MWS AI Agents Platform

Инструкция по установке экземпляра программного обеспечения

Содержание

О ПРОДУКТЕ

MWS AI Agents Platform – это платформа для создания AI решений, которые автоматизируют обслуживание клиентов через коммуникационные каналы и поддерживают омниканальное взаимодействие. Особенность MWS AI Agents Platform – набор готовых инструментов, не требующих от пользователей опыта в машинном обучении или навыков программирования. Благодаря этому клиенты платформы могут самостоятельно управлять жизненным циклом ботов и AI-агентов, создавать и обслуживать ML-модели и AI-сервисы. В зависимости от варианта поставки платформа MWS AI Agents Platform предоставляет следующие возможности:

- создание сценария бота или агента в удобном no-code-конструкторе;
- обработка логики бота на высокопроизводительном и отказоустойчивом движке;
- подключение ботов к каналам взаимодействия с поверхностями;
- подключение и использование LLM к ботам;
- создание и обучение ML-моделей типов Классификатор и NER;
- разметка данных для формирования и совершенствования обучающих датасетов;
- создание RAG-сервисов для быстрого поиска ответов на вопросы в регулярно обновляемой базе знаний.

Основные понятия

Агент
Автономная система, использующая LLM для сбора данных, анализа, принятия решений и выполнения задач различной сложности.

База знаний RAG
Набор документов, загруженных в систему, используемый RAG для выдачи релевантных ответов пользователю.

Бот
Программа для имитации общения с пользователями через голосовые или текстовые интерфейсы, предназначенная для предоставления информации или выполнения задач.

Веб-клиент MWS AI Agents Platform
Пользовательский веб-интерфейс для работы с платформой через браузер. Позволяет создавать и адаптировать ботов под конкретные требования.

Датасет
Набор данных для обучения, тестирования и оценки ML-моделей.

Движок (agent-engine)
Сервис для обработки запросов пользователей с помощью ботов, созданных в no-code-конструкторе.

Индекс RAG
Хранилище фактологической информации для использования в RAG, содержащее векторную часть (embedding) и текстовое представление, позволяющее проводить быстрый поиск релевантных записей.

Интент
Намерение пользователя, отражённое в поисковом запросе.

Краулинг (crawling)

Процесс автоматического сканирования, обхода и скачивания веб-страниц с помощью специальных сервисов – краулеров. Краулинг используется для сбора информации с веб-сайтов, анализа и индексации контента.

Классификатор (Classifier)
Модель машинного обучения, которая автоматически обучается и оптимизируется для решения задач классификации.

Модель машинного обучения (ML-модель)
Алгоритмическая конструкция для прогнозирования или классификации данных на основе обучения.

Паттерн
Формальное правило, описывающее ключевые слова и выражения для классификации запросов пользователя.

Проект

Совокупность сценариев разработанного бота или агента. Может содержать несколько версий.

Промпт
Подсказка для нейросети о том, что именно от неё требуется.

Рабочий процесс Langflow
Процесс в рамках фреймворка Langflow, интегрирующий несколько компонентов для обработки и генерации текста на основе пользовательских запросов и технологии RAG.

Сценарий
Логика работы навыка, направленная на достижение определённой цели в диалоге с клиентом. В no-code-конструкторе сценарий состоит из нод, связанных ребрами.

AI-сервис
Сервисы, использующие искусственный интеллект для обработки информации и решения задач в различных сферах.

AncSetFit (Anchored SetFit)

Расширение способа обучения ML-моделей SetFit, использующее «якорные» (anchor) примеры для каждого класса, что обеспечивает более стабильное обучение и лучшую генерализацию при экстремально малом количестве данных.

AutoML
Автоматизированный процесс создания и улучшения моделей машинного обучения.

Autoscaling (автомасштабирование)

Автоматическое изменение количества работающих экземпляров (инстансов) модели в зависимости от текущей нагрузки и потребления ресурсов.

Batching (батчинг)

Подход, при котором несколько отдельных запросов объединяются в одну группу (батч), чтобы выполнить их одновременно за одно обращение к модели.

Embedding (Эмбеддинг, вектор)
Векторное представление слова или фразы, полученное из моделей обработки естественного языка.

Fine-tuning

Дообучение предварительно обученной модели на новых данных путём обновления всех или части весов модели и/или добавления новых слоёв для адаптации к специфической задаче.

Langflow
Фреймворк для создания агентов с помощью no-code конструктора.

Large language model (LLM)
Языковая модель, обученная на больших объёмах текста для выполнения задач NLP.

Named Entity Recognition (NER)
Модель машинного обучения для выделения и классификации именованных сущностей в тексте.

Natural Language Processing (NLP)
Область искусственного интеллекта, занимающаяся обработкой и анализом естественного языка.

Retrieval Augmented Generation (RAG)
Технология ML, объединяющая поиск информации и генерацию текста для создания ответов на запросы.

SetFit

Метод эффективного обучения ML-моделей на малом количестве данных, использующий обучение через сравнение примеров из разных классов и последующее обучение классификатора на полученных векторных представлениях (эмбеддингах) без дообучения всей модели.

Slug (слаг)

Уникальный ключ, который представляет собой имя сценария строчными буквами. Вместо пробелов в качестве разделителей используются знаки нижнего подчеркивания.

РАЗВЕРТЫВАНИЕ

В разделе описываются требования к программному обеспечению и ресурсам, необходимым для развертывания платформы MWS AI Agents Platform. Приведены действия по подготовке к развертыванию, порядок установки и конфигурирования.

Требования к аппаратному и программному обеспечению

Требования к ПО

> Гарантируется работа платформы только на указанных версиях компонентов. Если версия не указана, используйте последнюю стабильную.

Компонент	Требование

Кластер Kubernetes	Не ниже версии 1.25.7 и не выше 1.31
	В k8s:
	• nginx ingress controller
	• cert-manager
	• GPU nodes на базе:
	  ○ Nvidia container toolkit версии 1.17.0
	  ○ Nvidia device plugin версии 0.17.0
	  ○ драйверов Nvidia A100 версии 550.144.03
	> Убедитесь, что у ноды есть доступ к карте. Доступ можно организовать через виртуализацию или путем установки на физическом оборудовании
	• Prometheus или VictoriaMetrics
	• OTEL Collector
	• Fluentbit
	Также рекомендуется использовать Persistent Volumes (PV):
	• для сбора метрик системой мониторинга Prometheus. При этом Prometheus может передавать метрики в другую систему для хранения и анализа данных, например VictoriaMetrics;
	• для модели Cotype. Она поставляется в виде большого образа либо образа, который при запуске загружает необходимые модели и требует большего объема временного хранилища на ноде или PV

Платформа потоковой передачи и обработки данных	Confluent Kafka 7.4.1-1 или Kafka версии 3.4.1
Хранение данных	PostrgeSQL 14.12 Redis 7.2.3 (авторизация поддерживается)

Компонент	Требование
	Milvus 2.5.14 (Helm Chart версии 4.2.56) Хранилище S3 (минимум 2ГБ + 1 ГБ на каждую модель классификации) ClickHouse 25.1.3.23
Мониторинг	Prometheus или VictoriaMetrics Grafana не ниже версии 11
Логирование	Для сбора логов вы можете использовать ПО, которое соответствует требованиям вашей компании
Трейсинг	Трейсы в приложениях экспортируются с помощью библиотек Opentelemetry. Для сбора трейсов нужно использовать OTEL collector и далее перенаправлять их в одну из выбранных систем – Jaeger, Grafana и т.д. Желательно через буфер – Opensearch, Tempo и т.д.

