SOLAR

Программный комплекс обнаружения и реагирования на мошеннические голосовые вызовы на основе анализа SIP/RTP трафика с применением технологий машинного обучения «Волна» (далее – Платформа «Волна»)

Руководство системного администратора

МОСКВА, 2025

Содержание

1. Общие сведения стр.3
2. Обновление компонентов ПО стр.4
3. Диагностика и локализация проблем в Docker Swarm стр.8
4. Эксплуатация инфраструктурных сервисов стр.19
5. Мониторинг платформы стр.30
6. Резервное копирование и восстановление стр.41
7. Безопасность стр.47
8. Приложения стр.50

1. Общие сведения

1.1 Описание платформы

Платформа "Волна" (Solar Antifraud Cloud) представляет собой систему обнаружения и предотвращения мошеннических звонков в режиме реального времени. Платформа предназначена для анализа VoIP-трафика с использованием технологий машинного обучения для выявления социальной инженерии и голосового мошенничества (vishing).

1.2 Архитектура компонентов

Платформа «Волна» построена на базе распределенной микросервисной архитектуры и включает следующие основные компоненты:

- **API** - центральный бэкенд для управления компонентами
- **CDR** – сервис обработки записей и детализации звонков
- **Classifier** – сервис транскрибирования и потоковой классификации речи
- **Importer** - сервис импорта голосовых данных из потоковой очереди (стриминг)
- **Notificator** - система управления оповещениями
- **Controller** – сервис отвечающий за управление жизненным циклом агентского ПО, реализующего функции коннектора к облачной платформе
- **UI** – графический интерфейс (пользовательский портал)

1.3 Технологический стек

- **Бэкенд:** FastAPI (Python 3.10-3.12)
- **Базы данных:** ClickHouse, PostgreSQL, SQLite
- **Сообщения:** Kafka/RedPanda
- **Контейнеризация:** Docker, Docker Compose, Docker Swarm
- **Мониторинг:** Prometheus, Grafana, Loki, Zabbix
- **Хранение:** S3-совместимое хранилище (Minio)
- **Безопасность:** HashiCorp Vault

- **CI/CD:** GitLab

1.4 Архитектура развертывания

Платформа развертывается в кластере Docker Swarm, обеспечивающем высокую доступность и масштабируемость компонентов. Компоненты распределяются по узлам кластера с учетом требований к ресурсам и специализации узлов (worker, ML-узлы с GPU).

2. Обновление компонентов ПО

2.1 Подготовка к обновлению

Перед началом обновления компонентов платформы необходимо выполнить следующие подготовительные действия:

1. **Проверка текущего состояния кластера:**

```
# Проверка статуса всех сервисов
docker stack services demo
docker stack ps demo

# Проверка состояния узлов кластера
docker node ls
docker node inspect <node-id>
```

2. **Резервное копирование критических данных:**
   - Резервное копирование базы данных ClickHouse
   - Резервное копирование конфигурации Vault
   - Сохранение текущей версии docker-compose.stack.yaml

3. **Проверка доступности новых образов:**

```
# Проверка доступности образа в registry
docker pull
registry.af.internal/antifraud/cloud/cdr:latest
docker pull
registry.af.internal/antifraud/cloud/classifier:la
test
# и т.д. для всех компонентов
```

2.2 Обновление сервисов в Docker Swarm

2.2.1 Обновление отдельного сервиса

Для обновления отдельного сервиса используется команда **docker service update:**

```
# Обновление образа сервиса
docker service update \
  --image
registry.af.internal/antifraud/cloud/cdr:v1.2.0 \
  demo_cdr

# Обновление с принудительным перезапуском
docker service update --force demo_cdr

# Обновление с изменением переменных окружения
docker service update \
  --env-add NEW_VAR=value \
  demo_cdr
```

2.2.2 Обновление через docker-compose.stack.yaml

Рекомендуемый способ обновления - через обновление файла конфигурации:

```
# 1. Обновить версии образов в
docker-compose.stack.yaml
# Например, изменить:
# image:
registry.af.internal/antifraud/cloud/cdr:latest
# на:
# image:
registry.af.internal/antifraud/cloud/cdr:v1.2.0

# 2. Развернуть обновленный стек
docker stack deploy -c docker-compose.stack.yaml demo

# 3. Проверить статус обновления
docker service ps demo_cdr
```

2.2.3 Rolling Update (постепенное обновление)

Docker Swarm по умолчанию использует стратегию rolling update:

```
# Настройка параметров обновления
docker service update \
  --update-delay 30s \
  --update-parallelism 1 \
  --update-failure-action rollback \
  demo_cdr

# Параметры:
# --update-delay: задержка между обновлением каждого контейнера
# --update-parallelism: количество контейнеров, обновляемых одновременно
# --update-failure-action: действие при ошибке (rollback, pause, continue)
```

2.3 Проверка работоспособности после обновления

После обновления необходимо проверить работоспособность обновленных сервисов:

```
# 1. Проверка статуса сервисов
docker service ps demo_cdr --no-trunc

# 2. Проверка healthcheck эндпоинтов
curl http://localhost:8800/healthcheck  # CDR
curl http://localhost:6061/healthcheck  # Importer
curl http://localhost:8803/healthcheck  # Notificator
curl http://localhost:6062/healthcheck  # Classifier

# 3. Проверка логов на наличие ошибок
docker service logs demo_cdr --tail 100 | grep -i error

# 4. Проверка метрик в Prometheus/Grafana
# Убедиться, что метрики собираются корректно
```

2.4 Откат изменений

В случае проблем после обновления можно выполнить откат:

```
# Откат к предыдущей версии образа
docker service rollback demo_cdr

# Или явное указание предыдущей версии
docker service update \
  --image
registry.af.internal/antifraud/cloud/cdr:v1.1.0 \
  demo_cdr

# Проверка статуса отката
docker service ps demo_cdr
```

2.5 Обновление инфраструктурных компонентов

Обновление ClickHouse

```
# 1. Резервное копирование данных
docker exec clickhouse clickhouse-client --query
"BACKUP DATABASE cdr TO Disk('backups',
'backup_before_update.zip')"

# 2. Обновление образа
docker service update --image
clickhouse/clickhouse-server:23.8 demo_clickhouse

# 3. Проверка версии после обновления
docker exec clickhouse clickhouse-client --query
"SELECT version()"
```

Обновление Redpanda

```
# Обновление Redpanda требует осторожности из-за
возможных изменений формата данных
docker service update --image
docker.redpanda.com/redpandadata/redpanda:v23.2.0
demo_redpanda
```

```
# Проверка статуса кластера
docker exec redpanda rpk cluster info
```

Обновление MinIO

```
docker service update --image
minio/minio:RELEASE.2024-01-01T00-00-00Z demo_minio

# Проверка доступности
curl http://localhost:9000/minio/health/live
```

2.6 Контрольный список обновления

- ✔ Выполнено резервное копирование всех критических данных
- ✔ Проверена доступность новых образов в registry
- ✔ Обновлен файл docker-compose.stack.yaml (если требуется)
- ✔ Выполнено обновление сервисов
- ✔ Проверены healthcheck эндпоинты всех сервисов
- ✔ Проверены логи на наличие ошибок
- ✔ Проверены метрики в системе мониторинга
- ✔ Выполнено тестирование основных функций платформы
- ✔ Документированы изменения и версии компонентов

3. Диагностика и локализация проблем в Docker Swarm

3.1 Мониторинг состояния кластера

3.1.1 Просмотр состояния узлов кластера

```
# Список всех узлов кластера
docker node ls

# Детальная информация об узле
docker node inspect <node-id> --pretty
```

```
# Просмотр меток узлов
docker node inspect <node-id> | grep -A 10 Labels
```

3.1.2 Просмотр состояния сервисов

```
# Список всех сервисов стека
docker stack services demo

# Детальная информация о сервисе
docker service inspect demo_cdr --pretty

# Задачи (tasks) сервиса
docker service ps demo_cdr --no-trunc

# Статус реплик сервиса
docker service ls | grep demo
```

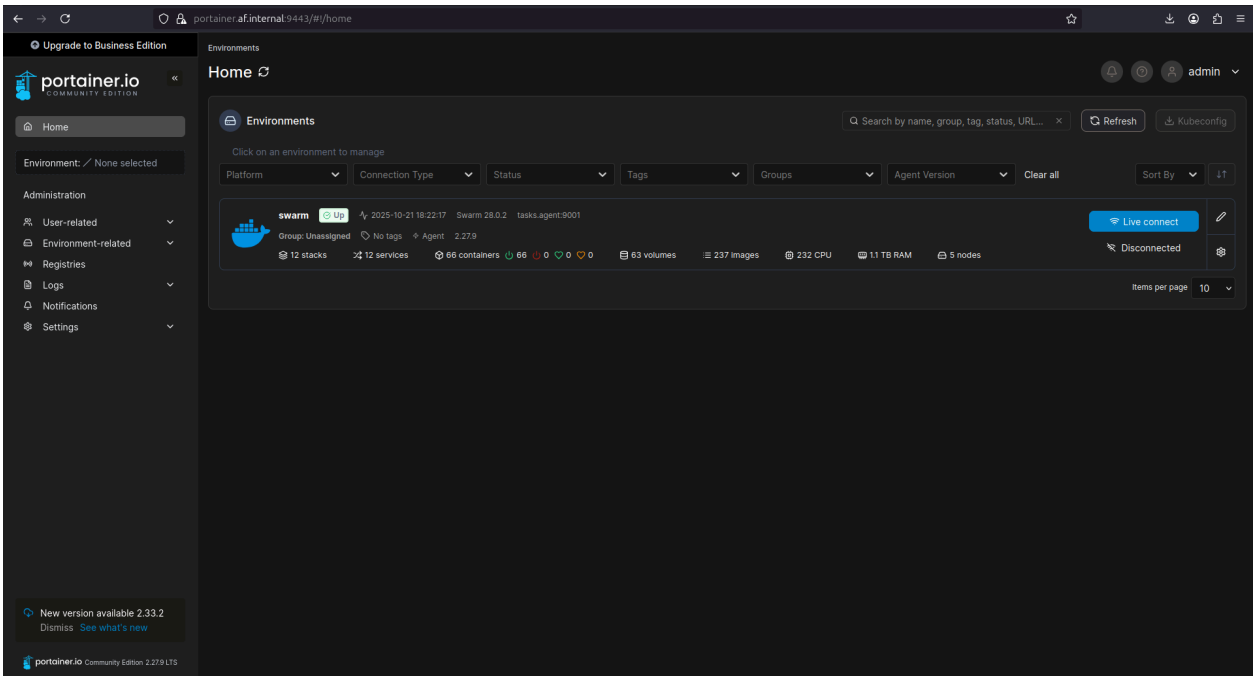


Рисунок 1 - Вывод состояния кластера Docker Swarm/ Графический интерфейс Portainer

3.2 Просмотр логов сервисов

3.2.1 Базовые команды для просмотра логов

```
# Логи конкретного сервиса
docker service logs demo_cdr --tail 100 -f
```

```
# Логи с фильтрацией по уровню
docker service logs demo_cdr 2>&1 | grep ERROR

# Логи всех реплик сервиса
docker service logs demo_cdr --tail 50 --raw

# Логи за определенный период времени
docker service logs demo_cdr --since 1h --tail 200
```

3.2.2 Анализ логов для диагностики

```
# Поиск ошибок в логах
docker service logs demo_cdr --tail 1000 | grep -i
"error\|exception\|failed"

# Подсчет ошибок
docker service logs demo_cdr --since 24h | grep -c
ERROR

# Поиск по конкретному call_id
docker service logs demo_cdr --tail 1000 | grep
"call_id_12345"

# Экспорт логов в файл для анализа
docker service logs demo_cdr --since 1h >
/tmp/cdr_logs_$(date +%Y%m%d_%H%M%S).log
```

3.2.3 Просмотр логов конкретного контейнера

```
# Получение ID контейнера
docker ps | grep demo_cdr

# Просмотр логов контейнера
docker logs <container-id> --tail 100 -f

# Просмотр логов с временными метками
docker logs <container-id> --timestamps --tail 100
```

