. Установка драйвера

Для установки необходимо последовательно выполнить следующие шаги:

a) По установке с использованием команды:

```
sudo apt-get install -y nvidia-open
```

b) Перезагрузить Систему, чтобы загрузить драйверы NVIDIA:

```
sudo reboot
```

c) Проверить корректность установки драйверов – в терминале выполнить команду nvidia-smi.

Ожидаемый результат – отображение в таблице описания всех видеокарт.

3.1.1.9. Установка docker engine

Для установки необходимо последовательно выполнить следующие шаги:

a) Добавить репозиторий docker:

```
# Add Docker's official GPG key:
 sudo apt-get update
 sudo apt-get install ca-certificates curl
 sudo install -m 0755 -d /etc/apt/keyrings
 sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg
-o /etc/apt/keyrings/docker.asc
 sudo chmod a+r /etc/apt/keyrings/docker.asc


 # Add the repository to Apt sources:
 echo \
  "deb [arch=$(dpkg --print-architecture) signed-
by=/etc/apt/keyrings/docker.asc]
https://download.docker.com/linux/ubuntu \
  $(. /etc/os-release &amp;&amp; echo "$VERSION_CODENAME")
stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
 sudo apt-get update
```

b) Обновить список пакетов в репозиториях:

```
sudo apt-get update
```

c) Установить необходимые пакеты:

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

d) Добавить пользователя в группу docker:

```
sudo usermod -aG docker $USER
```

3.1.1.10. Установка и настройка Container toolkit

Для настройки и установки необходимо последовательно выполнить следующие шаги:

a) Настроить репозиторий:

```
curl -fsSL https://nvidia.github.io/libnvidia-container/gpgkey
| sudo gpg --dearmor -o /usr/share/keyrings/nvidia-container-
toolkit-keyring.gpg \

&& curl -s -L https://nvidia.github.io/libnvidia-
container/stable/deb/nvidia-container-toolkit.list | \

sed 's#deb https://#deb [signed-by=/usr/share/keyrings/nvidia-
container-toolkit-keyring.gpg] https://#g' | \

sudo tee /etc/apt/sources.list.d/nvidia-container-toolkit.list
```

b) Обновить список пакетов в репозиториях:

```
sudo apt-get update
```

c) Установить NVIDIA Container Toolkit пакеты:

```
sudo apt-get install -y nvidia-container-toolkit
```

Примечание – действия следует выполнять через Apt.

3.1.1.11. Тестирование установки драйверов и компонентов для работы с GPU в docker

Для проверки того, что GPU передает в docker-контейнеры необходимо запустить тестовый контейнер с использованием команды:

```
sudo docker run --rm --runtime=nvidia --gpus all ubuntu
nvidia-smi
```

Ожидаемый результат:

```
+-----------------------------------------------------------------
+

 | NVIDIA-SMI 535.86.10 Driver Version: 535.86.10 CUDA
Version: 12.2 |

 |-------------------+----------+----------+

 | GPU Name Persistence-M| Bus-Id Disp.A | Volatile Uncorr.
ECC |

 | Fan Temp Perf Pwr:Usage/Cap| Memory-Usage | GPU-Util
Compute M. |

 | | | MIG M. |


|=====================+======================+================
|

 | 0 Tesla T4 On | 00000000:00:1E.0 Off | 0 |

 | N/A 34C P8 9W / 70W | 0MiB / 15109MiB | 0% Default |

 | | | N/A |

 +-------------------+----------+-------------+
```

```
 +-----------------------------------------+
 | Processes: |
 | GPU GI CI PID Type Process name GPU Memory |
 | ID ID Usage |

|=============================================================
|
 | No running processes found |
 +-------------------------------------------------------
+
```

Отображение ожидаемого результата означает, что установка завершена, драйверы работают корректно.

После окончания установки можно выполнить действия по развертыванию центра анализа.

Примечание – действия следует осуществлять на основании официальной документации, размещенной в сети Интернет по ссылке: https://docs.nvidia.com/datacenter/cloud-native/container-toolkit/latest/sample-workload.html.

3.1.2. Порядок разворачивания Реестра документов (vdc-rk-postgres01)

3.1.2.1. Общие сведения

Для разворачивания необходимо последовательно выполнить следующие шаги:

a) Создать машину в VMware vCloudDirector с названием vdc-rk-postgres01 на основе ОС Ubuntu 22.04.3 LTS и с характеристиками, приведенными в **Таблице 2-1** настоящего Руководства.

b) Выполнить настройку сетевых интерфейсов, указав IP-адрес 10.10.10.11.

c) Создать отдельного не привилегированного пользователя «adduser user-worker» для выполнения настроек.

d) Добавить пользователя в группу sudo.

3.1.2.2. Установка PostgreSQL

Для установки необходимо последовательно выполнить следующие шаги:

a) Установить Postgresql 14 из предоставленных дистрибутивов.

b) Проверить установку и запуск PostgreSQ с использованием команды:

```
sudo systemctl start postgresql
sudo systemctl enable postgresql
sudo systemctl status postgresql
```

3.1.2.3. Создание базы данных, пользователя, схемы и предоставления привилегий

Пример создания базы данных, пользователя, схемы и предоставления привилегий приведен ниже.

Пример

Необходимо последовательно выполнить следующие шаги:

a) Подключиться к PostgreSQL от имени пользователя postgres:

```
sudo -i -u postgres psql
```

b) Создать пользователя и базу данных graph_project.

c) Подключиться к созданной базе данных:

```
CREATE USER graph_project WITH PASSWORD 'password';
CREATE DATABASE graph_project;
\c graph_project;
```

d) Создать схему.

e) Предоставить все привилегии пользователю graph_project:

```
CREATE SCHEMA graph_project;
GRANT ALL PRIVILEGES ON DATABASE graph_project TO
graph_project;
GRANT ALL PRIVILEGES ON SCHEMA graph_project TO graph_project;
ALTER SCHEMA graph_project OWNER S TO graph_project;
```

Примечание – создание остальных баз данных graph_mark, graphui, async_gateway, graph_report, graph_search по аналогии с graph_project.

Также можно воспользоваться скриптом, создав sql-файл с кодом, который приведен ниже (с заменой пароля на свой), и выполнить его командой:

```
sudo -u postgres psql -U postgres -f /путь_к_скрипту:
```

```
-- Создание пользователя и базы данных graph_mark
CREATE USER graph_mark WITH PASSWORD 'password';
CREATE DATABASE graph_mark;
\c graph_mark;
CREATE SCHEMA graph_mark;
GRANT ALL PRIVILEGES ON DATABASE graph_mark TO graph_mark;
GRANT ALL PRIVILEGES ON SCHEMA graph_mark TO graph_mark;
ALTER SCHEMA graph_mark OWNER TO graph_mark;
\c postgres

-- Создание пользователя и базы данных graph_project
```

```
CREATE USER graph_project WITH PASSWORD 'password';
CREATE DATABASE graph_project;
\c graph_project;
CREATE SCHEMA graph_project;
GRANT ALL PRIVILEGES ON DATABASE graph_project TO
graph_project;
GRANT ALL PRIVILEGES ON SCHEMA graph_project TO graph_project;
ALTER SCHEMA graph_project OWNER TO graph_project;
\c postgres

-- Создание пользователя и базы данных graphui
CREATE USER graphui WITH PASSWORD 'password';
CREATE DATABASE graphui;
\c graphui;
CREATE SCHEMA graphui;
GRANT ALL PRIVILEGES ON DATABASE graphui TO graphui;
GRANT ALL PRIVILEGES ON SCHEMA graphui TO graphui;
ALTER SCHEMA graphui OWNER TO graphui;
\c postgres

-- Создание пользователя и базы данных async_gateway
CREATE USER async_gateway WITH PASSWORD 'password';
CREATE DATABASE async_gateway;
\c async_gateway;
CREATE SCHEMA async_gateway;
GRANT ALL PRIVILEGES ON DATABASE async_gateway TO
async_gateway;
GRANT ALL PRIVILEGES ON SCHEMA async_gateway TO async_gateway;
ALTER SCHEMA async_gateway OWNER TO async_gateway;
\c postgres

-- Создание пользователя и базы данных graph_report
CREATE USER graph_report WITH PASSWORD 'password';
CREATE DATABASE graph_report;
\c graph_report;
CREATE SCHEMA graph_report;
GRANT ALL PRIVILEGES ON DATABASE graph_report TO graph_report;
GRANT ALL PRIVILEGES ON SCHEMA graph_report TO graph_report;
ALTER SCHEMA graph_report OWNER TO graph_report;
\c postgres
```

```
-- Создание пользователя и базы данных graph_search
CREATE USER graph_search WITH PASSWORD 'password';
CREATE DATABASE graph_search;
\c graph_search;
CREATE SCHEMA graph_search;
GRANT ALL PRIVILEGES ON DATABASE graph_search TO graph_search;
GRANT ALL PRIVILEGES ON SCHEMA graph_search TO graph_search;
ALTER SCHEMA graph_search OWNER TO graph_search;
\q
```

3.1.3. Порядок разворачивания СУБД Графовой БД (vdc-rk-arangoDB)

3.1.3.1. Общие сведения

Для разворачивания необходимо последовательно выполнить следующие шаги:

a) Создать машину в VMware vCloudDirector с названием vdc-rk-arango на основе ОС Ubuntu 22.04.3 LTS и с характеристиками, приведенными в **Таблице 2-1** настоящего Руководства.

b) Выполнить настройку сетевых интерфейсов, указав IP-адрес 10.10.10.16.

c) Создать отдельного не привилегированного пользователя adduser sysman для выполнения настроек.

d) Добавить пользователя в группу sudo.

3.1.3.2. Настройка и развертывание кластерной версии ArangoDB3 (v3.11.6)

3.1.3.2.1. Настройка

Необходимо выполнить следующие шаги:

a) Перед началом настройки кластера определить место для хранения файлов базы данных, чтобы избежать использования данных кластера при аварийной ситуации, если ArangoDB запустится как одиночная нода. Также необходимо определить место для хранения логов.

Примечание – файлы конфигурации хранятся по пути /etc/arangodb3.

Для этого следует:

1. указать на текущий момент прослушиваемый адрес сервера:

```
[server]
endpoint = tcp://0.0.0.0:8529
#0.0.0.0 позволяет прослушивать и использовать все интерфейсы
на вм
```

Примечание – не следует переопределять все необходимые параметры через

хранящиеся файлы конфигурации, т.к. кластер будет запускаться через Starter. Из-за этого некоторые конфигурационные данные будет необходимо переписывать в юните.

2. создать общий каталог для хранения файлов базы данных, агентов и координатора, и установите необходимые права доступа:

```
mkdir -p /var/lib/arangodb3.cluster
chown -R arangodb:arangodb /var/lib/arangodb3.cluster
chmod -R 0700 /var/lib/arangodb3.cluster
```

3. создать общий каталог для файлов логирования:

```
mkdir -p /var/log/arangodb3
chown -R arangodb:arangodb /var/log/arangodb3
chmod -R 0700 /var/log/arangodb3
```

b) Настроить кластер ArangoDB:

1. создать токен аутентификации для возможности авторизации в кластере:

```
arangodb create jwt-secret -
secret=/etc/arangodb3/arangodb.secret
chmod 400 /etc/arangodb3/arangodb.secret
```

2. скопировать секретный файл на все машины, которые будут являться частью кластера.
3. проверить, что секретный файл является одинаковым на всех машинах;
4. создать юнит-файл systemd для запуска ArangoDB в кластерном режиме на всех машинах:

```
nano /etc/systemd/system/arangodb3-cluster.service
```

Содержимое юнит-файла должно быть следующим:

```
[Unit]
Description=ArangoDB Cluster Starter
After=network.target

[Service]
ExecStart=/usr/bin/arangodb \
    --log.dir=/var/log/arangodb3/ \
    --auth.jwt-secret=/etc/arangodb3/arangodb.secret \
    --starter.data-dir=/var/lib/arangodb3.cluster \
    --starter.join 10.10.10.16
Restart=on-failure
User=arangodb

[Install]
```

```
WantedBy=multi-user.target
```

5. обновить конфигурацию systemd:

```
systemctl daemon-reload
```

c) Запустить кластер:

1. для инициализации кластера следует запустить первую ноду – выполнить команду на любой из машин кластера:

```
systemctl start arangodb3-cluster.service
```

2. после старта первой ноды можете запускать оставшиеся ноды в любой последовательности;

3. проверить логи для подтверждения успешного старта кластер. При проверке будут отображены сообщения, подобные следующим:

```
|INFO| coordinator up and running (version 3.11.6).
component=arangodb pid=25132 type=coordinator

2024-06-28T13:18:26+03:00 |INFO| Your cluster can now be
accessed with a browser at `http://10.10.10.16:8529` or
component=arangodb pid=25132 type=coordinator
```

d) Установить пароль для пользователя root:

1. открыть файл конфигурации arangosh:

```
nano /etc/arangodb3/arangosh.conf
```

2. в секции [server] найти строку authentication = и изменить true на false:

```
[server]

authentication = false
```

3. остановить и снова запустить данную ноду:

```
systemctl stop arangodb3-cluster.service

systemctl start arangodb3-cluster.service
```

4. подключиться к ноде с помощью arangosh:

```
arangosh --server.endpoint tcp://localhost:8529
```

или используя IP-адрес сетевого интерфейса, к которому привязана данная нода:

```
arangosh --server.endpoint tcp://10.10.10.16:8529
```

5. после авторизации в консоли выполнить команду для установки нового пароля:

```
require("@arangodb/users").update("root", "новый_пароль")
```

Примечание – таким образом, без пароля вход в командную строку базы возможен только после инициализации кластера, когда пароль для root еще не установлен.