Минимальные требования к ресурсам

Состав компонентов отличается в зависимости от поставки. Ориентируйтесь на требования того блока, компоненты которого входят в вашу поставку.

Блок	Компонент	CPU, m	RAM, Mi	Disk, GB	GPU (A100 80GB), unit
Общие компоненты	web-backend	250	512		
	cms-frontend	250	256		
	KServe	250	512		
	ai-registry	250	512		
	automl-gateway	10	128		
	langflow-backend	2000	10240		
	langflow-frontend	2000	512		
	channel-registry	200	256		
	kafka (не входит в комплект поставки)	4000	8192	50	
	PostgreSQL (не входит в комплект поставки)	4000	16384	50	
	Redis (не входит в комплект поставки)	4000	8192		
No-code	agent-engine	200	256		
	bot-registry	250	256		
	go-webim-adapter	200	256		
	http-adapter	100	256		
	telegram-adapter	100	1024		

Блок	Компонент	CPU, m	RAM, Mi	Disk, GB	GPU (A100 80GB), unit
	mfaas	200	512		
AI-сервисы (AutoML)	automl-manager	200	256		
	automl-pipelines (образ поставляется в комплекте к automl-manager)	-	-		
	automl-serving (образ поставляется в комплекте к automl-manager)	-	-		
AI-сервисы (LLM / RAG)	rag-manager	250	2048		
	cotype	1000	8192		1
	embedding	100	4096		1 Примечание. На данный момент Cotype можно запустить вместе с embedding. Также embedding может работать на CPU, но с существенной потерей производительности
	milvus (не входит в комплект поставки)	4000	16384	50	
	crawler	200	512		
Разметка	labelx-web	250	512		
Аналитика и BI	*clickhouse (не входит в комплект поставки)*	4000	16384	100	
	DataFacade	250	256		

Подготовка к установке

1. Перед установкой ознакомьтесь с требованиями к аппаратному и программному обеспечению.
2. Разверните кластер Kubernetes.

   Используется один кластер Kubernetes, так как видеокарты доступны только в нем. Разделение выполняется на уровне пространства имен.

3. В кластер Kubernetes с помощью менеджера пакетов Helm установите nginx ingress controller, cert-manager, Nvidia device plugin. Также установите GPU ноды и необходимые драйвера.
4. Разверните компоненты хранения данных:
   - Confluent Kafka или Kafka;
   - PostrgeSQL;
   - Redis;

   - Milvus;
   - S3;
   - ClickHouse.
5. Установите компоненты мониторинга:
   - Prometheus или VictoriaMetrics;
   - Grafana.
6. В Kubernetes создайте пространства имен (namespace):

```
kubectl create ns dev
kubectl create ns prod
kubectl create ns kserve
```

> Kserve использует одно общее пространство имен (namespace) для dev и prod, так как оно инфраструктурное и его сложно разделять на dev и prod.

7. Сохраните на компьютере архивы с компонентами для установки.
8. Загрузите docker-образы в локальный реестр, который будет доступен из кластера Kubernetes.
9. Установите Cotype Pro 2.5 – большую языковую модель для генерации и редактирования текстов, суммирования и анализа информации. Инструкция по развертыванию входит в комплект поставки.

Установка платформы

1. По умолчанию для платформы уже заданы значения обязательных параметров для каждого компонента. При необходимости скорректируйте их. Для этого в файле values.yaml раскомментируйте строку с нужным параметром и укажите для него значение. При этом, если в параметре предполагается ссылка:
   - на базу данных PostgreSQL, то предварительно создайте ее;
   - на хранилище S3, то предварительно создайте соответствующий бакет.
2. Для сервисов embedding и automl-manager заранее разместите модели в хранилище S3.
3. Установите Kserve. Примеры команд:

```
kubectl apply -f ./kserve/ns-prepare.yaml
kubectl apply -n kserve -f ./kserve/kserve.yaml
kubectl apply -n kserve -f ./kserve/kserve-cluster-resources.yaml
kubectl apply -n kserve -f models-web-app.yaml
```

4. Установите каждый компонент с помощью Helm Chart. Вместо <SERVICE_NAME> укажите имя сервиса:

```
helm upgrade "<SERVICE_NAME>" \
    --install \
    --atomic \
    --namespace "dev" \
    --values "charts/<SERVICE_NAME>/values.yaml" \
    charts/<SERVICE_NAME>
```

Конфигурирование

По умолчанию настройки платформы уже заданы. При необходимости вы можете их изменить. Для этого в файле values.yaml раскомментируйте строку с нужным параметром и задайте для него значение. В таблицах приведены компоненты их параметры.

При необходимости включите пакетные запросы к модели и автомасштабирование, а также отключите системный датасет, если это влияет на качество обучения.

Общие компоненты

Компоненты	Параметры
web-backend	• **SERVICE_ENVIRONMENT** – параметры окружения; • Agent Engine configuration **AGENT_ENGINE_URL** – адрес сервиса agent-engine; • AI-Registry url: **AI_REGISTRY_URL** – URL до сервиса ai-registry; • Bot-Registry url: **BOT_REGISTRY_URL** – URL до сервиса bot-registry; **LANGFLOW_URL** – URL до сервиса langflow-backend • OpenTelemetry configuration: **OTEL_EXPORTER_OTLP_ENDPOINT** – адрес для отправки данных телеметрии; • Rag Manager configuration: **RAG_MANAGER_URL** – URL до сервиса rag-manager; • AutoML Manager configuration: **AUTOML_MANAGER_URL** – URL до сервиса automl-manager; • LabelX configuration: **DATA_FACADE_URL** – URL до сервиса data-facade; **LABELX_URL** – URL до сервиса разметки; **LABELX_JWT_SECRET** – секретный ключ для подписи и проверки JWT-токена; **LABELX_JWT_ALGORITHM** – алгоритм шифрования для подписи JWT-токена; **LABELX_ACCOUNT** – имя аккаунта. • crawler-gateway configuration: **CRAWLER_URL** – URL сервиса crawler-gateway; • **AUTOML_GATEWAY_URL –** URL сервиса automl-gateway
cms-frontend	• Langflow configuration: **LANGFLOW_FRONTEND_URL** – URL до сервиса langflow-frontend • **RULE_BLOCK_ENABLED** – включить возможность использовать блок активации **Правило**
KServe	kserve.yaml: • **data.credentials** – параметры подключения к S3 хранилищу models-web-app.yaml: • **spec.rules.host** – FQDN для KServe models web app inference-service-prepare.yaml – параметры подключения к S3 хранилищу
ai-registry	• **SERVICE_ENVIRONMENT** – переменные окружения;

