SOLAR

Программный комплекс обнаружения и реагирования на мошеннические голосовые вызовы на основе анализа SIP/RTP-трафика с применением технологий машинного обучения «Волна»

Инструкция по установке

МОСКВА, 2026

Содержание

1. Общие сведения
2. Системные требования
3. Предварительная подготовка
4. Установка компонентов платформы
5. Инициализация базы данных
6. Проверка работоспособности
7. Настройка мониторинга
8. Устранение неполадок

Ссылка для скачивания ПО и пароль будут предоставлены в отдельном архиве.

Требования к квалификации для демонстрации

Учитывая особенности заявляемого Программного комплекса обнаружения и реагирования на мошеннические голосовые вызовы на основе анализа SIP/RTP трафика с применением технологий машинного обучения «Волна», требования к установке и эксплуатации, а также возможность оказания сервисной поддержки исключительно силами компетентных специалистов разработчика и правообладателя, предлагаем провести демонстрацию функциональных характеристик, системы хранения исходного кода и процессов эксплуатации программного обеспечения в формате видеоконференцсвязи.

Демонстрация может быть организована по согласованию с экспертом Минцифры в рабочем порядке, в любое удобное время и с использованием любого подходящего технического решения для ВКС.

Контактные данные ответственного лица за организацию демонстрации программного обеспечения: (Журавлев Андрей Анатольевич, Руководитель развития направления Антифрод, +7 916 641 2701, an.zhuravlev@rt-solar.ru).

1. Общие сведения

1.1 Описание ПО

Программный комплекс обнаружения и реагирования на мошеннические голосовые вызовы на основе анализа SIP/RTP трафика с применением технологий машинного обучения «Волна» (далее - Платформа "Волна") 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/Swarm
- **Мониторинг:** Prometheus, Grafana, Loki, Zabbix
- **Хранение:** S3-совместимое хранилище (Minio)
- **Безопасность:** HashiCorp Vault
- **CI/CD:** GitLab

2. Системные требования

2.1 Требования к серверу приложения

Операционные системы

- Семейство Linux: РЕД ОС, ОС Альт, Ubuntu 22.04/24.04, Debian, CentOS, Astra Linux

Минимальные аппаратные требования

- Процессор: 4 ядра, 2.4 ГГц или выше
- Оперативная память: 8 ГБ RAM
- Дисковое пространство: 100 ГБ SSD
- Сеть: Ethernet 1 Гбит/с

Программное обеспечение

- Docker 20.10+
- Docker Compose 2.0+
- Docker Swarm (для кластерного развертывания)
- Python 3.10-3.12 (для разработки)

2.2 Требования к автоматизированному рабочему месту пользователя

Операционные системы

- Windows 10/11
- macOS 12+
- Linux (Ubuntu 20.04+, CentOS 8+)

Аппаратные требования

- Процессор: 2 ядра, 2.0 ГГц или выше
- Оперативная память: 4 ГБ RAM
- Дисковое пространство: 1 ГБ свободного места
- Дисплей: разрешение 1024x768 или выше

Программное обеспечение

- Веб-браузер: Chrome 90+, Firefox 88+, Safari 14+, Edge 90+
- Доступ к сети Интернет для работы с веб-интерфейсом

2.3 Сетевые требования

- Доступ к серверу приложения по HTTPS (порт 443)
- Пропускная способность: не менее 10 Мбит/с
- Открытые порты для сервисов:
  - 8000 - API-сервис
  - 8800 - CDR-сервис
  - 8803 - Notificator-сервис
  - 6061 - Importer-сервис

- 6062 - Classifier-сервис
- 9092 - Kafka/RedPanda
- 8123 - ClickHouse
- 5432 - PostgreSQL