3.3 Диагностика сетевых проблем

3.3.1 Проверка сетей Docker

```
# Список всех сетей
docker network ls

# Детальная информация о сети
docker network inspect demo-backend-network

# Проверка подключения контейнеров к сети
docker network inspect demo-backend-network | grep -A
5 Containers
```

3.3.2 Тестирование сетевого подключения

```
# Тест подключения между контейнерами
docker exec -it <container-id> ping
<target-container-name>

# Проверка DNS разрешения
docker exec -it <container-id> nslookup clickhouse

# Проверка доступности портов
docker exec -it <container-id> nc -zv clickhouse 8123

# Проверка подключения к внешним сервисам из контейнера
docker exec -it <container-id> curl -v
http://clickhouse:8123/ping
```

3.3.3 Диагностика проблем с overlay сетями

```
# Проверка состояния overlay сетей
docker network inspect demo-backend-network --format
'{{json .}}' | jq

# Проверка маршрутизации на узлах
docker node inspect <node-id> | grep -A 20 Swarm

# Пересоздание сети (крайняя мера)
docker network rm demo-backend-network
```

```
docker network create --driver overlay
demo-backend-network
```

3.4 Анализ использования ресурсов

3.4.1 Мониторинг ресурсов узлов

```
# Использование ресурсов узлами
docker node ls
docker stats $(docker ps -q)

# Детальная информация об использовании ресурсов узлом
docker node inspect <node-id> | grep -A 10 Resources

# Проверка доступных ресурсов на узле
docker system df -v
```

3.4.2 Мониторинг ресурсов контейнеров

```
# Статистика использования ресурсов контейнерами
docker stats --no-stream

# Статистика конкретного контейнера
docker stats <container-id> --no-stream

# Использование дискового пространства
docker system df
docker volume ls
docker volume inspect <volume-name>
```

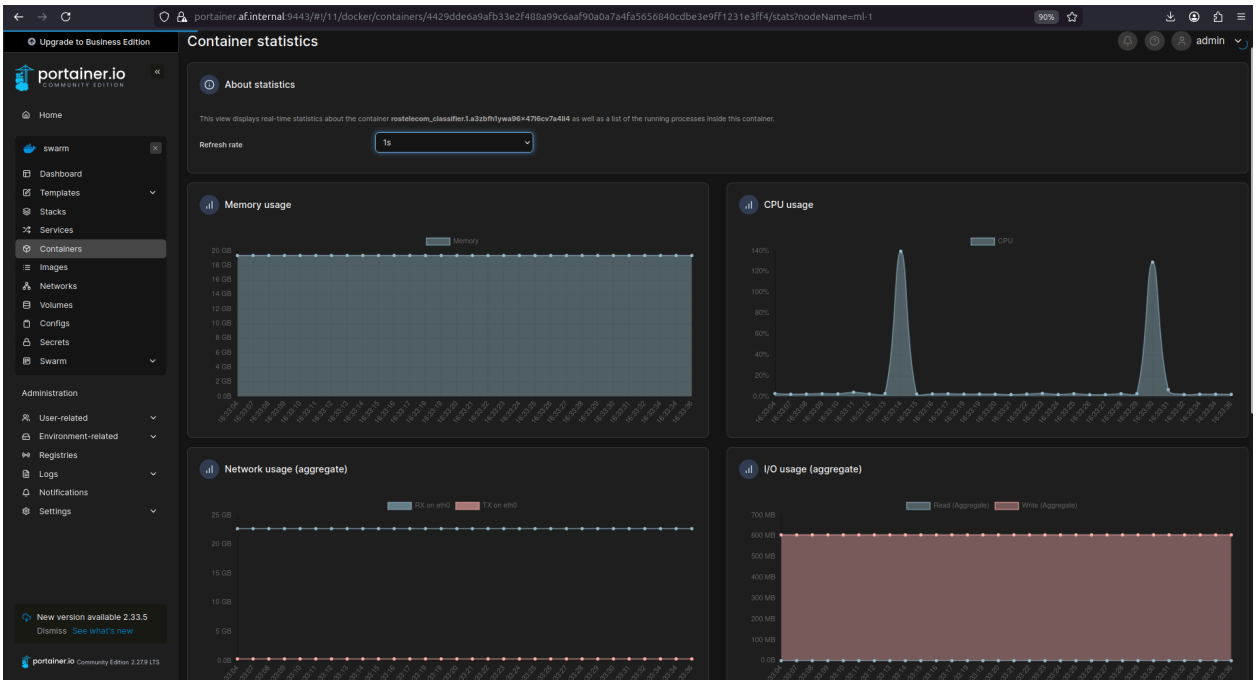


Рисунок 2 – Мониторинг реурсов кластера Docker Swarm. Portainer.

3.5 Типичные проблемы и их решения

3.5.1 Сервис не запускается или постоянно перезапускается

Симптомы:

- Статус сервиса показывает **"Restarting"** или **"Failed"**
- В логах присутствуют ошибки запуска

Диагностика:

```
# Проверка статуса задач сервиса
docker service ps demo_cdr --no-trunc

# Просмотр логов последней попытки запуска
docker service logs demo_cdr --tail 200

# Проверка конфигурации сервиса
docker service inspect demo_cdr --pretty | grep -A 20 "Command\|Env"
```

Возможные причины и решения:

1. **Ошибки в переменных окружения:**

```
# Проверка переменных окружения
docker service inspect demo_cdr --pretty | grep -A
30 Env

# Исправление переменных окружения
docker service update --env-rm BAD_VAR demo_cdr
docker service update --env-add CORRECT_VAR=value
demo_cdr
```

2. **Проблемы с подключением к зависимостям:**

```
# Проверка доступности ClickHouse
docker exec -it <cdr-container-id> curl
http://clickhouse:8123/ping

# Проверка доступности Kafka
docker exec -it <cdr-container-id> nc -zv redpanda
9092
```

3. **Недостаточно ресурсов:**

```
# Проверка доступных ресурсов на узле
docker node inspect <node-id> | grep -A 10
Resources

# Увеличение лимитов ресурсов
docker service update \
--limit-memory 16G \
--limit-cpu 8 \
demo_cdr
```

3.5.2 Проблемы с подключением к Vault

Симптомы:

- Ошибки аутентификации в Vault
- Сервисы не могут получить секреты

Диагностика:

```
# Проверка доступности Vault
curl -k ${VAULT_ADDR}/v1/sys/health

# Проверка переменных окружения Vault
```

```
docker service inspect demo_cdr --pretty | grep -A 10
VAULT

# Проверка правильности Role ID и Secret ID
vault read auth/approle/role/antifraud/role-id
```

Решение:

```
# Обновление Role ID и Secret ID в сервисе
docker service update \
  --env-add VAULT_ROLE_ID=<new-role-id> \
  --env-add VAULT_SECRET_ID=<new-secret-id> \
  demo_cdr

# Или синхронизация настроек через API сервиса
curl -X PUT http://localhost:8800/settings/sync
```

3.5.3 Classifier не использует GPU

Симптомы:

- Низкая производительность Classifier
- В логах отсутствуют упоминания GPU

Диагностика:

```
# Проверка доступности GPU в контейнере
docker exec -it <classifier-container-id> nvidia-smi

# Проверка настройки generic resources на узле
docker node inspect <node-id> | grep -A 10
GenericResources

# Проверка переменных окружения GPU
docker service inspect demo_classifier --pretty | grep
CUDA
```

Решение:

```
# Убедиться, что узел имеет GPU и настроен
nvidia-container-toolkit
# Проверка на узле:
nvidia-smi
docker run --rm --gpus all nvidia/cuda:11.0-base
```

```
nvidia-smi

# Обновление сервиса с правильными настройками GPU
docker service update \
  --env-add CUDA_VISIBLE_DEVICES=0,1 \
  --env-add NVIDIA_VISIBLE_DEVICES=0,1 \
  demo_classifier
```

3.5.4 Проблемы с записью в ClickHouse

Симптомы:

- Ошибки записи данных в ClickHouse
- Данные не появляются в таблицах

Диагностика:

```
# Проверка подключения к ClickHouse
docker exec -it clickhouse clickhouse-client --query
"SELECT version()"

# Проверка прав доступа пользователя
docker exec -it clickhouse clickhouse-client --query
"SHOW GRANTS FOR admin"

# Проверка существования таблиц
docker exec -it clickhouse clickhouse-client --query
"SHOW TABLES FROM cdr"

# Проверка последних записей
docker exec -it clickhouse clickhouse-client --query
"SELECT count() FROM cdr.calls"
```

Решение:

```
# Пересоздание таблиц при необходимости
docker exec -i clickhouse clickhouse-client <
clickhouse.sql

# Проверка и исправление прав доступа
docker exec -it clickhouse clickhouse-client --query
"GRANT ALL ON cdr.* TO admin"
```

3.5.5 Проблемы с сетью между сервисами

Симптомы:

- Сервисы не могут связаться друг с другом
- Таймауты при обращении к другим сервисам

Диагностика:

```
# Проверка сетей
docker network inspect demo-backend-network

# Проверка DNS разрешения
docker exec -it <container-id> nslookup clickhouse

# Проверка доступности портов
docker exec -it <container-id> nc -zv clickhouse 8123
```

Решение:

```
# Переподключение сервиса к сети
docker service update --network-rm
demo-backend-network demo_cdr
docker service update --network-add
demo-backend-network demo_cdr

# Пересоздание сети (крайняя мера)
docker network rm demo-backend-network
docker network create --driver overlay
demo-backend-network
docker stack deploy -c docker-compose.stack.yaml demo
```

3.6 Использование инструментов диагностики

3.6.1 Docker Events для мониторинга событий

```
# Просмотр событий кластера в реальном времени
docker events --filter 'service=demo_cdr'

# Просмотр событий за последний час
docker events --since 1h --filter 'type=container'
```

3.6.2 Инспекция контейнеров

```
# Детальная информация о контейнере
docker inspect <container-id>

# Проверка конфигурации контейнера
docker inspect <container-id> --format '{{json
.Config}}' | jq

# Проверка сетевых настроек
docker inspect <container-id> --format '{{json
.NetworkSettings}}' | jq
```

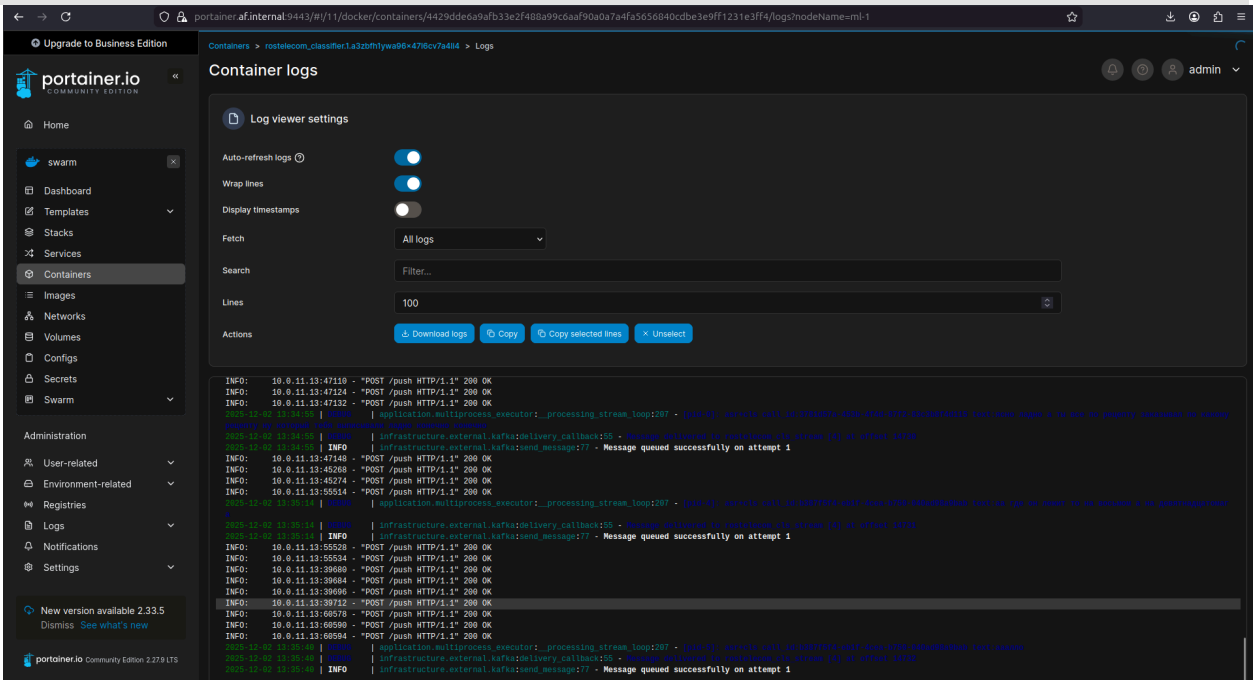


Рисунок 3 – Мониторинг состояния контейнера Docker Swarm. Portainer.