Компоненты	Параметры
	• OpenTelemetry configuration: **OTEL_EXPORTER_OTLP_ENDPOINT** – адрес для отправки данных телеметрии; • PostgreSQL configuration: **POSTGRES_SERVERS** – список серверов PostgreSQL; **POSTGRES_USER** – имя пользователя; **POSTGRES_PASSWORD** – пароль; **POSTGRES_DB_NAME** – название базы данных
bot-registry	• Service configuration: **CHANNELS_SERVING_HOST** – адрес хоста обработки запросов; • OpenTelemetry configuration: **OTEL_EXPORTER_OTLP_ENDPOINT** – адрес для отправки данных телеметрии; • PostgreSQL configuration: **POSTGRES_SERVERS** – список серверов PostgreSQL; **POSTGRES_USER** – имя пользователя; **POSTGRES_PASSWORD** – пароль; **POSTGRES_DB_NAME** – имя базы данных • **AUTOML_GATEWAY_URL –** URL сервиса automl-gateway
langflow-frontend	• **FRONTEND_PORT** – порт фронтенда; • **WEB_BACKEND_API_URL** – URL до сервиса web-backend; • **BACKEND_URL** – URL до сервиса langflow-backend
langflow-backend	• **LANGFLOW_SECRET_KEY** – секретный ключ для шифрования конфиденциальных данных, таких как ключи API. Подробнее о генерации секретного ключа см. в документации Langflow статью Authentication. Если значение переменной не указано, то ключ генерируется Langflow при каждом запуске пода. В этом случае Langflow не может расшифровать глобальные переменные новым сгенерированным ключом. Аналогично при масштабировании, когда число подов сервиса Langflow становится больше 1, новые поды не могут расшифровать глобальные переменные, созданные другим подом. В каждом из них свой ключ. Значение переменной должно совпадать на всех экземплярах сервиса. Если Langflow уже запущен и переменная **LANGFLOW_SECRET_KEY** не задана: 1. Очистите таблицу variables в базе данных. При этом удалятся **все** глобальные переменные. 2. Установите переменную **LANGFLOW_SECRET_KEY** с новым ключом. 3. Перезапустите под. • PostgreSQL configuration: **LF_CHART_EXTERNALDB_HOST** – имя хоста PostgreSQL; **LF_CHART_EXTERNALDB_PORT** – номер порта PostgreSQL; **LF_CHART_EXTERNALDB_DATABASE** – название базы данных; **LF_CHART_EXTERNALDB_USER** – имя пользователя; **LF_CHART_EXTERNALDB_PASSWORD** – пароль пользователя; • OpenTelemetry configuration: **OTEL_EXPORTER_OTLP_ENDPOINT** – конечная точка для отправки данных трассировки;

Компоненты	Параметры
	OTEL_EXPORTER_OTLP_TRACES_INSECURE – признак того, что следует экспортировать данные трассировки через небезопасное соединение без https. Возможные значения: **True**, **False**; **OTEL_EXPORTER_OTLP_TRACES_PROTOCOL** – протокол для передачи данных трассировки (grpc, http/protobuf); **OTEL_EXPORTER_OTLP_TRACES_COMPRESSION** – стратегия сжатия передаваемых данных трассировки (gzip или none); **OTEL_SERVICE_NAME** – имя сервиса, отправляющего данные трассировки; **OTEL_RESOURCE_ATTRIBUTES** – атрибуты ресурсов, которые производят данные трассировки; **OTEL_TRACING_LIST_LENGTH_LIMIT** – по умолчанию Langflow привязывает ключи **input.value** и **output.value** к трассировке и отображает список документов. Можно ограничить длину списка через переменную (по умолчанию — 5). Если элементов больше лимита, список сворачивается с указанием размера (**"length:<>"**). **Примечание**. Для стабильной работы Langflow должно быть не менее 500 свободных соединений в пуле PostgreSQL Настройки агентов: • **LANGFLOW_MCP_DEFAULT_SSE_SERVER** – адрес SSE-сервера, который используется по умолчанию в блоке **MCP Connection**; • **LANGFLOW_STREAM_CHUNK_SEPARATORS** – строка символов, по которой ответ от модели разделяется на чанки, если агент запускается в стриминговом режиме. По умолчанию " " (пробел). Указанная строка символов будет относиться к первому чанку; • **LANGFLOW_STREAM_CHUNK_DEBOUNCE_SECONDS** – время в секундах, по истечении которого отправляется следующий чанк при работе агента в стриминговом режиме. Чанк отправляется, даже если не встретился разделительный символ. По умолчанию **1.0** (1 секунда)

No-code-конструктор

Компоненты	Параметры
agent-engine	• Otel configuration: **OTEL_EXPORTER_OTLP_ENDPOINT** – адрес для отправки данных телеметрии; • **OTEL_SERVICE_NAME** – имя сервиса, отправляющего данные трассировки; • **OTEL_EXPORTER_OTLP_TRACES_INSECURE** – признак того, что следует экспортировать данные трассировки через небезопасное соединение без https. Возможные значения: **True**, **False**; • **OTEL_EXPORTER_OTLP_TRACES_PROTOCOL** – протокол для передачи данных трассировки (grpc, http/protobuf); • **OTEL_EXPORTER_OTLP_TRACES_COMPRESSION** – стратегия сжатия передаваемых данных трассировки (gzip или none); • PostgreSQL configuration: **POSTGRES_SERVERS** – список серверов БД; **POSTGRES_USER** – имя пользователя;

Компоненты	Параметры
	POSTGRES_PASSWORD – пароль; **POSTGRES_DB** – имя базы данных; • Kafka configuration: **KAFKA_BOOTSTRAP_SERVER** – адрес брокера Kafka; **DIALOG_HISTORY_KAFKA_TOPIC** – название топика; **DIALOG_HISTORY_SOURCE** – источник данных в Kafka; • Events API configuration: **EVENTS_API_ENABLED** **EVENTS_INPUT_KAFKA_TOPIC** • Priority Activator: **PRIORITY_ACTIVATOR** – приоритет условия активации. Укажите условие активации, которое нужно проверить в первую очередь. Возможные значения: **intent** – первым выбирается активационный блок с интентом независимо от порядка сценариев; **match** – первым выбирается активационный блок с регулярным выражением независимо от порядка сценариев; **Rule** – активация выполняется по заданным правилам; **none** – активационные блоки проверяются в порядке их добавления • **EXTERNAL_PYTHON_EXECUTOR_URL** – адрес сервиса mfaas. Если обработка кода выполняется в стороннем сервисе, укажите в качестве значения null; • **ON_ERROR_STUB_ANSWER** – сообщение, которое отображается при возникновении технической ошибки. По умолчанию «Возникла техническая ошибка»; • **RULE_MATCHER_URL** – адрес сервиса rule-matcher • **DEFAULT_ACTIVATION_RULES_THRESHOLD** – пороговое значение активации по правилу. Значение игнорируется, если для перехода к сценарию в препроцессинге используется функция defer_jump_to
channel-registry	• Common Kafka configuration **KAFKA_BROKER_ADDRESS** – адрес брокера Kafka; • Channel-config-telegram producer configuration **CHANNEL_CONFIG_TELEGRAM_REDEFINED_TOPIC_NAME** • Channel-config-webim producer configuration **CHANNEL_CONFIG_WEBIM_REDEFINED_TOPIC_NAME** • Channel-config-http producer configuration **CHANNEL_CONFIG_HTTP_REDEFINED_TOPIC_NAME** • Channel-config-langflow producer configuration **CHANNEL_CONFIG_LANGFLOW_REDEFINED_TOPIC_NAME** • Channels configuration **CHANNEL_HTTP_REDEFINED_TOPIC_NAME** **CHANNEL_LANGFLOW_REDEFINED_TOPIC_NAME** **CHANNEL_TELEGRAM_REDEFINED_TOPIC_NAME** **CHANNEL_WEBIM_REDEFINED_TOPIC_NAME** • OpenTelemetry configuration **OTEL_SERVICE_NAME** – имя сервиса, отправляющего данные трассировки; **OTEL_RESOURCE_ATTRIBUTES** – атрибуты ресурсов, которые производят данные трассировки;

