Инфраструктурный слой LLM Gateway в Yandex Cloud

Шуба С.А.

2026

Цель работы

Требовалось реализовать инфраструктурный слой для обращения к нескольким LLM через единый REST API в Yandex Cloud.

Основная идея:

- не обращаться к моделям напрямую;
- сделать единый сервис-переходник;
- обеспечить работу с ролями `scorer`, `auditor`, `validator`;
- добавить работу с данными и логами через Object Storage.

Общая идея решения

Пользователь / разработчик

↓

CPU VM в Yandex Cloud

↓

LLM Gateway на FastAPI

↓

Yandex Cloud AI Studio API

Как работает gateway

Пользователь отправляет запрос не в Yandex Cloud напрямую, а в собственный gateway.

Gateway:

- принимает запрос;
- определяет роль модели;
- выбирает нужный backend;
- возвращает ответ в едином формате.

Это позволяет:

- скрыть внутренние ключи доступа;
- централизовать вызов LLM;
- использовать один API для разных ролей.

Что было настроено в Yandex Cloud

- создана отдельная сеть;
- подготовлены подсети;
- настроены security groups;
- создана виртуальная машина `llm-gateway-1`;
- настроен SSH-доступ;
- подготовлено Python-окружение и `venv`;
- развернуто FastAPI-приложение.

Сетевой доступ

Для ограничения сетевого доступа были подготовлены группы безопасности:

- `sg-alb`
- `sg-gateway`
- `sg-models`

LLM Gateway

На виртуальной машине было реализовано FastAPI-приложение.

Основные методы:

- `GET /health`
- `POST /llm/generate`
- `POST /v1/generate` (как целевой формат)

Для проверки использовалась модель:

`qwen3-235b-a22b-fp8`

Доступ к Yandex Cloud AI Studio

- был создан service account `sa-gateway-ai`;
- был создан API key со scope `yc.ai.languageModels.execute`;
- была назначена роль `ai.languageModels.user`;
- на VM были добавлены переменные окружения:

```
YC_FOLDER_ID
YC_AI_API_KEY
CLIENT_API_KEY
```

Работа с данными через Object Storage

Кроме обращения к LLM, требовалось организовать работу с данными и логами так, чтобы они не хранились только локально.

Для этого был выбран Yandex Object Storage.

Структура хранения:

`raw/parsed_new/`	исходные данные парсера
`prepared/parsed_sources/`	подготовленные источники
`output/results/`	итоговые результаты
`logs/gateway/`	логи gateway
`logs/pipeline/`	логи пайплайна

Методы для работы с файлами и логами

- `POST /files/upload`
- `GET /files/list`
- `GET /files/download`
- `POST /logs/write`

Таким образом, через один gateway можно:

- вызывать LLM;
- загружать файлы;
- получать список файлов;
- скачивать результаты;
- сохранять логи.

Smoke-test

Для проверки был подготовлен тестовый Python-скрипт.

Он проверял:

- `/health`
- `/llm/generate`
- `/files/upload`
- `/files/list`
- `/files/download`
- `/logs/write`

Все основные методы вернули успешный ответ 200.

Результат проверки

Проверка	Статус	Результат
/health	200	gateway доступен
/llm/generate	200	вызов LLM работает
/files/upload	200	файл загружен
/files/list	200	список файлов получен
/files/download	200	файл скачан
/logs/write	200	лог записан

Что получилось в итоге

- развернут gateway на VM в Yandex Cloud;
- реализован единый API для ролей моделей;
- настроено обращение к LLM через Yandex Cloud AI Studio;
- добавлена работа с Object Storage;
- выполнен smoke-test;
- подтверждена работоспособность базового инфраструктурного слоя.