3.1.3.2.2. Создание пользователей и баз данных через веб-интерфейс

Для создания пользователей и баз данных можно воспользоваться веб-интерфейсом ArangoDB – необходимо выполнить следующие шаги:

a) Открыть браузер и перейдите по ссылке: Ошибка! Недопустимый объект гиперссылки..

b) Войти в Систему под пользователем root.

c) Перейти в раздел «Users» и создать пользователя graph_admin.

d) Перейти в раздел «Databases» и создайть базы данных:

- sfera;
- project_db;
- report_db;

e) Назначить пользователя graph_admin для управления этими базами данных.

3.1.3.2.3. Создание резервных копий и восстановление из них

Для создания резервных копий и восстановления из них необходимо использовать утилиты командной строки, поставляемые в пакете с arangodb3: arangodump и arangorestore.

Пароль от пользователя root (пароль от root пользователя arangodb) запрашивается после вызова данных команд.

Примеры создания резервных копий приведены ниже.

Примеры

1. *Создание резервной копии всех баз данных включая системные коллекции(fullbackup):*

```
arangodump --server.endpoint tcp:// 10.10.10.16:8529 --
server.username root --all-databases true --include-system-
collections --output-directory /path/to/fullbackup/
```

2. *Создание резервной копии конкретной базы данных:*

```
arangodump --server.endpoint tcp:// 10.10.10.16:8529 --
server.username root –server.database sfera --output-directory
/path/to/backup-sfera/
```

3. *Восстановление из резервной копии конкретной базы данных:*

```
arangorestore --server.endpoint tcp://10.10.10.16:8529 --
server.username root --server.database sfera --input-directory
/path/to/backup-sfera/
```

4. *Восстановление из резервной копии всех баз данных:*

```
arangorestore --server.endpoint tcp://10.10.10.16:8529 --
server.username root --create-database true --input-directory
/path/to/backup-sfera/
```

3.1.3.2.4. Автоматизация резервного копирования по крону

Для создания резервной копии 1 раз в сутки в 23:00, например, необходимо выполнить следующие шаги:

a) Добавить запись в крон:

```
00 23 * * *  /path/to/create_Full_ArangoDB_backup.sh >
/path/to/backup_logs/`date +\%Y-\%m-\%d-\%H:\%M:\%S`-
backup.log
```

b) Создать скрипт с данным наполнением по пути /path/to/create_Full_ArangoDB_backup.sh:

```
#! /bin/sh +x
set -e

Arango_endpoint="ssl://10.10.10.16:8529"
Arango_password="password"

Arango_Backup=/path/to/data
Arango_BackupArch=/path/to/archive
Arango_BackupLogs=/path/to/backup_logs/

today=`date +'%d-%b-%y_%H-%M-%S'`
day=`date +'%u'`

export today day Arango_Backup Arango_BackupArch

cd $Arango_Backup

rm -rf $Arango_Backup/*

echo "================================="
echo "================================="
echo "Create full Arango database backup"
echo "================================="
echo "================================="

arangodump \
--server.endpoint $Arango_endpoint \
--server.password "$Arango_password" \
--server.username root \
--all-databases true \
--include-system-collections \
--output-directory "Arango_Full_db_backup_$today"

echo "Backup all db's completed under directory
Arango_Full_db_backup_$today"
```

```
echo "Zip db_dump and cleanup tmp files"

zip -r Full_db_backup_$today.zip *

mv $Arango_Backup/Full_db_backup_$today.zip
$Arango_BackupArch/


rm -rf $Arango_Backup/*


find $Arango_BackupArch/ -type f -name "*.zip" -mtime +7 -exec
rm -f {} \;

find $Arango_BackupLogs/ -type f -name "*backup.log" -mtime +7
-exec rm -f {} \;

echo -e "\n"

echo -e "\n"

echo
"============================================================
============================================================
=="

echo "Pls find Full postgres backup at
$Arango_BackupArch/Full_db_backup_$today.zip"

echo "W A R N I N G: ZIP Files older 7 days will be deleted
from $Arango_BackupArch directory automatically"

echo "Backup completed"

echo
"============================================================
================================================="
```

3.1.4. Порядок разворачивания объектного хранилища S3 (vdc-rk-minio)

Для разворачивания необходимо последовательно выполнить следующие шаги:

a) Создать машину в VMware vCloudDirector с названием vdc-rk-minio на основе ОС Ubuntu 22.04.3 LTS и с характеристиками, приведенными в **Таблице 2-1** настоящего Руководства.

b) Выполнить настройку сетевых интерфейсов, указав IP адрес 10.10.10.19.

c) Создать отдельного не привилегированного пользователя adduser user-worker для выполнения настроек.

d) Добавить пользователя в группу sudo.

3.1.5. Добавление прав sudo пользователю

Для добавления прав необходимо в файле /etc/sudoers.d/user-worker добавить и сохранить следующую строку:

```
user-worker ALL=(ALL) NOPASSWD: ALL
```

3.1.6. Установка одиночного экземпляра MinIO

3.1.6.1. Загрузка и установка MinIO

Для загрузки и установки необходимо последовательно выполнить следующие шаги:

a) Скачать исполняемый файл MinIO:

```
wget https://dl.min.io/server/minio/release/linux-amd64/minio
```

b) Сделать файл исполняемым:

```
chmod +x minio
```

c) Переместить файл в каталог /usr/local/bin/ для удобного запуска:

```
sudo mv minio /usr/local/bin/
```

3.1.6.2. Создание директории для данных и конфигурации

Для создания директории необходимо последовательно выполнить следующие шаги:

a) Создать директорию, где будут храниться данные MinIO:

```
sudo mkdir /media/minio-data
```

b) Создать пользователя и группу для MinIO:

```
sudo useradd -r minio-user -s /sbin/nologin
```

c) Назначить созданному пользователю права на директорию:

```
sudo chown -R minio-user:minio-user /mnt/data
```

3.1.6.3. Настройка MinIO как службы

Для настройки необходимо последовательно выполнить следующие шаги:

a) Создать конфигурационный файл службы:

```
sudo nano /etc/systemd/system/minio.service
```

b) Вставить в файл следующий контент:

```
[Unit]
 Description=MinIO Object Storage
 After=network.target


 [Service]
 User=minio-user
 Group=minio-user
 ExecStart=/usr/local/bin/minio server /media/minio-data --
console-address ":9001"
 Restart=always
```

```
LimitNOFILE=65536
Environment="MINIO_ROOT_USER=admin"
Environment="MINIO_ROOT_PASSWORD=strongpassword"

[Install]
WantedBy=multi-user.target
```

c) Сохранить файл.
d) Закрыть файл.
e) Включить и осуществить запуск службы:

```
sudo systemctl daemon-reload
sudo systemctl enable minio
sudo systemctl start minio
```

3.1.6.4. Проверка статуса MinIO

Для проверки статуса запуска необходимо использовать команду:

```
sudo systemctl status minio
```

3.1.6.5. Доступ к веб-интерфейсу MinIO

После запуска можно открыть веб-браузер и перейти по ссылке: http://10.10.10.19:9001.

Необходимо использовать учетные данные, указанные в файле службы:

- имя пользователя: admin;
- пароль: strongpassword.

3.1.7. Порядок разворачивания FTP-сервера для пакетной загрузки документов (vdc-rk-ftp01)

3.1.7.1. Общие сведения

Для разворачивания необходимо последовательно выполнить следующие шаги:

a) Создать машину в VMware vCloudDirector с названием vdc-rk-tp01 на основе ОС Ubuntu 22.04.3 LTS и с характеристиками, приведенными в **Таблице 2-1** настоящего Руководства.
b) Выполнить настройку сетевых интерфейсов, указав IP-адрес 10.10.10.17.
c) Создать отдельного не привилегированного пользователя adduser user-worker для выполнения настроек.
d) Добавить пользователя в группу sudo.

3.1.7.2. Добавление пользователю прав sudo

Для добавления прав необходимо в файле /etc/sudoers.d/user-worker добавить и сохранить следующую строку:

```
user-worker ALL=(ALL) NOPASSWD: ALL
```

3.1.7.3. Установка vsftpd

Для установки необходимо последовательно выполнить следующие шаги:

a) Установить пакет:

```
sudo apt-get install vsftpd
```

b) Изменить конфигурационный файл /etc/vsftpd.conf:

```
listen=YES
anonymous_enable=NO
local_enable=YES
write_enable=YES
local_umask=022
dirmessage_enable=YES
xferlog_enable=YES
connect_from_port_20=YES
ftpd_banner=Welcome to ftp
chroot_local_user=YES
allow_writeable_chroot=YES
local_root=/home/$USER
user_sub_token=$USER
userlist_enable=YES
userlist_deny=NO
userlist_file=/etc/vsftpd.users
ls_recurse_enable=YES
secure_chroot_dir=/var/run/vsftpd
pam_service_name=vsftpd
rsa_cert_file=/etc/ssl/certs/ssl-cert-snakeoil.pem
rsa_private_key_file=/etc/ssl/private/ssl-cert-snakeoil.key
```

c) Создать файл для хранения пользователей, которым будет разрешен доступ к FTP.

d) Добавить в созданный файл строку с названием FTP-пользователя:

```
echo "ftp-user" >> /etc/vsftpd.users
```

e) Перезапустить сервис vsftpd:

```
sudo systemctl restart vsftpd
```

f) Проверить соответствие статуса работы сервиса статусу «active»:

```
sudo systemctl status vsftpd
```

g) Проверить подключение к FTP-серверу с другого сервера:

```
sudo apt -y install lftp
```

h) Подключиться с использованием команды:

```
lftp -u ftp-user <ftp.server.ip>
```

i) Проверить доступность просмотра содержимого FTP:

```
ls
```

j) Загрузить файл на FTP:

```
put -a test.txt
```

3.1.8. Порядок разворачивания FTP-сервера для загрузки правил (vdc-rk-ftp02)

3.1.8.1. Общие сведения

Для разворачивания необходимо последовательно выполнить следующие шаги:

a) Создать машину в VMware vCloudDirector с названием vdc-rk-aez02 на основе ОС Ubuntu 22.04.3 LTS и с характеристиками, приведенными в **Таблице 2-1** настоящего Руководства.
b) Выполнить настройку сетевых интерфейсов, указав IP-адрес 10.10.10.18.
c) Создать отдельного не привилегированного пользователя adduser user-worker для выполнения настроек.
d) Добавить пользователя в группу sudo.