Компоненты	Параметры
	OTEL_EXPORTER_OTLP_ENDPOINT – адрес для отправки данных телеметрии; • PostgreSQL configuration **POSTGRES_HOST** – имя хоста Postrgres; **POSTGRES_PORT** – порт; **POSTGRES_USER** – имя пользователя; **POSTGRES_PASSWORD** – пароль; **POSTGRES_DB_NAME** – имя базы данных
go-webim-adapter	• Common Kafka configuration: **KAFKA_BROKER_ADDRESS** – адрес брокера Kafka; • Channel-config-webim consumer configuration: **CHANNEL_CONFIG_WEBIM_REDEFINED_TOPIC_NAME** • Channel-webim consumer configuration: **CHANNEL_WEBIM_REDEFINED_TOPIC_NAME** • Engine-request producer configuration: **ENGINE_REQUEST_REDEFINED_TOPIC_NAME** • OpenTelemetry configuration: **OTEL_SERVICE_NAME** – имя сервиса для передачи данных телеметрии; **OTEL_RESOURCE_ATTRIBUTES** – атрибуты ресурсов, которые производят данные трассировки; **OTEL_EXPORTER_OTLP_ENDPOINT** – конечная точка для отправки данных трассировки
http-adapter	• Common Kafka configuration: **KAFKA_BROKER_ADDRESS** – адрес брокера Kafka; • Channel-config-http consumer configuration: **CHANNEL_CONFIG_HTTP_REDEFINED_TOPIC_NAME** • Channel-http consumer configuration: **CHANNEL_HTTP_REDEFINED_TOPIC_NAME** • Engine-request producer configuration: **ENGINE_REQUEST_REDEFINED_TOPIC_NAME** • OpenTelemetry configuration: **OTEL_SERVICE_NAME** – имя сервиса для передачи данных телеметрии; **OTEL_RESOURCE_ATTRIBUTES** – атрибуты ресурсов, которые производят данные трассировки; **OTEL_EXPORTER_OTLP_ENDPOINT** – конечная точка для отправки данных трассировки
telegram-adapter	• Common Kafka configuration: **KAFKA_BROKER_ADDRESS** – адрес брокера Kafka; • Channel-config-telegram consumer configuration: **CHANNEL_CONFIG_TELEGRAM_REDEFINED_TOPIC_NAME** • Channel-telegram consumer configuration: **CHANNEL_TELEGRAM_REDEFINED_TOPIC_NAME** • Engine-request producer configuration: **ENGINE_REQUEST_REDEFINED_TOPIC_NAME** • OpenTelemetry configuration: **OTEL_SERVICE_NAME** – имя сервиса для передачи данных телеметрии; **OTEL_RESOURCE_ATTRIBUTES** – атрибуты ресурсов, которые производят данные трассировки;

Компоненты	Параметры
	OTEL_EXPORTER_OTLP_ENDPOINT – конечная точка для отправки данных трассировки

AI-сервисы (AutoML)

Компоненты	Параметры
automl-manager	• Service configuration: **SERVICE_ENVIRONMENT** – переменные окружения; • OpenTelemetry configuration: **OTEL_EXPORTER_OTLP_ENDPOINT** – адрес для отправки данных телеметрии; • Service specific envs: **AI_REGISTRY_URL** – URL сервиса ai-registry; • Serving image: **K8S_SERVING_IMAGE** – путь до образа; **K8S_SERVING_NAMESPACE** – имя пространства для запуска automl-serving; **K8S_SERVING_SERVICE_ACCOUNT_NAME** – имя аккаунта; **K8S_SERVING_IMAGE_PULL_SECRET** – секрет для работы с образом; • Training image: **K8S_TRAINING_IMAGE** – путь до образа; **K8S_TRAINING_JOB_TTL_SECONDS** – время жизни джобы пайплайна на тренировку модели в секундах. • Kserve configuration: **KSERVE_URL** – URL до сервиса KServe; • Serving batching & scaling configuration: **K8S_SERVING_SCALING__MIN_REPLICAS** – минимальное количество инстансов моделей для развертывания; **K8S_SERVING_SCALING__MAX_REPLICAS** – максимальное количество инстансов моделей для развертывания; **K8S_SERVING_SCALING__SCALE_METRIC** – метрика для масштабирования. Возможные значения: "cpu", "memory"; **K8S_SERVING_SCALING__SCALE_TARGET** – целевое значение метрики для запуска автоскейлинга; **K8S_SERVING_BATCHING__MAX_BATCH_SIZE** – максимальный размер батча для группировки запросов; **K8S_SERVING_BATCHING__MAX_LATENCY** – время набора пачки в мс, по истечении которого запросы отправятся в модель; **K8S_SERVING_BATCHING__TIMEOUT** – таймаут в секундах на выполнение запросов в модели; • MinIO configuration: **S3_BUCKET** – имя бакета для хранения результатов; **S3_CHECKPOINTS_BUCKET** – имя бакета для хранения чекпоинтов; **S3_CHECKPOINTS_PATH** – путь для хранения чекпоинтов; **S3_USER** – имя пользователя Minio; **S3_PASSWORD** – пароль; **S3_URI** – URL для подключения к хранилищу S3; • LabelX configuration: **LABELING_URL** – URL до сервиса разметки; **LABELING_JWT_SECRET** – секретный ключ, используемый для подписи и проверки JWT-токена;

Компоненты	Параметры
	LABELING_JWT_ALGORITHM – алгоритм шифрования, применяемый для подписи JWT-токена; **LABELING_ACCOUNT** – имя аккаунта
automl-gateway	**INSTANCE_URL_TEMPLATE –** шаблон URL для проксирования запросов к модулям инференса (автоматического логического вывода моделей). Пример: "http://%s-%s-<platform_namespace>-predictor.<kserve_inference_namespace>/v1/models/automl-model:predict" • Замените <platform_namespace> на имя namespace, где установлены все сервисы платформы • Замените <kserve_inference_namespace> на имя namespace, содержащего KServe inference (InferenceServices)

AI-сервисы (RAG)

Компоненты	Параметры
rag-manager	• Service configuration: **EMBEDDING_HOST** – имя хоста embedding; **EMBEDDING_PORT** – порт сервиса embedding; **AI_REGISTRY_URL** – URL до сервиса ai-registry; **LANGFLOW_BACKEND_URL** – URL до сервиса langflow-backend; **LLM_URL** – URL до сервиса cotype; **MODEL_NAME** – название LLM; **RAG_MANAGER_URL** – URL до сервиса rag-manager; • MinIO configuration: **MINIO_HOST** – имя хоста хранилища S3; **MINIO_PORT** – порт сервиса хранилища S3; **MINIO_USE_HTTPS** – признак того, что используется протокол HTTPS; **MINIO_BUCKET_NAME** – имя бакета; **MINIO_ACCESS_KEY** – ключ доступа; **MINIO_SECRET_KEY** – секретный ключ • PostgreSQL configuration: **POSTGRES_HOST** – имя хоста PostgreSQL; **POSTGRES_PORT** – порт PostgreSQL; **POSTGRES_SERVERS** – список серверов PostgreSQL; **POSTGRES_USER** – имя пользователя; **POSTGRES_PASSWORD** – пароль; **POSTGRES_DB_NAME** – имя базы данных; • Vector DB configuration: **VECTOR_DB_HOST** – имя хоста векторной базы данных; **VECTOR_DB_PORT** – порт векторной базы данных; **VECTOR_DB_NAME** – имя базы данных; **VECTOR_DB_USER** – имя пользователя; **VECTOR_DB_PASSWORD** – пароль • OpenTelemetry configuration: **OTEL_EXPORTER_OTLP_ENDPOINT** – адрес для отправки данных трассировки; **OTEL_EXPORTER_OTLP_TRACES_INSECURE** – признак того, что следует экспортировать данные трассировки через небезопасное соединение без https. Возможные значения: **True**, **False**; **OTEL_EXPORTER_OTLP_TRACES_PROTOCOL** – протокол для передачи