3. Предварительная подготовка

3.1 Установка Docker и Docker Swarm

Установка Docker

Для Ubuntu/Debian:

```
# Обновление списка пакетов
sudo apt-get update

# Установка зависимостей
sudo apt-get install -y \
    ca-certificates \
    curl \
    gnupg \
    lsb-release

# Добавление официального GPG ключа Docker
sudo mkdir -p /etc/apt/keyrings
curl -fsSL
https://download.docker.com/linux/ubuntu/gpg | sudo
gpg --dearmor -o /etc/apt/keyrings/docker.gpg

# Настройка репозитория
echo \
  "deb [arch=$(dpkg --print-architecture)
signed-by=/etc/apt/keyrings/docker.gpg]
https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) stable" | sudo tee
/etc/apt/sources.list.d/docker.list > /dev/null

# Установка Docker Engine
sudo apt-get update
sudo apt-get install -y docker-ce docker-ce-cli
containerd.io docker-compose-plugin
```

```
# Проверка установки
docker --version
docker compose version
```

Для CentOS/RHEL:

```
# Установка зависимостей
sudo yum install -y yum-utils

# Добавление репозитория Docker
sudo yum-config-manager --add-repo
https://download.docker.com/linux/centos/docker-ce.rep
o

# Установка Docker Engine
sudo yum install -y docker-ce docker-ce-cli
containerd.io docker-compose-plugin

# Запуск Docker
sudo systemctl start docker
sudo systemctl enable docker

# Проверка установки
docker --version
docker compose version
```

Инициализация Docker Swarm

```
# Инициализация Swarm на управляющем узле (manager)
docker swarm init

# Сохранение токена для присоединения worker-узлов
# Команда выведет команду для присоединения
worker-узлов, например:
# docker swarm join --token SWMTKN-1-xxx
192.168.1.100:2377

# На worker-узлах выполните команду присоединения
# docker swarm join --token <TOKEN> <MANAGER_IP>:2377
```

```
# Проверка статуса Swarm
docker node ls
```

3.2 Настройка сетей Docker

Создайте необходимые сети для платформы:

```
# Создание внешней сети для frontend
docker network create --driver overlay
demo-frontend-network

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

# Проверка создания сетей
docker network ls
```

3.3 Настройка HashiCorp Vault

Установка Vault

```
# Скачивание Vault
wget
https://releases.hashicorp.com/vault/1.15.0/vault_1.15
.0_linux_amd64.zip
unzip vault_1.15.0_linux_amd64.zip
sudo mv vault /usr/local/bin/

# Проверка установки
vault --version
```

Настройка Vault для платформы

```
# Запуск Vault в режиме разработки (для тестирования)
vault server -dev

# В другом терминале настройте переменные окружения
export VAULT_ADDR='http://127.0.0.1:8200'
export VAULT_TOKEN='your-dev-token'

# Создание mount point для проекта
vault secrets enable -path=demo kv-v2
```

```
# Создание секретов для платформы
vault kv put demo/settings \
    kafka_url="kafka:9092" \
    clickhouse_host="clickhouse" \
    clickhouse_port="8123" \
    clickhouse_database="cdr" \
    clickhouse_user="admin" \
    clickhouse_password="your_password"

# Создание AppRole для аутентификации
vault auth enable approle

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

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

# Получение Role ID и Secret ID
vault read auth/approle/role/antifraud/role-id
vault write -f auth/approle/role/antifraud/secret-id
```

Важно: Сохраните Role ID и Secret ID для использования в конфигурации платформы.

3.4 Подготовка хранилищ данных

Установка и настройка ClickHouse

```
# Запуск ClickHouse через Docker
docker run -d \
    --name clickhouse \
    --network demo-backend-network \
```

```
    -p 8123:8123 \
    -p 9000:9000 \
    -e CLICKHOUSE_DB=cdr \
    -e CLICKHOUSE_USER=admin \
    -e CLICKHOUSE_PASSWORD=your_password \
    clickhouse/clickhouse-server:latest

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

Установка и настройка PostgreSQL

