Руководство по развертыванию Telegram Store MVP

Версия документа: 1.0
Дата: 2025-11-30

Содержание

Требования к серверу

Минимальные требования для MVP

Application Server (FastAPI + Telegram Bot):
- **CPU:** 2 vCPU
- **RAM:** 4 GB
- **Disk:** 20 GB SSD
- **OS:** Ubuntu 22.04 LTS или новее
- **Network:** Статический IP адрес, открытые порты 80, 443

Database Server (PostgreSQL):
- **CPU:** 2 vCPU
- **RAM:** 4 GB
- **Disk:** 40 GB SSD (с учетом роста данных)
- **OS:** Ubuntu 22.04 LTS

Рекомендации:
- Для production рекомендуется разделить application и database на разные серверы
- Использовать managed database сервис (AWS RDS, DigitalOcean Managed Databases) для

упрощения администрирования
- Минимум 10 GB свободного места для логов и временных файлов

Требования к сети

- Доменное имя (например, `api.telegram-shop.example`)
- DNS записи настроены на IP адрес сервера
- Firewall правила:
- Входящие: 22 (SSH), 80 (HTTP), 443 (HTTPS)
- Исходящие: 443 (для Telegram API, Google APIs)
- Между серверами: 5432 (PostgreSQL) если БД на отдельном сервере

Подготовка сервера

1. Обновление системы

```
# Подключение к серверу
ssh root@your-server-ip

# Обновление пакетов
apt update && apt upgrade -y

# Установка базовых утилит
apt install -y curl wget git vim htop ufw
```

2. Создание пользователя для приложения

```
# Создание пользователя
adduser telegram-shop
usermod -aG sudo telegram-shop

# Переключение на нового пользователя
su - telegram-shop
```

3. Настройка Firewall (UFW)

```
# Включение UFW
sudo ufw allow OpenSSH
sudo ufw allow 80/tcp
sudo ufw allow 443/tcp
sudo ufw enable

# Проверка статуса
sudo ufw status
```

Установка зависимостей

1. Python 3.11+

```
# Установка Python 3.11
sudo apt install -y software-properties-common
sudo add-apt-repository ppa:deadsnakes/ppa -y
sudo apt update
sudo apt install -y python3.11 python3.11-venv python3.11-dev

# Установка pip
curl -sS https://bootstrap.pypa.io/get-pip.py | python3.11

# Проверка версии
python3.11 --version
```

2. PostgreSQL 15+

```
# Добавление репозитория PostgreSQL
sudo sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt $(lsb_release -cs)-pgdg
main" > /etc/apt/sources.list.d/pgdg.list'
wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo apt-key ad
d -

# Установка PostgreSQL
sudo apt update
sudo apt install -y postgresql-15 postgresql-contrib-15

# Проверка статуса
sudo systemctl status postgresql
```

3. Nginx

```
# Установка Nginx
sudo apt install -y nginx

# Запуск и автозагрузка
sudo systemctl start nginx
sudo systemctl enable nginx

# Проверка
sudo systemctl status nginx
```

4. Docker и Docker Compose (опционально)

```
# Установка Docker
curl -fsSL https://get.docker.com -o get-docker.sh
sudo sh get-docker.sh

# Добавление пользователя в группу docker
sudo usermod -aG docker $USER

# Установка Docker Compose
sudo curl -L "https://github.com/docker/compose/releases/latest/download/docker-com-
pose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose

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

5. Redis (для FSM хранилища)

```
# Установка Redis
sudo apt install -y redis-server

# Настройка для production
sudo sed -i 's/supervised no/supervised systemd/' /etc/redis/redis.conf

# Перезапуск
sudo systemctl restart redis-server
sudo systemctl enable redis-server

# Проверка
redis-cli ping
# Ожидаемый ответ: PONG
```

Настройка PostgreSQL

1. Создание базы данных и пользователя

```
# Переключение на пользователя postgres
sudo -u postgres psql

# В psql консоли:
CREATE DATABASE telegram_shop;
CREATE USER telegram_shop_user WITH ENCRYPTED PASSWORD 'your_secure_password_here';
GRANT ALL PRIVILEGES ON DATABASE telegram_shop TO telegram_shop_user;

# Для PostgreSQL 15+ дополнительно:
\c telegram_shop
GRANT ALL ON SCHEMA public TO telegram_shop_user;
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO telegram_shop_user;
GRANT ALL PRIVILEGES ON ALL SEQUENCES IN SCHEMA public TO telegram_shop_user;

# Выход
\q
```

2. Настройка удаленного доступа (если БД на отдельном сервере)

```
# Редактирование postgresql.conf
sudo vim /etc/postgresql/15/main/postgresql.conf

