**Инструкция для установки программного обеспечения**

«Система управления знаниями AI-Solver»

Страниц: 17

[**1.** **Введение** 3](#_Toc208933941)

[**2.** **Назначение системы** 4](#_Toc208933942)

[**3.** **Установка и настройка** 6](#_Toc208933943)

[**3.1.** **Минимальные требования к системе** 6](#_Toc208933944)

[**3.2.** **Создание общей папки** 7](#_Toc208933945)

[**3.3.** **Установка Docker** 9](#_Toc208933946)

[**3.4.** **Создание кластера** 10](#_Toc208933947)

[**3.5.** **Установка Docker registry** 10](#_Toc208933948)

[**3.6.** **Установка БД** 11](#_Toc208933949)

[**3.7.** **Установка главного модуля** 15](#_Toc208933950)

[**3.8.** **Установка ETL модуля** 16](#_Toc208933951)

[**3.9.** **Установка серверов обработки файлов** 16](#_Toc208933952)

[**3.10.** **Установка LLM модуля** 16](#_Toc208933953)

[**3.11.** **Установка LLM моделей** 16](#_Toc208933954)

# **Введение**

Инструкция предназначена для администраторов системы управления знаниями AI-Solver. Система обеспечивает загрузку данных из источников, обработку (оптическое распознавание текста, перевод аудио файлов в текстовые, извлечение сущностей из текста, хранение, и поиск документов. В инструкции подробно описаны ключевые разделы системы, включая настройку, администрирование, работу с документами, клиентами и мониторинг состояния системы.

Администратор должен иметь понимание как работать со следующими функциями:

* Установка и настройка компонентов AI-Solver
* Настройка вкладок и документов в админ панели
* Настройка схем для коллекций в Apache Solr
* Контроль работы коннекторов в Apache Nifi
* Интеграция с MS AD. Настройка keycloack

Для работы с системой администратору необходимо иметь базовые знания в области работы со следующими программными средствами (ПС):

* Операционная система Ubuntu 22.04;
* Система управления доступом Keycloak
* Облачное хранилище объектов MinIO
* Система Apache Solr;
* Векторное хранилище Qdrant
* Брокеры сообщений RabbitMQ и Apache Kafka
* Docker, Docker Swarm
* Conda, Miniconda

# **Назначение системы**

AI-Solver – это программный продукт, предназначенный для сбора, хранения, анализа и распространения структурированных и неструктурированных знаний внутри организации. Цель системы — повышение эффективности работы и поиска решений за счёт централизованного доступа к корпоративным знаниям, автоматизации обработки данных и интеллектуального анализа информации.

Основные возможности системы:

1. Сбор данных из разнородных источников.

2. Интеллектуальная обработка контента

* + Извлечение текста из документов
  + OCR (распознавание текста в сканированных изображениях и документах)
  + STT (расшифровка аудио и видео в текст)
  + Извлечение сущностей (имена, даты, фамилии, термины из справочников)
  + Классификация документов
  + Генерация сводок с применение AI

3. Хранение и индексация в Apache Solr

4. Удобный доступ через Web-интерфейс, с возможность ограничения доступа в зависимости от роли пользователя

5. Поиск документов по их атрибутам, с использованием полнотекстового поиска и применением сложных условий фильтрации

7. Админ панель, позволяющая гибко настроить отображение документа, фильтры

8. Загрузка и просмотр оригинала документа из источника либо S3 хранилища

9. Возможность строить связи и графы по документам

10. Поиск документов с применением AI, AI-чат по документу

11. Возможность интеграции с Service Desk для предоставления предварительного решения проблемы на основе данных из системы