```
# Запуск PostgreSQL через Docker
docker run -d \
    --name postgres \
    --network demo-backend-network \
    -p 5432:5432 \
    -e POSTGRES_DB=antifraud \
    -e POSTGRES_USER=admin \
    -e POSTGRES_PASSWORD=your_password \
    postgres:15

# Проверка подключения
docker exec -it postgres psql -U admin -d antifraud -c
"SELECT version();"
```

3.5 Настройка Kafka/RedPanda

```
# Запуск RedPanda через Docker
docker run -d \
    --name redpanda \
    --network demo-backend-network \
    -p 9092:9092 \
    -p 9644:9644 \
    docker.redpanda.com/redpandadata/redpanda:latest \
    redpanda start \
    --kafka-addr
internal://0.0.0.0:9092,external://0.0.0.0:19092 \
    --advertise-kafka-addr
internal://redpanda:9092,external://localhost:19092 \
```

```
    --pandaproxy-addr
internal://0.0.0.0:8082,external://0.0.0.0:18082 \
    --advertise-pandaproxy-addr
internal://redpanda:8082,external://localhost:18082 \
    --schema-registry-addr
internal://0.0.0.0:8081,external://0.0.0.0:18081 \
    --rpc-addr redpanda:33145 \
    --advertise-rpc-addr redpanda:33145 \
    --smp 1 \
    --memory 1G \
    --mode dev-container \
    --default-log-level=info

# Проверка работы
docker exec -it redpanda rpk cluster info
```

3.6 Настройка S3-совместимого хранилища (Minio)

```
# Запуск Minio через Docker
docker run -d \
    --name minio \
    --network demo-backend-network \
    -p 9000:9000 \
    -p 9001:9001 \
    -e MINIO_ROOT_USER=minioadmin \
    -e MINIO_ROOT_PASSWORD=minioadmin \
    -v /mnt/nfs/docker-data/minio:/data \
    minio/minio server /data --console-address ":9001"

# Создание bucket для аудиофайлов
# Доступ к консоли: http://localhost:9001
# Создайте bucket с именем "audio" через веб-интерфейс
```

4. Установка компонентов платформы

4.1 Подготовка конфигурационных файлов

Создание файла stack.env

Создайте файл stack.env в директории с конфигурацией:

```
# Создание директории для конфигурации
mkdir -p /opt/demo/config
cd /opt/demo/config

# Создание файла stack.env
cat > stack.env <<EOF
PROJECT=demo
AF_PROJECT=demo
VAULT_ENABLE=yes
VAULT_SKIP_VERIFY=true
VAULT_ADDR=https://vault.af.internal
VAULT_MOUNT_POINT=demo
VAULT_SECRET_PATH=settings
VITE_PUBLIC_URL=http://demo.cloud.af.internal

EOF
```

Важно: Замените значения на актуальные для вашей инфраструктуры.

Создание docker-compose.stack.yaml

Создайте файл docker-compose.stack.yaml:

```
version: '3.9'

services:
  cdr:
    image:
registry.af.internal/antifraud/cloud/cdr:latest
    entrypoint: /opt/entrypoint.sh
    command: /opt/.venv/bin/fastapi run main.py --host
0.0.0.0 --port 8800
    healthcheck:
      test: curl -sS http://127.0.0.1:8800/healthcheck
|| echo 1
      interval: 30s
      timeout: 3s
      retries: 12
    env_file: stack.env
    environment:
```