# Изменить:
listen_addresses = '*'  # или конкретный IP

# Редактирование pg_hba.conf
sudo vim /etc/postgresql/15/main/pg_hba.conf

# Добавить строку (замените IP на адрес application сервера):
host    telegram_shop    telegram_shop_user    10.0.0.5/32    md5

# Перезапуск PostgreSQL
sudo systemctl restart postgresql
```

3. Оптимизация PostgreSQL для production

```
# Редактирование postgresql.conf
sudo vim /etc/postgresql/15/main/postgresql.conf

# Рекомендуемые настройки для 4GB RAM:
shared_buffers = 1GB
effective_cache_size = 3GB
maintenance_work_mem = 256MB
checkpoint_completion_target = 0.9
wal_buffers = 16MB
default_statistics_target = 100
random_page_cost = 1.1
effective_io_concurrency = 200
work_mem = 10MB
min_wal_size = 1GB
max_wal_size = 4GB
max_connections = 100

# Перезапуск
sudo systemctl restart postgresql
```

4. Создание схемы базы данных

```
# Клонирование репозитория проекта
cd /home/telegram-shop
git clone https://github.com/your-org/telegram-shop-mvp.git
cd telegram-shop-mvp

# Создание виртуального окружения
python3.11 -m venv venv
source venv/bin/activate

# Установка зависимостей
pip install -r requirements.txt

# Применение миграций (используя Alembic)
alembic upgrade head
```

Настройка приложения

1. Структура проекта

```
/home/telegram-shop/telegram-shop-mvp/
├── src/
│   ├── api/          # FastAPI приложение
│   ├── bot/          # Telegram bot (aiogram)
│   ├── models/       # SQLAlchemy модели
│   ├── schemas/      # Pydantic схемы
│   ├── services/     # Бизнес-логика
│   └── integrations/ # Google Sheets, Analytics
├── alembic/          # Миграции БД
├── admin-panel/      # React приложение
├── .env.example      # Пример переменных окружения
├── requirements.txt
└── docker-compose.yml
```

2. Настройка переменных окружения

```
# Копирование примера
cp .env.example .env

# Редактирование .env
vim .env
```

Содержимое `.env`:

```
# ==================== APPLICATION ====================
APP_ENV=production
DEBUG=False
SECRET_KEY=your-super-secret-key-min-32-chars-long
API_HOST=0.0.0.0
API_PORT=8000

# ==================== DATABASE ====================
DATABASE_URL=postgresql+asyncpg://tele-
gram_shop_user:your_secure_password_here@localhost:5432/telegram_shop

# ==================== TELEGRAM ====================
TELEGRAM_BOT_TOKEN=1234567890:ABCdefGHIjklMNOpqrsTUVwxyz
TELEGRAM_WEBHOOK_URL=https://api.telegram-shop.example/webhook
TELEGRAM_WEBHOOK_SECRET=random-secret-token-for-webhook-validation
TELEGRAM_ADMIN_CHANNEL_ID=-1001234567890

# ==================== JWT ====================
JWT_SECRET_KEY=another-super-secret-key-for-jwt-tokens
JWT_ALGORITHM=HS256
JWT_ACCESS_TOKEN_EXPIRE_MINUTES=30
JWT_REFRESH_TOKEN_EXPIRE_DAYS=7

# ==================== REDIS ====================
REDIS_URL=redis://localhost:6379/0

# ==================== GOOGLE SHEETS ====================
GOOGLE_SERVICE_ACCOUNT_FILE=/home/telegram-shop/telegram-shop-mvp/secrets/google-
service-account.json

# ==================== ANALYTICS ====================
GOOGLE_ANALYTICS_MEASUREMENT_ID=G-XXXXXXXXXX
GOOGLE_ANALYTICS_API_SECRET=your-ga4-api-secret
YANDEX_METRIKA_TAG_ID=12345678

# ==================== CORS ====================
CORS_ORIGINS=https://admin.telegram-shop.example,https://telegram-shop.example

# ==================== LOGGING ====================
LOG_LEVEL=INFO
LOG_FILE=/var/log/telegram-shop/app.log
```

3. Генерация секретных ключей

```
# Генерация SECRET_KEY
python3 -c "import secrets; print(secrets.token_urlsafe(32))"

# Генерация JWT_SECRET_KEY
python3 -c "import secrets; print(secrets.token_urlsafe(32))"

# Генерация TELEGRAM_WEBHOOK_SECRET
python3 -c "import secrets; print(secrets.token_hex(16))"
```

4. Настройка Google Service Account