# **Установка и настройка**

## **Минимальные требования к системе**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Имя машины | Устанавливаемы сервисы | ОС | RAM | Disk | Примечание |
| 1 | suz-01 | registry  db  solr-node-1  zookeper | Ubuntu 22.04 | 16GB | 1TB |  |
| 2 | suz-02 | keycloack  redis  search-backend  search-frontend  service-call-analyzer  view-server | Ubuntu 22.04 | 16GB | 200GB |  |
| 3 | suz-03 | nifi  nifi-registry  zookeeper  kafka  kafka-ui  file-processor  rabbitmq  ocr-server  ocr-worker | Ubuntu 22.04 | 16GB | 200GB |  |
| 4 | suz-04 | stt-server  stt-worker  stt-worker-punctuator  ner-server  ner-worker-1  ner-worker-2  ner-worker-3 | Ubuntu 22.04 | 32GB | 200GB |  |
| 5 | suz-05 | insight-ai-server  insight-ai-indexer  qdrant  redis  rabbitmq  text-gen-web-ui | Ubuntu 22.04 | 16GB | 200GB | Требуется видеокарта для запуска LLM (не менее 24GB VRAM) |

## **Создание общей папки**

На машине suz-01 установите NFS-сервер:

|  |
| --- |
| sudo apt update && sudo apt install nfs-kernel-server -y |

Настройте /etc/exports:

|  |
| --- |
| echo "/share 172.24.1.0/24(rw,sync,no\_subtree\_check,no\_root\_squash)" | sudo tee -a /etc/exports |

Примените изменения:

|  |
| --- |
| sudo exportfs -a  sudo systemctl restart nfs-server |

На остальных машинах установите NFS-клиент:

|  |
| --- |
| sudo apt install nfs-common -y |

Создайте точки монтирования:

|  |
| --- |
| sudo share -p /share |

Настройте /etc/fstab:

|  |
| --- |
| echo "172.24.1.132:/share /share nfs rw,sync,hard,intr 0 0" | sudo tee -a /etc/fstab |

Выполните монтирование:

|  |
| --- |
| sudo mount -a |

Создайте директории:

|  |
| --- |
| sudo mkdir -p /share/volumes/db  sudo mkdir -p /share/volumes/nifi\_data/nifi/database\_repository  sudo mkdir -p /share/volumes/nifi\_data/nifi/flowfile\_repository  sudo mkdir -p /share/volumes/nifi\_data/nifi/content\_repository  sudo mkdir -p /share/volumes/nifi\_data/nifi/provenance\_repository  sudo mkdir -p /share/volumes/nifi\_data/nifi/state  sudo mkdir -p /share/volumes/nifi\_data/nifi/logs  sudo mkdir -p /share/volumes/nifi\_data/nifi/conf  sudo mkdir -p /share/volumes/nifi\_data/nifi/lib  sudo mkdir -p /share/volumes/nifi\_data/registry/flow-storage  sudo mkdir -p /share/volumes/nifi\_data/registry/database  sudo mkdir -p /share/volumes/fp\_data/zookeper  sudo mkdir -p /share/volumes/fp\_data/kafka  sudo mkdir -p /share/volumes/fp\_data/fp/config  sudo mkdir -p /share/volumes/fp\_data/fp/exchange  sudo mkdir -p /share/volumes/fp\_data/fp/temp  sudo mkdir -p /share/volumes/fp\_data/fp/failed  sudo mkdir -p /share/volumes/fp\_data/fp/damaged  sudo mkdir -p /share/volumes/fp\_data/fp/scripts  sudo mkdir -p /share/volumes/fp\_data/fp/logs  sudo mkdir -p /share/volumes/fp\_data/fp/tessdata  sudo mkdir -p /share/volumes/fp\_data/fp/cert  sudo mkdir -p /share/volumes/rabbit\_data  sudo mkdir -p /share/volumes/ocr\_data/exchange  sudo mkdir -p /share/volumes/ocr\_data/tessdata  sudo mkdir -p /share/volumes/stt\_data/stt/exchange/out  sudo mkdir -p /share/volumes/stt\_data/stt/exchange/tmp  sudo mkdir -p /share/volumes/stt\_data/stt/models  sudo mkdir -p /share/volumes/stt\_data/punctuator/models  sudo mkdir -p /share/volumes/stt\_data/punctuator/logs  sudo mkdir -p /share/volumes/ner\_data/redis  sudo mkdir -p /share/volumes/insight\_ai\_data/insight\_ai  sudo mkdir -p /share/volumes/insight\_ai\_data/redis  sudo mkdir -p /share/volumes/insight\_ai\_data/rabbit  sudo mkdir -p /share/volumes/insight\_ai\_data/qdrant  sudo mkdir -p /share/volumes/zk\_data  sudo mkdir -p /share/volumes/search\_redis\_data  sudo mkdir -p /share/volumes/search\_data/log  sudo mkdir -p /share/volumes/registry  find /share/volumes/ -type d -exec sudo chmod 775 {} \;  find /share/volumes/ -type f -exec sudo chmod 664 {} \;  sudo chown -R nobody:nogroup /share/volumes/ |