Компоненты	Параметры
	данных трассировки (grpc, http/protobuf); **OTEL_EXPORTER_OTLP_TRACES_COMPRESSION** – стратегия сжатия передаваемых данных трассировки (gzip или none); **OTEL_SERVICE_NAME** – имя сервиса, отправляющего данные трассировки; **OTEL_RESOURCE_ATTRIBUTES** – атрибуты ресурсов, которые производят данные трассировки
cotype	• **GPUUTIL** – доля в использовании GPU (от 0 до 1); **EAGER** – отключает оптимизации CUDA graphs и выполняет вычисления последовательно; **MACHINE_PORT** – номер порта; **ADDITIONAL_ARGS** – дополнительные аргументы • OpenTelemetry configuration: **OTEL_EXPORTER_OTLP_TRACES_INSECURE** – признак того, что следует экспортировать данные трассировки через небезопасное соединение без https. Возможные значения: **True**, **False**; **OTEL_EXPORTER_OTLP_ENDPOINT** – конечная точка для отправки данных трассировки; **OTEL_EXPORTER_OTLP_TRACES_PROTOCOL** – протокол для передачи данных трассировки (grpc, http/protobuf); **OTEL_EXPORTER_OTLP_TRACES_COMPRESSION** – стратегия сжатия передаваемых данных трассировки (gzip или none); **OTEL_SERVICE_NAME** – имя сервиса, отправляющего данные трассировки
embedding	• MinIO configuration: **PRIMARY_S3_ENDPOINT_URL** – URL до хранилища с моделями; **PRIMARY_S3_BUCKET** – имя бакета; **PRIMARY_S3_ACCESS_KEY_ID** – идентификатор ключа; **PRIMARY_S3_SECRET_ACCESS_KEY** – ключ доступа; • OpenTelemetry configuration: **OTEL_EXPORTER_OTLP_ENDPOINT** – конечная точка для отправки данных трассировки; **OTEL_RESOURCE_ATTRIBUTES** – атрибуты ресурсов, которые производят данные трассировки
crawler-gateway	• Service configuration: **APP_HOST** – хост сервиса crawler-gateway; **APP_PORT** – порт сервиса crawler-gateway; **APP_LOG_LEVEL** – уровень логирования; **APP_DEBUG** – режим отладки сервиса; **APP_WORKERS_NUM** – количество рабочих процессов, которые будут одновременно обрабатывать входящие запросы; **APP_WORKERS_TIMEOUT** – таймаут рабочих процессов. • Kafka configuration: **APP_KAFKA_HOST** – хост брокера Kafka; **APP_KAFKA_PORT** – порт брокера Kafka. • OpenTelemetry configuration: **APP_OTLP_URL** – конечная точка для отправки данных трассировки; **APP_OTLP_INSECURE** – признак того, что следует экспортировать данные трассировки через небезопасное соединение – без https. Возможные значения: **True**, **False**.

Компоненты	Параметры
	• Kafka`s topic name: **APP_EXECUTOR_CRAWLING_TASK_QUEUE** – название топика Kafka, принимающего задачу от crawler-gateway. • secrets: **APP_SERVICE_TOKEN** – токен, используемый для авторизации сервисов при обращении к API crawler-gateway
crawler-executor	• Service configuration: **LOG_LEVEL** – уровень логирования; **MAX_DEPTH** – максимальная глубина рекурсивного обхода сайта; **PROCESSES** – количество параллельных рабочих процессов краулера; **NAME** – имя исполнителя; **TASK_NAME** – тип задачи для идентификации в очереди и логах; **SE_CACHE_PATH** – путь к папке, используемой для кэширования данных Selenium. • Kafka configuration: **KAFKA_HOST** – хост брокера Kafka; **KAFKA_PORT** – порт брокера Kafka. • Kafka`s topic name: **QUEUE** – название Kafka-топика, из которого сервис crawler-executor получает задачи на выполнение краулинга. • rag-manager configuration: **RAG_MANAGER_URL** – URL-адрес сервиса rag-manager. • OpenTelemetry configuration: **OTLP_URL** – конечная точка для отправки данных трассировки; **OTLP_INSECURE** – признак того, что следует экспортировать данные трассировки через небезопасное соединение – без https. Возможные значения: **True**, **False**. • AI-registry configuration: **AI_REGISTRY_URL** – URL-адрес сервиса ai-registry. • secrets: **EXPORTER_URL** – URL для отправки результатов краулинга сайта; **TOKEN** – токен, для авторизации при обращении к API crawler-gateway (должны содержать то же значение, что и параметр **APP_SERVICE_TOKEN** сервиса crawler-gateway)

Разметка

Компоненты	Параметры
labelx-web	• **ROOT_URL** – внешний URL сервиса разметки; • **SENTRY_DSN** – URL для отправки алертов (опционально); • PostgreSQL configuration: **DB_HOST** – имя хоста PostgresSQL; **DB_PORT** – порт PostgresSQL; **DB_NAME** – имя базы данных; **DB_USER** – имя пользователя; **DB_PASS** – пароль

Аналитика и BI

Компоненты	Параметры
data-facade	• Kafka configuration: **KAFKA_BROKER_LIST** – адреса брокеров Kafka; **KAFKA_TOPIC_LIST** – список топиков; **KAFKA_GROUP_NAME** – имя группы; • OpenTelemetry configuration: **TRACING_AGENT_HOST** – адрес для отправки данных телеметрии; **TRACING_AGENT_PORT** – порт для отправки данных телеметрии • ClickHouse secrets: **CLICKHOUSE_SERVER** – URL до сервиса clickhouse; **CLICKHOUSE_PORT** – порт сервиса clickhouse; **CLICKHOUSE_USER** – имя пользователя; **CLICKHOUSE_PASSWORD** – пароль пользователя; **CLICKHOUSE_DATABASE** – имя базы данных; • PostgreSQL configuration – подключение к базе labelx_web: **POSTGRES_HOST** – имя хоста PostgreSQL; **POSTGRES_PORT** – порт PostgreSQL; **POSTGRES_DATABASE** – имя базы данных; **POSTGRES_USER** – имя пользователя; **POSTGRES_PASSWORD** – пароль

Настройка пакетных запросов к модели и автомасштабирования

Сервис automl-manager поддерживает пакетную отправку клиентских запросов в модель (батчинг) и автомасштабирование экземпляров развернутой модели (автоскейлинг). По умолчанию эти функции выключены.

Чтобы включить батчинг, необходимо при развертывании automl-manager задать значения следующим переменным:

- **K8S_SERVING_BATCHING__MAX_BATCH_SIZE**
- **K8S_SERVING_BATCHING__MAX_LATENCY**
- **K8S_SERVING_BATCHING__TIMEOUT**

Чтобы включить автоскейлинг, требуется прописать переменные:

- **K8S_SERVING_SCALING__MIN_REPLICAS**
- **K8S_SERVING_SCALING__MAX_REPLICAS**
- **K8S_SERVING_SCALING__SCALE_METRIC**
- **K8S_SERVING_SCALING__SCALE_TARGET**

При установленном в переменной **K8S_SERVING_SCALING__SCALE_METRIC** значении "cpu" рекомендуется в переменной **K8S_SERVING_SCALING__SCALE_TARGET** установить значение "75".

Для настройки фиксированного масштабирования достаточно задать значение только переменной **K8S_SERVING_SCALING__MIN_REPLICAS**.

Отключение системного датасета при обучении моделей NER

При обучении модели NER пользовательский датасет автоматически обогащается системными (предопределёнными) сущностями. В некоторых случаях это может непреднамеренно влиять на

качество обучения, вводя шум (ложные метки), bias (предвзятости) или конфликты с пользовательскими сущностями, снижая точность модели в специализированных доменах.