```
# Создание директории для секретов
mkdir -p /home/telegram-shop/telegram-shop-mvp/secrets
chmod 700 /home/telegram-shop/telegram-shop-mvp/secrets

# Загрузка JSON ключа сервисного аккаунта
# (скачайте из Google Cloud Console)
scp google-service-account.json telegram-shop@your-server:/home/telegram-shop/
telegram-shop-mvp/secrets/

# Установка прав
chmod 600 /home/telegram-shop/telegram-shop-mvp/secrets/google-service-account.json
```

Docker и Docker Compose

1. Dockerfile для Backend API

`Dockerfile`:

```
FROM python:3.11-slim

WORKDIR /app

# Установка системных зависимостей
RUN apt-get update && apt-get install -y \
    gcc \
    postgresql-client \
    && rm -rf /var/lib/apt/lists/*

# Копирование requirements
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

# Копирование приложения
COPY . .

# Создание пользователя без root прав
RUN useradd -m -u 1000 appuser && chown -R appuser:appuser /app
USER appuser

# Expose порт
EXPOSE 8000

# Запуск приложения
CMD ["uvicorn", "src.api.main:app", "--host", "0.0.0.0", "--port", "8000", "--work-
ers", "4"]
```

2. Docker Compose конфигурация

`docker-compose.yml`:

```yaml
version: '3.8'

services:
  postgres:
    image: postgres:15-alpine
    container_name: telegram_shop_db
    environment:
      POSTGRES_DB: telegram_shop
      POSTGRES_USER: telegram_shop_user
      POSTGRES_PASSWORD: ${DB_PASSWORD}
    volumes:
      - postgres_data:/var/lib/postgresql/data
      - ./backups:/backups
    ports:
      - "5432:5432"
    restart: unless-stopped
    healthcheck:
      test: ["CMD-SHELL", "pg_isready -U telegram_shop_user"]
      interval: 10s
      timeout: 5s
      retries: 5

  redis:
    image: redis:7-alpine
    container_name: telegram_shop_redis
    ports:
      - "6379:6379"
    volumes:
      - redis_data:/data
    restart: unless-stopped
    healthcheck:
      test: ["CMD", "redis-cli", "ping"]
      interval: 10s
      timeout: 3s
      retries: 5

  api:
    build: .
    container_name: telegram_shop_api
    env_file:
      - .env
    depends_on:
      postgres:
        condition: service_healthy
      redis:
        condition: service_healthy
    ports:
      - "8000:8000"
    volumes:
      - ./src:/app/src
      - ./secrets:/app/secrets:ro
      - ./logs:/app/logs
    restart: unless-stopped
    command: uvicorn src.api.main:app --host 0.0.0.0 --port 8000 --workers 4

  bot:
    build: .
    container_name: telegram_shop_bot
    env_file:
      - .env
    depends_on:
      - api
```

```
      - redis
    volumes:
      - ./src:/app/src
      - ./logs:/app/logs
    restart: unless-stopped
    command: python -m src.bot.main

  nginx:
    image: nginx:alpine
    container_name: telegram_shop_nginx
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - ./nginx/nginx.conf:/etc/nginx/nginx.conf:ro
      - ./nginx/ssl:/etc/nginx/ssl:ro
      - ./admin-panel/build:/usr/share/nginx/html:ro
      - ./certbot/conf:/etc/letsencrypt:ro
      - ./certbot/www:/var/www/certbot:ro
    depends_on:
      - api
    restart: unless-stopped

volumes:
  postgres_data:
  redis_data:
```

3. Запуск через Docker Compose

```
# Сборка образов
docker-compose build

# Запуск всех сервисов
docker-compose up -d

# Проверка статуса
docker-compose ps

# Просмотр логов
docker-compose logs -f api

# Остановка
docker-compose down
```

Настройка Nginx

1. Конфигурация Nginx как Reverse Proxy

`/etc/nginx/sites-available/telegram-shop`:

```
# Upstream для FastAPI
upstream fastapi_backend {
    server 127.0.0.1:8000;
    # Для Docker:
    # server api:8000;
}

# HTTP -> HTTPS редирект
server {
    listen 80;
    listen [::]:80;
    server_name api.telegram-shop.example;

    # Для Let's Encrypt ACME challenge
    location /.well-known/acme-challenge/ {
        root /var/www/certbot;
    }

    location / {
        return 301 https://$server_name$request_uri;
    }
}

# HTTPS сервер для API
server {
    listen 443 ssl http2;
    listen [::]:443 ssl http2;
    server_name api.telegram-shop.example;

    # SSL сертификаты
    ssl_certificate /etc/letsencrypt/live/api.telegram-shop.example/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/api.telegram-shop.example/privkey.pem;