3.1.8.2. Добавление пользователю прав sudo

Для добавления прав необходимо в файле /etc/sudoers.d/user-worker добавить и сохранить следующую строку:

```
user-worker ALL=(ALL) NOPASSWD: ALL
```

3.1.8.3. Установка vsftpd

Для установки необходимо последовательно выполнить следующие шаги:

a) Установить пакет:

```
sudo apt-get install vsftpd
```

b) Изменить конфигурационный файл /etc/vsftpd.conf:

```
listen=YES
anonymous_enable=NO
local_enable=YES
write_enable=YES
local_umask=022
dirmessage_enable=YES
xferlog_enable=YES
connect_from_port_20=YES
ftpd_banner=Welcome to ftp
chroot_local_user=YES
allow_writeable_chroot=YES
local_root=/home/$USER
user_sub_token=$USER
userlist_enable=YES
```

```
userlist_deny=NO
userlist_file=/etc/vsftpd.users
ls_recurse_enable=YES
secure_chroot_dir=/var/run/vsftpd
pam_service_name=vsftpd
rsa_cert_file=/etc/ssl/certs/ssl-cert-snakeoil.pem
rsa_private_key_file=/etc/ssl/private/ssl-cert-snakeoil.key
```

c) Создать файл для хранения пользователей, которым предоставляется доступ к FTP.
d) Добавить в созданный файл строку с названием FTP-пользователя:

```
echo "ftp-user" >> /etc/vsftpd.users
```

e) Перезапустить сервис vsftpd:

```
sudo systemctl restart vsftpd
```

f) Проверить соответствие статуса работы сервиса статусу active:

```
sudo systemctl status vsftpd
```

g) Проверить подключение к FTP-серверу с другого сервера:

```
sudo apt -y install lftp
```

h) Подключиться с использованием команды:

```
lftp -u ftp-user <ftp.server.ip>
```

i) Проверить доступность просмотра содержимого FTP:

```
ls
```

j) Загрузить файла на FTP:

```
put -a test.txt
```

3.1.9. Порядок разворачивания kubernetes

3.1.9.1. Общие сведения

Для разворачивания необходимо последовательно выполнить следующие шаги:

a) Создать машину в VMware vCloudDirector с названием vdc-rk-k8s01 на основе ОС Ubuntu 22.04.3 LTS и с характеристиками, приведенными в **Таблице 2-1** настоящего Руководства.
b) Выполнить настройку сетевых интерфейсов, указав IP-адрес 10.10.10.12.
c) Создать отдельного не привилегированного пользователя adduser user-worker для выполнения настроек.
d) Добавить пользователя в группу sudo.

3.1.9.2. Подготовка VM к установке kubernetes

3.1.9.2.1. Обновление операционной системы

Необходимо обновить все пакеты Системы до актуальных версий:

```
sudo apt update && sudo apt upgrade -y
```

3.1.9.2.2. Настройка имени хоста и hosts-файла

Для функционирования кластера в нормальном режиме следует настроить уникальное имя хоста для каждого узла.

Также необходимо:

a) Добавить запись в файл /etc/hosts для разрешения имен узлов:

```
sudo nano /etc/hosts
```

b) Добавить следующую строку (изменить IP и имя хоста на необходимые значения):

```
10.10.10.12 vdc-rk-k8s01
10.10.10.13 vdc-rk-k8s02
10.10.10.14 vdc-rk-k8s03
10.10.10.15 vdc-rk-k8s04
```

3.1.9.2.3. Отключение swap

Для временного отключения swap необходимо использовать команду:

```
sudo swapoff -a
```

Для постоянного отключения swap закомментируйте или удалите строку, содержащую swap, в файле /etc/fstab:

```
sudo nano /etc/fstab
```

Примечание – необходимо отключить swap, т.к. kubernetes не поддерживает работу с включенным swap.

3.1.9.3. Настройка сетевых параметров – включение iptables для перенаправления трафика

Для включения необходимо последовательно выполнить следующие шаги:

a) Проверить, что параметры netfilter позволяют пересылать трафик:

```
sudo modprobe br_netfilter
```

b) Создать конфигурационный файл:

```
cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf br_netfilter
EOF
```

c) Включить необходимые параметры:

```
cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf net.bridge.bridge-
nf-call-ip6tables = 1 net.bridge.bridge-nf-call-iptables = 1
EOF
```

d) Принять изменения:

```
sudo sysctl --system
```

Примечание – для kubernetes необходима корректная работа iptables.

3.1.9.4. Установка docker

Для установки необходимо последовательно выполнить следующие шаги:

a) Добавить репозиторий docker:

```
# Add Docker's official GPG key:
sudo apt-get update
sudo apt-get install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -
o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc


# Add the repository to Apt sources:
echo \
  "deb [arch=$(dpkg --print-architecture) signed-
by=/etc/apt/keyrings/docker.asc]
https://download.docker.com/linux/ubuntu \
  $(. /etc/os-release &amp;&amp; echo "$VERSION_CODENAME")
stable" | \
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
```

b) Обновить список пакетов в репозиториях:

```
sudo apt-get update
```

c) Установить необходимые пакеты:

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

d) Добавить пользователя в группу docker:

```
sudo usermod -aG docker $USER
```

Примечание – для kubernetes необходимо наличие контейнерной среды для запуска контейнеров. Можно использовать docker, а также и другие, например, containerd.

3.1.9.5. Настройка конфигурации docker для работы с cgroup

Для корректной работы с kubernetes следует настроить cgroup driver. Для этого необходимо последовательно выполнить следующие шаги:

a) Создать или изменить файл /etc/docker/daemon.json:

```
sudo nano /etc/docker/daemon.json
```

b) Добавить следующие параметры:

```
{
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": {
  "max-size": "100m"
  },
```

```
  "storage-driver": "overlay2"
 }
```

c) Применить изменения и перезапустить docker:

```
sudo systemctl restart docker
```

3.1.9.6. Открытие необходимых портов

Для корректного функционирования кластера kubernetes необходимо открыть следующие порты:

- на master -узле:
  - 6443: API сервер kubernetes;
  - 2379-2380: etcd серверы;
  - 10250: kubelet API;
  - 10251: kube-scheduler;
  - 10252: kube-controller-manager;
- worker -узлах:
  - 10250: kubelet API
  - 30000-32767: NodePort для доступа к сервисам.

Также следует открыть порты в firewall:

```
sudo ufw allow 6443
sudo ufw allow 2379:2380/tcp
sudo ufw allow 10250
sudo ufw allow 10251
sudo ufw allow 10252
 sudo ufw allow 30000:32767/tcp
```

3.1.9.7. Синхронизация времени

Для корректной работы кластера необходимо, чтобы время на всех узлах было синхронизировано.

Для установки NTP необходимо выполнить:

```
sudo apt install -y ntp
sudo systemctl enable ntp sudo systemctl start ntp
```

3.1.9.8. Установка необходимых инструментов

3.1.9.8.1. Проверка утилит

Для проверки установки на сервере необходимых утилит следует выполнить:

```
sudo apt install -y apt-transport-https ca-certificates curl
```

3.1.9.8.2. Настройка доступа по ssh между всеми машинами по ключу

Для настройки необходимо последовательно выполнить следующие шаги:

a) Выполнить генерацию SSH-ключей на управляющей машине – команду для создания SSH-ключей на машине, которая будет использоваться для развертывания:

```
ssh-keygen -t ed25519 -C "коментарий по поводу ключа"
```

Примечание – по умолчанию ключи будут сохранены в файле ~/.ssh/id_ed25519. При создании можно оставить поля пароля пустыми, чтобы не вводить его каждый раз при подключении.

b) Выполнить копирование публичного ключа на все узлы в кластере с использованием команд ssh-copy-id:

```
ssh-copy-id user-worker@10.10.10.12
ssh-copy-id user-worker@10.10.10.13
ssh-copy-id user-worker@10.10.10.14
ssh-copy-id user-worker@10.10.10.14
```

Ожидаемый результат – отображено предложение ввести пароль для каждого узла, после чего публичный ключ будет добавлен в файл ~/.ssh/authorized_keys на каждом узле, а также можно подключаться к узлам по ключам без пароля.

3.1.9.8.3. Проверка подключения по SSH

Для проверки возможности подключения к каждому узлу без ввода пароля после копирования ключей необходимо выполнить:

```
ssh user-worker@10.10.10.14
```

В случае успешного подключения можно выйти с использованием команды exit и повторить действие для остальных узлов.

3.1.9.8.4. Проверка обратного подключения между узлами

Для проверки возможности подключения узлов друг к другу можно повторить процесс генерации ключей и копирования на каждом узле или в ручном режиме добавить публичные ключи каждого узла в файл ~/.ssh/authorized_keys на других узлах.

3.1.9.8.5. Добавление прав sudo пользователю

Для добавления прав необходимо в файле /etc/sudoers.d/user-worker добавить и сохранить следующую строку:

```
user-worker ALL=(ALL) NOPASSWD: ALL
```

3.1.9.8.6. Иные условия

Необходимо повторить все вышеуказанные настройки на каждой VM под k8s кластер.

3.1.9.9. Установка kubernetes

3.1.9.9.1. Подготовка виртуальных машин и создание *SnapShot*

Перед началом развертывания кластера необходимо проверить соответствие VM следующим требованиям:

- на всех машинах установлена ОС Ubuntu 22.04;
- серверы настроены и выполнено отключение swap;
- выполнена настройка сеть;
- обеспечен доступ по SSH с одного узла на другие.

Внимание! Перед началом установки следует сделать *SnapShot* всех виртуальных машин через vCloud Director. Для этого необходимо последовательно выполнить следующие шаги:

a) Зайти в интерфейс vCloud Director.
b) Перейти к требуемой виртуальной машине.
c) Выбрать Create Snapshot.
d) Проверить, что созданный SnapShot можно использовать для восстановления в случае возникновения проблем.
e) Повторить процесс для всех машин, которые будут участвовать в кластере.

3.1.9.9.2. Подготовка машины для развертывания Kubespray

Для установки kubernetes с использованием kubespray необходимо наличие одной машины (локальная или отдельная VM), с которой будет выполняться развертывание.

3.1.9.9.3. Установка необходимых утилит на машине для развертывания

Для установки необходимо выполнить:

```
sudo apt update &amp;&amp; sudo apt install -y python3-pip git
sshpass
```

3.1.9.9.4. Клонирование репозитория kubespray

Для клонирования репозитория kubespray необходимо выполнить:

```
git clone https://github.com/kubernetes-sigs/kubespray.git
 cd kubespray
```

3.1.9.9.5. Установка зависимостей

Для установки требуемых Python-зависимостей необходимо выполнить:

```
sudo pip3 install -r requirements.txt
```

3.1.9.9.6. Настройка инвентаря

Для настройки необходимо скопировать пример инвентарного файла, а также выполнить его настройку:

```
cp -rfp inventory/sample inventory/rks-cluster
```

3.1.9.9.7. Генерация файла hosts.yaml

Необходимо использовать скрипт inventory builder для автоматического создания конфигурационного файла:

```
declare -a IPS=(10.10.10.12 10.10.10.13 10.10.10.14
10.10.10.15)

 CONFIG_FILE=inventory/rks-cluster/hosts.yaml python3
contrib/inventory_builder/inventory.py ${IPS[@]}
```

Также следует в ручном режиме выполнить настройку роли узлов (например, master node и worker node), отредактировать файл inventory/rks-cluster/hosts.yaml:

```
all:
  hosts:
  vdc-rk-k8s01:
  ansible_host: 10.10.10.12
  vdc-rk-k8s02:
  ansible_host: 10.10.10.13
  vdc-rk-k8s03:
  ansible_host: 10.10.10.14
  vdc-rk-k8s04:
  ansible_host: 10.10.10.15
  children:
  kube_control_plane:
  hosts:
  vdc-rk-k8s01:
  kube_node:
  hosts:
  vdc-rk-k8s02:
  vdc-rk-k8s03:
  vdc-rk-k8s04:
  etcd:
  hosts:
  vdc-rk-k8s01:
  k8s_cluster:
  children:
  kube_control_plane:
  kube_node:
  calico_rr:
  hosts: {}
```

3.1.9.9.8. Настройка конфигурации kubespray

Для настройки необходимых параметров следует отредактировать файл конфигурации inventory/rks-cluster/group_vars/all/all.yml:

```
kube_network_plugin: calico
```

3.1.9.9.9. Отключение установки docker в kubespray

В случае, если docker уже установлен, то следует отключить установку docker и зависимостей через kubespray. Для этого необходимо последовательно выполнить следующие шаги:

a) Открыть файл inventory/rks-cluster/group_vars/k8s_cluster/docker.yml.
b) Найти или добавить параметр для передачи kubespray команды не выполнять установку docker, если он уже установлен на узле:

```
docker_skip_install: true
```

3.1.9.9.10. Запуск процесса развертывания

Для запуска необходимо последовательно выполнить следующие шаги:

a) Проверить доступность всех серверов для ansible:

```
ansible all -i inventory/rks-cluster/hosts.yaml -m ping
```

b) Выполнить команду:

```
ansible-playbook -i inventory/rks-cluster/hosts.yaml --become
--become-user=user-worker cluster.yml
```

Примечание – опция become позволяет Ansible выполнять команды с привилегиями root.