Отключить системный датасет можно через переменную **NER_TRAINING_SETTINGS__CHECKPOINTS** сервиса automl-manager.

Установите её значение в конфигурации Helm-чарта automl-manager, как указано в примерах, и перезапустите сервис перед обучением.

Пример с включённым обучением на системных сущностях:

NER_TRAINING_SETTINGS__CHECKPOINTS: '[{"NAME":"ruBert-base","JOB_ENV_NAME":"MAIN_CHECKPOINT"},{"NAME":"v1.0.0_tadner_pretrain","JOB_ENV_NAME":"TAD_NER_CHECKPOINT"},{"NAME":"v1.0.0_finetuner_system_ner","JOB_ENV_NAME":"FINETUNER_SYSTEM_NER_CHECKPOINT"}]'

Пример с отключённым обучением на системных сущностях:

NER_TRAINING_SETTINGS__CHECKPOINTS: '[{"NAME":"ruBert-base","JOB_ENV_NAME":"MAIN_CHECKPOINT"},{"NAME":"v1.0.0_tadner_pretrain","JOB_ENV_NAME":"TAD_NER_CHECKPOINT"}]'

Мониторинг

В платформе предусмотрена возможность оценивать текущее состояние сервисов. Для этого в комплект поставки входят архивы с темплейтами (шаблонами) дашбордов в формате JSON. Чтобы отслеживать состояние сервисов с помощью дашбордов, предварительно установите систему сбора метрик Prometheus или VictoriaMetrics, а также платформу мониторинга и анализа данных Grafana (не входит в комплект поставки). Данные на дашбордах позволяют быстро определить работоспособность сервисов и в случае возникновения инцидентов вовремя принять меры.

> В полученном JSON-файле укажите источники данных в виде ссылок в формате используемой системы сбора метрик.

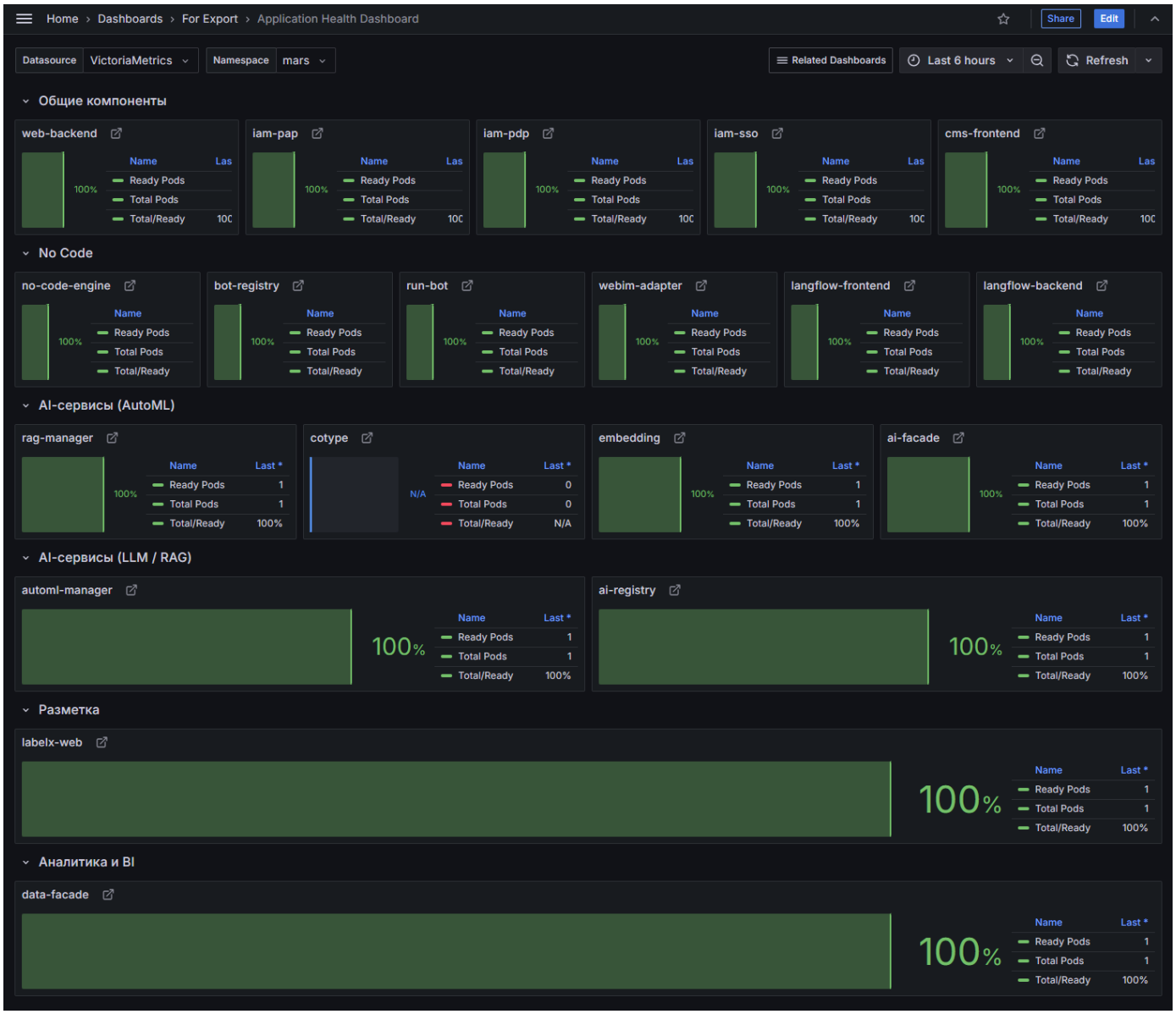
Подробнее о настройках и работе с дашбордами см. в документации Grafana.

Для мониторинга используются:

- **Application Health Dashboard** – дашборд состояния работоспособности сервисов;
- **Application summary** – дашборд с метриками конкретного сервиса. К нему можно перейти из панелей дашборда **Application Health Dashboard**.

Application Health Dashboard

Панели дашборда разделены на группы. На каждой панели отображается информация о состоянии подов конкретного сервиса. Чтобы перейти к дашборду **Application summary** с детальной информацией по сервису, нажмите на кнопку .