    # SSL настройки
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers HIGH:!aNULL:!MD5;
    ssl_prefer_server_ciphers on;
    ssl_session_cache shared:SSL:10m;
    ssl_session_timeout 10m;

    # Security headers
    add_header Strict-Transport-Security "max-age=31536000; includeSubDomains" always;
    add_header X-Frame-Options "SAMEORIGIN" always;
    add_header X-Content-Type-Options "nosniff" always;
    add_header X-XSS-Protection "1; mode=block" always;

    # Логи
    access_log /var/log/nginx/telegram-shop-api-access.log;
    error_log /var/log/nginx/telegram-shop-api-error.log;

    # Максимальный размер загружаемых файлов
    client_max_body_size 50M;

    # Proxy к FastAPI
    location / {
        proxy_pass http://fastapi_backend;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;

        # Таймауты
```

```
        proxy_connect_timeout 60s;
        proxy_send_timeout 60s;
        proxy_read_timeout 60s;
    }

    # Webhook endpoint (дополнительная защита)
    location /webhook {
        proxy_pass http://fastapi_backend;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;

        # Ограничение доступа только с IP Telegram
        # (опционально, список IP: https://core.telegram.org/bots/webhooks#the-short-
version)
        # allow 149.154.160.0/20;
        # allow 91.108.4.0/22;
        # deny all;
    }
}

# HTTPS сервер для Admin Panel
server {
    listen 443 ssl http2;
    listen [::]:443 ssl http2;
    server_name admin.telegram-shop.example;

    ssl_certificate /etc/letsencrypt/live/admin.telegram-shop.example/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/admin.telegram-shop.example/privkey.pem;

    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers HIGH:!aNULL:!MD5;
    ssl_prefer_server_ciphers on;

    root /var/www/telegram-shop/admin-panel/build;
    index index.html;

    # React Router (SPA)
    location / {
        try_files $uri $uri/ /index.html;
    }

    # Кэширование статики
    location ~* \.(js|css|png|jpg|jpeg|gif|ico|svg|woff|woff2|ttf|eot)$ {
        expires 1y;
        add_header Cache-Control "public, immutable";
    }
}
```

2. Активация конфигурации

```
# Создание символической ссылки
sudo ln -s /etc/nginx/sites-available/telegram-shop /etc/nginx/sites-enabled/

# Проверка конфигурации
sudo nginx -t

# Перезагрузка Nginx
sudo systemctl reload nginx
```

SSL сертификаты (Let’s Encrypt)

1. Установка Certbot

```
# Установка Certbot
sudo apt install -y certbot python3-certbot-nginx
```

2. Получение сертификатов

```
# Для API домена
sudo certbot --nginx -d api.telegram-shop.example

# Для Admin Panel домена
sudo certbot --nginx -d admin.telegram-shop.example

# Следуйте инструкциям на экране
```

3. Автоматическое обновление сертификатов

```
# Тест автообновления
sudo certbot renew --dry-run

# Certbot автоматически создает cron job для обновления
# Проверка:
sudo systemctl status certbot.timer
```

4. Ручное обновление (если необходимо)

```
# Обновление всех сертификатов
sudo certbot renew

# Перезагрузка Nginx после обновления
sudo systemctl reload nginx
```

Systemd для автозапуска

1. Systemd unit для FastAPI

`/etc/systemd/system/telegram-shop-api.service`:

```
[Unit]
Description=Telegram Shop FastAPI Application
After=network.target postgresql.service redis.service
Requires=postgresql.service redis.service

[Service]
Type=notify
User=telegram-shop
Group=telegram-shop
WorkingDirectory=/home/telegram-shop/telegram-shop-mvp
Environment="PATH=/home/telegram-shop/telegram-shop-mvp/venv/bin"
EnvironmentFile=/home/telegram-shop/telegram-shop-mvp/.env

ExecStart=/home/telegram-shop/telegram-shop-mvp/venv/bin/gunicorn \
    -k uvicorn.workers.UvicornWorker \
    -w 4 \
    -b 0.0.0.0:8000 \
    --timeout 60 \
    --access-logfile /var/log/telegram-shop/access.log \
    --error-logfile /var/log/telegram-shop/error.log \
    src.api.main:app

Restart=always
RestartSec=10

[Install]
WantedBy=multi-user.target
```

2. Systemd unit для Telegram Bot

`/etc/systemd/system/telegram-shop-bot.service`:

```
[Unit]
Description=Telegram Shop Bot
After=network.target telegram-shop-api.service redis.service
Requires=telegram-shop-api.service redis.service