3.1.9.9.11. Проверка состояния кластера

Для проверки корректности работы кластера после завершения установки необходимо последовательно выполнить следующие шаги:

a) Подключиться к master node:

```
ssh user-worker@vdc-rk-k8s01
```

b) Выполнить команду для проверки состояния узлов:

```
kubectl get nodes
```

Ожидаемый результат – вывод списка из четырех узлов, что подтверждает факт разворачивания kubernetes.

c) Можно проверить, что все компоненты Системы установлены и работают, с использованием команды:

```
kubectl get pod -A
```

Ожидаемый результат – вывод список всех подов Системы во всех namespace и статус у каждого пода – Running.

3.1.9.10. Установка local-path StorageClass

Для использования local-path необходимо проверить local-path Storagев в кластере. Для этого необходимо выполнить следующие шаги:

a) Создайть манифест local-path-storage.yaml:

```
apiVersion: storage.k8s.io/v1
 kind: StorageClass
 metadata:
  name: local-path
 provisioner: rancher.io/local-path
 reclaimPolicy: Retain
 volumeBindingMode: WaitForFirstConsumer
```

b) Применить его:

```
kubectl apply -f local-path-storage.yaml
```

3.2. Развертывание компонентов системы

3.2.1. Развертывание компонента Airflow

3.2.1.1. Установка helm chart для Airflow

Необходимо добавить репозиторий helm и обновить его:

```
helm repo add apache-airflow https://airflow.apache.org
helm repo update
```

3.2.1.2. Создание Namespace

Для создания отдельного namespace для Airflow необходимо выполнить:

```
kubectl create namespace airflow
```

3.2.1.3. Метка для конкретной ноды

Для установки метки на ноду, чтобы назначать на неё поды Airflow необходимо выполнить:

```
kubectl label nodes vdc-rk-k8s04 airflow-node=true
```

3.2.1.4. Настройка values.yaml для Airflow

Для настройки хранения данных и привязке к ноде необходимо создать файл values.yaml:

```
# values.yaml

 # Настройка базы данных
 postgresql:
   enabled: true
   postgresqlPassword: airflow

 # Настройка Redis (для Celery Executor)
 redis:
   enabled: true
```

```
# Использование Celery Executor
executor: CeleryExecutor

# Настройка администратора пользователя Airflow
airflow:
  users:
    - username: admin
      password: admin
      role: Admin

# Настройка хранилища данных (PV)
persistence:
  enabled: true
  size: 10Gi
  accessMode: ReadWriteOnce
  storageClass: local-path

# Настройка привязки подов к конкретной ноде
nodeSelector:
  airflow-node: "true"

# Настройка хранилища DAGs (PVC с local-path)
dags:
  persistence:
    enabled: true
    existingClaim: ""
    size: 5Gi
    storageClass: local-path
    accessMode: ReadWriteOnce

# Настройка внешнего доступа (Ingress)
ingress:
  enabled: true
  web:
    annotations:
      nginx.ingress.kubernetes.io/rewrite-target: /
    hosts:
      - airflow.svc.cluster.local
    tls: []
```

3.2.1.5. Установка Airflow с использованием helm

После создания файла values.yaml можно выполнить установку Airflow:

```
helm install airflow apache-airflow/airflow -n airflow -f
values.yaml
```

3.2.1.6. Проверка развертывания

Необходимо проверить запуск и работу подов:

```
kubectl get pods -n airflow
```

3.2.1.7. Доступ к веб-интерфейсу Airflow

После установки Airflow можно получить доступ к веб-интерфейсу через Ingress по ссылке: http://airflow.svc.cluster.local.

Если Ingress не работает, то следует использовать порт-форвардинг для временного доступа:

```
kubectl port-forward svc/airflow-web 8080:8080 -n airflow
```

3.2.2. Развертывание Keycloak

3.2.2.1. Общие сведения

Для установки необходимо выполнить следующие шаги:

a) Создать namespace и секреты для Keycloak.
b) Настроить PersistentVolume (PV) и PersistentVolumeClaim (PVC) для хранения данных Keycloak.
c) Создать базу данных PostgreSQL для хранения данных Keycloak.
d) Установить Keycloak, используя helm, с учетом зависимости от базы данных.
e) Настроить доступ к Keycloak через NodePort.

Примечания

1. *Keycloak разворачивается как микросервис внутри kubernetes, и его web интерфейс публикуется как nodePort на определенном worker node).*
2. *Для публикации можно использовать Порт 30083 (запрошен для сетевого взаимодействия с АРМ администратора / инженера ИИ).*

3.2.2.2. Создание Namespace и секретов

Для создания Namespace необходимо выполнить:

```
# keycloak-namespace.yaml
apiVersion: v1
kind: Namespace
metadata:
  name: keycloak
```

```
---
# keycloak-secret.yaml
apiVersion: v1
kind: Secret
metadata:
  name: keycloak-db-secret
  namespace: default
type: Opaque
data:
      DB_USER: cG9zdGdyZXM=  # base64("postgres")
      DB_PASSWORD: cGFzc3dvcmQ=  # base64("password")
```

Для создания секретов необходимо выполнить:

```
kubectl apply -f keycloak-namespace.yaml
kubectl apply -f keycloak-secret.yaml
```

3.2.2.3. Настройка PV и PVC для Keycloak

Необходимо выполнить:

```
# keycloak-pv.yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: keycloak-logs-pv
spec:
  capacity:
    storage: 5Gi
  accessModes:
  - ReadWriteOnce
  hostPath:
    path: /var/log/keycloak/
---
# keycloak-pvc.yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: keycloak-logs-pvc
  namespace: default
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 5Gi
```

Для создания PV и PVC необходимо выполнить:

```
kubectl apply -f keycloak-pv.yaml
kubectl apply -f keycloak-pvc.yaml
```

3.2.2.4. Настройка PostgreSQL для Keycloak

В имеющемся Postgresql-сервер необходимо создать базу данных для Keycloak и пользователя, который обладает полными правами на указанную базу данных с помощью dbeaver или psql.

3.2.2.5. Создание Deployment для Keycloak

Необходимо выполнить:

```
# keycloak-deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: keycloak
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: keycloak
  template:
    metadata:
      labels:
        app: keycloak
    spec:
      containers:
      - name: keycloak
        image: quay.io/keycloak/keycloak:latest
        args: ["start", "--optimized", "--log=console,file",
"--log-file=/opt/keycloak/logs/"]
        env:
        - name: KEYCLOAK_ADMIN
          valueFrom:
            configMapKeyRef:
              name: keycloak-config
              key: KEYCLOAK_USER
        - name: KEYCLOAK_ADMIN_PASSWORD
          valueFrom:
            configMapKeyRef:
              name: keycloak-config
              key: KEYCLOAK_PASSWORD
        - name: DB_USER
          valueFrom:
            secretKeyRef:
              name: keycloak-db-secret
              key: DB_USER
        - name: DB_PASSWORD
          valueFrom:
            secretKeyRef:
              name: keycloak-db-secret
```

```
              key: DB_PASSWORD
        ports:
        - containerPort: 8080
          name: http
        volumeMounts:
        - name: keycloak-data
          mountPath: /opt/keycloak/data
        - name: keycloak-logs
          mountPath: /opt/keycloak/logs
      volumes:
      - name: keycloak-data
        emptyDir: {}
      - name: keycloak-logs
        persistentVolumeClaim:
          claimName: keycloak-logs-pvc
```

Для запуска Keycloak необходимо выполнить:

```
kubectl apply -f keycloak-deployment.yaml
```

3.2.2.6. Создание NodePort Service для Keycloak

Необходимо выполнить:

```
# keycloak-service.yaml
apiVersion: v1
kind: Service
metadata:
  name: keycloak
  namespace: default
spec:
  selector:
    app: keycloak
  ports:
  - protocol: TCP
    port: 8080
    targetPort: 8080
    nodePort: 30083
  type: NodePort
```

Для создания NodePort Service необходимо выполнить:

```
kubectl apply -f keycloak-service.yaml
```

3.2.2.7. Создание ConfigMap для Keycloak

Необходимо добавить ConfigMap с настройками для Keycloak:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: keycloak-config
  namespace: default
data:
  KEYCLOAK_USER: admin
  KEYCLOAK_PASSWORD: admin
```

Для ConfigMap необходимо выполнить:

```
kubectl apply -f keycloak-configmap.yaml
```

3.2.2.8. Включение ведения журнала в файл

По умолчанию запись в файл отключена. Для включения необходимо в файле юниты systemd /etc/systemd/system/keycloak.service добавить параметры к запуску Keycloak. Тоже самое можно сделать в deployment kubernetes:

```
--log="console,file"
```

Примечание – файл журнала с именем keycloak.log создается в каталоге data/log вашей установки Keycloak.

3.2.2.9. Настройка расположения и имени файла журнала

Для изменения места создания файла журнала и его имени необходимо выполнить следующие шаги:

a) Создать каталог, доступный для записи, для хранения файла журнала.

Примечание – если каталог не доступен для записи, то Keycloak запускается правильно, но выдаст ошибку, и файл журнала не будет создан.

b) Добавить параметры в юнит-файл или в deployment keycloak:

```
--log="console,file" --log-file=/var/log/keycloak/keycloak.log
```

Примечание – предварительно необходимо создать папку в /var/log/.

3.2.2.10. Настройка уровня журнала файлов

Уровень журнала для обработчика файлового журнала можно указать с помощью свойства --log-file-level следующим образом:

```
--log-file-level=warn
```

3.2.2.11. Настройка логирования в Keycloak, развернутым в Kubernetes

Для изменения конфигурации логирования Keycloak необходимо выполнить следующие шаги:

a) Добавить в параметры запуска Keycloak флаги для включения логирования в файл и указания пути:

```
--log="console,file" --log-file=/var/log/keycloak/keycloak.log
```

b) Проверить, что в Deployment Keycloak настроен volume для хранения логов, доступный извне.

Пример приведен ниже.

Пример

```
volumes:
  - name: keycloak-logs
    hostPath:
      path: /data/logs/keycloak
      type: DirectoryOrCreate

containers:
  - name: keycloak
    volumeMounts:
      - name: keycloak-logs
        mountPath: /var/log/keycloak
```

Ожидаемый результат – файл keycloak.log находится на узле kubernetes в директории /data/logs/keycloak.

3.2.2.12. Настройка MaxPatrol для доступа к файлу логов

Для MaxPatrol необходимо организовать сбор файла журнала через общий доступ к файлам (NFS/SMB).

Необходимо выполнить:

a) Настроить общий доступ к папке /data/logs/keycloak через NFS или SMB – установить NFS-сервер или настроить экспорт папки.

Пример приведен ниже.

Пример

Для экспорта для NFS необходимо:

- *в файле /etc/exports добавить:*

```
/data/logs/keycloak *(rw,sync,no_subtree_check)
```

- *применить изменения:*

```
exportfs -arv
```

b) Настройть MaxPatrol для подключения к папке через NFS:
   1. Добавить точку монтирования в MaxPatrol:

```
mount -t nfs <kubernetes_node_ip>:/data/logs/keycloak
/mnt/keycloak-logs
```

2. Указать в настройках MaxPatrol путь к файлу лога:

```
/mnt/keycloak-logs/keycloak.log
```

3.2.3. Дополнительные настройки – привязка компонентов к ноде

В случае необходимости привязки в values.yaml только определенных компонентов (например, только веб-сервер или worker), следует использовать nodeSelector или affinity в разделах web, scheduler, worker:

```
web:
   nodeSelector:
     airflow-node: "true"


 worker:
   nodeSelector:
     airflow-node: "true"
```

Также необходимо выполнить команду:

```
helm upgrade airflow apache-airflow/airflow -n airflow -f
values.yaml
```

3.2.4. Развертывание компонента интерфейса ИОИ

3.2.4.1. Компоненты, которые должны быть развернуты в kubernetes

Компоненты, которые должны быть развернуты в kubernetes, приведены в **Таблице 3-1** настоящего Руководства.

Таблица 3-1 *– Компоненты, которые должны быть развернуты в kubernetes*

Name	Namespace
backend-api	rks-ioi
backend-auth	rks-ioi
backend-mark	rks-ioi
backend-metadata-service	rks-ioi
backend-project	rks-ioi
backend-report-service	rks-ioi
backend-s3-adapter	rks-ioi
backend-search	rks-ioi

Name	Namespace
graph-ui	rks-ioi

Каждый микросервис загружается в kubernetes с помощью docker-образа микросервиса и yaml файла с описанием объектов данного сервиса для kubernetes.

Готовые docker-образы находятся в папке /_ _, которые необходимо загрузить в docker registry. Для этого в папке следует запустить скрипт upload_images_to_registry.sh.

Пример скрипта приведен ниже.

Пример

```
#!/bin/bash
docker load < backend-api.tar
docker load < backend-auth.tar
docker load < backend-mark.tar
docker load < backend-metadata-service
docker load < backend-project.tar
docker load < backend-report-service.tar
docker load < backend-s3-adapter.tar
docker load < backend-search.tar
docker load < graph-ui.tar

docker push localhost:5000/backend-api:latest
docker push localhost:5000/backend-auth:latest
docker push localhost:5000/backend-mark:latest
docker push localhost:5000/backend-metadata-service:latest
docker push localhost:5000/backend-project:latest
docker push localhost:5000/backend-report-service:latest
docker push localhost:5000/backend-s3-adapter:latest
docker push localhost:5000/backend-search:latest
docker push localhost:5000/graph-ui:latest
```

3.2.4.2. Развертывание сервисов в kubernetes

Для развертывания необходимо последовательно выполнить следующие шаги:

a) Создать namespace:

```
kubectl create namespace rks-ioi
```

b) Для развертывания всех сервисов запустить команду:

```
kubectl apply -f mirrion/
```

Примечание – в папке ioi/ находятся файлы манифестов для развертывания объектов kubernetes.

c) Через некоторое проверить работу подов всех микросервисов в namespace rks-ioi:

```
kubectl get pod -n rks-ioi
```

Ожидаемый результат – статус Running.

3.2.4.3. Дополнительная инструкция по развертыванию компонентов ИОИ

Список необходимых сервисов приведен ниже:

- *Backend-сервисы:*
  - *Backend-api;*
  - *Backend-mark;*
  - *Backend-metadata;*
  - *Backend-project;*
  - *Backend-search;*
  - *Backend-s3-adapter;*
  - *Backend-auth;*
- *Frontend-сервис: Graph-ui.*

Пример файлов конфигурации и рекомендуемые значения параметров для манифестов в k8s приведены ниже.

Пример

- *специфические параметры при настройке ingress отсутствуют;*
- *при инициализации проекта необходимо запускать сервисы в следующей последовательности: маркировок, метаданных, API, далее последовательность не имеет значение;*
- *примерное потребление ресурсов приведено в* ***Таблице 3-2*** *настоящего Руководства.*

Таблица 3-2 *– Примерное потребление ресурсов*

Сервис	Мин. ОЗУ	Мин.ЦПУ
Backend-mark	0.5Gb	0.3m
Backend-api	1Gb	0.3m
Backend-metadata	1Gb	0.3m
Backend-project	2Gb	0.3m
Backend-search	1Gb	0.3m

Сервис	Мин. ОЗУ	Мин.ЦПУ
Graph-websocket	0.5Gb	0.3m
Backend-s3-adapter	1Gb	0.3m
Backend-auth	0.5Gb	0.3m
Graph-ui	256Mb	0.1m

При установке сервисов в Linux как демонов (daemon), необходимо. выгрузить jar-файлы и создать юниты, в которых будет описан их запуск/перезапуск.

Пример юнита приведен ниже.

Пример

```
[Unit]
Description=Java App Ioi
[Service]
Environment="JASYPT_ENCRYPTOR_PASSWORD=graph-backend"
EnvironmentFile=/path/to/envfile
 #путь к env файлу для каждого сервиса свой
ExecStart=/usr/bin/java -jar /path/to/your/app.jar
 #путь к java файлу приложения
User=in-asu-ioi
Restart=on-failure
[Install]
WantedBy=multi-user.target
```

При использовании манифестов в k8s для каждого сервиса должны быть созданы секреты, конфигпамы, сервисы и деплоймент конфигурации. Создание рассмотрено ниже на примере сервиса backend-api:

Примеры

1. ***Манифест configmap.yaml:***

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: backend-api-config
  namespace: rks-dev
data:
  STORE_PASSWORD: "changeit"
  DATABASE_URL: "jdbc:postgresql://10.10.10.10:5432/graphui"
  DATABASE_USERNAME: "graphui"
  PROJECT_NAME: "VTB"
```

```
  ARANGO_TIMEOUT: "30000"
  ARANGO_USER: "graph_admin"
  ARANGO_MAX_CONNECTIONS: "30"
  ARANGO_DATABASE: "graph_db"
  SEARCH_TYPE: "ANALYZER"
  ASYNC_DATA_LIFETIME: "5"
  KAFKA_TRANSACTION_PREFIX: "backend-api-dev-tx"
  GROUP_ID: "backend-api-dev-group"
  ASYNC_RESULT_TOPIC: "result-topic-dev"
  FIND_PATH_TOPIC: "find-path-dev"
  APPLICATION_NAME: "backend-api-dev"
  SWAGGER_URL: "http://10.10.10.10:8081/api"
  ARANGO_GRAPH_MAX_LIFETIME_DAYS: "14"
  COLLECTION_HISTORY_DAYS: "100"
  COLLECTION_HISTORY_CRON: "0 0 */2 * * *"
```

2. ***Манифест deployment.yaml:***

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: backend-api
  labels:
    app: backend-api
  namespace: rks-dev
spec:
  replicas: 1
  strategy:
    type: Recreate
  selector:
    matchLabels:
      app: backend-api
  template:
    metadata:
      labels:
        app: backend-api
    spec:
      affinity:
        podAntiAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
          - labelSelector:
```

```
              matchExpressions:
              - key: app
                operator: In
                values:
                - backend-api
            topologyKey: "kubernetes.io/hostname"
      tolerations:
      - key: "node.kubernetes.io/disk-pressure"
        operator: "Exists"
        effect: "NoSchedule"
      containers:
      - name: backend-api
        image: 10.10.10.10:30005/rks-docker-
snapshot/graph/ui/backend-api:latest
        ports:
        - containerPort: 8080
        # volumeMounts:
        # - name: keystore-volume
        #   mountPath: "/etc/keystore"
        #   readOnly: true
        # - name: ssl-volume
        #   mountPath: "/etc/pki/ca-trust/extracted/java"
        #   readOnly: true
        resources:
          limits:
            cpu: "2.5"
            memory: "2048Mi"
          requests:
            cpu: "1.4"
            memory: "1024Mi"
        livenessProbe:
          httpGet:
            path: /actuator/health
            port: 8080
          initialDelaySeconds: 60
          periodSeconds: 20
          timeoutSeconds: 10
          failureThreshold: 5
        readinessProbe:
          httpGet:
            path: /actuator/health
```

```
            port: 8080
          initialDelaySeconds: 30
          periodSeconds: 20
          timeoutSeconds: 10
          successThreshold: 1
        envFrom:
        - configMapRef:
            name: backend-api-config
        - configMapRef:
            name: common-setting-config
        - secretRef:
            name: backend-api-secret
      # volumes:
      # - name: keystore-volume
      #   secret:
      #     secretName: keystore-secret
      # - name: ssl-volume
      #   secret:
      #     secretName: ssl-secret
```

3. ***Манифест secret.yaml:***

```
apiVersion: v1
kind: Secret
metadata:
  name: backend-api-secret
  namespace: rks-dev
type: Opaque
stringData:
  DATABASE_PASSWORD: "strong_password_for_example_12332145"
  ARANGO_PASSWORD: "strong_password_for_example_12332145"
  JASYPT_ENCRYPTOR_PASSWORD: "graph-backend"
```

4. ***Манифест srvc.yaml:***

```
apiVersion: v1
kind: Service
metadata:
  name: backend-api-service
  namespace: rks-dev
  labels:
    app: backend-api
spec:
  selector:
```

```
    app: backend-api
  ports:
  - protocol: TCP
    port: 9090
    nodePort: 31001
    targetPort: 8080
  type: NodePort
```

5. ***Манифест с общими настройками для всех сервисов:***

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: common-setting-config
  namespace: rks-dev
data:
  SSL_ENABLE: "FALSE"
  STORE_PATH: "/etc/keystore/keystore.p12"
  STORE_PASSWORD: "changeit"
  TRUST_STORE_PATH: "/etc/keystore/keystore.p12"
  TRUST_STORE_PASSWORD: "changeit"
  KEY_ALIAS: "graph"

  PORT: "8080"

  DATABASE_TIMEOUT: "30000"
  DATABASE_MAX_CONNECTIONS: "15"

  KEYCLOAK_URL: "http://10.10.10.10:8083"
  KEYCLOAK_REALM: "Graph"
  KEYCLOAK_RESOURCE: "graph-backend"
  KEYCLOAK_ISSUER_URI: "http:// 10.10.10.10:8083/realms/Graph"
  KEYCLOAK_REDIRECT_URL: "http://10.10.10.10:8081"
  KEYCLOAK_RESOURCE_SECRET: "9GRkr0ZJEXnSPp4ygpJswHDvZm7aWf7X"
  JASYPT_ENCRYPTOR_PASSWORD: "graph-backend"
  KEYCLOAK_USER: "admin"
  CLIENT_SECRET: "9GRkr0ZJEXnSPp4ygpJswHDvZm7aWf7X"

  KIBANA_ENABLED: "FALSE"

  ARANGO_SSL_ENABLE: "FALSE"
  CONNECTION_TIMEOUT: "30000"
```

```
  READ_TIMEOUT: "30000"
  ALLOW_ALL_CORS: "true"
  ARANGO_HOSTS: "10.10.10.10:8084"
  ASYNC_DATA_LIFETIME: "14"

  SPRING_ACTIVE_PROFILES: "keycloak"
  PROFILE: "keycloak"

  WEBSOCKET_TOPIC: "websocket-dev"
  PREGEL_REGISTER_JOB_TOPIC: "pregel-register-job-dev"
  GRAPH_INFLUENCE_REGISTER_JOB: "graph-influence-register-job-
dev"
  PREGEL_SERVICE_NAME: "pregel-service-dev"
  RESET_TASK_CALLBACK_TOPIC: "reset-task-callback-dev"

  KAFKA_SECURITY_PROTOCOL: "PLAINTEXT"
  KAFKA_TRUST_STORE_LOCATION: ""
  KAFKA_TRUST_STORE_PASSWORD: ""
  KAFKA_KEY_STORE_LOCATION: ""
  KAFKA_KEY_STORE_PASSWORD: ""
  KAFKA_KEY_STORE_TYPE: ""
  KAFKA_TRUST_STORE_TYPE: ""
  BOOTSTRAP_SERVERS: "10.10.10.10:9200"

  URL_BACKEND_API: "http://backend-api-service:9090"
  URL_AUTH_SERVICE: "http://backend-auth-service:9090"
  URL_PROJECT_SERVICE: "http://backend-project-service:9090"
  URL_MARK_SERVICE: "http://backend-mark-service:9090"
  URL_PREGEL_SERVICE: "http://backend-pregel-service:9090"
  URL_METADATA_SERVICE: "http://backend-metadata-service:9090"
  URL_BACKEND_SERVICE: "http://backend-api-service:9090"
  URL_ASYNC_SERVICE: "http://backend-async-gateway-
service:9090"

  GRAPH-CORE-API: "http://backend-api-service:9090"
  KAFKA_CONCURRENCY: "4"
```

Все остальные манифест-файлы для других сервисов находятся в папке поставки РК СВИД – Исходные коды / Пользовательский интерфейс / k8s-manifest.