3.7 Контрольный список диагностики

При возникновении проблем выполните следующие шаги:

- ✔ Проверен статус всех узлов кластера (docker node ls)
- ✔ Проверен статус всех сервисов (docker stack services demo)

- ✔ Просмотрены логи проблемного сервиса (docker service logs)
- ✔ Проверена доступность зависимостей (ClickHouse, Kafka, Vault)
- ✔ Проверено использование ресурсов (docker stats)
- ✔ Проверены сетевые подключения между контейнерами
- ✔ Проверены переменные окружения сервиса
- ✔ Проверены healthcheck эндпоинты
- ✔ Проверены метрики в системе мониторинга
- ✔ Документирована проблема и решение

4. Эксплуатация инфраструктурных сервисов

4.1 Redpanda (Kafka)

4.1.1 Управление топиками

```
# Список всех топиков
docker exec -it redpanda rpk topic list

# Создание нового топика
docker exec -it redpanda rpk topic create sip_stream \
  --partitions 3 \
  --replicas 2

# Описание топика
docker exec -it redpanda rpk topic describe sip_stream

# Удаление топика (осторожно!)
docker exec -it redpanda rpk topic delete sip_stream
```

4.1.2 Мониторинг Redpanda

```
# Информация о кластере
docker exec -it redpanda rpk cluster info
```

```
# Статистика брокера
docker exec -it redpanda rpk cluster info --brokers

# Просмотр метрик
docker exec -it redpanda rpk cluster info --metrics

# Проверка здоровья кластера
docker exec -it redpanda rpk cluster health
```

4.1.3 Управление сообщениями

```
# Просмотр сообщений из топика
docker exec -it redpanda rpk topic consume sip_stream
--num 10

# Отправка тестового сообщения
docker exec -it redpanda rpk topic produce sip_stream
\
  --key test-key \
  --value
'{"call_id":"test-123","from":"+79001234567","to":"+79
007654321"}'

# Просмотр offset'ов consumer групп
docker exec -it redpanda rpk group describe cdr
```

4.1.4 Резервное копирование и восстановление

```
# Экспорт сообщений из топика
docker exec -it redpanda rpk topic consume sip_stream
\
  --num 1000 \
  --format json > /backup/sip_stream_$(date
+%Y%m%d).json

# Восстановление сообщений в топик
cat /backup/sip_stream_20250101.json | \
  docker exec -i redpanda rpk topic produce sip_stream
--format json
```

4.1.5 Оптимизация производительности

```
# Настройка retention политики
docker exec -it redpanda rpk topic alter-config
sip_stream \
  --set retention.ms=604800000 \
  --set retention.bytes=10737418240

# Увеличение количества партиций для масштабирования
docker exec -it redpanda rpk topic alter sip_stream
--partitions 6
```

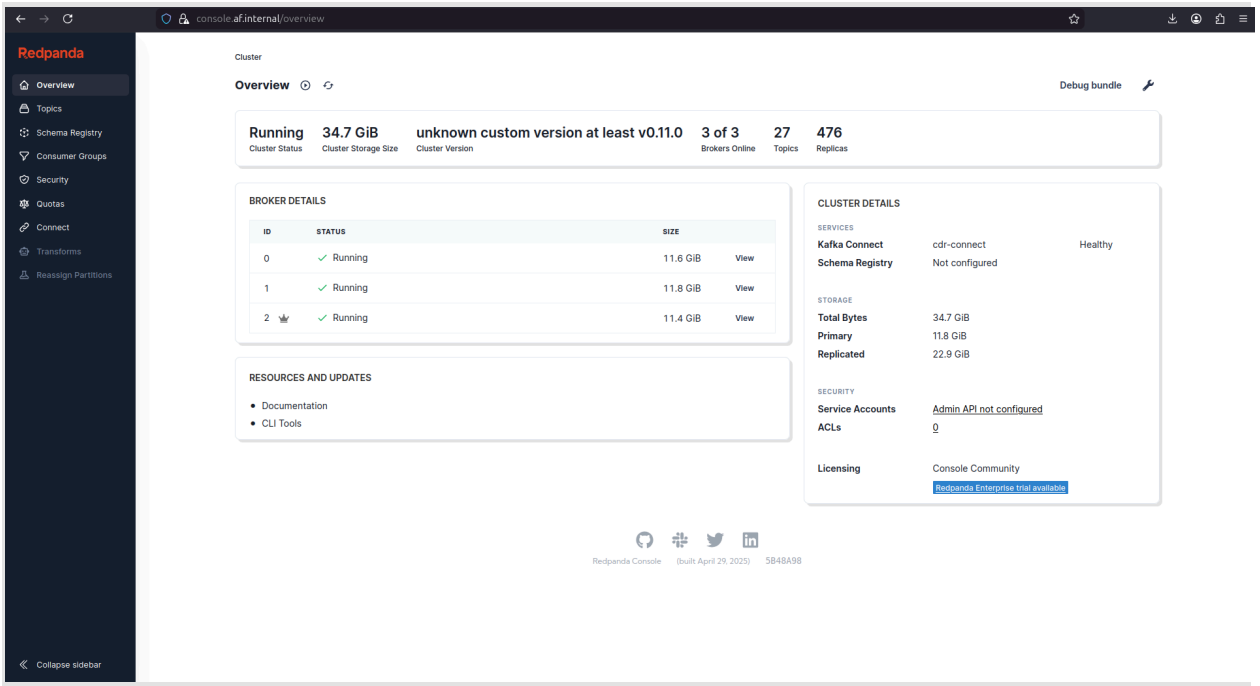


Рисунок 4 – Redpanda Console. Мониторинг состояния кластера

4.2 ClickHouse

4.2.1 Управление таблицами

```
# Подключение к ClickHouse
docker exec -it clickhouse clickhouse-client

# Список всех таблиц
docker exec -it clickhouse clickhouse-client --query
"SHOW TABLES FROM cdr"

# Структура таблицы
docker exec -it clickhouse clickhouse-client --query
"DESCRIBE TABLE cdr.calls"

# Создание новой таблицы
docker exec -it clickhouse clickhouse-client <<EOF
CREATE TABLE IF NOT EXISTS cdr.test_table (
    id UInt64,
    name String,
    created_at DateTime
) ENGINE = MergeTree
```

```
ORDER BY id;
EOF
```

4.2.2 Оптимизация запросов

```
# Анализ производительности запроса
docker exec -it clickhouse clickhouse-client --query "
EXPLAIN PLAN SELECT * FROM cdr.calls WHERE start_time
> now() - INTERVAL 1 DAY"

# Просмотр статистики таблицы
docker exec -it clickhouse clickhouse-client --query "
SELECT
    table,
    formatReadableSize(sum(bytes)) as size,
    sum(rows) as rows
FROM system.parts
WHERE database = 'cdr'
GROUP BY table"

# Проверка фрагментации данных
docker exec -it clickhouse clickhouse-client --query "
SELECT
    table,
    count() as parts,
    sum(rows) as rows
FROM system.parts
WHERE database = 'cdr' AND active
GROUP BY table"
```

4.2.3 Управление партициями

```
# Просмотр партиций таблицы
docker exec -it clickhouse clickhouse-client --query "
SELECT
    partition,
    count() as parts,
    sum(rows) as rows,
    formatReadableSize(sum(bytes)) as size
FROM system.parts
```

```
WHERE database = 'cdr' AND table = 'calls'
GROUP BY partition
ORDER BY partition DESC"

# Удаление старых партиций (осторожно!)
docker exec -it clickhouse clickhouse-client --query "
ALTER TABLE cdr.calls DROP PARTITION '2024-01-01'"
```

4.2.4 Резервное копирование

```
# Создание резервной копии базы данных
docker exec clickhouse clickhouse-client --query "
BACKUP DATABASE cdr TO Disk('backups', 'backup_$(date
+%Y%m%d_%H%M%S).zip')"

# Создание резервной копии конкретной таблицы
docker exec clickhouse clickhouse-client --query "
BACKUP TABLE cdr.calls TO Disk('backups',
'calls_backup_$(date +%Y%m%d).zip')"

# Восстановление из резервной копии
docker exec clickhouse clickhouse-client --query "
RESTORE DATABASE cdr FROM Disk('backups',
'backup_20250101_120000.zip')"
```

4.2.5 Мониторинг производительности

```
# Активные запросы
docker exec -it clickhouse clickhouse-client --query "
SELECT
    query_id,
    user,
    query,
    elapsed,
    read_rows,
    read_bytes,
    formatReadableSize(read_bytes) as read_size
FROM system.processes
ORDER BY elapsed DESC"
```

```
# Медленные запросы из логов
docker exec -it clickhouse clickhouse-client --query "
SELECT
    query_start_time,
    query_duration_ms,
    query,
    user,
    exception
FROM system.query_log
WHERE type = 'QueryFinish' AND query_duration_ms >
1000
ORDER BY query_duration_ms DESC
LIMIT 10"
```

4.3 MinIO (S3-совместимое хранилище)

4.3.1 Управление bucket'ами

```
# Доступ к консоли MinIO через браузер
# http://localhost:9001
# Логин: minioadmin / minioadmin (по умолчанию)

# Использование mc (MinIO Client)
# Установка mc
wget
https://dl.min.io/client/mc/release/linux-amd64/mc
chmod +x mc
sudo mv mc /usr/local/bin/

# Настройка подключения
mc alias set local http://localhost:9000 minioadmin
minioadmin

# Список bucket'ов
mc ls local

# Создание bucket'а
mc mb local/audio-backup
```

```
# Удаление bucket'а (осторожно!)
mc rb local/old-bucket
```

4.3.2 Управление объектами

```
# Загрузка файла
mc cp /path/to/file.wav local/audio/file.wav

# Загрузка директории
mc cp --recursive /path/to/audio/ local/audio/

# Список объектов в bucket'е
mc ls local/audio --recursive

# Скачивание файла
mc cp local/audio/file.wav /path/to/download/

# Удаление объекта
mc rm local/audio/file.wav

# Удаление объектов старше определенного периода
mc find local/audio --older-than 30d --exec "mc rm {}"
```

4.3.3 Политики доступа

```
# Создание политики доступа только на чтение
cat > /tmp/readonly-policy.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["s3:GetObject"],
      "Resource": ["arn:aws:s3:::audio/*"]
    }
  ]
}
EOF

mc admin policy create local readonly-policy
/tmp/readonly-policy.json
```

```
# Создание пользователя
mc admin user add local readonly-user
readonly-password

# Привязка политики к пользователю
mc admin policy attach local readonly-policy --user
readonly-user
```

4.3.4 Мониторинг хранилища

```
# Статистика использования хранилища
mc du local/audio

# Информация о bucket'е
mc stat local/audio

# Проверка целостности данных
mc diff local/audio /backup/audio

# Мониторинг через API
curl http://localhost:9000/minio/health/live
curl http://localhost:9000/minio/health/ready
```