Копируем модели OCR из дистрибутива:

|  |
| --- |
| cp -r ./models/tessdata/\* /share/volumes/fp\_data/fp/tessdata/ && \  cp -r ./models/tessdata/\* /share/volumes/ocr\_data/tessdata/ |

Копируем настройки LLM:

|  |
| --- |
| sudo mkdir -p /share/volumes/insight\_ai\_data/insight\_ai/configs  cp -r ./configs/insight\_ai/\* /share/volumes/insight\_ai\_data/insight\_ai/configs |

Создадим конфигурационный файл для RabbitMQ:

|  |
| --- |
| echo "consumer\_timeout=20700000" | sudo tee /share/volumes/insight\_ai\_data/rabbit/rabbit.conf  echo "consumer\_timeout=20700000" | sudo tee /share/volumes/rabbit\_data/rabbit.conf |

Загружаем STT модели:

|  |
| --- |
| sudo mkdir -p /share/volumes/stt\_data/stt/models/rubig  wget https://alphacephei.com/vosk/models/vosk-model-ru-0.42.zip -P /share/volumes/stt\_data/stt/models/rubig  unzip -j /share/volumes/stt\_data/stt/models/rubig/vosk-model-ru-0.42.zip -d /share/volumes/stt\_data/stt/models/rubig  rm /share/volumes/stt\_data/stt/models/rubig/vosk-model-ru-0.42.zip  sudo mkdir -p /share/volumes/stt\_data/punctuator/models/  sudo apt-get install git-lfs  git lfs install  git clone <https://huggingface.co/DeepPavlov/rubert-base-cased> /share/volumes/stt\_data/punctuator/models/ |

Копируем конфигурационные файлы для fileprocessor:

|  |
| --- |
| sudo mkdir -p /share/volumes/insight\_ai\_data/insight\_ai/configs  cp -r ./configs/fp/\* /share/volumes/fp\_data/fp/config |

## **Установка Docker**

Скачайте и установите docker:

|  |
| --- |
| 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 cp ~/gpg /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 \    $(. /etc/os-release && echo "${UBUNTU\_CODENAME:-$VERSION\_CODENAME}") stable" | \    sudo tee /etc/apt/sources.list.d/docker.list > /dev/null  sudo apt-get update  sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin -y |

Добавьте пользователя в группу docker

|  |
| --- |
| sudo groupadd docker  sudo usermod -aG docker $USER  newgrp docker |

## **Создание кластера**

Инициализировать новый Docker Swarm-кластер на узле suz-01:

|  |
| --- |
| docker swarm init |

Подключить остальные узлы к Docker Swarm-кластеру с использованием полученного токена:

|  |
| --- |
| docker swarm join --token <токен> |

Добавить метки к узлам:

|  |
| --- |
| docker node update --label-add TAG=suz-01 suz-01  docker node update --label-add TAG=suz-02 suz-02  docker node update --label-add TAG=suz-03 suz-03  docker node update --label-add TAG=suz-04 suz-04  docker node update --label-add TAG=suz-05 suz-05 |