Сведения о переменных окружения приведены в **Таблице 3-3** настоящего Руководства.

Таблица 3-3 *– Переменные окружения*

Название параметра	Пример значения	Описание
SSL_ENABLE	TRUE/FALSE	Включение SSL шифрования сервера (https протокол)
STORE_PATH	/opt/back/ssl/keystore.p12	Путь к хранилищу, содержащему приватный ключ сервера для SSL протокола и публичные доверенные ключи (сертификаты). Хранилище в формате p12
STORE_PASSWORD	123	Пароль от хранилища приватных/публичных ключей заданных в STORE_PATH
KEY_ALIAS	graph	Алиас ключа, импортированного в хранилище приватных/публичных ключей заданных в STORE_PATH. Ссылка: https://security.stackexchange.com/questions/123944/what-is-the-purpose-role-of-the-alias-attribute-in-java-keystore-files
PORT	8080	Порт, на котором будет запущено приложение (сервер)
DATABASE_TIMEOUT	30000	Таймаут соединения к реляционной базе данных (в мс)

Название параметра	Пример значения	Описание
DATABASE_MAX_CONNECTIONS	30	Максимально возможное количество одновременных соединений к реляционной базе данных
DATABASE_URL	jdbc: postgresql://localhost:6432/graphui	Адрес соединения с реляционной базой данных в формате jdbc (ссылка: https://docs.oracle.com/cd/E17952_01/connector-j-8.0-en/connector-j-reference-jdbc-url-format.html)
DATABASE_USERNAME	user	Пользователь реляционной базы данных, от имени которого будет выполнятся соединение
DATABASE_PASSWORD	123 или зашифрованный ENC (rtuouq+neOBt+BPq1JdyD01cBxmlxhjOfFEj27Hqbys=)	Пароль от пользователя (DATABASE_USERNAME) реляционной базы данных. Может быть в открытом или зашифрованном виде. Шифруется через jasypt (ссылка – https://www.devglan.com/online-tools/jasypt-online-encryption-decryption)
PROJECT_NAME	ORG	Наименование заказчика проекта. У каждого может быть свой. Влияет на наполнение базы данных. Под каждого заказчика – отдельное наполнение, зависящее от этого параметра

Название параметра	Пример значения	Описание
PROFILE	sfera	Режим запуска приложения
ARANGO_HOSTS	10.56.165.64:8529	TCP адрес сервера arangodb. Формат – host:port
ARANGO_TIMEOUT	30000	Таймаут жизни соединения к базе данных arangodb
ARANGO_USER	graph_admin	Пользователь, от которого будут выполнятся запросы в arangodb
ARANGO_PASSWORD	123 или ENC (DzaG9SZjqX0VEzFPnGycdMXm5Ym9tHh19hH+j/IoPec=)	Пароль от пользователя (ARANGO_USER) arangodb. Может быть в открытом или зашифрованном виде. Шифруется через jasypt (ссылка – https://www.devglan.com/online-tools/jasypt-online-encryption-decryption)
ARANGO_MAX_CONNECTIONS	30	Максимальное количество одновременных соединений к arangodb
ARANGO_DATABASE	graph_db	Название базы данных, в которой будут выполняться запросы в arangodb
ARANGO_SSL_ENABLE	true/false	Включение или отключение защищенного (ssl) соединения к arangodb
TRUST_STORE_PATH	/opt/back/ssl/keystore.p12	Путь к хранилищу публичных ключей (сертификатов) для защищенного соединения к arangodb

Название параметра	Пример значения	Описание
TRUST_STORE_PASSWORD	123	Пароль от хранилища публичных ключей (сертификатов) для защищенного соединения к arangodb
SEARCH_TYPE	ANALYZER	Режим работы поисковой строки (при поиске по индексу): – PHRASE – поиск по вхождению строки, только при 100% вхождении с учетом регистра ввода; – ANALYZER – при использовании этого режима ввод токенизируется и происходит поиск с использованием анализатора (в этом случае регистр роли не играет)
JASYPT_ENCRYPTOR_PASSWORD	my-secret	Кодовое слово для шифрования паролей методом jasypt. secret key (по ссылке – https://www.devglan.com/online-tools/jasypt-online-encryption-decryption)
URL_AUTH_SERVICE	https://10.1.1.1:8082	http адрес микросервиса аутентификации. backend-auth
ALLOWED_ORIGINS	*	-
KIBANA_ENABLED	false	Включение/выключение логов в json формате
PREGEL_REGISTER_JOB_TOPIC	pregel-register-job-dev	Название топика kafka для регистрации задач прегеля
PREGEL_SERVICE_NAME	pregel-service-dev	Название сервиса задач прегеля

Название параметра	Пример значения	Описание
ASYNC_VIEW_TOPIC	async-websocket-view-dev	Название топика kafka, в который отправляются прочитанные асинхронные задачи
URL_ASYNC_SERVICE	http://graph-ui-service.oslb-dev02.corp.dev.vtb/api/async-gateway-api	http url сервиса запуска асинхронных задач
WEBSOCKET_TOPIC	websocket_topic	Топик kafka, куда приходят события оповещения пользователя через WebSocket
KAFKA_CONCURRENCY	100	Количество параллельных обработчиков на каждый топик kafka Может быть не установлен, по умолчанию 100
ASYNC_DATA_LIFETIME	14	Время жизни результата асинхронных задач в днях, по истечению будет стерто
GROUP_ID	backend-api-dev-group	Идентификатор группы консьюмеров kafka (должен быть уникальным для каждого сервиса)
KAFKA_TRANSACTION_PREFIX	backend-api-dev-tx	Префикс идентификатора транзакций в kafka (должен быть уникальным для каждого сервиса)
APPLICATION_NAME	backend-search	Наименование микросервиса
RESET_TASK_CALLBACK_TOPIC	reset_topic	Технический топик для сброса отмененных задач; удаляет им статус «отменено»
REPORT_GENERATE_TOPIC	report-generate	Название топика для асинхронной генерации отчетов

Название параметра	Пример значения	Описание
USER_FIO_FIELD	givenName	Наименование поля из JWT токена, из которого сервис отчетов будет извлекать ФИО (можно не заполнять, по умолчанию given_name)
SCREENSHOT_LIMIT	10	Максимальное количество скриншотов для генерации отчета
SPRING_ACTIVE_PROFILES	sfera	Активный профиль микросервиса (sfera/keycloack)
PROJECT_HISTORY_STEPS_NUMBER	100	Максимальное количество шагов истории на проект
PROJECT_HISTORY_CLEAN_ENABLED	true	Включить/выключить удаление лишней истории проекта
PROJECT_HISTORY_CLEAN_CRON	0 0 1 * * ?	Расписание для удаления лишней истории проекта
ENABLE_LIQUIBASE	false	Включение/отключение инструмента миграции данных
DDL_AUTO	false	Проверка схемы данных на соответствие модели (validate – включено, false – отключено)
ARANGO_GRAPH_MAX_LIFETIME_DAYS	7	Максимальное количество дней хранения кэша графа (после будет очищено и запрошено заново)
COLLECTION_HISTORY_DAYS	100	Настройка планировщика на удаление из коллекций из аранги с префиксом _HISTORY, где date_from старше 100 дней

Название параметра	Пример значения	Описание
COLLECTION_HISTORY_CRON	0 0 */2 * * *	Время запуска планировщика в формате крона
FIND_PATH_TOPIC	find-path-dev	Название топика kafka для асинхронных задач поиска пути
ASYNC_RESULT_TOPIC	result-topic-dev	Название топика kafka для «отбивки» по результатам выполнения асинхронной задачи
PREGEL_TOPIC	pregel-topic-dev	Название топика kafka для асинхронных задач прегеля
DATA_IMPORT_TOPIC	data-import-dev	Название топика kafka для асинхронных задач поиска загрузки данных из файлов
BACKEND_REPORT_SERVICE_NAME	backend-report-service-dev	Имя микросервиса генерации отчетов; должно быть уникально, т.к. используется асинхронным шлюзом при отмене задач
GRAPH-CORE-API	http://backend-api-service:9090	Http url сервиса backend-ap
ARANGO_GRAPH_MAX_LIFETIME_DAYS	7	Максимальное количество дней хранения кэша графа (после будет очищено и запрошено заново)

3.2.5. Развертывание микросервиса OCR transform

3.2.5.1. Общие условия

Для развертывания микросервиса необходим установленный docker на виртуальной машине.

3.2.5.2. Установка необходимых пакетов

3.2.5.2.1. Обновление списка пакетов системы

Перед установкой пакетов рекомендуется обновить списки пакетов. Для этого необходимо открыть терминал и выполнить команду:

```
sudo apt update
```

3.2.5.2.2. Установка пакетов

Для установки необходимо выполнить:

```
sudo apt install libreoffice libmagic1 pandoc libpq5 -y
```

3.2.5.2.3. Проверка установки

После установки можно проверить правильность установки пакетов. Для этого используйте команды:

- для LibreOffice:

```
libreoffice --version
```

- для libmagic:

```
dpkg -s libmagic1 | grep Version
```

- для pandoc:

```
pandoc --version
```

- для libpq:

```
dpkg -s libpq5 | grep Version
```

3.2.5.2.4. Сборка проекта

Для сборки проекта необходимо последовательно выполнить следующие шаги:

a) Распаковать архив с репозиторием ocr-transform.
b) Распаковать архив с моделями, которые необходимо положить в папку репозитория build/model.

Примечание – модели: craft_mlt_25k.pth, cyrillic.pth, cyrillic_g2.pth, english_g2.pth, latin.pth.

c) Запустить сборку docker-контейнера:

```
docker save ru.datatech/ocr-transform:latest > ocr-
transform_latest.tar
```

```
#копируем на удаленную машину если нужно, а так можно с
локальной машины

#запушить образ в docker registry

scp -P 23 ocr-transform_latest.tar  user@10.10.10.x:/tmp
```

d) На удаленной машине выполнить:

```
docker load < ocr-transform_latest.tar



#перетегируем наш контейнер

 docker tag ru.datatech/ocr-transform:latest
localhost:5000/ocr-transform:latest

#отправим в docker registry наш контейнер

 docker push localhost:5000/ocr-transform:latest
```

e) Создать необходимые объекты в kubernetes:

1. Создать секрет:

```
kubectl apply -f ocr-transform-secret.yml
```

2. Создать configmap:

```
kubectl apply -f ocr-transform-config.yaml
```

f) Задеплоить в kubernetes:объект deployment:

```
kubectl apply -f ocr-transform-deployment.yaml
```

g) Проверить запуск и работу Системы:

```
kubectl get pod -A
```

Ожидаемый результат – статус Running.

3.2.6. Развертывание сервиса DWH

Для сервиса dwh требуется наличие PostgreSQL (развертывание выполнено на предыдущих шагах).

В архиве с исходными кодами необходимо запустить скрипт для развертывания базы и заполнения справочников под сервис dwh (/db/ivd_dwh_init.sql).

Примечание – в файле указана процедура запуска файла по частям от одного пользователя, далее – часть от другого пользователя.

Также потребуется создание базы данных внутри Arangodb.

Пример скрипта для создания базы приведен ниже.