```
      PROJECT: ${AF_PROJECT}
      VAULT_ENABLE: ${VAULT_ENABLE}
      VAULT_SKIP_VERIFY: ${VAULT_SKIP_VERIFY}
      VAULT_ADDR: ${VAULT_ADDR}
      VAULT_ROLE_ID: ${VAULT_ROLE_ID}
      VAULT_SECRET_ID: ${VAULT_SECRET_ID}
      VAULT_MOUNT_POINT: ${VAULT_MOUNT_POINT}
      VAULT_SECRET_PATH: ${VAULT_SECRET_PATH}
    volumes:
      - /mnt/nfs/docker-data/vault:/usr/bin/vault
    networks:
      - demo-backend-network
    deploy:
      restart_policy:
        condition: on-failure
      placement:
        constraints:
          - "node.labels.TAG == worker"
      resources:
        limits:
          cpus: '4'
          memory: 8G
        reservations:
          cpus: '4'
          memory: 4G

  importer:
    image:
registry.af.internal/antifraud/cloud/importer:latest
    entrypoint: /opt/entrypoint.sh
    command: /opt/.venv/bin/fastapi run main.py --host
0.0.0.0 --port 6061
    healthcheck:
      test: curl -sS http://127.0.0.1:6061/healthcheck
|| echo 1
      interval: 30s
      timeout: 3s
      retries: 12
```

```
    env_file: stack.env
    environment:
      PROJECT: ${AF_PROJECT}
      VAULT_ENABLE: ${VAULT_ENABLE}
      VAULT_SKIP_VERIFY: ${VAULT_SKIP_VERIFY}
      VAULT_ADDR: ${VAULT_ADDR}
      VAULT_ROLE_ID: ${VAULT_ROLE_ID}
      VAULT_SECRET_ID: ${VAULT_SECRET_ID}
      VAULT_MOUNT_POINT: ${VAULT_MOUNT_POINT}
      VAULT_SECRET_PATH: ${VAULT_SECRET_PATH}
    volumes:
      - /mnt/nfs/docker-data/vault:/usr/bin/vault
    networks:
      - demo-backend-network
    deploy:
      restart_policy:
        condition: any
      placement:
        constraints:
          - "node.hostname == ml-1"
      resources:
        limits:
          cpus: '4'
          memory: 8G
        reservations:
          cpus: '4'
          memory: 4G

  notificator:
    image:
registry.af.internal/antifraud/cloud/notificator:lates
t
    entrypoint: /opt/entrypoint.sh
    command: /opt/.venv/bin/fastapi run main.py --host
0.0.0.0 --port 8803
    healthcheck:
      test: curl -sS http://127.0.0.1:8803/healthcheck
|| echo 1
```

```
      interval: 30s
      timeout: 3s
      retries: 12
    env_file: stack.env
    environment:
      PROJECT: ${AF_PROJECT}
      VAULT_ENABLE: ${VAULT_ENABLE}
      VAULT_SKIP_VERIFY: ${VAULT_SKIP_VERIFY}
      VAULT_ADDR: ${VAULT_ADDR}
      VAULT_ROLE_ID: ${VAULT_ROLE_ID}
      VAULT_SECRET_ID: ${VAULT_SECRET_ID}
      VAULT_MOUNT_POINT: ${VAULT_MOUNT_POINT}
      VAULT_SECRET_PATH: ${VAULT_SECRET_PATH}
    volumes:
      - /mnt/nfs/docker-data/vault:/usr/bin/vault
    networks:
      - demo-backend-network
    deploy:
      restart_policy:
        condition: any
      placement:
        constraints:
          - "node.labels.TAG == worker"
      resources:
        limits:
          cpus: '6'
          memory: 12G
        reservations:
          cpus: '4'
          memory: 8G

  classifier:
    image:
registry.af.internal/antifraud/cloud/classifier:latest
    hostname: demo_classifier-1
    entrypoint: /app/entrypoint.sh
    command: ["uvicorn", "main:app", "--host",
"0.0.0.0", "--port", "6062"]
```

