

NovaData LMS: регламент эксплуатации и восстановления

Область: lms-novadata , SQLChecker , code_checker

Документ описывает базовые операции сопровождения, диагностику типовых проблем и порядок восстановления после сбоев.

Связанные документы:

- README_DOCUMENTATION.md - индекс комплекта документации и рекомендуемый порядок чтения.
- PROJECT_DOCUMENTATION.md - общая техническая документация проекта.
- DEPLOYMENT_GUIDE.md - инструкция по разворачиванию.
- ENVIRONMENT_VARIABLES.md - схема переменных окружения.
- API_CONTRACTS.md - API-контракты LMS/checkers.
- PROGRAMMING_TASK_AUTHOR_GUIDE.md - руководство по созданию programming-заданий.
- SQL_TASK_AUTHOR_GUIDE.md - руководство по созданию SQL-заданий.

1. Состав эксплуатируемых сервисов

Компонент	Тип запуска	Назначение
LMS / Unicorn	systemd	Основное Django-приложение
Celery worker	systemd	Фоновые задачи LMS
Celery beat	systemd	Планировщик периодических задач
Redis	systemd/package service	Broker/result backend Celery
MySQL	systemd/package service	Основная БД LMS
SQLChecker API	Docker Compose	Проверка SQL-заданий
SQLChecker PostgreSQL	Docker Compose	Временные БД для SQL-проверок

Компонент	Тип запуска	Назначение
SQLChecker ClickHouse	Docker Compose	Проверка ClickHouse-заданий
code_checker API	Docker Compose	API шаблонов и ручных запусков
code_checker worker	Docker Compose	Асинхронная проверка programming-заданий
Docker sandbox images	Docker images	Изолированное выполнение кода студентов
S3/MinIO/Selectel	внешний сервис	Хранение файлов и видео

2. Ежедневный health-check

2.1. LMS

```
sudo systemctl status gunicorn --no-pager -l
sudo systemctl status celery --no-pager -l
sudo systemctl status celery-beat --no-pager -l
sudo systemctl status redis-server --no-pager -l
```

Проверить HTTP-ответ LMS:

```
curl -I http://LMS_HOST:8000/
```

Проверить Django:

```
cd /var/www/lms
source venv/bin/activate
python manage.py check
```

2.2. SQLChecker

```
cd /opt/nd_lms/SQLChecker
docker compose ps
curl http://SQL_CHECKER_HOST:8000/health
curl http://SQL_CHECKER_HOST:8000/stats
```

Ожидаемый /health :

```
{
  "status": "healthy",
  "database_connected": true,
  "active_tasks": 0,
  "uptime_seconds": 0.0
}
```

2.3. code_checker

```
cd /opt/nd_lms/code_checker
docker compose ps
docker compose logs --tail=100 worker
curl http://CODE_CHECKER_HOST:5001/info
```

В логах worker не должно быть:

```
FATAL ERROR: LMS_BASE_URL or LMS_API_TOKEN is not configured.
WORKER ERROR: Could not connect to LMS
```

2.4. Очередь проверок

Проверить, что очередь не растёт бесконтрольно:

```
cd /var/www/lms
source venv/bin/activate
python manage.py shell
```

```
from lms_project.apps.common.models import SubmissionQueue, SubmissionHistory
SubmissionQueue.objects.values("status", "task_type").order_by("status", "task_type").count()
SubmissionQueue.objects.filter(status="pending").count()
SubmissionHistory.objects.count()
```

Если pending-задания накапливаются и не уходят в историю, нужно проверить `code_checker worker` .

3. Логи

3.1. LMS

Основные команды:

```
sudo journalctl -u gunicorn -f
sudo journalctl -u celery -f
sudo journalctl -u celery-beat -f
```

Файловый лог ошибок Django:

```
/var/www/lms/log/error.log
```

Лог результатов validator API:

```
/var/www/lms/validator_results.log
```

Что искать:

```
grep -i "error" /var/www/lms/log/error.log
grep -i "traceback" /var/www/lms/log/error.log
tail -f /var/www/lms/validator_results.log
```

3.2. SQLChecker

```
cd /opt/nd_lms/SQLChecker
docker compose logs -f sql-checker-api
docker compose logs -f postgres-db
docker compose logs -f clickhouse-db
```

Что искать:

- ошибки подключения к PostgreSQL;
- ошибки подключения к ClickHouse;
- exceptions при /check ;
- накопление временных БД;
- HTTP 500.