## **Установка Docker registry**

Разверните стэк сервисов на основе конфигурационного файла:

|  |
| --- |
| docker stack deploy -c ./stacks/registry.yaml registry |

Загрузите все образы в docker registry:

|  |
| --- |
| docker load -i ./images/main/search\_backend.tar  docker tag search\_backend:latest suz-01.company.ru/search-backend:latest  docker push suz-01.company.ru:5000/search-backend:latest  docker load -i ./images/main/search\_frontend.tar  docker tag search\_frontend:latest suz-01.company.ru/search-frontend:latest  docker push suz-01.company.ru:5000/search-frontend:latest  docker load -i ./images/main/service\_call\_analyzer.tar  docker tag service\_call\_analyzer:latest suz-01.company.ru/service-call-analyzer:latest  docker push suz-01.company.ru:5000/service-call-analyzer:latest  docker load -i ./images/main/view\_server.tar  docker tag view\_server:latest suz-01.company.ru/view-server:latest  docker push suz-01.company.ru:5000/view-server:latest  docker load -i ./images/etl/file\_processor.tar  docker tag file\_processor:latest suz-01.company.ru/file-processor:latest  docker push suz-01.company.ru:5000/file-processor:latest  docker load -i ./images/llm/insight\_ai.tar  docker tag insight\_ai:latest suz-01.company.ru/insight-ai:latest  docker push suz-01.company.ru:5000/insight-ai:latest  docker load -i ./images/functional/ner.tar  docker tag ner:latest suz-01.company.ru/ner:latest  docker push suz-01.company.ru:5000/ner:latest  docker load -i ./images/functional/ocr\_server.tar  docker tag ocr\_server:latest suz-01.company.ru/ocr-server:latest  docker push suz-01.company.ru:5000/ocr-server:latest  docker load -i ./images/functional/ocr\_worker.tar  docker tag ocr\_worker:latest suz-01.company.ru/ocr-worker:latest  docker push suz-01.company.ru:5000/ocr-worker:latest  docker load -i ./images/functional/stt\_server.tar  docker tag stt\_server:latest suz-01.company.ru/stt-server:latest  docker push suz-01.company.ru:5000/stt-server:latest  docker load -i ./images/functional/stt\_worker.tar  docker tag stt\_worker:latest suz-01.company.ru/stt-worker:latest  docker push suz-01.company.ru:5000/stt-worker:latest  docker load -i ./images/functional/stt\_worker\_punctuator.tar  docker tag stt\_worker\_punctuator:latest suz-01.company.ru/stt-worker-punctuator:latest  docker push suz-01.company.ru:5000/stt-worker-punctuator:latest  docker image prune -a -f |

## **Установка БД**

Разверните стэк сервисов на основе конфигурационного файла:

|  |
| --- |
| docker stack deploy -c ./stacks/db.yaml registry |

Создайте БД search. Выполните SQL

|  |
| --- |
| **CREATE** **SCHEMA** properties **AUTHORIZATION** **"search"**;  **CREATE** **SCHEMA** **"search"** **AUTHORIZATION** **"search"**;  -- properties.caption definition  -- Drop table  -- DROP TABLE properties.caption;  **CREATE** **TABLE** properties.caption (  captiongroup **varchar** **NULL**,  "name" **varchar** **NULL**,  alias **varchar** **NULL**,  id serial4 **NOT** **NULL**  );  **CREATE** **TABLE** properties.descriptions (  id serial4 **NOT** **NULL**,  "table" **varchar** **NULL**,  param **varchar** **NULL**,  "name" **varchar** **NULL**,  description **varchar** **NULL**  );  **CREATE** **TABLE** properties.doc (  **"name"** **varchar** **NOT** **NULL**,  extend **varchar** **NULL**,  hfields **varchar** **NULL**,  ffields **varchar** **NULL**,  titlefields **varchar** **NULL**,  datefields **varchar** **NULL**,  extensionfields **varchar** **NULL**,  metafields **varchar** **NULL**,  attachmentdatabases **varchar** **NULL**,  groupfield **varchar** **NULL**,  id serial4 **NOT** **NULL**,  docnumfields **varchar** **NULL**,  linked **varchar** **NULL**,  subtitle **varchar** **NULL**  );  **CREATE** **TABLE** properties.doc\_group (  **"name"** **varchar** **NOT** **NULL**,  docgroup **varchar** **NULL**,  downloadmask **varchar** **NULL**,  id serial4 **NOT** **NULL**  );  **CREATE** **TABLE** properties.target (  id serial4 **NOT** **NULL**,  target **varchar** **NOT** **NULL**,  databases **varchar** **NULL**,  alloweddatabases **varchar** **NULL**,  printfields **varchar** **NULL**,  filters **varchar** **NULL**,  filterchains **varchar** **NULL**,  titlefields **varchar** **NULL**,  fieldtext **varchar** **NULL**,  roles **varchar** **NULL**,  amountfilters **varchar** **NULL**,  datefilters **varchar** **NULL**,  searchfields **varchar** **NULL**,  pathsfilters **varchar** **NULL**,  commands **varchar** **NULL**,  **CONSTRAINT** target\_pk **PRIMARY** **KEY** (id)  );  **CREATE** **TABLE** "search".ai\_logs (  "uuid" **uuid** **NOT** **NULL**,  username **varchar**(255) **NOT** **NULL**,  department **varchar**(255) **NULL**,  request\_text **text** **NOT** **NULL**,  target **varchar**(255) **NULL**,  response\_answer **text** **NULL**,  request\_timestamp **timestamp** **NOT** **NULL**,  response\_timestamp **timestamp** **NULL**,  processing\_time\_ms **int8** **NULL**,  status **varchar**(20) **NULL**,  error\_message **text** **NULL**,  from\_web **bool** **DEFAULT** **false** **NOT** **NULL**,  request\_type **varchar**(50) **DEFAULT** 'UNKNOWN'::**character** **varying** **NOT** **NULL**,  "cost" **int4** **DEFAULT** 1 **NULL**,  "token" **varchar** **NULL**,  **CONSTRAINT** ai\_logs\_pkey **PRIMARY** **KEY** (uuid)  );  **CREATE** **INDEX** idx\_ai\_logs\_request\_timestamp **ON** **search**.ai\_logs **USING** btree (request\_timestamp);  **CREATE** **INDEX** idx\_ai\_logs\_status **ON** **search**.ai\_logs **USING** btree (status);  **CREATE** **INDEX** idx\_ai\_logs\_username **ON** **search**.ai\_logs **USING** btree (username);  **CREATE** **INDEX** idx\_ai\_logs\_username\_timestamp **ON** **search**.ai\_logs **USING** btree (username, request\_timestamp);  **CREATE** **TABLE** **"search"**.analysis\_tasks (  task\_id **varchar**(255) **NOT** **NULL**,  service\_call\_uuid **varchar**(255) **NOT** **NULL**,  status **varchar**(20) **NOT** **NULL**,  progress **int4** **NULL**,  error\_message **text** **NULL**,  **"result"** **text** **NULL**,  created\_at **timestamp** **DEFAULT** CURRENT\_TIMESTAMP **NOT** **NULL**,  started\_at **timestamp** **NULL**,  completed\_at **timestamp** **NULL**,  **CONSTRAINT** analysis\_tasks\_pkey **PRIMARY** **KEY** (task\_id),  **CONSTRAINT** analysis\_tasks\_progress\_check **CHECK** (((progress >= 0) **AND** (progress <= 100))),  **CONSTRAINT** analysis\_tasks\_status\_check **CHECK** (((status)::**text** = **ANY** ((**ARRAY**['PENDING'::**character** **varying**, 'IN\_PROGRESS'::**character** **varying**, 'COMPLETED'::**character** **varying**, 'FAILED'::**character** **varying**])::**text**[])))  );  **CREATE** **INDEX** idx\_analysis\_tasks\_created\_at **ON** **search**.analysis\_tasks **USING** btree (created\_at);  **CREATE** **INDEX** idx\_analysis\_tasks\_service\_call\_uuid **ON** **search**.analysis\_tasks **USING** btree (service\_call\_uuid);  **CREATE** **INDEX** idx\_analysis\_tasks\_started\_at **ON** **search**.analysis\_tasks **USING** btree (started\_at);  **CREATE** **INDEX** idx\_analysis\_tasks\_status **ON** **search**.analysis\_tasks **USING** btree (status);  **CREATE** **INDEX** idx\_analysis\_tasks\_status\_created\_at **ON** **search**.analysis\_tasks **USING** btree (status, created\_at);  **CREATE** **TABLE** **"search"**.asutd\_roles (  id serial4 **NOT** **NULL**,  **"role"** **varchar** **NULL**,  **CONSTRAINT** asutd\_roles\_pkey **PRIMARY** **KEY** (id)  );  **CREATE** **TABLE** **"search"**.asutd\_roles\_tmp (  kind\_exact\_id **varchar** **NULL**,  organization\_id **varchar** **NULL**,  user\_login **varchar** **NULL**,  user\_domain **varchar** **NULL**  );  **CREATE** **TABLE** **"search"**.history (  id serial4 **NOT** **NULL**,  username **varchar** **NULL**,  params **varchar** **NULL**,  datetime **timestamp** **NULL**,  **CONSTRAINT** history\_pk **PRIMARY** **KEY** (id)  );  **CREATE** **TABLE** **"search"**.ocr\_files (  id serial4 **NOT** **NULL**,  username **varchar** **NULL**,  filename **varchar** **NULL**,  create\_dt **timestamp** **NULL**,  **"token"** **varchar** **NULL**,  file\_status **varchar** **NULL**,  **"result"** **varchar** **NULL**,  **CONSTRAINT** ocr\_files\_pkey **PRIMARY** **KEY** (id)  );  **CREATE** **TABLE** **"search"**.saved\_searches (  **"uuid"** **uuid** **NOT** **NULL**,  username **varchar** **NULL**,  **"name"** **varchar** **NULL**,  query **varchar** **NULL**,  **CONSTRAINT** saved\_searches\_pkey **PRIMARY** **KEY** (**uuid**)  );  **CREATE** **TABLE** **"search"**.stt\_files (  id serial4 **NOT** **NULL**,  username **varchar** **NULL**,  filename **varchar** **NULL**,  create\_dt **timestamp** **NULL**,  **"token"** **varchar** **NULL**,  **"result"** **varchar** **NULL**,  file\_status **varchar** **NULL**,  **CONSTRAINT** stt\_files\_pkey **PRIMARY** **KEY** (id)  );  **CREATE** **TABLE** **"search"**.ticket\_analyze\_tasks (  task\_id **uuid** **NOT** **NULL**,  ticket\_id **varchar**(255) **NOT** **NULL**,  username **varchar**(255) **NOT** **NULL**,  status **varchar**(20) **NOT** **NULL**,  start\_time **timestamp** **NULL**,  end\_time **timestamp** **NULL**,  processing\_time **int8** **NULL**,  **"result"** **text** **NULL**,  error\_message **text** **NULL**,  **CONSTRAINT** pk\_ticket\_analyze\_tasks **PRIMARY** **KEY** (task\_id)  );  **CREATE** **TABLE** **"search"**.translate\_files (  id serial4 **NOT** **NULL**,  username **varchar** **NULL**,  filename **varchar** **NULL**,  create\_dt **timestamp** **NULL**,  **"token"** **varchar** **NULL**,  **"result"** **varchar** **NULL**,  file\_status **varchar** **NULL**,  **CONSTRAINT** translate\_files\_pkey **PRIMARY** **KEY** (id)  );  **CREATE** **TABLE** **"search"**.users (  id serial4 **NOT** **NULL**,  username **varchar** **NULL**,  **CONSTRAINT** users\_pkey **PRIMARY** **KEY** (id)  );  **CREATE** **TABLE** **"search"**.users\_asutd\_roles (  user\_id **int8** **NOT** **NULL**,  asutd\_role\_id **int8** **NOT** **NULL**,  **CONSTRAINT** users\_asutd\_roles\_pkey **PRIMARY** **KEY** (user\_id, asutd\_role\_id),  **CONSTRAINT** asutd\_roles\_fk **FOREIGN** **KEY** (asutd\_role\_id) **REFERENCES** **"search"**.asutd\_roles(id),  **CONSTRAINT** users\_fk **FOREIGN** **KEY** (user\_id) **REFERENCES** **"search"**.users(id)  ); |