```
    healthcheck:
      test: curl -sS http://127.0.0.1:6062/healthcheck
|| echo 1
      interval: 30s
      timeout: 3s
      retries: 12
    env_file: stack.env
    environment:
      PROJECT: ${AF_PROJECT}
      VAULT_ENABLE: ${VAULT_ENABLE}
      VAULT_SKIP_VERIFY: ${VAULT_SKIP_VERIFY}
      VAULT_ADDR: ${VAULT_ADDR}
      VAULT_ROLE_ID: ${VAULT_ROLE_ID}
      VAULT_SECRET_ID: ${VAULT_SECRET_ID}
      VAULT_MOUNT_POINT: ${VAULT_MOUNT_POINT}
      VAULT_SECRET_PATH: ${VAULT_SECRET_PATH}

    volumes:
      - /mnt/nfs/docker-data/vault:/usr/bin/vault
      - /mnt/nfs/docker-data/.cache:/root/.cache
    networks:
      - demo-backend-network
    deploy:
      restart_policy:
        condition: on-failure
      placement:
        constraints:
          - "node.hostname == ml-1"
      resources:
        limits:
          cpus: '12'
          memory: 24G
        reservations:
          cpus: '4'
          memory: 8G

networks:
  demo-frontend-network:
```

```
    external: true

  demo-backend-network:
    external: true
```

Примечание: Убедитесь, что образы доступны в вашем registry или замените пути на актуальные.

4.2 Развертывание через Docker Swarm

Подготовка узлов Swarm

```
# На управляющем узле создайте метки для узлов
docker node update --label-add TAG=worker <node-id>


# Проверка меток
docker node inspect <node-id> | grep -A 5 Labels
```

Развертывание стека

```
# Переход в директорию с конфигурацией
cd /opt/demo/config

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

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

4.3 Конфигурация микросервисов

API-сервис

API-сервис требует дополнительной настройки переменных окружения. Добавьте в stack.env:

```
# API настройки
API_DEBUG=false
API_HOST=0.0.0.0
API_PORT=8000
DATABASE_URL=sqlite:///antifraud.db
```

CDR-сервис

CDR-сервис требует настройки подключения к ClickHouse и Kafka. Убедитесь, что в Vault настроены:

- CDR_KAFKA_URL - адрес Kafka брокера
- CDR_HOST - адрес ClickHouse
- CDR_PORT - порт ClickHouse (по умолчанию 8123)
- CDR_DATABASE - имя базы данных
- CDR_USER - пользователь ClickHouse
- CDR_PASSWORD - пароль ClickHouse

Classifier-сервис

Classifier-сервис требует CPU для работы.

Importer-сервис

Importer-сервис требует настройки подключения к внешним источникам данных. Настройте в Vault:

- IMPORTER_KAFKA_URL - адрес Kafka
- IMPORTER_PBX_HOST - адрес PBX (Asterisk)
- IMPORTER_PBX_PORT - порт PBX

Notificator-сервис

Notificator-сервис требует настройки каналов уведомлений. Настройте в Vault:

- NOTIFICATOR_TELEGRAM_BOT_TOKEN - токен Telegram бота
- NOTIFICATOR_TELEGRAM_CHAT_ID - ID чата для уведомлений
- NOTIFICATOR_EMAIL_SMTP_HOST - SMTP сервер
- NOTIFICATOR_EMAIL_SMTP_PORT - SMTP порт
- NOTIFICATOR_EMAIL_FROM - адрес отправителя
- NOTIFICATOR_EMAIL_TO - адреса получателей

5. Инициализация базы данных

5.1 Создание таблиц ClickHouse

Подключитесь к ClickHouse и выполните SQL-скрипт:

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

Выполнение SQL-скрипта

Создайте файл **clickhouse_init.sql**:

```
CREATE TABLE IF NOT EXISTS calls (
    call_id String,
    start_time DateTime,
    stop_time DateTime,
    duration UInt32,
    direction String,
    geolocation String,
    from String,
    from_ip String,
    from_codek String,
    from_freq String,
    from_channels String,
    to String,
    to_ip String,
    to_codek String,
    to_freq String,
    to_channels String,
) ENGINE = ReplacingMergeTree
ORDER BY call_id;

CREATE TABLE IF NOT EXISTS frauds (
    call_id String,
    is_valid UInt8 DEFAULT 0,
    notification_time DateTime,
    metric Float32,
    keywords Array(String),
) ENGINE = ReplacingMergeTree
ORDER BY call_id;

CREATE TABLE IF NOT EXISTS frauds_log (
    call_id String,
    tstamp DateTime,
    is_fraud UInt8 DEFAULT 0,
    text String,
```

```
    metric Float32,
    keywords Array(String),
    org String DEFAULT '',
    category String DEFAULT '',
) ENGINE = MergeTree
ORDER BY tstamp
TTL tstamp + INTERVAL 3 DAY;

CREATE TABLE IF NOT EXISTS fraudwords
(
    id UInt64,
    text String,
    score Float64,
    created_at DateTime,
    updated_at DateTime,
)
ENGINE = ReplacingMergeTree(updated_at)
ORDER BY id
SETTINGS index_granularity = 8192;
```

Выполните скрипт:

```
docker exec -i clickhouse clickhouse-client <
clickhouse_init.sql
```

5.2 Настройка PostgreSQL для Controller

```
# Подключение к PostgreSQL
docker exec -it postgres psql -U admin -d antifraud

# Применение миграций (если используется Alembic)
# Миграции применяются автоматически при запуске
Controller-сервиса
```

6. Проверка работоспособности

6.1 Проверка healthcheck эндпоинтов

Проверьте доступность каждого сервиса:

```
# CDR-сервис
curl http://localhost:8800/healthcheck
```

```
# Importer-сервис
curl http://localhost:6061/healthcheck

# Notificator-сервис
curl http://localhost:8803/healthcheck

# Classifier-сервис
curl http://localhost:6062/healthcheck

# API-сервис (если развернут)
curl http://localhost:8000/healthcheck
```

Ожидаемый ответ: {"status": "ok"} или аналогичный JSON с информацией о статусе.

6.2 Проверка подключений к зависимостям

Проверка ClickHouse

```
docker exec -it clickhouse clickhouse-client --query
"SELECT count() FROM system.tables WHERE
database='cdr'"
```

Проверка Kafka/RedPanda

```
docker exec -it redpanda rpk topic list
```

Проверка Vault

```
vault status
vault kv get demo/settings
```

6.3 Тестирование основных функций

Тест обработки CDR

Отправьте тестовое событие в Kafka:

```
# Создание тестового топика
docker exec -it redpanda rpk topic create test-cdr

# Отправка тестового сообщения
docker exec -it redpanda rpk topic produce test-cdr
--key test-key --value
```

```
'{"call_id":"test-123","from":"+79001234567","to":"+79
007654321"}'
```

Проверьте логи CDR-сервиса:

```
docker service logs demo_cdr --tail 50
```

Тест Classifier

Отправьте тестовый запрос на транскрибацию:

```
curl -X POST http://localhost:6062/classify \
  -H "Content-Type: application/json" \
  -d '{
    "audio_url": "http://example.com/test.wav",
    "call_id": "test-123"
  }'
```

7. Настройка мониторинга

7.1 Интеграция с Prometheus/Grafana

Настройка Prometheus

Создайте конфигурацию Prometheus для сбора метрик:

```
# prometheus.yml
global:
  scrape_interval: 15s

scrape_configs:
  - job_name: 'demo-cdr'
    static_configs:
      - targets: ['cdr:8800']

  - job_name: 'demo-classifier'
    static_configs:
      - targets: ['classifier:6062']

  - job_name: 'demo-notificator'
    static_configs:
      - targets: ['notificator:8803']
```

Настройка Grafana

1. Импортируйте дашборды для мониторинга сервисов
2. Настройте источники данных (Prometheus, Loki)

3. Создайте алерты для критических метрик

7.2 Настройка логирования (Loki)

Настройте сбор логов в Loki:

```
# docker-compose для Loki
version: '3.9'
services:
  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

volumes:
  loki-data:
```

7.3 Настройка Zabbix (опционально)