3.3. code_checker

```
cd /opt/nd_lms/code_checker
docker compose logs -f worker
docker compose logs -f api
```

Что искать:

- FATAL ERROR ;
- WORKER ERROR ;
- Could not connect to LMS ;
- ошибки Docker socket;
- ошибки отсутствующих sandbox images;
- timeout выполнения кода;
- compilation/runtime errors.

4. Базовые операции перезапуска

4.1. LMS

```
sudo systemctl restart gunicorn
sudo systemctl restart celery
sudo systemctl restart celery-beat
```

После перезапуска:

```
sudo systemctl status gunicorn --no-pager -l  
sudo systemctl status celery --no-pager -l
```

4.2. Redis

```
sudo systemctl restart redis-server  
sudo systemctl status redis-server --no-pager -l
```

После перезапуска Redis нужно проверить Celery:

```
sudo systemctl restart celery celery-beat
```

4.3. SQLChecker

```
cd /opt/nd_lms/SQLChecker  
docker compose restart sql-checker-api
```

Если проблема в БД:

```
docker compose restart postgres-db  
docker compose restart clickhouse-db  
docker compose restart sql-checker-api
```

4.4. code_checker

```
cd /opt/nd_lms/code_checker  
docker compose restart worker  
docker compose restart api
```

Если проблема в sandbox images:

```
docker compose --profile build_only build  
docker compose restart worker
```

5. Типовые инциденты и восстановление

5.1. LMS не открывается

Симптомы:

- `curl -I http://LMS_HOST:8000/` не отвечает;
- пользователь видит 502/503/connection refused.

Проверка:

```
sudo systemctl status gunicorn --no-pager -l
sudo journalctl -u gunicorn -n 200 --no-pager
cd /var/www/lms && source venv/bin/activate && python manage.py check
```

Восстановление:

```
sudo systemctl restart gunicorn
```

Если ошибка связана с БД:

```
sudo systemctl status mysql --no-pager -l
sudo systemctl restart mysql
python manage.py migrate --check
sudo systemctl restart gunicorn
```

5.2. Celery-задачи не выполняются

Симптомы:

- не обновляется прогресс;
- не обрабатываются видео;
- периодические задачи не стартуют.

Проверка:

```
sudo systemctl status redis-server --no-pager -l
sudo systemctl status celery --no-pager -l
sudo journalctl -u celery -n 200 --no-pager
```

Восстановление:

```
sudo systemctl restart redis-server  
sudo systemctl restart celery  
sudo systemctl restart celery-beat
```

Проверить, что worker слушает очереди:

```
celery,progress_updates,video_processing
```

5.3. SQL-задания не проверяются

Симптомы:

- /api/sql/validate/ возвращает 500;
- студент видит ошибку проверки SQL;
- в LMS логах ошибка подключения к SQLChecker.

Проверка:

```
curl http://SQL_CHECKER_HOST:8000/health  
cd /opt/nd_lms/SQLChecker  
docker compose ps  
docker compose logs --tail=200 sql-checker-api
```

Проверить в LMS:

```
SQL_CHECKER_URL  
SQL_CHECKER_TIMEOUT
```

Текущий риск: SQL_CHECKER_URL задан в settings.py жестко. Если адрес SQLChecker изменился, LMS продолжит обращаться по старому адресу.

Восстановление:

```
cd /opt/nd_lms/SQLChecker  
docker compose restart sql-checker-api  
docker compose restart postgres-db clickhouse-db
```

После восстановления:

```
curl http://SQL_CHECKER_HOST:8000/health
sudo systemctl restart gunicorn
```

5.4. Programming-задания зависли в pending

Симптомы:

- студент видит, что решение долго проверяется;
- В `submission_queue` растёт количество `pending` ;
- `submission_history` не пополняется.

Проверка:

```
cd /opt/nd_lms/code_checker
docker compose ps
docker compose logs --tail=300 worker
```

Проверить доступ worker к LMS:

```
curl -H "X-API-Key: TOKEN" "http://LMS_HOST:8000/api/validator/submissions/?status=pending"
```

Проверить переменные:

```
LMS_BASE_URL
LMS_API_TOKEN
VALIDATOR_PROGRAMMING_TOKEN
```

Восстановление:

```
cd /opt/nd_lms/code_checker
docker compose restart worker
```

Если worker не видит Docker:

```
docker ps
ls -l /var/run/docker.sock
docker compose logs --tail=200 worker
```

Проверить, что в compose проброшен:

```
/var/run/docker.sock:/var/run/docker.sock
```

5.5. Programming-задания падают из-за отсутствия sandbox image

Симптомы:

- в логах code_checker ошибка Docker image not found;
- проверки конкретного языка всегда завершаются ошибкой.