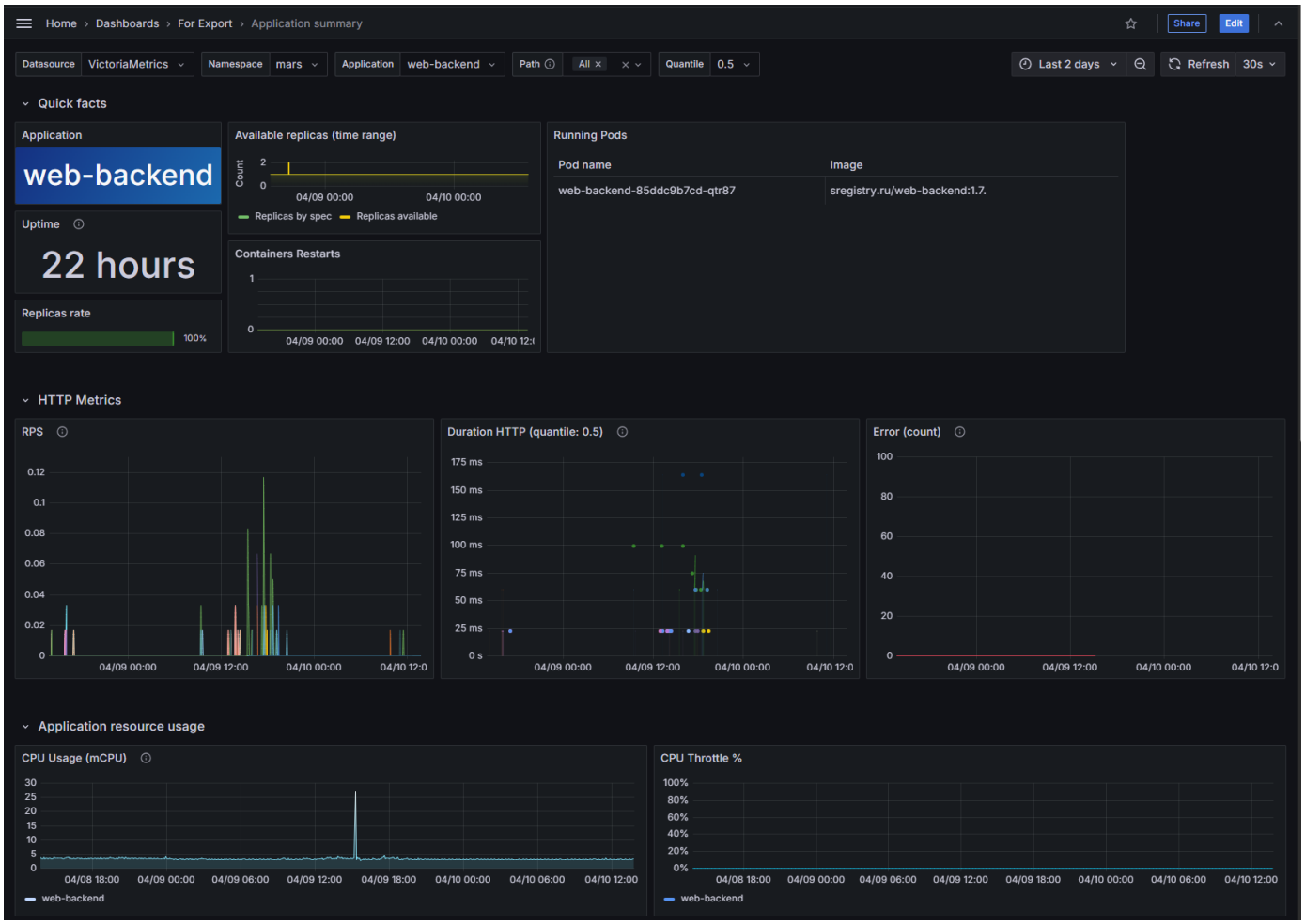


Application summary

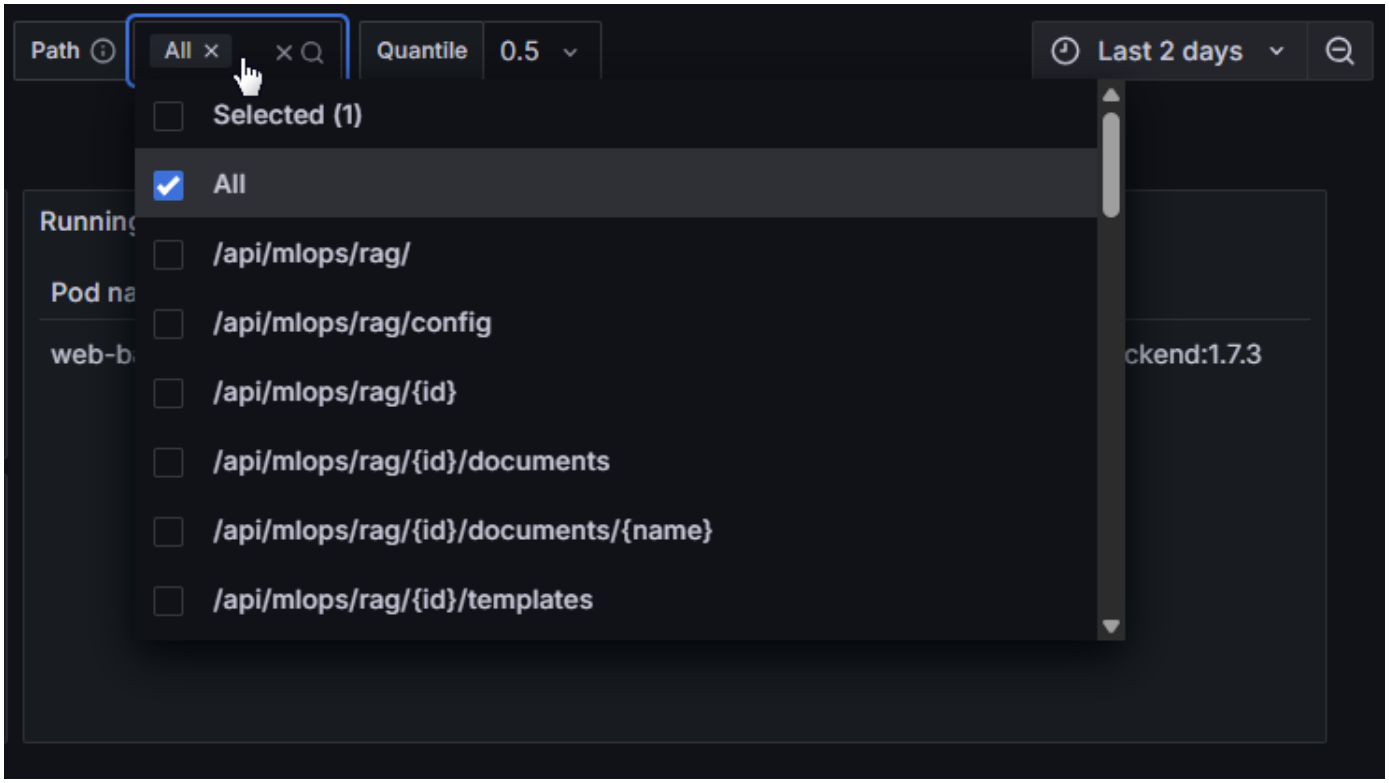
На дашборде отображается название сервиса, имя пода и другая информация, например:

- длительность обработки HTTP-запросов и количество ошибок (HTTP Metrics);
- загрузка процессора, оперативной памяти, потребление ресурсов сервисами (Application resource usage);

- ресурсы, потребляемые подами (Pod's resourse usage).



При необходимости отфильтруйте данные по конкретным HTTP-запросам. Для этого выберите нужные эндпоинты методов в выпадающем списке **Path**:



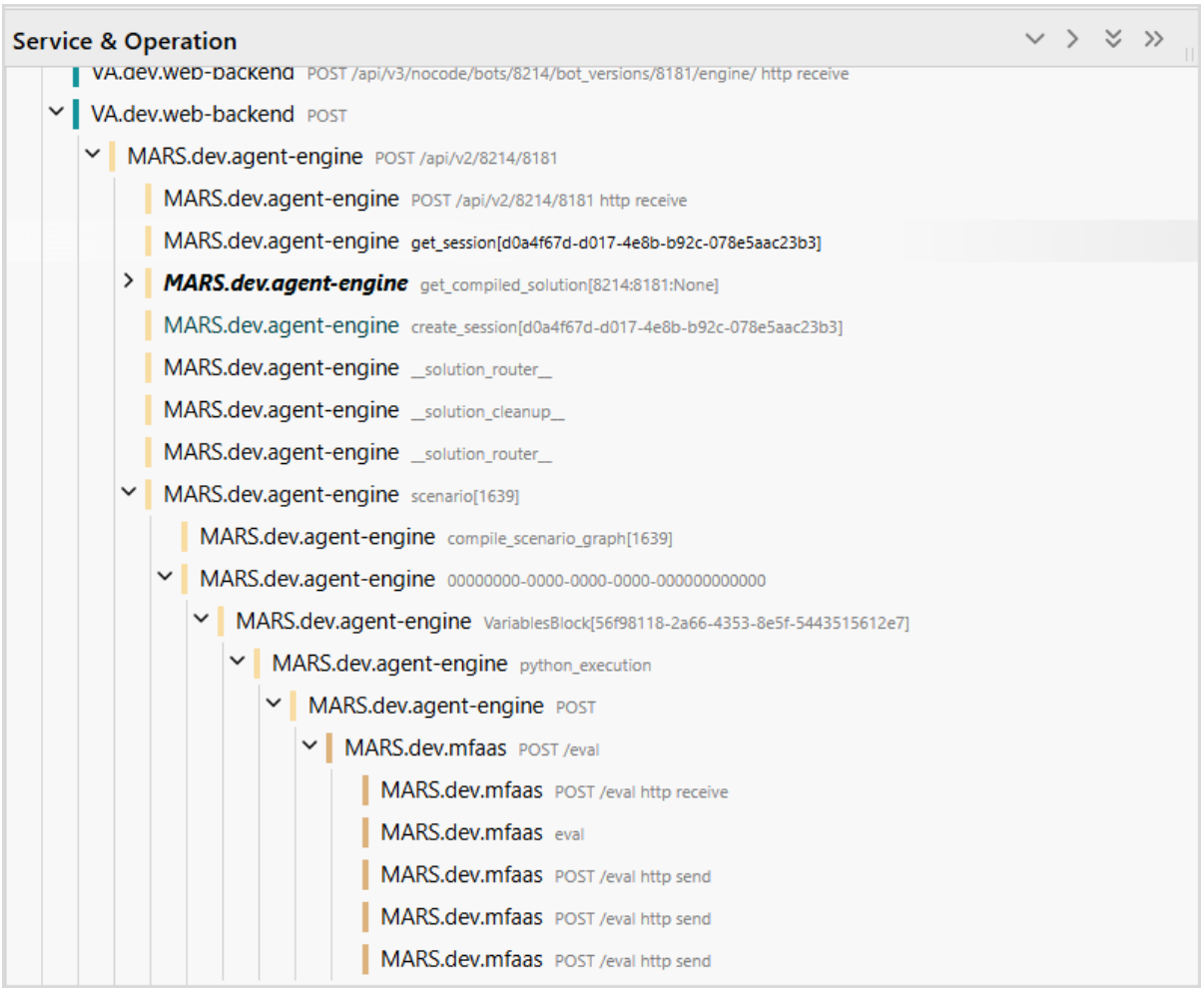
Трассировка

Платформа MWS AI Agents Platform поддерживает систему трассировки. Трассировка предоставляет детализированную информацию о прохождении запросов через различные компоненты платформы. Система основана на инструменте OpenTelemetry. Интеграция с компонентами платформы выполняется через OTLP (OpenTelemetry Protocol), данные о спанах экспортируются по gRPC.

Для просмотра и анализа трассировок используйте Jaeger.

В качестве примера. Метод POST /api/v3/nocode/bots/{bot_id}/bot_versions/{bot_version_id}/engine/ передает метаданные трассировки прохождения запроса через компоненты сценария бота при выполнении запроса в виджете тестирования. Для этого в сервисе реализована генерация и отправка спанов в Jaeger.

Таким образом в Jaeger UI возможно получить и проанализировать детали обработки.



Логирование сервисов

Чтобы своевременно принимать меры по устранению неисправностей, рекомендуется отслеживать лог-файлы системы. Для этого используйте возможности k8s:

```
kubectl logs -n <namespace> deployment/<name-of-component> # logs of deployment
kubectl logs -n <namespace> -f deployment/<name-of-component> # follow logs
```

Логи движка agent-engine

Логи сервиса agent-engine представляют собой последовательную запись процесса выполнения движком сценария бота.

Уровень логирования устанавливается с помощью переменной **LOG_LEVEL** (по умолчанию **LOG_LEVEL=INFO**).
Формат вывода можно задать в переменной **LOG_JSON_FORMAT**.

Вот этапы процесса работы движка, которые можно найти в логах.

Начало выполнения запроса к движку

- Фиксируется начало обработки запроса с уникальным идентификатором (request_id).
- Содержит информацию о сессии, пользователе и тексте сообщения.

Создание новой сессии

- agent-engine создает новую сессию при новом диалоге с ботом.
- Содержится идентификатор сессии (session_id).

Запуск препроцессинга

- Фиксируется запуск сценария (execution_phase=PREPROCESSING).
- Фиксируются выполнение нод и блоков.

Основной сценарий

- Фиксируется старт сценария "Default NoMatch Scenario" (scenario_id: 2947).
- Последовательно фиксируются исполняемые ноды и блоки сценария.

HTTP-запрос

- Фиксируется HTTP-запрос к внешнему сервису, например, классификатору.
- Содержит параметры запроса.
- При ошибках обработки запроса.
- При отправке сообщений на callback URL.

Структура логов

Поле	Описание
message	Текстовое сообщение о происходящем действии. Примеры: • Created new session – начало сессии; • Execute engine call – информация о начале обработки запроса пользователя; • Executing node – выполненная нода в сценарии; • Executing block – выполненный блока сценария;

Поле	Описание
	• Http request succeeded – успешный http запрос; • An error occurred during request – ошибка в ходе обработки запроса; • An error occurred in node {id ноды} block {id блока} – ошибка в ноде и блоке; • Error in block {id блока} – ошибка в блоке; • Error in Agent block {id блока} – ошибка в блоке агента; • Error in Python code execution – ошибка при выполнении блока с python-скриптом
time	Временная метка события в формате yyyy-MM-dd HH:mm:ss
timestamp	Временная метка в формате Unix-время с миллисекундами
logger_name	Имя логгера
level	Уровень логирования (по умолчанию – INFO)
thread_id	Идентификатор потока, в котором выполнялась операция
process_id	Идентификатор процесса
metadata	Метаданные, включая trace_id, для отслеживания запроса в OpenTelemetry
extra	Дополнительная контекстная информация, которая зависит от типа лога
traceback (при LOG_LEVEL = ERROR)	Последовательность вызовов функций, которая привела к ошибке python-скрипта

Значение поля extra

Основные поля	Описание
request_id	Уникальный идентификатор запроса
session_id	Уникальный идентификатор сессии, связанной с данным запросом
user_id	Уникальный идентификатор пользователя, отправившего запрос
message_original_text	Первоначальный текст сообщения, который был отправлен пользователем
channel_id	Идентификатор канала, через который было отправлено сообщение
message_id	Уникальный идентификатор сообщения
solution_id	Идентификатор решения, соответствующий обработке запроса
solution_version_id	Версия решения, соответствующая обработке запроса
node_id	Уникальный идентификатор ноды, когда выполняется нода
node_name	Имя узла, которое имеет значение для описательного контекста
execution_phase	Фаза выполнения сценария: "PREPROCESSING" или "MAIN"

Основные поля	Описание
scenario_id	Идентификатор сценария, связанного с текущим запросом
scenario_name	Имя сценария, также используемое для контекстуальности (например, "Запись к врачу")
block_type	Тип блока, который выполняется (например, "VariablesBlock", "AnswerBlock", "HttpRequestBlock")
block_id	Уникальный идентификатор выполняемого блока
status_code	HTTP-статус код, возвращенный при выполнении HTTP-запроса (например, 200 для успешного запроса)