Для интеграции с Zabbix:

1. Установите Zabbix Agent на узлах Swarm
2. Настройте шаблоны для мониторинга Docker контейнеров
3. Создайте триггеры для критических событий

8. Устранение неполадок

8.1 Типичные проблемы и решения

Проблема: Сервисы не запускаются

Решение:

```
# Проверка логов сервиса
docker service logs demo_cdr --tail 100

# Проверка статуса сервиса
docker service ps demo_cdr

# Проверка доступности зависимостей
docker exec -it clickhouse clickhouse-client --query
```

```
"SELECT 1"
docker exec -it redpanda rpk cluster info
```

Проблема: Ошибки подключения к Vault
Решение:

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

# Проверка переменных окружения
docker service inspect demo_cdr --pretty | grep -A 20
Env

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

Проблема: Ошибки записи в ClickHouse
Решение:

```
# 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"
```

8.2 Логи и диагностика

Просмотр логов сервисов

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

# Логи всех сервисов стека
docker stack services demo --format "{{.Name}}" |
xargs -I {} docker service logs {} --tail 50
```

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

Диагностика сети

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

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

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

Диагностика ресурсов

```
# Использование ресурсов узлами
docker node ls
docker node inspect <node-id> | grep -A 10 Resources

# Статистика контейнеров
docker stats $(docker ps -q)
```

8.3 Восстановление после сбоев

Перезапуск сервисов

```
# Перезапуск конкретного сервиса
docker service update --force demo_cdr

# Перезапуск всего стека
docker stack rm demo
docker stack deploy -c docker-compose.stack.yaml demo
```

Восстановление данных

```
# Резервное копирование ClickHouse
docker exec clickhouse clickhouse-client --query
"BACKUP DATABASE cdr TO Disk('backups', 'backup_$(date
```

```
+%Y%m%d).zip')"

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

Приложение А: Переменные окружения

Основные переменные проекта

Переменная	Описание	Пример значения
PROJECT	Имя проекта Swarm	demo
AF_PROJECT	Имя проекта антифрода	demo
VAULT_ENABLE	Включить использование Vault	yes
VAULT_SKIP_VERIFY	Пропустить проверку SSL сертификата Vault	true
VAULT_ADDR	Адрес Vault сервера	https://vault.af.internal
VAULT_MOUNT_POINT	Точка монтирования Vault	demo
VAULT_SECRET_PATH	Путь к секретам	settings
VAULT_ROLE_ID	Role ID для AppRole	(из Vault)
VAULT_SECRET_ID	Secret ID для AppRole	(из Vault)

Переменные CDR-сервиса

Переменная	Описание	Значение по умолчанию
CDR_KAFKA_URL	URL Kafka брокера	kafka:9092
CDR_HOST	Адрес ClickHouse	clickhouse
CDR_PORT	Порт ClickHouse	8123
CDR_DATABASE	Имя базы данных	cdr
CDR_TIMEOUT	Таймаут операций (сек)	60
CDR_SAVE_WAV	Включить запись аудио	true
CDR_AUDIO_DIR	Каталог аудиофайлов	/opt/audio
CDR_S3_URL	URL S3 хранилища	(пусто)
CDR_S3_BUCKET	Имя S3 bucket	(пусто)

Приложение В: Полезные команды

Управление Docker Swarm

```
# Просмотр всех сервисов
docker service ls

# Детальная информация о сервисе
docker service inspect demo_cdr

# Масштабирование сервиса
docker service scale demo_cdr=3

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

Мониторинг

```
# Статус стека
docker stack ps demo

# Использование ресурсов
docker stats

# События Swarm
docker events
```

Заключение

После выполнения всех шагов инструкции платформа "Волна" должна быть полностью развернута и готова к работе. Убедитесь, что все сервисы работают корректно, выполнив проверки из раздела 6. При возникновении проблем обращайтесь к разделу 8 "Устранение неполадок" или к технической поддержке.