4.3.5 Резервное копирование

```
# Синхронизация bucket'а с локальной директорией
mc mirror local/audio /backup/audio

# Репликация между MinIO серверами
mc replicate add local/audio \
  --remote-bucket remote/audio \
  --remote-endpoint https://remote-minio:9000

# Автоматическое резервное копирование через cron
# Добавить в crontab:
# 0 2 * * * mc mirror local/audio /backup/audio-$(date
+\%Y\%m\%d)
```

4.4 HashiCorp Vault

4.4.1 Управление секретами

```
# Проверка статуса Vault
vault status

# Установка переменных окружения
export VAULT_ADDR='https://vault.af.internal'
export VAULT_TOKEN='**************'

# Просмотр секретов
vault kv get demo/settings

# Создание/обновление секретов
vault kv put demo/settings \
    kafka_url="redpanda:9092" \
    clickhouse_host="clickhouse" \
    clickhouse_password="**************"

# Удаление секрета
vault kv delete demo/settings
```

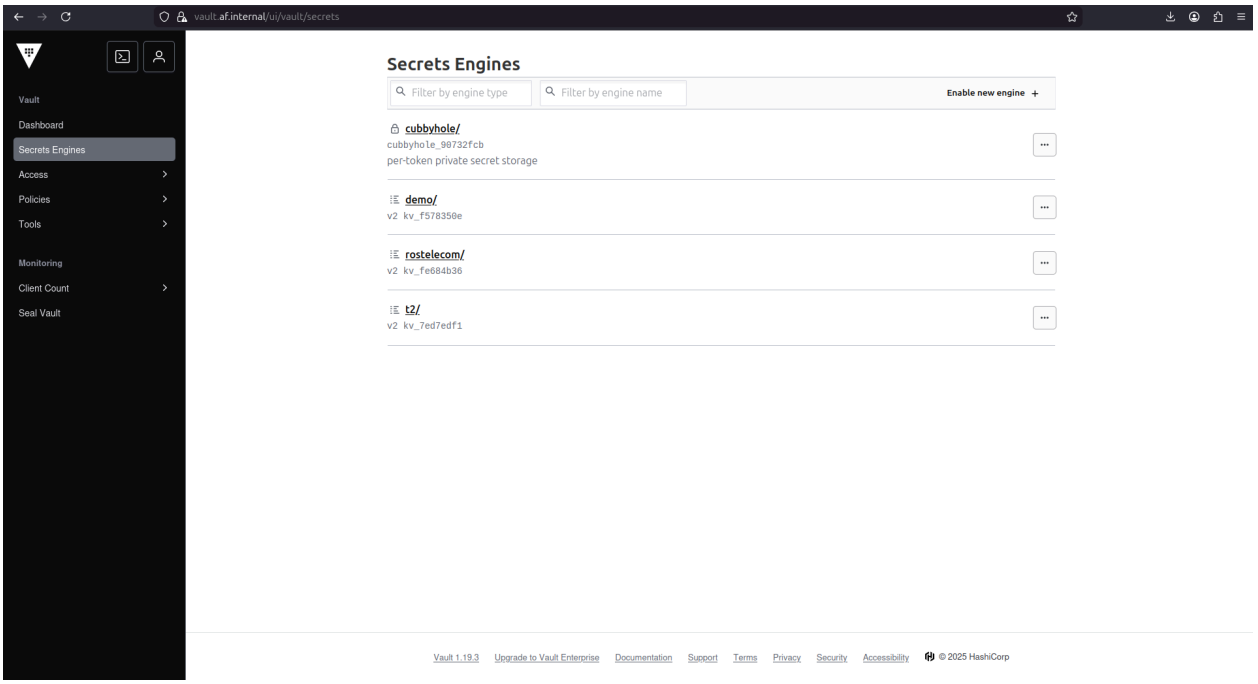


Рисунок 5 – Vault. Хранилища секретов

4.4.2 Управление AppRole

```
# Список всех AppRole
vault list auth/approle/role

# Создание нового AppRole
vault write auth/approle/role/antifraud \
    token_policies="antifraud-policy" \
    token_ttl=1h \
    token_max_ttl=4h

# Получение Role ID
vault read auth/approle/role/antifraud/role-id

# Генерация нового Secret ID
vault write -f auth/approle/role/antifraud/secret-id

# Просмотр настроек AppRole
vault read auth/approle/role/antifraud
```

4.4.3 Ротация секретов

```
# Ротация пароля ClickHouse
# 1. Создать новый пароль в ClickHouse
docker exec -it clickhouse clickhouse-client --query
"ALTER USER admin IDENTIFIED BY '***************'"

# 2. Обновить пароль в Vault
vault kv put demo/settings
clickhouse_password="***************"

# 3. Синхронизировать настройки в сервисах
curl -X PUT http://localhost:8800/settings/sync

# Ротация Secret ID для AppRole
vault write -f auth/approle/role/antifraud/secret-id
# Обновить VAULT_SECRET_ID в docker-compose.stack.yaml
или через docker service update
```

4.4.4 Резервное копирование Vault

```
# Экспорт всех секретов (требует root токен)
vault operator export -format=json >
vault_backup_$(date +%Y%m%d).json

# Резервное копирование через API
curl \
    --header "X-Vault-Token: $VAULT_TOKEN" \
    --request GET \
    $VAULT_ADDR/v1/sys/export \
    > vault_backup_$(date +%Y%m%d).json

# Восстановление из резервной копии
vault operator import vault_backup_20250101.json
```

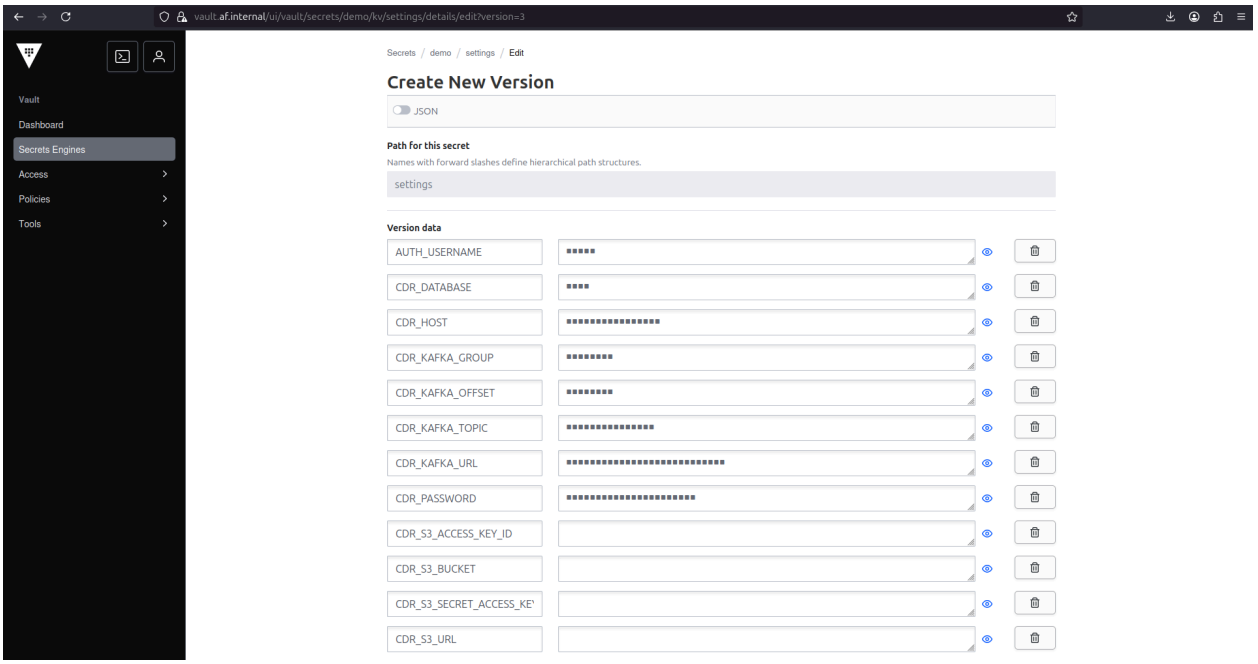


Рисунок 6 – Vault. Управление секретами и переменными окружения экземпляра сервисов платофрмы "Волна"

4.4.5 Аудит и мониторинг

```
# Включение аудита
vault audit enable file
file_path=/var/log/vault_audit.log

# Просмотр логов аудита
```

```
tail -f /var/log/vault_audit.log | jq

# Мониторинг использования токенов
vault list auth/token/accessors

# Информация о токене
vault token lookup <token>
```

4.6 Контрольный список эксплуатации инфраструктурных сервисов

Ежедневные задачи:

- ✔ Проверка статуса всех инфраструктурных сервисов
- ✔ Мониторинг использования дискового пространства
- ✔ Проверка логов на критические ошибки
- ✔ Проверка метрик производительности

Еженедельные задачи:

- ✔ Анализ производительности запросов ClickHouse
- ✔ Проверка retention политик в Redpanda
- ✔ Анализ использования хранилища MinIO
- ✔ Проверка политик доступа в Vault

Ежемесячные задачи:

- ✔ Резервное копирование всех критических данных
- ✔ Оптимизация таблиц ClickHouse
- ✔ Ротация секретов в Vault
- ✔ Анализ и очистка старых данных

5. Мониторинг платформы

5.1 Настройка Prometheus для сбора метрик

5.1.1 Конфигурация Prometheus

Создайте файл **prometheus.yml:**

```
global:
  scrape_interval: 15s
  evaluation_interval: 15s
  external_labels:
    cluster: 'demo-production'

rule_files:
  - "alert_rules.yml"

scrape_configs:
  # Мониторинг Docker Swarm
  - job_name: 'docker-swarm'
    static_configs:
      - targets: ['localhost:9323']
    metrics_path: '/metrics'

  # Мониторинг CDR-сервиса
  - job_name: 'demo-cdr'
    static_configs:
      - targets: ['cdr:8800']
    metrics_path: '/metrics'
    scrape_interval: 30s

  # Мониторинг Classifier-сервиса
  - job_name: 'demo-classifier'
    static_configs:
      - targets: ['classifier:6062']
    metrics_path: '/metrics'
    scrape_interval: 30s

  # Мониторинг Notificator-сервиса
  - job_name: 'demo-notificator'
    static_configs:
      - targets: ['notificator:8803']
    metrics_path: '/metrics'
    scrape_interval: 30s

  # Мониторинг Importer-сервиса
```

```
  - job_name: 'demo-importer'
    static_configs:
      - targets: ['importer:6061']
    metrics_path: '/metrics'
    scrape_interval: 30s

  # Мониторинг ClickHouse
  - job_name: 'clickhouse'
    static_configs:
      - targets: ['clickhouse:9363']
    metrics_path: '/metrics'

  # Мониторинг Redpanda
  - job_name: 'redpanda'
    static_configs:
      - targets: ['redpanda:9644']
    metrics_path: '/metrics'

  # Мониторинг MinIO
  - job_name: 'minio'
    static_configs:
      - targets: ['minio:9000']
    metrics_path: '/minio/v2/metrics/cluster'
```

5.1.2 Развертывание Prometheus в Docker Swarm

Добавьте в **docker-compose.stack.yaml**:

```
  prometheus:
    image: prom/prometheus:latest
    volumes:
      -
./prometheus.yml:/etc/prometheus/prometheus.yml
      - prometheus-data:/prometheus
    command:
      - '--config.file=/etc/prometheus/prometheus.yml'
      - '--storage.tsdb.path=/prometheus'
      -
'--web.console.libraries=/usr/share/prometheus/console
_libraries'
```

```
      -
'--web.console.templates=/usr/share/prometheus/console
s'
    ports:
      - "9090:9090"
    networks:
      - demo-backend-network
    deploy:
      placement:
        constraints:
          - "node.labels.TAG == worker"
```

5.1.3 Проверка сбора метрик

```
# Проверка доступности Prometheus
curl http://localhost:9090/-/healthy

# Просмотр собранных метрик
curl http://localhost:9090/api/v1/targets

# Проверка конкретной метрики
curl 'http://localhost:9090/api/v1/query?query=up'
```