Проверка:

```
docker images | grep code-checker
cd /opt/nd_lms/code_checker
docker compose logs --tail=200 worker
```

Восстановление:

```
cd /opt/nd_lms/code_checker
docker compose --profile build_only build
docker compose restart worker
```

Если проблема только в одном языке, проверить соответствие app/config.py и docker-compose.yml .

Известный риск: java_base default может не совпадать с доступным Docker image.

5.6. Неверный token validator API

Симптомы:

- worker получает HTTP 403;
- GET /api/validator/submissions/ не возвращает задания;

- в логах LMS ошибка доступа.

Проверка:

```
grep VALIDATOR_PROGRAMMING_TOKEN /var/www/lms/lms_project/.env  
grep LMS_API_TOKEN /opt/nd_lms/code_checker/.env
```

Значения должны совпадать.

Восстановление:

```
sudo systemctl restart gunicorn  
cd /opt/nd_lms/code_checker  
docker compose restart worker
```

5.7. S3/MinIO недоступен

Симптомы:

- не загружаются видео или изображения;
- ошибки boto3/S3 в логах;
- подписанные ссылки не открываются.

Проверка:

```
grep S3_ /var/www/lms/lms_project/.env  
sudo journalctl -u gunicorn -n 200 --no-pager  
tail -n 200 /var/www/lms/log/error.log
```

Проверить:

- S3_ACCESS_KEY ;
- S3_SECRET_KEY ;
- S3_BUCKET_NAME ;
- S3_ENDPOINT ;
- доступность bucket;
- права на чтение/запись.

Восстановление:

```
sudo systemctl restart gunicorn
sudo systemctl restart celery
```

Если менялись ключи S3, перезапуск обязателен.

5.8. Видео не обрабатывается

Симптомы:

- видео долго остается в processing;
- в логах ошибки FFmpeg;
- Celery task завершается с ошибкой.

Проверка:

```
ffmpeg -version
ffprobe -version
sudo journalctl -u celery -n 300 --no-pager
```

Проверить env:

```
VIDEO_ASSETS_DIR
VIDEO_WATERMARK_DIR
VIDEO_INTRO_FILE
VIDEO_PROCESSING_MODE
VIDEO_GPU_DEVICE
```

Восстановление:

```
sudo systemctl restart celery
```

Если задача застряла, использовать management command:

```
cd /var/www/lms
source venv/bin/activate
python manage.py cleanup_video_tasks
```

5.9. Временные БД SQLChecker не очищаются

Симптомы:

- PostgreSQL или ClickHouse занимают слишком много места;
- в SQLChecker были аварийные остановки во время проверок.

Проверка PostgreSQL:

```
cd /opt/nd_lms/SQLChecker
docker compose exec postgres-db psql -U user -d postgres -c "\l"
```

Ищем базы с префиксами вида:

```
check_
```

Восстановление:

- сначала остановить активные проверки;
- удалить только временные базы, которые не используются;
- перезапустить SQLChecker.

Пример:

```
docker compose restart sql-checker-api
```

Удаление БД вручную выполнять только после проверки, что это временная БД SQLChecker, а не рабочая БД.

6. Работа с очередью проверок

6.1. Посмотреть pending-задания

```
cd /var/www/lms
source venv/bin/activate
python manage.py shell
```

```
from lms_project.apps.common.models import SubmissionQueue
SubmissionQueue.objects.filter(status="pending").count()
SubmissionQueue.objects.filter(status="pending").order_by("created_at").values("id", "user_id",
```

6.2. Посмотреть историю

```
from lms_project.apps.common.models import SubmissionHistory
SubmissionHistory.objects.order_by("-completed_at").values("queue_id", "user_id", "task_id", "t
```

6.3. Повторная обработка зависших заданий

Если worker не забрал pending-задания, сначала нужно восстановить worker.