Пример

```
sudo arangosh   # enter root (arangodb) password



#Create database.
```

```
db._createDatabase("graph_db");
db._createDatabase("project_db");
db._createDatabase("graph_metadata");
db._createDatabase("report_db");
db._createDatabase("graphui");



#Create user
var users = require("@arangodb/users");
users.save("dwh-client", "2KMaKFtA2V6yfZax");



#Grant access to the database created above.
users.grantDatabase("dwh-client", "graph_db");
users.grantDatabase("dwh-client", "project_db");
users.grantDatabase("dwh-client", "graph_metadata");
users.grantDatabase("dwh-client", "report_db");
users.grantDatabase("dwh-client", "graphui");



# list databases
db._databases()



# check connection
arangosh --server.username "dwh-client" --server.database
"graph_db"
```

Для создания базы в PostgreSQL необходимо запустить SQL-скрипт:

a) Создать пользователя и базу данных ivd_dwh:

```
CREATE USER ivd_etl_user WITH PASSWORD '';
CREATE DATABASE ivd_dwh;
\c ivd_dwh;
CREATE SCHEMA ivd_dwh;
GRANT ALL PRIVILEGES ON DATABASE ivd_dwh TO givd_etl_user;
GRANT ALL PRIVILEGES ON SCHEMA ivd_dwh TO ivd_etl_user;
ALTER SCHEMA ivd_dwh OWNER TO ivd_etl_user;
```

b) Создать пользователя и базу данных graph_mark:

```
CREATE USER graph_mark_user WITH PASSWORD '';
CREATE DATABASE graph_mark;
\c graph_mark;
CREATE SCHEMA graph_mark;
```

```
GRANT ALL PRIVILEGES ON DATABASE graph_mark TO
graph_mark_user;
GRANT ALL PRIVILEGES ON SCHEMA graph_mark TO graph_mark_user;
ALTER SCHEMA graph_mark OWNER TO graph_mark_user;
```

c) Создать пользователя и базу данных graph_project:

```
CREATE USER graph_project_user WITH PASSWORD '';
CREATE DATABASE graph_project;
\c graph_project;
CREATE SCHEMA graph_project;
GRANT ALL PRIVILEGES ON DATABASE graph_project TO
graph_project_user;
GRANT ALL PRIVILEGES ON SCHEMA graph_project TO
graph_project_user;
ALTER SCHEMA graph_project OWNER TO graph_project_user;
```

d) Создать пользователя и базу данных graphui:

```
CREATE USER graphui_user WITH PASSWORD '';
CREATE DATABASE graphui;
\c graphui;
CREATE SCHEMA graphui;
GRANT ALL PRIVILEGES ON DATABASE graphui TO graphui_user;
GRANT ALL PRIVILEGES ON SCHEMA graphui TO graphui_user;
ALTER SCHEMA graphui OWNER TO graphui_user;
```

e) Создать пользователя и базу данных graph_metadata:

```
CREATE USER graph_metadata_user WITH PASSWORD '';
CREATE DATABASE graph_metadata;
\c graph_metadata;
CREATE SCHEMA graph_metadata;
GRANT ALL PRIVILEGES ON DATABASE graph_metadata TO
graph_metadata_user;
GRANT ALL PRIVILEGES ON SCHEMA graph_metadata TO
graph_metadata_user;
ALTER SCHEMA graph_metadata OWNER TO graph_metadata_user;
```

3.2.7. Развертывание БЯМ

3.2.7.1. Репозитории с кодом

Необходимо скачать в полученном архиве репозитории с кодом и разархивировать.

3.2.7.2. Сборка и запуск компонентов

Запуск компонентов должен осуществляться путем сборки docker-контейнеров.

3.2.7.3. Задание переменных окружения для получения конфигов из FTP, доступа к БЯМ и БД

Для того, чтобы задать креды для подключения в S3 при запуске компонентов в docker-контейнере необходимо создать .env файл в корневом каталоге с исходными кодами:

```
# Имя пользователя для доступа к FTP
FTP_USERNAME=<username>
# Пароль пользователя для доступа к FTP
FTP_PASSWORD=<password>
# Имя пользователя для доступа к внешней БД системы ИВД
EXT_DB_USERNAME=<username>
# Пароль пользователя для доступа к внешней БД системы ИВД
EXT_DB_PASSWORD=<username>
# Ключ доступа к API большой языковой модели (LLM)
# см. раздел Установка и запуск vLLM
LLM_API_KEY=<secret_key>
# Путь до конфигурационного файла system_check
PATH_CONFIG_SYSTEMCHECK=<ftp://hostame:port/путь/до/файла-настройки-systemcheck.yaml>
# Путь до конфигурационного файла docsplit
PATH_CONFIG_DOCSPLIT=<ftp://hostame:port/путь/до/файла-настройки-docsplit.yaml>
# Путь до каталога хранения лог файлов компонента.
# Не указывайте / в конце пути
# Компонент сохранит в нем файл с именем docsplit.log и system_check.log
# Не указывайте имя файла в пути
PATH_LOG_DIR=</путь/до/каталога/логов>
# Путь до модели - укажите абсолютный путь
PATH_TO_MODEL=</путь/до/модели>
```

3.2.7.4. Запуск компонентов из docker-контейнеров

Необходимо собрать и запустить контейнеры, используя docker-compose файл:

```
docker compose up -d --build
```

3.2.7.5. Запуск и установка vLLM

Необходимо выполнить:

a) Разместить файлы модели из дистрибутива поставки Системы на машине, предназначенной для хостинга vLLM.

Примечание – путь до модели указывается в параметре model при старте vLLM.

b) Установить vLLM на машине, предназначенной для хостинга vLLM.

Внимание! Необходимо установить следующие зависимости в отдельном окружении

python:

```
pip install transformers==4.43.3 vllm==0.5.3.post1
```

c) Установить требуемые драйверы графических процессоров на машину, предназначенную для хостинга vLLM.

Примечание – в случае использования графических процессоров NVIDIA H100 необходимо устанавливать следующие драйверы: `Driver Version: 545.23.08  CUDA Version: 12.3`.

d) Создать для работы vLLM пользовательский сервис linux – vllm.service:

```
nano ~/.config/systemd/user/vllm.service
```

e) Вставить в файл следующее содержимое:

```
[Unit]
Description=vLLM API Server
After=network.target


[Service]
Environment=PATH=/home/dsuser/.local/bin:/usr/local/bin:/usr/b
in:/bin
Type=simple
ExecStart=/usr/bin/python3 -m
vllm.entrypoints.openai.api_server \
    --port 3423 \
    --model <path/to/model> \
    --api-key <sectet_api_key> \
    --tensor-parallel-size 2 \
    --disable-custom-all-reduce \
    --uvicorn-log-level debug
Restart=always
RestartSec=3


[Install]
WantedBy=default.target
```

f) Указать путь до каталога с моделью – <path/to/model> в параметре PATH_TO_MODEL env файла (описание приведено в п. 3.2.7.3 настоящего Руководства, а также в конфигурационном файле system_check.yaml (раздел common_params: llm_connection: model)).

g) Указать ключ для доступа к API vLLM – <sectet_api_key> в параметре LLM_API_KEY env файла (описание приведено в п. 3.2.7.3 настоящего Руководства):

```
chmod 644 ~/.config/systemd/user/vllm.service
systemctl --user daemon-reload
systemctl --user enable vllm.service
```

h) Проверить статус:

```
systemctl --user status vllm.service
```

i) Посмотреть логи:

```
journalctl --user -u vllm.service
```

j) Остановить:

```
systemctl --user stop vllm.service
```

k) Рестарт:

```
systemctl --user restart vllm.service
```

l) Для старта сервиса даже без логина пользователя:

```
loginctl enable-linger $USER
```

3.2.7.6. Проверка работы компонентов

3.2.7.6.1. Проверка docsplit

Для проверки работы компонента Docsplit_consumer необходимо выполнить подключение к БД, используя секреты из .env файла и параметры docsplit.yaml:

```
host: из docsplit.yaml
port: из docsplit.yaml
username: из .env файла
password: из .env файла
database: из docsplit.yaml
```

Также необходимо выполнить запрос:

```
INSERT INTO "event".evnt_rgtr (
    evnt_type_id, doc_id, evnt_context, created_at,
    created_by_code, evnt_type_name
) values (
   null, null,
   '
   {"file_name": "имя_файла_документа.doc",
    "doc_id": null, "ver_id": null,
    "ver_txt": {"content": [
        "Текст документа", "Продолжение текста документа"
        ],
    "doc_title_page": "",
    "doc_structure": "[]"}}
   ', LOCALTIMESTAMP, '', 'full_doc_text_placed'
);
```

Ожидаемый результат – новая запись в event.evnt_rgtr с evnt_type_name = "doc_sectioned" и "doc_sectioned_vector".

3.2.7.6.2. Проверка system_check

Необходимо добавить в таблицу event.evnt_rgtr новое событие (запись):

```
INSERT INTO "event".evnt_rgtr (
```

```
    evnt_type_id, doc_id, evnt_context, created_at,
    created_by_code, evnt_type_name
) values (
   null, null,
   '
   {"file_name": "имя_файла_документа.doc",
    "doc_id": null, "ver_id": null,
    "chunks_list": [
      "Текст документа - чанк 1", " Продолжение текста
документа - чанк 2"
      ],
    "doc_title_page": "",
    "doc_structure": "[]",
    "attr_profile": [{"attribute_id": 1, "profile_id": 0},
{"attribute_id": 2, "profile_id": 0}, {"attribute_id": 3,
"profile_id": 0}, {"attribute_id": 4, "profile_id": 0},
{"attribute_id": 5, "profile_id": 0}]}
   ', LOCALTIMESTAMP, '', 'system_check_request'
);
```

Ожидаемый результат – новая запись в event.evnt_rgtr с evnt_type_name="system_check_event".

3.2.7.7. Дополнительная инструкция по docsplit

3.2.7.7.1. Общие сведения

Компонент предназначен для обработки входящего текста:

- извлечения текста из входящего события;
- разделения текста на части размером (в символах) в заданном диапазоне и формирования массива из частей текста;
- преобразования каждой части из массива частей текста в векторное представление;
- отправки события, содержащего массив с частями текста в установленном формате;
- отправки события, содержащего массив с частями векторного представления текста в установленном формате.

Компонент получает события путем периодического чтения заданной таблицы внешней БД. Компонент возвращает события в виде записей в заданную таблицу внешней БД.

3.2.7.7.2. Состав компонента

Компонент реализован с использованием скриптов python со следующими

зависимостями:

- python==3.12;
- tiktoken==0.7.0;
- fastapi==0.114.1;
- langchain_openai==0.2.1;
- langchain-text-splitters==0.3.0;
- uvicorn==0.30.6;
- PyYAML==6.0.2;
- langchain_community==0.3.0;
- ransformers==4.45.1;
- nltk==3.9.1;
- pandas;
- pymorphy3;
- setuptools;
- scikit-learn; omegaconf;
- dependency-injector;
- psycopg_binary;
- requests==2.32.3;
- pydantic==2.9.1;
- python-dotenv==1.0.1;
- psycopg==3.2.2;
- sqlalchemy==2.0.34;
- fsspec==2024.9.0;
- s3fs==2024.9.0;
- paramiko==3.5.0.

Компонент функционально разделен:

- на подкомпонент чтения и записи в таблицу событий;
- подкомпонент для реализации функции векторизации и разделения текста;
- HTTP-сервер Uvicorn для хостинга подкомпонента с реализацией функции векторизации и разделения текста.

Примечание – подкомпоненты запускаются в параллельных потоках из скрипта main_docsplit.py.

Компонент состоит из исходных кодов, размещенных в следующих скриптах и каталогах:

- src/common_utils;
- src/llm_utils;
- src/empty_log_config.json;
- src/handlers;
- src/routers;
- src/services;
- src/containers.py;
- src/common_utils;
- src/consumer;
- src/main_docsplit.py.

3.2.7.7.3. Формат интерфейсов компонента

Чтение записей о событиях производится при помощи подключаемого модуля Pgq-cleint, поставляемого в виде wheel-сборки вместе с компонентом.

Для корректной работы модуля Pgq-cleint внешняя БД должна иметь схему event с таблицами evnt_rgtr, evnt_offset и evnt_type.

3.2.7.7.4. Входящее событие «Документ распознан»

При обнаружении в таблице БД новой записи с именем события full_doc_text_placed (конфигурируется) Система ожидает формат контекста события (поле evnt_context в таблице событий), сведения о котором приведены в **Таблице 3-4** настоящего Руководства.

***Таблица 3-4** – Контекст события в формате JSON, необходимый для работы компонента*

Имя поля	Тип значения	Описание
ver_txt	dict	Словарь с содержимым документа в формате списка из plain-text, титульной страницей и структурой документа
ver_id	str	ID версии документа

Имя поля	Тип значения	Описание
profile_id	int | null	ID профиля настроек для выполнения фрагментации. Если null, то используется профиль по умолчанию (это id алгоритма – alg_id; для разных алгоритмов разный профиль)

Формат текста, получаемый в поле ver_txt:

```
{
'content': List[str] | None, # Содержимое документа
                              # в формате plain-text
'doc_title_page': str | None, # Содержимое первой (титульной)
                              # страницы документа
                              # (извлекается ocr_transform)
'doc_structure': dict | None, # Структура документа,
извлекается
                              # компонентом ocr_transform
}
```

3.2.7.7.5. Формируемое исходящее событие «Определены фрагменты документа»

Исходящее событие имеет тип doc_sectioned (поле evnt_rgtr.evnt_type_name).