5.2 Создание дашбордов в Grafana

5.2.1 Настройка источника данных

1. Войдите в Grafana (обычно http://localhost:3000)
2. Перейдите в Configuration → Data Sources
3. Добавьте Prometheus как источник данных:
   - URL: http://prometheus:9090
   - Access: Server (default)

5.2.2 Создание дашборда для мониторинга сервисов

Дашборд "Обзор платформы Волна":

```
{
  "dashboard": {
    "title": "Волна - Обзор платформы",
    "panels": [
      {
        "title": "Статус сервисов",
        "targets": [
```

```
          {
            "expr": "up{job=~'demo-.*'}",
            "legendFormat": "{{job}}"
          }
        ],
        "type": "stat"
      },
      {
        "title": "Запросы к CDR-сервису",
        "targets": [
          {
            "expr":
"rate(http_requests_total{job='demo-cdr'}[5m])",
            "legendFormat": "{{method}} {{endpoint}}"
          }
        ],
        "type": "graph"
      },
      {
        "title": "Использование памяти",
        "targets": [
          {
            "expr":
"container_memory_usage_bytes{name=~'demo.*'}",
            "legendFormat": "{{name}}"
          }
        ],
        "type": "graph"
      }
    ]
  }
}
```

5.2.3 Ключевые метрики для мониторинга

Метрики производительности:

- http_request_duration_seconds - время обработки запросов
- http_requests_total - количество запросов
- process_cpu_seconds_total - использование CPU

- process_resident_memory_bytes - использование памяти

Метрики бизнес-логики:

- cdr_calls_processed_total - количество обработанных звонков
- cdr_calls_failed_total - количество ошибок обработки
- classifier_processing_duration_seconds - время обработки классификации
- notificator_notifications_sent_total - количество отправленных уведомлений

Метрики инфраструктуры:

- clickhouse_query_duration_seconds - время выполнения запросов
- kafka_messages_consumed_total - количество потребленных сообщений
- minio_disk_usage_bytes - использование дискового пространства

5.3 Настройка алертов

5.3.1 Создание правил алертов

Создайте файл **alert_rules.yml**:

```
groups:
  - name: demo_alerts
    interval: 30s
    rules:
      # Алерт о недоступности сервиса
      - alert: ServiceDown
        expr: up{job=~'demo-.*'} == 0
        for: 2m
        labels:
          severity: critical
        annotations:
          summary: "Сервис {{ $labels.job }}
недоступен"
          description: "Сервис {{ $labels.job }} не
отвечает более 2 минут"

      # Алерт о высокой загрузке CPU
```

```
      - alert: HighCPUUsage
        expr:
rate(process_cpu_seconds_total{job=~'demo-.*'}[5m]) >
0.8
        for: 5m
        labels:
          severity: warning
        annotations:
          summary: "Высокая загрузка CPU в {{
$labels.job }}"
          description: "Использование CPU превышает
80% в течение 5 минут"

      # Алерт о высокой загрузке памяти
      - alert: HighMemoryUsage
        expr:
(process_resident_memory_bytes{job=~'demo-.*'} /
process_virtual_memory_max_bytes) > 0.9
        for: 5m
        labels:
          severity: warning
        annotations:
          summary: "Высокая загрузка памяти в {{
$labels.job }}"
          description: "Использование памяти превышает
90%"

      # Алерт о большом количестве ошибок
      - alert: HighErrorRate
        expr:
rate(http_requests_total{job=~'demo-.*',status=~'5..'}
[5m]) > 10
        for: 5m
        labels:
          severity: critical
        annotations:
          summary: "Высокий уровень ошибок в {{
$labels.job }}"
```

```
          description: "Более 10 ошибок в секунду в
течение 5 минут"

      # Алерт о проблемах с ClickHouse
      - alert: ClickHouseDown
        expr: up{job='clickhouse'} == 0
        for: 1m
        labels:
          severity: critical
        annotations:
          summary: "ClickHouse недоступен"
          description: "ClickHouse не отвечает более 1
минуты"

      # Алерт о проблемах с Kafka
      - alert: KafkaDown
        expr: up{job='redpanda'} == 0
        for: 1m
        labels:
          severity: critical
        annotations:
          summary: "Redpanda/Kafka недоступен"
          description: "Redpanda не отвечает более 1
минуты"

      # Алерт о переполнении диска
      - alert: DiskSpaceLow
        expr: (node_filesystem_avail_bytes /
node_filesystem_size_bytes) < 0.1
        for: 5m
        labels:
          severity: warning
        annotations:
          summary: "Мало свободного места на диске"
          description: "Свободного места менее 10% на
{{ $labels.mountpoint }}"
```

5.3.2 Настройка уведомлений

В Grafana настройте каналы уведомлений:

1. Перейдите в Alerting → Notification channels
2. Добавьте каналы уведомлений:
   - Email
   - Slack
   - Telegram (через webhook)
   - PagerDuty

Пример настройки Email:

```
notifications:
  - name: email_admin
    type: email
    settings:
      addresses: admin@af.internal
      subject: "[Волна] {{ .GroupLabels.alertname }}"
```

5.4 Мониторинг производительности компонентов

5.4.1 Мониторинг CDR-сервиса

Ключевые метрики:

- Количество обработанных CDR записей
- Время обработки записи
- Количество ошибок обработки
- Использование памяти и CPU
- Размер очереди Kafka

Запросы для Grafana:

```
# Скорость обработки CDR
rate(cdr_records_processed_total[5m])

# Среднее время обработки
histogram_quantile(0.95,
rate(cdr_processing_duration_seconds_bucket[5m]))

# Процент ошибок
rate(cdr_errors_total[5m]) /
rate(cdr_records_processed_total[5m]) * 100
```

5.4.2 Мониторинг Classifier-сервиса

Ключевые метрики:

- Количество обработанных звонков
- Время транскрибации
- Время классификации
- Использование GPU
- Размер очереди запросов

Запросы для Grafana:

```
# Скорость обработки
rate(classifier_audio_processed_total[5m])

# Время транскрибации
histogram_quantile(0.95,
rate(classifier_transcription_duration_seconds_bucket[
5m]))

# Использование GPU
nvidia_gpu_utilization{container="classifier"}
```

5.4.3 Мониторинг Notificator-сервиса

Ключевые метрики:

- Количество отправленных уведомлений по каналам
- Время доставки уведомлений
- Количество ошибок доставки
- Размер очереди событий

Запросы для Grafana:

```
# Количество уведомлений по каналам
sum by (channel)
(rate(notificator_notifications_sent_total[5m]))

# Процент успешных доставок
rate(notificator_notifications_success_total[5m]) /
rate(notificator_notifications_sent_total[5m]) * 100
```

5.5 Анализ логов через Loki

5.5.1 Настройка Loki

Добавьте в **docker-compose.stack.yaml:**

```
  loki:
    image: grafana/loki:latest
    ports:
      - "3100:3100"
    volumes:
      - ./loki-config.yml:/etc/loki/local-config.yaml
      - loki-data:/loki
    command: -config.file=/etc/loki/local-config.yaml
    networks:
      - demo-backend-network

  promtail:
    image: grafana/promtail:latest
    volumes:
      -
/var/lib/docker/containers:/var/lib/docker/containers:
ro
      - /var/run/docker.sock:/var/run/docker.sock:ro
      - ./promtail-config.yml:/etc/promtail/config.yml
    command: -config.file=/etc/promtail/config.yml
    networks:
      - demo-backend-network
```

5.5.2 Настройка Promtail

Создайте **promtail-config.yml**:

```
server:
  http_listen_port: 9080
  grpc_listen_port: 0

positions:
  filename: /tmp/positions.yaml

clients:
  - url: http://loki:3100/loki/api/v1/push

scrape_configs:
  - job_name: docker
    docker_sd_configs:
```

```
      - host: unix:///var/run/docker.sock
        refresh_interval: 5s
    relabel_configs:
      - source_labels:
['__meta_docker_container_name']
        regex: '/(.*)'
        target_label: 'container'
      - source_labels:
['__meta_docker_container_label_com_docker_swarm_servi
ce_name']
        target_label: 'service'
```

5.5.3 Использование LogQL в Grafana

Примеры запросов:

```
# Все ошибки из CDR-сервиса
{service="demo_cdr"} |= "ERROR"

# Ошибки за последний час
{service="demo_cdr"} |= "ERROR" [1h]

# Подсчет ошибок по типу
sum by (level) (count_over_time({service="demo_cdr"}
|= "ERROR" [5m]))

# Поиск по call_id
{service="demo_cdr"} |~ "call_id_12345"
```

5.6 Контрольный список мониторинга

Ежедневные проверки:

- ✔ Проверка статуса всех сервисов в Grafana
- ✔ Просмотр критических алертов
- ✔ Анализ метрик производительности
- ✔ Проверка использования ресурсов

Еженедельные проверки:

- ✔ Анализ трендов производительности
- ✔ Проверка эффективности алертов
- ✔ Обновление дашбордов при необходимости
- ✔ Анализ логов на аномалии

Ежемесячные проверки:

- ✔ Аудит настроек мониторинга
- ✔ Оптимизация запросов Prometheus
- ✔ Обновление правил алертов
- ✔ Резервное копирование конфигурации мониторинга

6. Резервное копирование и восстановление

6.1 Стратегии резервного копирования

6.1.1 Резервное копирование ClickHouse

Автоматическое резервное копирование:

Создайте скрипт **/opt/scripts/backup_clickhouse.sh:**

```
#!/bin/bash
BACKUP_DIR="/backup/clickhouse"
DATE=$(date +%Y%m%d_%H%M%S)
RETENTION_DAYS=30

# Создание резервной копии
docker exec clickhouse clickhouse-client --query "
BACKUP DATABASE cdr TO Disk('backups',
'backup_${DATE}.zip')"

# Копирование резервной копии на внешний носитель
docker cp
clickhouse:/var/lib/clickhouse/disks/backups/backup_${
DATE}.zip \
  ${BACKUP_DIR}/backup_${DATE}.zip

# Удаление старых резервных копий
```

```
find ${BACKUP_DIR} -name "backup_*.zip" -mtime
+${RETENTION_DAYS} -delete

echo "Резервное копирование ClickHouse завершено:
backup_${DATE}.zip"
```

Настройка cron для автоматического резервного копирования:

```
# Добавить в crontab (ежедневно в 2:00)
0 2 * * * /opt/scripts/backup_clickhouse.sh >>
/var/log/backup_clickhouse.log 2>&1
```

6.1.2 Резервное копирование Redpanda/Kafka

Экспорт сообщений из топиков:

```
#!/bin/bash
BACKUP_DIR="/backup/kafka"
DATE=$(date +%Y%m%d_%H%M%S)
TOPICS=("sip_stream" "cdr_events" "notifications")

for topic in "${TOPICS[@]}"; do
  docker exec redpanda rpk topic consume ${topic} \
    --num 1000000 \
    --format json >
${BACKUP_DIR}/${topic}_${DATE}.json

  # Сжатие архива
  gzip ${BACKUP_DIR}/${topic}_${DATE}.json
done

echo "Резервное копирование Kafka завершено"
```

6.1.3 Резервное копирование MinIO

Синхронизация bucket'ов:

```
#!/bin/bash
BACKUP_DIR="/backup/minio"
DATE=$(date +%Y%m%d_%H%M%S)

# Синхронизация bucket'a audio
```

```
mc mirror local/audio ${BACKUP_DIR}/audio_${DATE}/

# Создание архива
tar -czf ${BACKUP_DIR}/audio_${DATE}.tar.gz
${BACKUP_DIR}/audio_${DATE}/

# Удаление временной директории
rm -rf ${BACKUP_DIR}/audio_${DATE}/

echo "Резервное копирование MinIO завершено"
```