## **Установка главного модуля**

Разверните стэк сервисов на основе конфигурационного файла:

|  |
| --- |
| docker stack deploy -c ./stacks/main.yaml registry |

Настройте Keycloack согласно официальной документации на ПО. Настройки реалма экспортируйте из файла ./configs/keycloack/realm-export

Настройте и загрузите коллекции в Apache Solr. Пример настройки коллекции ./examples/solr\_schema/

## **Установка ETL модуля**

Разверните стэк сервисов на основе конфигурационного файла:

|  |
| --- |
| docker stack deploy -c ./stacks/etl.yaml registry |

## **Установка серверов обработки файлов**

Разверните стэк сервисов на основе конфигурационного файла:

|  |
| --- |
| docker stack deploy -c ./stacks/functional.yaml registry |

## **Установка LLM модуля**

Разверните стэк сервисов на основе конфигурационного файла:

|  |
| --- |
| docker stack deploy -c ./stacks/llm.yaml registry |

## **Установка LLM моделей**

Установим LLM модели на машине suz-05.

Загрузите и установите Miniforge3:

|  |
| --- |
| curl -sL "<https://github.com/conda-forge/miniforge/releases/latest/download/Miniforge3-Linux-x86_64.sh>" > "Miniforge3.sh"  bash Miniforge3.sh |

Создайте виртуальное окружение:

|  |
| --- |
| conda create -n textgen python=3.11  conda activate textgen |

Устанановите Pytorch 2.6.0 для GPU Nvidia:

|  |
| --- |
| pip3 install torch==2.6.0 --index-url https://download.pytorch.org/whl/cu124 |

Клонируйте репозиторий text-generation-webui и устанавливите зависимости для GPU Nvidia и CPU с AVX2:

|  |
| --- |
| git clone https://github.com/oobabooga/text-generation-webui  cd text-generation-webui  pip install -r ./requirements/full/requirements.txt |

В директорию моделей (например /home/owner/models) загружаем модели YandexGPT-5-Lite-8B-instruct-Q4\_K\_M.gguf и Vikhr-Nemo-12B-Instruct-R-21-09-24-Q4\_K\_M.gguf

Создайте сервисы:

|  |
| --- |
| sudo nano /etc/systemd/system/textgen\_1.service |

|  |
| --- |
| [Unit]  Description=Text Generation WebUI  After=network.target    [Service]  User=admin\_askit  WorkingDirectory=/home/owner/text-generation-webui  ExecStart=/home/owner/miniforge3/envs/textgen/bin/python server.py --model-dir /home/owner/models --model YandexGPT-5-Lite-8B-instruct-Q4\_K\_M.gguf  --listen --listen-port 5055 --listen-host 0.0.0.0 --api --api-port 5056  Restart=always  Environment="PATH=/home/owner/miniconda3/envs/textgen/bin"    [Install]  WantedBy=multi-user.target |

|  |
| --- |
| sudo nano /etc/systemd/system/textgen\_2.service |

|  |
| --- |
| [Unit]  Description=Text Generation WebUI  After=network.target    [Service]  User=admin\_askit  WorkingDirectory=/home/owner/text-generation-webui  ExecStart=/home/owner/miniforge3/envs/textgen/bin/python server.py --model-dir /home/owner/models –model Vikhr-Nemo-12B-Instruct-R-21-09-24-Q4\_K\_M.gguf --listen --listen-port 5057 --listen-host 0.0.0.0 --api --api-port 5058  Restart=always  Environment="PATH=/home/owner/miniconda3/envs/textgen/bin"    [Install]  WantedBy=multi-user.target |

Включите и запустите сервисы:

|  |
| --- |
| sudo systemctl daemon-reload  sudo systemctl enable textgen\_1  sudo systemctl start textgen\_1  sudo systemctl enable textgen\_2  sudo systemctl start textgen\_2 |