Формат JSON, передаваемый для формирования события «Определены фрагменты документа», приведен в **Таблице 3-5**Error! Reference source not found. настоящего Руководства.

Таблица 3-5 *– Формат JSON, передаваемый для формирования события «Определены фрагменты документа»*

Имя поля	Тип значения	Описание
ver_prt_txt	dict	Фрагменты документа с титульной страницей и структурой документа
profile_id	int	ID профиля настроек для выполнения фрагментации (это id алгоритма; для разных алгоритмов разный профиль)

Имя поля	Тип значения	Описание
status	int	ID статуса исполнения (0 – успешно, > 0 – код ошибки)
error_msg	str | null	Сообщение об ошибке
ver_id	str	ID версии документа

Формат словаря ver_prt_txt с фрагментами документа, формируемый компонентом:

```
{
'chunks_list': {
'chunk_txt': str, # Текст чанка
'chunk_id': str, # Порядковый номер чанка
} , # Список фрагментов текста документа
'doc_title_page': str | None, # Содержимое первой (титульной)
                              # страницы документа
                              # (извлекается ocr_transform)
'doc_structure': str | None, # Структура документа,
извлекается
                              # компонетом ocr_transform
}
```

3.2.7.7.6. Формируемое исходящее событие «Определено векторизированное представление фрагментов документа»

Исходящее событие имеет тип doc_sectioned_vector (поле evnt_rgtr.evnt_type_name).

Формат JSON, передаваемый для формирования события «Определено векторизированное представление фрагментов документа», приведен в **Таблице 3-6** настоящего Руководства.

Таблица 3-6 *– Формат JSON, передаваемый для формирования события «Определено векторизированное представление фрагментов документа»*

Имя поля	Тип значения	Описание
ver_prt_vector	dict	Векторизированное представление частей текста документа
profile_id	int	ID профиля настроек для выполнения фрагментации (это id алгоритма; для разных алгоритмов разный профиль)

Имя поля	Тип значения	Описание
status	int	ID статуса исполнения (0 – успешно, > 0 – код ошибки)
error_msg	str | null	Сообщение об ошибке
ver_id	str	ID версии документа

Формат словаря ver_prt_txt с фрагментами документа, формируемый компонентом:

```
{
'chunks_list': [ # Список векторизированных
                 # фрагментов текста документа
{
'chunk_vector': float, # Векторизированное
                       # представление чанка
'chunk_id': str, # Порядковый номер представления
}
] ,
}
```

3.2.7.7.7. Запуск компонента

Запуск компонента должен осуществляться путем сборки docker-контейнера.

3.2.7.7.8. Задание переменных окружения для передачи требуемых паролей и токенов доступа к FTP, БД

Для того, чтобы задать креды для подключения к FTP, БД а также путь до конфигурационного файла необходимо создать .env файл в каталоге с файлом docker-compose.yml и заполнить его следующими ключами:

```
# Имя пользователя для доступа к FTP
FTP_USERNAME=<username>
# Пароль пользователя для доступа к FTP
FTP_PASSWORD=<password>
# Имя пользователя для доступа к внешней БД системы ИВД
EXT_DB_USERNAME=<username>
# Пароль пользователя для доступа к внешней БД системы ИВД
EXT_DB_PASSWORD=<username>
# Путь до конфигурационного файла docsplit
PATH_CONFIG_DOCSPLIT=<ftp://hostame:port/путь/до/файла-
настройки-
docsplit.yaml>
# Путь до каталога хранения лог файлов компонента.
# Не указывайте / в конце пути
```

```
# Компонент сохранит в нем файл с именем docsplit.log
# Не указывайте имя файла в пути
PATH_LOG_DIR=</путь/до/каталога/логов>
```

3.2.7.7.9. Первоначальная настройка конфигурационного файла

Исходный код конфигурационного файла предоставлен по пути src/config/doscplit.yaml.

Для настройки необходимо последовательно выполнить следующие шаги:

a) Отредактировать конфигурационный файл, задав значения полей, а именно: заполнить поля drivername, host, port, database в разделе external_db: connection согласно описаниям, предоставленным в исходном коде конфигурационного файла.
b) Разместить отредактированный конфигурационный файл в FTP-фолдере по пути, указанному в env-файле.
c) При необходимости отредактировать настройки разделения текста и векторизации:
    1. профиль с ключом 0 удалять не допускается;
    2. добавить новый профиль настройки в разделе profiles (скопировать профиль с ключом 0 как образец);
    3. ключ профиля должен быть целым числом от 1 до 65535;
    4. установить параметр is_default = true для нового профиля;
    5. установить параметр is_default = false для всех остальных профилей;
    6. установить требуемые настройки в разделе config нового профиля.

3.2.7.7.10. Запуск компонента из docker-контейнера

Запуск компонента docsplit допускается только совместно с компонентом system_check с использованием общего конфигурационного файла docker-compse.yml.

Необходимо собрать и запустить контейнеры, используя docker-compose файл:

```
docker compose up -d --build
```

3.2.7.7.11. Проверка работы компонентов

Для проверки работы компонента необходимо последовательно выполнить следующие шаги:

a) Выполнить подключение к БД Системы.
b) Добавить в таблицу event.evnt_rgtr новое событие (запись):

```
INSERT INTO "event".evnt_rgtr (
evnt_type_id, doc_id, evnt_context, created_at,
created_by_code, evnt_type_name
) values (
null, null,
'
{"file_name": "имя_файла_документа.doc",
"doc_id": null, "ver_id": null,
"ver_txt": {"content": [
"Текст документа", "Продолжение текста документа"
```

```
],
"doc_title_page": "",
"doc_structure": "[]"}}
', LOCALTIMESTAMP, '', 'full_doc_text_placed'
);
```

Ожидаемый результат – новая запись в event.evnt_rgtr с evnt_type_name="doc_sectioned" и "doc_sectioned_vector".

3.2.8. Развертывание системы мониторинга

3.2.8.1. Общие свеления

Для развертывания системы мониторинга, состоящей из Prometheus и Grafana, необходимы манифест файлы.

Для этого необходимо последовательно выполнить следующие шаги:

a) В поставке найти архив с компонентами мониторинга в составе:

- docker-образ Grafana, в виде tgz файла;
- docker-образ Prometheus, в виде tgz файла;
- архив с файлами манифестов для развертывания Prometheus, Graphana и файлами jsot dashboards для Grafana.

b) Файлы docker-образов импортировать в локальный репозиторий на компьютере.
c) Залить изменения файлов в локальный nexus docker hosted репозиторий.
d) Установить манифесты в kubernetes с помощью kubectl apply.
e) Выполнить в Grafana импорт файлов dashboards.
f) Проверить работу сервисов:

```
kubectl get pod -n monitoring
```

3.2.8.2. Dashboards для Grafana

3.2.8.2.1. Общие сведения

Необходимые dashboards для Grafana можно взять и загрузить уже в работающую Grafana (Компоненты инфраструктуры → kubernetes → monitoring manifests → Grafana-dashboards).

Ожидаемый результат – список Dashboards в Grafana:

- kubernetes Deployment metrics;
- kubernetes Nodes;
- Uptime за последние 24 часа;
- Node Exporter Full;
- PostgreSQL Exporter.

Примечание – каждая инфопанель (dashboard) содержить в себе графики на основе метрик, получаемых из Prometheus.

3.2.8.2.2. Kubernetes Deployment metrics

На данной инфопанели расположены графики работы кластера kubernetes в целом и в рамках установленных в нем deployments (развернутых/запущенных микросервисов).

С помощью данных графиков можно посмотреть нагрузку, оказываемую развернутым миркосервисом на кластер kubernetes.

3.2.8.2.3. Kubernetes Nodes

Данная инфопанель показывает все серверы Системы, где можно посмотреть нагрузку на каждый сервер отдельно.

3.2.8.2.4. Uptime за последние 24 часа

Инфопанель uptime показывает виджеты по микросервисам только тех компонентов Системы, которые разрабатывались (исключая компоненты инфраструктурные, такие как: Airflow, Keycloak и т.д., и которые развернуты в kubernetes).

Каждый виджет показывает время старта pod микросервиса, вычтенное из текущих даты и времени (разница от момента старта микросервиса до настоящего времени).

3.2.8.2.5. Node Exporter Full

Инфопанель для более детально информации по серверам Системы.

3.2.8.2.6. PostgreSQL Exporter

Инфопанель с мониторингом работы баз данных для PostgreSQL; предназначена для просмотра объемов баз данных, нагрузки и скорости работы PostgreSQL в разрезе каждой БД.

4. Инструкция по сборке исходных кодов системы «Система интеллектуального анализа документов»

4.1. Запуск компонентов непосредственно в среде разработки БЯМ

Для запуска Docsplit необходимо выполнить:

```
uvicorn docsplit_app:app --host 0.0.0.0 --port 9061 --workers
4
```

Для запуска Docsplit_consumer необходимо разместить конфигурационный файл из src/iad_repo/config/docsplit.yaml в бакете S3, а также выполнить:

```
sh start_consumer.sh docsplit_consumer.py
s3://<bucket_name>/path/to/docsplit.yaml logs/docsplit.log
```

Последним аргументом передается путь до лог-файла компонента.

Скрипт start_consumer.sh дополнительно вызывает скрипт start_ssh.sh в котором создается ssh-proxy с подключением к серверу с Postgres.

Для запуска System_check необходимо выполнить:

```
uvicorn system_check_app:app --host 0.0.0.0 --port 9062 --
workers 4
```

Для System_check_consumer необходимо разместить конфигурационный файл из src/iad_repo/config/system_check.yaml в бакете S3, а также выполнить:

```
sh start_consumer.sh system_check_consumer.py
s3://<bucket_name>/path/to/system_check.yaml
logs/system_check.log
```

Последним аргументом передается путь до лог-файла компонента.

Скрипт start_consumer.sh дополнительно вызывает скрипт start_ssh.sh, в котором создается ssh-proxy с подключением к серверу с Postgres.

4.2. Проверка работы компонентов

4.2.1. Общие сведения

Документация по API компонентов Docsplit и System_check находится по адресам:

- http://localhost:9061/docs – Docsplit;
- http://localhost:9062/docs – System_check.

Для проверки работы компонента Docsplit_consumer необходимо выполнить подключение к БД:

```
host: localhost
port: 5432
username: ivd_etl_user
```

```
password: nTxBTAmrMoTrQn4a
database: ivd_dwh
```

Перед подключением к БД должен быть запущен ssh прокси (скрипт start_ssh.sh).

4.2.2. Проверка doc_split_consumer.py

Необходимо добавить в таблицу event.evnt_rgtr новое событие (запись):

```
INSERT INTO "event".evnt_rgtr (
     evnt_type_id, doc_id, evnt_context, created_at,
     created_by_code, evnt_type_name
 ) values (
    3, null,
    '{"profile_id": 1 , "ver_txt": {"content": "<ВСТАВЬТЕ
ТЕКСТ ДОКУМЕНТА>"}}',
    LOCALTIMESTAMP, '', 'full_doc_text_placed'
```

Далее следует ожидать новую запись в event.evnt_rgtr с evnt_type_name="doc_sectioned".

4.2.3. Проверка system_check_consumer.py

Необходимо добавить в таблицу event.evnt_rgtr новое событие (запись):

```
INSERT INTO "event".evnt_rgtr (
     evnt_type_id, doc_id, evnt_context, created_at,
     created_by_code, evnt_type_name
 ) values (
    3, null,
    '[{"attribute_id": 1, "profile_id": 0}]',
    LOCALTIMESTAMP, '', 'req_system_check'
 );
```

Далее следует ожидать новую запись в event.evnt_rgtr с evnt_type_name="system_check_event".

5. Журнал автотестирования системы «Система интеллектуального анализа документов»

Журнал автотестирования Системы заполняется в соответствии с формой, которая представлена в **Таблице 5-1** настоящего Руководства.

Таблица 5-1 *– Форма журнала автотестирования*

Дата тестирования	Наименование теста	Результат теста	Комментарий