6.1.4 Резервное копирование Vault

Экспорт секретов:

```
#!/bin/bash
BACKUP_DIR="/backup/vault"
DATE=$(date +%Y%m%d_%H%M%S)
VAULT_ADDR="https://vault.af.internal"
VAULT_TOKEN="***************"

export VAULT_ADDR
export VAULT_TOKEN

# Экспорт всех секретов
vault operator export -format=json >
${BACKUP_DIR}/vault_backup_${DATE}.json

# Шифрование резервной копии
gpg --encrypt --recipient admin@example.com
${BACKUP_DIR}/vault_backup_${DATE}.json

# Удаление незашифрованной копии
rm ${BACKUP_DIR}/vault_backup_${DATE}.json

echo "Резервное копирование Vault завершено"
```

6.1.5 Резервное копирование конфигурации

Сохранение конфигурационных файлов:

```
#!/bin/bash
BACKUP_DIR="/backup/config"
DATE=$(date +%Y%m%d_%H%M%S)

# Создание архива конфигурации
tar -czf ${BACKUP_DIR}/config_${DATE}.tar.gz \
  /opt/demo/config/docker-compose.stack.yaml \
  /opt/demo/config/stack.env \
  /opt/demo/config/prometheus.yml \
  /opt/demo/config/alert_rules.yml

echo "Резервное копирование конфигурации завершено"
```

6.2 Восстановление данных

6.2.1 Восстановление ClickHouse

```
# Остановка сервисов, использующих ClickHouse
docker service scale demo_cdr=0
docker service scale demo_notificator=0

# Восстановление из резервной копии
docker exec clickhouse clickhouse-client --query "
RESTORE DATABASE cdr FROM Disk('backups',
'backup_20250101_120000.zip')"

# Или восстановление конкретной таблицы
docker exec clickhouse clickhouse-client --query "
RESTORE TABLE cdr.calls FROM Disk('backups',
'calls_backup_20250101.zip')"

# Проверка восстановленных данных
docker exec clickhouse clickhouse-client --query "
SELECT count() FROM cdr.calls"

# Запуск сервисов
docker service scale demo_cdr=3
docker service scale demo_notificator=2
```

6.2.2 Восстановление Kafka

```
# Восстановление сообщений в топик
cat /backup/kafka/sip_stream_20250101.json.gz | gunzip
| \
  docker exec -i redpanda rpk topic produce sip_stream
--format json

# Проверка восстановленных сообщений
docker exec redpanda rpk topic consume sip_stream
--num 10
```

6.2.3 Восстановление MinIO

```
# Распаковка архива
tar -xzf /backup/minio/audio_20250101.tar.gz -C /tmp/

# Восстановление в MinIO
mc mirror /tmp/audio_20250101/ local/audio-restored/

# Проверка восстановленных данных
mc ls local/audio-restored --recursive
```

6.2.4 Восстановление Vault

```
# Расшифровка резервной копии
gpg --decrypt vault_backup_20250101.json.gpg >
vault_backup_20250101.json

# Восстановление секретов
vault operator import vault_backup_20250101.json

# Проверка восстановленных секретов
vault kv get demo/settings
```

6.3 План действий при аварии (Disaster Recovery)

6.3.1 Классификация инцидентов

Критический инцидент (полная недоступность платформы):

1. Оценка масштаба проблемы
2. Уведомление команды

3. Активация плана восстановления
4. Восстановление из резервных копий
5. Проверка работоспособности
6. Постмортем анализ

Средний инцидент (частичная недоступность):

1. Локализация проблемы
2. Восстановление проблемного компонента
3. Проверка работоспособности
4. Документирование инцидента

6.3.2 Процедура восстановления после полного сбоя

```
# 1. Восстановление инфраструктурных сервисов
# ClickHouse
docker stack deploy -c
docker-compose.infrastructure.yaml demo-infra
docker exec clickhouse clickhouse-client --query
"RESTORE DATABASE cdr FROM ..."

# Redpanda
docker service create --name redpanda ...
docker exec redpanda rpk topic create sip_stream
--partitions 3 --replicas 2

# MinIO
docker service create --name minio ...
mc mb local/audio

# Vault
docker service create --name vault ...
vault operator import vault_backup.json

# 2. Восстановление сервисов платформы
docker stack deploy -c docker-compose.stack.yaml demo

# 3. Проверка работоспособности
curl http://localhost:8800/healthcheck
curl http://localhost:6062/healthcheck
```

```
# 4. Восстановление данных
# (см. разделы выше)
```

6.3.3 Контрольный список восстановления

- ✔ Определен тип и масштаб инцидента
- ✔ Уведомлена команда поддержки
- ✔ Создана резервная копия текущего состояния (если возможно)
- ✔ Восстановлены инфраструктурные сервисы
- ✔ Восстановлены данные из резервных копий
- ✔ Восстановлены сервисы платформы
- ✔ Выполнена проверка работоспособности всех компонентов
- ✔ Проведено тестирование основных функций
- ✔ Проверены метрики и логи
- ✔ Документирован инцидент и процедура восстановления
- ✔ Проведен постмортем анализ

7. Безопасность

7.1 Управление секретами через Vault

7.1.1 Ротация секретов

Рекомендуемая частота ротации:

- Пароли баз данных: каждые 90 дней
- API ключи: каждые 180 дней
- Secret ID для AppRole: каждые 30 дней
- TLS сертификаты: согласно сроку действия

Процедура ротации пароля ClickHouse:

```
# 1. Создание нового пароля
NEW_PASSWORD=$(openssl rand -base64 32)
```

```
# 2. Обновление пароля в ClickHouse
docker exec -it clickhouse clickhouse-client --query "
ALTER USER admin IDENTIFIED BY '${NEW_PASSWORD}'"

# 3. Обновление пароля в Vault
vault kv put demo/settings
clickhouse_password="${NEW_PASSWORD}"
# Примечание: ${NEW_PASSWORD} содержит сгенерированный
пароль из предыдущего шага

# 4. Синхронизация настроек в сервисах
curl -X PUT http://localhost:8800/settings/sync
curl -X PUT http://localhost:8803/settings/sync

# 5. Проверка работоспособности
curl http://localhost:8800/healthcheck
```

7.1.2 Управление политиками доступа

```
# Создание политики для чтения настроек
vault policy write read-settings - <<EOF
path "demo/settings" {
  capabilities = ["read"]
}
EOF

# Создание политики для записи настроек
vault policy write write-settings - <<EOF
path "demo/settings" {
  capabilities = ["read", "create", "update"]
}
EOF

# Привязка политики к AppRole
vault write auth/approle/role/antifraud-readonly \
    token_policies="read-settings" \
    token_ttl=1h
```

7.2 Обновление сертификатов

7.2.1 Обновление TLS сертификатов

```
# 1. Генерация нового сертификата
openssl req -x509 -newkey rsa:4096 \
  -keyout /etc/ssl/private/demo.key \
  -out /etc/ssl/certs/demo.crt \
  -days 365 -nodes

# 2. Обновление сертификата в Vault
vault kv put demo/tls \
    cert=@/etc/ssl/certs/demo.crt \
    key=@/etc/ssl/private/demo.key

# 3. Перезапуск сервисов с новым сертификатом
docker service update --force demo_api
docker service update --force demo_ui
```

7.2.2 Автоматическое обновление сертификатов

Настройте автоматическое обновление через cert-manager или аналогичный инструмент для автоматического управления сертификатами Let's Encrypt.

7.3 Аудит доступа

7.3.1 Включение аудита в Vault

```
# Включение файлового аудита
vault audit enable file
file_path=/var/log/vault_audit.log

# Включение syslog аудита
vault audit enable syslog tag="vault" facility="AUTH"

# Просмотр логов аудита
tail -f /var/log/vault_audit.log | jq
```

7.3.2 Мониторинг доступа к API

Настройте сбор логов доступа к API сервисам и анализ подозрительной активности:

```
# Анализ логов доступа
grep "401\|403" /var/log/demo/api.log | \
  awk '{print $1, $7}' | \
  sort | uniq -c | sort -rn
```

7.3.3 Регулярный аудит безопасности

Ежемесячные проверки:

- ✔ Проверка списка активных пользователей и токенов
- ✔ Анализ логов доступа на подозрительную активность
- ✔ Проверка актуальности секретов
- ✔ Обновление политик доступа при необходимости
- ✔ Проверка обновлений безопасности компонентов

7.4 Контрольный список безопасности

Ежедневно:

- ✔ Проверка критических алертов безопасности
- ✔ Мониторинг неудачных попыток аутентификации

Еженедельно:

- ✔ Анализ логов доступа
- ✔ Проверка актуальности секретов

Ежемесячно:

- ✔ Ротация секретов согласно политике
- ✔ Аудит доступа и политик
- ✔ Проверка обновлений безопасности
- ✔ Обновление сертификатов при необходимости

8. Приложения

8.1 Полезные команды Docker Swarm

8.1.1 Управление стеком

```
# Развертывание стека
docker stack deploy -c docker-compose.stack.yaml demo

# Просмотр сервисов стека
docker stack services demo

# Просмотр задач стека
docker stack ps demo

# Удаление стека
docker stack rm demo

# Масштабирование сервиса
docker service scale demo_cdr=5
```

8.1.2 Управление сервисами

```
# Обновление сервиса
docker service update --image new-image:tag demo_cdr

# Перезапуск сервиса
docker service update --force demo_cdr

# Масштабирование сервиса
docker service scale demo_cdr=3

# Удаление сервиса
docker service rm demo_cdr

# Просмотр логов сервиса
docker service logs demo_cdr --tail 100 -f
```

8.1.3 Управление узлами

```
# Список узлов
docker node ls
```

```
# Детальная информация об узле
docker node inspect <node-id> --pretty

# Добавление метки узлу
docker node update --label-add TAG=worker <node-id>

# Удаление метки
docker node update --label-rm TAG <node-id>

# Дренирование узла (перемещение задач на другие узлы)
docker node update --availability drain <node-id>

# Возврат узла в активное состояние
docker node update --availability active <node-id>
```

8.1.4 Управление сетями

```
# Список сетей
docker network ls

# Создание overlay сети
docker network create --driver overlay
demo-backend-network

# Удаление сети
docker network rm demo-backend-network

# Инспекция сети
docker network inspect demo-backend-network
```

8.2 Шпаргалка по инфраструктурным сервисам

8.2.1 ClickHouse

```
# Подключение
docker exec -it clickhouse clickhouse-client

# Просмотр баз данных
SHOW DATABASES;

# Просмотр таблиц
```

```
SHOW TABLES FROM cdr;

# Выполнение запроса
SELECT count() FROM cdr.calls;

# Создание резервной копии
BACKUP DATABASE cdr TO Disk('backups', 'backup.zip');

# Восстановление
RESTORE DATABASE cdr FROM Disk('backups',
'backup.zip');
```

8.2.2 Redpanda/Kafka

```
# Список топиков
docker exec -it redpanda rpk topic list

# Создание топика
docker exec -it redpanda rpk topic create test-topic
--partitions 3

# Просмотр сообщений
docker exec -it redpanda rpk topic consume test-topic
--num 10

# Отправка сообщения
docker exec -it redpanda rpk topic produce test-topic
--value "test message"

# Информация о кластере
docker exec -it redpanda rpk cluster info
```

8.2.3 MinIO

```
# Список bucket'ов
mc ls local

# Создание bucket'а
mc mb local/new-bucket

# Загрузка файла
```

```
mc cp file.wav local/bucket/file.wav

# Список объектов
mc ls local/bucket --recursive

# Скачивание файла
mc cp local/bucket/file.wav ./file.wav

# Удаление объекта
mc rm local/bucket/file.wav
```

8.2.4 Vault

```
# Статус Vault
vault status

# Просмотр секрета
vault kv get demo/settings

# Создание/обновление секрета
vault kv put demo/settings key=value

# Удаление секрета
vault kv delete demo/settings

# Список AppRole
vault list auth/approle/role

# Получение Role ID
vault read auth/approle/role/antifraud/role-id

# Генерация Secret ID
vault write -f auth/approle/role/antifraud/secret-id
```

8.3 Контрольные списки для операций

8.3.1 Ежедневные операции

- Проверка статуса всех сервисов (docker stack services demo)
- Просмотр критических алертов в Grafana
- Проверка использования дискового пространства

- Просмотр логов на критические ошибки
- Проверка метрик производительности

8.3.2 Еженедельные операции

- Анализ производительности запросов ClickHouse
- Проверка retention политик в Redpanda
- Анализ использования хранилища MinIO
- Проверка политик доступа в Vault
- Обновление документации при необходимости

8.3.3 Ежемесячные операции

- Резервное копирование всех критических данных
- Оптимизация таблиц ClickHouse
- Ротация секретов в Vault
- Анализ и очистка старых данных
- Аудит безопасности
- Проверка обновлений компонентов
- Обновление планов резервного копирования

8.3.4 Операции при обновлении

- Резервное копирование перед обновлением
- Проверка совместимости версий
- Тестирование обновления на тестовой среде
- Планирование окна обслуживания
- Уведомление пользователей
- Выполнение обновления
- Проверка работоспособности после обновления
- Мониторинг метрик после обновления
- Документирование изменений

8.4 Переменные окружения

Все сервисы платформы "Волна" используют переменные окружения для конфигурации. При запуске сервиса переменные окружения могут быть установлены напрямую или импортированы из HashiCorp Vault (если VAULT_ENABLE=yes). Значения из Vault переопределяют значения переменных окружения, установленные при запуске контейнера.