Если запись имеет статус `processing`, но worker уже не работает, возможный ручной возврат в очередь:

```
from lms_project.apps.common.models import SubmissionQueue
SubmissionQueue.objects.filter(status="processing").update(status="pending")
```

Ручной возврат выполнять только после проверки, что старый worker не продолжает обработку. Иначе возможна двойная проверка одного решения.

6.4. Удаление испорченной записи

Если payload поврежден и задача не может быть обработана:

```
from lms_project.apps.common.models import SubmissionQueue
item = SubmissionQueue.objects.get(id="UUID")
item.status = "failed"
item.save(update_fields=["status"])
```

Удалять запись из очереди вручную стоит только после сохранения диагностической информации.

7. Backup и восстановление данных

7.1. MySQL LMS

Backup:

```
mysqldump -u lms_user -p lms > lms_$(date +%F_%H-%M).sql
```

Restore:

```
mysql -u lms_user -p lms < lms_YYYY-MM-DD_HH-MM.sql
```

После restore:

```
cd /var/www/lms
source venv/bin/activate
python manage.py migrate
sudo systemctl restart gunicorn celery celery-beat
```

7.2. S3/MinIO

Нужно регулярно сохранять:

- загруженные видео;
- изображения;
- презентации;
- материалы шагов;
- файлы SQL dump, если они хранятся в S3.

Регламент зависит от провайдера S3. Минимально должны быть включены:

- versioning или регулярный backup bucket;
- контроль прав доступа;
- проверка восстановления тестового файла.

7.3. SQLChecker volumes

Docker volumes:

```
postgres_data
clickhouse_data
```

В production нужно определить, являются ли эти данные временными или требуют backup. Если SQLChecker использует только временные БД для проверок, backup может быть не нужен, но volumes должны очищаться контролируемо.

8. Обновление проекта

8.1. LMS

```
cd /var/www/lms
source venv/bin/activate
pip install -r requirements.txt
python manage.py migrate
python manage.py collectstatic --noinput
sudo systemctl restart gunicorn celery celery-beat
```

Проверка:

```
python manage.py check
sudo journalctl -u gunicorn -n 100 --no-pager
```

8.2. SQLChecker

```
cd /opt/nd_lms/SQLChecker
docker compose build
docker compose up -d
curl http://localhost:8000/health
```

8.3. code_checker

```
cd /opt/nd_lms/code_checker
docker compose build api worker
docker compose --profile build_only build
docker compose up -d api worker
docker compose logs --tail=100 worker
```

9. Минимальный мониторинг

Нужно мониторить:

- доступность LMS HTTP;
- status systemd-сервисов `gunicorn` , `celery` , `celery-beat` , `redis-server` , `mysql` ;
- `SQLChecker /health` ;
- `SQLChecker /stats` ;
- состояние контейнеров `SQLChecker` ;
- состояние контейнеров `code_checker` ;
- рост `submission_queue` ;
- ошибки в `validator_results.log` ;
- свободное место на диске;
- размер Docker volumes;
- доступность S3.

Минимальные пороги:

Метрика	Условие тревоги
LMS HTTP	Нет ответа более 1-2 минут
<code>SQLChecker /health</code>	<code>status != healthy</code>
<code>code_checker worker</code>	Контейнер не запущен
<code>submission_queue pending</code>	Растет дольше 10-15 минут
Диск	Свободно менее 15-20%
Ошибки validator	Повторяющиеся 500/403

10. Регламент реакции на инциденты

1. Зафиксировать время, симптом и затронутый сервис.
2. Проверить `health/status` соответствующего сервиса.
3. Посмотреть последние 100-300 строк логов.
4. Проверить связанные зависимости: БД, Redis, Docker, S3, сеть.
5. Выполнить минимальный безопасный `restart` только затронутого сервиса.
6. Проверить, исчез ли симптом.
7. Если проблема с данными, сделать backup перед ручными правками.

8. После восстановления зафиксировать причину и выполненные действия.

11. Что нельзя делать без отдельного подтверждения

- Удалять рабочую MySQL БД LMS.
- Очищать Docker volumes без понимания, какие данные в них находятся.
- Массово удалять записи `submission_queue`.
- Ручно менять `submission_history` без backup.
- Менять токены валидаторов без синхронного обновления `code_checker/.env`.
- Пересобирать sandbox images во время активной проверки большого потока задач без предупреждения пользователей.
- Отключать Redis без перезапуска Celery после восстановления.