8.4.1 Общие параметры

Эти параметры используются всеми сервисами платформы:

Переменная	Описание	Пример значения	Обязательная
PROJECT	Идентификатор проекта/окружения	demo	Да
VAULT_ENABLE	Включить использование Vault для получения конфигурации	yes, no	Нет
VAULT_SKIP_VERIFY	Пропустить проверку SSL сертификата Vault	true, false	Нет
VAULT_ADDR	Адрес сервера Vault	https://vault.af.internal	Да (если VAULT_ENABLE=yes)
VAULT_MOUNT_POINT	Точка монтирования секретов в Vault	demo	Да (если VAULT_ENABLE=yes)
VAULT_SECRET_PATH	Путь к секретам в Vault	settings	Да (если VAULT_ENABLE=yes)
VAULT_ROLE_ID	Role ID для аутентификации через AppRole	**********	Да (если VAULT_ENABLE=yes)
VAULT_SECRET_ID	Secret ID для аутентификации через AppRole	**********	Да (если VAULT_ENABLE=yes)
VITE_PUBLIC_URL	Публичный URL для фронтенда	http://demo.cloud.af.internal	Нет
AUTH_USERNAME	Имя пользователя для базовой аутентификации	admin	Нет

8.4.2 Параметры CDR-сервиса

Параметры, специфичные для сервиса обработки записей детализации звонков:

Переменная	Описание	Пример значения	Значение по умолчанию

CDR_DATABASE	Имя базы данных ClickHouse	demo	cdr
CDR_HOST	Адрес сервера ClickHouse	ch-1.af.internal	clickhouse
CDR_PORT	Порт ClickHouse	8123	8123
CDR_USER	Пользователь ClickHouse	demo	admin
CDR_PASSWORD	Пароль пользователя ClickHouse	**********	(пусто)
CDR_KAFKA_URL	Адрес брокера Kafka/RedPanda	redpanda-1.af.internal:9092	redpanda:9092
CDR_KAFKA_TOPIC	Топик Kafka для SIP-потока	demo_sip_stream	sip_stream
CDR_KAFKA_GROUP	Группа потребителей Kafka	demo_cdr	cdr
CDR_KAFKA_OFFSET	Стратегия чтения сообщений из Kafka	earliest, latest	earliest
CDR_TIMEOUT	Таймаут операций CDR (секунды)	10	60
CDR_SAVE_WAV	Включить запись аудиофайлов	true, false	true
CDR_AUDIO_DIR	Каталог для сохранения аудиофайлов	/opt/audio	/opt/audio
CDR_S3_URL	URL S3-совместимого хранилища	http://minio:9000	(пусто)
CDR_S3_BUCKET	Имя bucket'а в S3	audio	(пусто)
CDR_S3_ACCESS_KEY_ID	Access Key ID для S3	**********	(пусто)

CDR_S3_SECRET_ACCESS_KEY	Secret Access Key для S3	**********	(пусто)

8.4.3 Параметры CLASSIFIER-сервиса

Параметры для сервиса транскрибации и классификации речи:

Переменная	Описание	Пример значения	Значение по умолчанию
CLASSIFIER_KAFKA_URL	Адрес брокера Kafka/RedPanda	redpanda-1.af.internal:9092	redpanda:9092
CLASSIFIER_KAFKA_TOPIC	Входной топик Kafka для SIP-потока	demo_sip_stream	sip_stream
CLASSIFIER_KAFKA_OUT_TOPIC	Выходной топик Kafka для результатов классификации	demo_cls_stream	cls_stream
CLASSIFIER_KAFKA_GROUP	Группа потребителей Kafka	demo_classifier	classifier
CLASSIFIER_KAFKA_OFFSET	Стратегия чтения сообщений из Kafka	earliest, latest	earliest
CLASSIFIER_MODEL_TYPE	Тип модели транскрибации	v2_ctc	v2_ctc
CLASSIFIER_USE_ONNX	Использовать ONNX модели	true, false	false
CLASSIFIER_ONNX_DIR	Директория с ONNX моделями	infrastructure/gigaam/onnx	onnx
CLASSIFIER_CLASSIFY_MODEL	Модель классификации	summary	summary

CLASSIFIER_SAMPLING_RATE	Частота дискретизации аудио (Гц)	16000	16000
CLASSIFIER_NUM_CPU_PROC	Количество CPU процессов для обработки	4	2
CLASSIFIER_NUM_GPU_PROC	Количество GPU процессов для обработки	2	1
CLASSIFIER_EXPIRE_TIME	Время жизни кэша (секунды)	60	300
CLASSIFIER_AUDIO_STORE_CACHE_SIZE	Размер кэша аудио (количество элементов)	1000	1000
CLASSIFIER_AUDIO_STORE_THRESHOLD	Порог размера кэша аудио (байты)	200000	1000000
CLASSIFIER_LLM_USE	Использовать LLM для обработки	true, false	false

8.4.4 Параметры IMPORTER-сервиса

Параметры для сервиса импорта данных:

Переменная	Описание	Пример значения	Значение по умолчанию
IMPORTER_KAFKA_URL	Адрес брокера Kafka/RedPanda	redpanda-1.af.internal:9092	redpanda:9092
IMPORTER_KAFKA_TOPIC	Топик Kafka для SIP-потока	demo_sip_stream	sip_stream

IMPORTER_KAFKA_GROUP	Группа потребителей Kafka	demo_importer	importer
IMPORTER_KAFKA_OFFSET	Стратегия чтения сообщений из Kafka	earliest, latest	earliest
IMPORTER_HTTP_URL	URL для отправки данных в Classifier	http://demo_classifier:6062/push	http://classifier:6062/push
IMPORTER_MAX_CONCURRENT_REQUESTS	Максимальное количество одновременных запросов	1000	100
IMPORTER_CACHE_TTL	Время жизни кэша (секунды)	300	300
IMPORTER_BLACKLIST_ENABLE	Включить использование черного списка	true, false	true
IMPORTER_WHITELIST_ENABLE	Включить использование белого списка	true, false	false

8.4.5 Параметры NOTIFICATOR-сервиса

Параметры для сервиса уведомлений:

Параметры Kafka:

Переменная	Описание	Пример значения	Значение по умолчанию
NOTIFICATOR_KAFKA_URL	Адрес брокера Kafka/RedPanda	redpanda-1.af.internal:9092	redpanda:9092
NOTIFICATOR_KAFKA_TOPIC	Топик Kafka для результатов классификации	demo_cls_stream	cls_stream

NOTIFICATOR_KAFKA_GROUP	Группа потребителей Kafka	demo_notificator	notificator
NOTIFICATOR_KAFKA_OFFSET	Стратегия чтения сообщений из Kafka	earliest, latest	earliest
NOTIFICATOR_KAFKA_POLL_INTERVAL	Интервал опроса Kafka (секунды)	0.1	0.1

Параметры классификации:

Переменная	Описание	Пример значения	Значение по умолчанию
NOTIFICATOR_CLASSIFY_URL	URL сервиса классификации	http://demo_classifier:7778/detect	http://classifier:7778/detect
NOTIFICATOR_CLASSIFY_MODEL	Модель классификации	summary	summary
NOTIFICATOR_CLASSIFY_THRESHOLD	Порог для определения мошенничества	0.8	0.8
NOTIFICATOR_CLASSIFY_FRAUD_LEVEL	Уровень для определения мошенничества	0.85	0.85
NOTIFICATOR_CLASSIFY_NOTFRAUD_LEVEL	Уровень для определения не-мошенничества	0.5	0.5
NOTIFICATOR_CLASSIFY_NOTFRAUD_PERIOD	Период для определения не-мошенничества (секунды)	180	180
NOTIFICATOR_CLASSIFY_METRIC_WINDOW	Окно для агрегации метрик	3	3

NOTIFICATOR_CLASSIFY_METRIC_AGGREGATE	Метод агрегации метрик	last, avg, max	last
NOTIFICATOR_CLASSIFY_FLEVEL_INTERVAL	Интервал обновления уровня мошенничества (секунды)	60	60
NOTIFICATOR_CLASSIFY_LAST_NWORDS	Количество последних слов для анализа	100	100
NOTIFICATOR_CLASSIFY_CACHE_TIMEOUT	Таймаут кэша классификации (секунды)	300	300
NOTIFICATOR_CLASSIFY_SAVE_FRAUD_LOG	Сохранять логи мошеннических вызовов	true, false	true

Параметры фильтрации:

Переменная	Описание	Пример значения	Значение по умолчанию
NOTIFICATOR_FILTER_ENABLE	Включить фильтрацию	true, false	false
NOTIFICATOR_FILTER_THRESHOLD	Порог фильтрации	0.1	0.1
NOTIFICATOR_FILTER_LARGE_MODEL_WEIGHT	Вес большой модели	0.7	0.7
NOTIFICATOR_FILTER_LITTLE_MODEL_WEIGHT	Вес маленькой модели	0.3	0.3

Параметры Fraudwords (ключевые слова мошенничества):

Переменная	Описание	Пример значения	Значение по умолчанию
NOTIFICATOR_FRAUDWORDS_HEURISTIC	Использовать эвристику для fraudwords	true, false	true
NOTIFICATOR_FRAUDWORDS_HEURISTIC_SCORE	Базовый score для эвристики	0.14	0.14
NOTIFICATOR_FRAUDWORDS_HEURISTIC_SCORE_MAX	Максимальный score для эвристики	0.18	0.18
NOTIFICATOR_FRAUDWORDS_HEURISTIC_MIN_RATIO	Минимальный процент совпадения для эвристики	85	85
NOTIFICATOR_FRAUDWORDS_DEBUG_PHRASE	Отладочные фразы для тестирования	бабкина радость; бабки на радость	(пусто)

Параметры ключевых слов:

Переменная	Описание	Пример значения	Значение по умолчанию
NOTIFICATOR_KEYWORDS_MAX_COUNT	Максимальное количество ключевых слов	12	10

NOTIFICATOR_KEYWORDS_SCORE_THRESHOLD	Порог score для ключевых слов	0.55	0.5
NOTIFICATOR_KEYWORDS_SORTED	Сортировать ключевые слова	true, false	true
NOTIFICATOR_KEYWORDS_FILTER_SIMILAR	Фильтровать похожие ключевые слова	true, false	true
NOTIFICATOR_KEYWORDS_CORRECT_ERRORS	Исправлять ошибки в ключевых словах	true, false	true

Параметры Stubwords (стоп-слова):

Переменная	Описание	Пример значения	Значение по умолчанию
NOTIFICATOR_STUBWORDS_FILTER	Включить фильтрацию stubwords	true, false	false
NOTIFICATOR_STUBWORDS_MIN_RATIO	Минимальный процент для stubwords	85	85
NOTIFICATOR_STUBWORDS_DECREMENT_SCORE	Уменьшение score для stubwords	0.15	0.15

Параметры уведомлений:

Переменная	Описание	Пример значения	Значение по умолчанию
NOTIFICATOR_NOTIFICATION_COUNT	Количество уведомлений для одного вызова	1	1
NOTIFICATOR_NOTIFICATION_PERIOD	Период между уведомлениями (секунды)	3600	3600
NOTIFICATOR_NOTIFICATION_PLAY_AUDIO_PERIOD	Период проигрывания аудио (секунды)	60	60
NOTIFICATOR_NOTIFICATION_WEBHOOK_URL	URL для webhook уведомлений	http://example.com/webhook	(пусто)

Параметры Telegram:

Переменная	Описание	Пример значения	Значение по умолчанию
NOTIFICATOR_TELEGRAM_BOT_TOKEN	Токен Telegram бота	**********	(пусто)
NOTIFICATOR_TELEGRAM_CHAT_ID	ID чата для уведомлений	**********	(пусто)
NOTIFICATOR_TELEGRAM_THREAD_ID	ID треда для уведомлений	**********	(пусто)
NOTIFICATOR_TELEGRAM_MAX_MESSAGE_SIZE	Максимальный размер	1024	4096

	сообщения (байты)		
NOTIFICATOR_TELEGRAM_SEND_TEXT	Отправлять текст транскрибации	true, false	true
NOTIFICATOR_TELEGRAM_SEND_KEYWORDS	Отправлять ключевые слова	true, false	true
NOTIFICATOR_TELEGRAM_SEND_AUDIO	Отправлять аудиофайлы	true, false	false

Параметры Email:

Переменная	Описание	Пример значения	Значение по умолчанию
NOTIFICATOR_EMAIL_SERVER	SMTP сервер	smtp.example.com	(пусто)
NOTIFICATOR_EMAIL_PORT	SMTP порт	587	587
NOTIFICATOR_EMAIL_USERNAME	Имя пользователя SMTP	notify@af.internal	(пусто)
NOTIFICATOR_EMAIL_PASSWORD	Пароль SMTP	**********	(пусто)
NOTIFICATOR_EMAIL_FROM	Адрес отправителя	notify@af.internal	(пусто)
NOTIFICATOR_EMAIL_TO	Адрес получателя	info@af.internal	(пусто)

Параметры CDR:

Переменная	Описание	Пример значения	Значение по умолчанию
NOTIFICATOR_CDR_ENABLE	Включить сохранение CDR	true, false	true

Параметры API:

Переменная	Описание	Пример значения	Значение по умолчанию
NOTIFICATOR_API_BASE_URL	Базовый URL API сервиса	http://demo_api:8000	http://api:8000

Параметры управления вызовами (PBX):

Переменная	Описание	Пример значения	Значение по умолчанию
NOTIFICATOR_CALL_MUTE	Включить заглушку вызовов	true, false	false
NOTIFICATOR_CALL_MUTE_PERIOD	Период заглушки (секунды)	3600	3600
NOTIFICATOR_CALL_MUTE_URL	URL для управления заглушкой	http://demo_importer:6061/mute	(пусто)

8.4.6 Параметры PBX

Параметры для интеграции с телефонной станцией (Asterisk):

Переменная	Описание	Пример значения	Значение по умолчанию
PBX_URL	URL API телефонной станции	http://10.0.1.110:8088	(пусто)
PBX_USERNAME	Имя пользователя для доступа к PBX	antifraud	(пусто)
PBX_PASSWORD	Пароль для доступа к PBX	**********	(пусто)
PBX_PLAY_SOUND	Проигрывать звуковые предупреждения	true, false	false
PBX_FRAUD_SOUND	Тип звука для мошеннических вызовов	hangup, warning	hangup

PBX_CALL_HANGUP	Отключать мошеннические вызовы	true, false	false

8.4.7 Параметры GPU

Параметры для управления использованием GPU в Classifier-сервисе:

Переменная	Описание	Пример значения	Значение по умолчанию
CUDA_VISIBLE_DEVICES	Список видимых GPU устройств (через запятую)	2,3	0
NVIDIA_VISIBLE_DEVICES	Список видимых NVIDIA GPU устройств	2,3	all

Примечание: Эти переменные должны совпадать. Используются для ограничения доступа Classifier-сервиса к конкретным GPU устройствам в многопользовательской среде.

8.4.8 Использование Vault для управления переменными окружения

При установке VAULT_ENABLE=yes сервисы платформы автоматически загружают конфигурацию из HashiCorp Vault при старте. Значения из Vault переопределяют значения переменных окружения, установленные в docker-compose.stack.yaml или stack.env.

Процесс загрузки конфигурации:

1. Сервис запускается с базовыми переменными окружения (VAULT_*)
2. При старте сервис подключается к Vault используя AppRole аутентификацию
3. Загружаются секреты из пути: {VAULT_MOUNT_POINT}/{VAULT_SECRET_PATH}
4. Значения из Vault переопределяют переменные окружения
5. Сервис продолжает работу с обновленной конфигурацией

Пример настройки секретов в Vault:

```
# Установка переменных окружения для Vault
export VAULT_ADDR='https://vault.af.internal'
export VAULT_TOKEN='**************'

# Сохранение конфигурации для проекта demo
vault kv put demo/settings \
    CDR_HOST="ch-1.af.internal" \
    CDR_USER="demo" \
    CDR_PASSWORD="**************" \
    CDR_KAFKA_URL="redpanda-1.af.internal:9092" \
    CLASSIFIER_KAFKA_URL="redpanda-1.af.internal:9092"
\
    NOTIFICATOR_TELEGRAM_BOT_TOKEN="**************" \
    NOTIFICATOR_TELEGRAM_CHAT_ID="**************"

# Просмотр сохраненных настроек
vault kv get demo/settings
```

Преимущества использования Vault:

- Централизованное управление секретами
- Ротация секретов без перезапуска сервисов (через API /settings/sync)
- Аудит доступа к секретам
- Шифрование секретов
- Разделение прав доступа через политики

Синхронизация настроек без перезапуска:

```
# Синхронизация настроек CDR-сервиса с Vault
curl -X PUT http://localhost:8800/settings/sync

# Синхронизация настроек Notificator-сервиса
curl -X PUT http://localhost:8803/settings/sync

# Синхронизация настроек Classifier-сервиса
curl -X PUT http://localhost:6062/settings/sync
```

8.4.9 Пример файла stack.env

Ниже приведен пример полного файла **stack.env** с комментариями:

```
# Общие параметры проекта
PROJECT="demo"
AF_PROJECT="demo"

# Параметры Vault
VAULT_ENABLE="yes"
VAULT_SKIP_VERIFY="true"
VAULT_ADDR="https://vault.af.internal"
VAULT_MOUNT_POINT="demo"
VAULT_SECRET_PATH="settings"
VAULT_ROLE_ID="****************************"
VAULT_SECRET_ID="****************************"

# Параметры фронтенда
VITE_PUBLIC_URL="http://demo.cloud.af.internal"

# Параметры аутентификации
AUTH_USERNAME="admin"

# Параметры CDR-сервиса
CDR_DATABASE="demo"
CDR_HOST="ch-1.af.internal"
CDR_USER="demo"
CDR_PASSWORD="**************"
CDR_KAFKA_URL="redpanda-1.af.internal:9092"
CDR_KAFKA_TOPIC="demo_sip_stream"
CDR_KAFKA_GROUP="demo_cdr"
CDR_KAFKA_OFFSET="earliest"
CDR_TIMEOUT=10
CDR_SAVE_WAV="false"
CDR_S3_URL=""
CDR_S3_BUCKET=""
CDR_S3_ACCESS_KEY_ID=""
CDR_S3_SECRET_ACCESS_KEY=""
```

```
# Параметры CLASSIFIER-сервиса
CLASSIFIER_KAFKA_URL="redpanda-1.af.internal:9092"
CLASSIFIER_KAFKA_TOPIC="demo_sip_stream"
CLASSIFIER_KAFKA_OUT_TOPIC="demo_cls_stream"
CLASSIFIER_KAFKA_GROUP="demo_classifier"
CLASSIFIER_KAFKA_OFFSET="earliest"
CLASSIFIER_MODEL_TYPE="v2_ctc"
CLASSIFIER_USE_ONNX="false"
CLASSIFIER_CLASSIFY_MODEL="summary"
CLASSIFIER_NUM_CPU_PROC=4
CLASSIFIER_NUM_GPU_PROC=2
CLASSIFIER_EXPIRE_TIME=60
CLASSIFIER_AUDIO_STORE_CACHE_SIZE=1000
CLASSIFIER_AUDIO_STORE_THRESHOLD=200000
CLASSIFIER_LLM_USE="false"

# Параметры IMPORTER-сервиса
IMPORTER_KAFKA_URL="redpanda-1.af.internal:9092"
IMPORTER_KAFKA_TOPIC="demo_sip_stream"
IMPORTER_KAFKA_GROUP="demo_importer"
IMPORTER_KAFKA_OFFSET="earliest"
IMPORTER_HTTP_URL="http://demo_classifier:6062/push"
IMPORTER_MAX_CONCURRENT_REQUESTS=1000
IMPORTER_CACHE_TTL=300
IMPORTER_BLACKLIST_ENABLE="true"
IMPORTER_WHITELIST_ENABLE="false"

# Параметры NOTIFICATOR-сервиса
NOTIFICATOR_KAFKA_URL="redpanda-1.af.internal:9092"
NOTIFICATOR_KAFKA_TOPIC="demo_cls_stream"
NOTIFICATOR_KAFKA_GROUP="demo_notificator"
NOTIFICATOR_KAFKA_OFFSET="earliest"
NOTIFICATOR_KAFKA_POLL_INTERVAL=0.1
NOTIFICATOR_CLASSIFY_URL="http://demo_classifier:7778/
detect"
NOTIFICATOR_CLASSIFY_MODEL="summary"
NOTIFICATOR_CLASSIFY_THRESHOLD=0.8
```

```
NOTIFICATOR_CLASSIFY_FRAUD_LEVEL=0.85
NOTIFICATOR_CLASSIFY_NOTFRAUD_LEVEL=0.5
NOTIFICATOR_TELEGRAM_BOT_TOKEN="**************"
NOTIFICATOR_TELEGRAM_CHAT_ID="**************"
NOTIFICATOR_TELEGRAM_THREAD_ID="**************"
NOTIFICATOR_API_BASE_URL="http://demo_api:8000"
NOTIFICATOR_CDR_ENABLE="true"
NOTIFICATOR_CALL_MUTE="true"
NOTIFICATOR_CALL_MUTE_PERIOD=3600
NOTIFICATOR_CALL_MUTE_URL="http://demo_importer:6061/m
ute"

# Параметры PBX
PBX_URL="http://10.0.1.110:8088"
PBX_USERNAME="antifraud"
PBX_PASSWORD="**************"
PBX_PLAY_SOUND="true"
PBX_FRAUD_SOUND="hangup"
PBX_CALL_HANGUP="true"

# Параметры GPU
CUDA_VISIBLE_DEVICES="2,3"
NVIDIA_VISIBLE_DEVICES="2,3"
```

Примечание: Секретные значения (пароли, токены) должны храниться в Vault, а не в файле **stack.env**. Файл stack.env используется только для базовой конфигурации, которая не содержит секретов.

8.5 Контакты и поддержка

Техническая поддержка:

- Email: support@rt-solar.ru
- Телефон: +7 (XXX) XXX-XX-XX
- Документация: https://docs.example.com

Экстренные контакты:

- On-call инженер: +7 (XXX) XXX-XX-XX
- Менеджер проекта: +7 (XXX) XXX-XX-XX

Заключение

Данная инструкция системного администратора содержит основные процедуры эксплуатации платформы "Волна". Регулярное следование рекомендациям и контрольным спискам обеспечит стабильную работу платформы и минимизирует риски возникновения проблем.

При возникновении вопросов или проблем, не описанных в данной инструкции, обращайтесь в службу технической поддержки.