[Service]
Type=simple
User=telegram-shop
Group=telegram-shop
WorkingDirectory=/home/telegram-shop/telegram-shop-mvp
Environment="PATH=/home/telegram-shop/telegram-shop-mvp/venv/bin"
EnvironmentFile=/home/telegram-shop/telegram-shop-mvp/.env

ExecStart=/home/telegram-shop/telegram-shop-mvp/venv/bin/python -m src.bot.main

Restart=always
RestartSec=10

[Install]
WantedBy=multi-user.target
```

3. Создание директории для логов

```
# Создание директории
sudo mkdir -p /var/log/telegram-shop

# Установка владельца
sudo chown telegram-shop:telegram-shop /var/log/telegram-shop
```

4. Запуск и управление сервисами

```
# Перезагрузка systemd
sudo systemctl daemon-reload

# Запуск сервисов
sudo systemctl start telegram-shop-api
sudo systemctl start telegram-shop-bot

# Включение автозапуска
sudo systemctl enable telegram-shop-api
sudo systemctl enable telegram-shop-bot

# Проверка статуса
sudo systemctl status telegram-shop-api
sudo systemctl status telegram-shop-bot

# Просмотр логов
sudo journalctl -u telegram-shop-api -f
sudo journalctl -u telegram-shop-bot -f

# Перезапуск
sudo systemctl restart telegram-shop-api
sudo systemctl restart telegram-shop-bot
```

Настройка Telegram Webhook

1. Установка webhook через API

```
# Используя curl
curl -X POST "https://api.telegram.org/bot<YOUR_BOT_TOKEN>/setWebhook" \
  -H "Content-Type: application/json" \
  -d '{
    "url": "https://api.telegram-shop.example/webhook",
    "secret_token": "your-webhook-secret-from-env",
    "max_connections": 40,
    "allowed_updates": ["message", "callback_query"]
  }'
```

2. Проверка webhook

```
# Получение информации о webhook
curl "https://api.telegram.org/bot<YOUR_BOT_TOKEN>/getWebhookInfo"

# Ожидаемый ответ:
# {
#   "ok": true,
#   "result": {
#     "url": "https://api.telegram-shop.example/webhook",
#     "has_custom_certificate": false,
#     "pending_update_count": 0,
#     "max_connections": 40
#   }
# }
```

3. Удаление webhook (для отладки)

```
# Удаление webhook
curl "https://api.telegram.org/bot<YOUR_BOT_TOKEN>/deleteWebhook"
```

Мониторинг и логирование

1. Настройка ротации логов

`/etc/logrotate.d/telegram-shop`:

```
/var/log/telegram-shop/*.log {
    daily
    rotate 14
    compress
    delaycompress
    notifempty
    create 0640 telegram-shop telegram-shop
    sharedscripts
    postrotate
        systemctl reload telegram-shop-api > /dev/null 2>&1 || true
        systemctl reload telegram-shop-bot > /dev/null 2>&1 || true
    endscript
}
```

2. Мониторинг с помощью htop

```
# Установка htop
sudo apt install -y htop

# Запуск
htop
```

3. Мониторинг PostgreSQL

```
# Подключение к БД
sudo -u postgres psql telegram_shop

# Проверка активных соединений
SELECT count(*) FROM pg_stat_activity;

# Проверка размера БД
SELECT pg_size_pretty(pg_database_size('telegram_shop'));

# Проверка размера таблиц
SELECT
    schemaname,
    tablename,
    pg_size_pretty(pg_total_relation_size(schemaname||'.'||tablename)) AS size
FROM pg_tables
WHERE schemaname = 'public'
ORDER BY pg_total_relation_size(schemaname||'.'||tablename) DESC;
```

4. Настройка Prometheus + Grafana (опционально)

```
# Установка Prometheus
sudo apt install -y prometheus

# Установка Grafana
sudo apt-get install -y software-properties-common
sudo add-apt-repository "deb https://packages.grafana.com/oss/deb stable main"
wget -q -O - https://packages.grafana.com/gpg.key | sudo apt-key add -
sudo apt update
sudo apt install -y grafana

# Запуск
sudo systemctl start prometheus
sudo systemctl start grafana-server
sudo systemctl enable prometheus
sudo systemctl enable grafana-server
```

Добавление метрик в FastAPI:

```
# requirements.txt
prometheus-fastapi-instrumentator

# src/api/main.py
from prometheus_fastapi_instrumentator import Instrumentator

app = FastAPI()

Instrumentator().instrument(app).expose(app)
```

5. Мониторинг с помощью Sentry (опционально)

```
# Установка Sentry SDK
pip install sentry-sdk[fastapi]
```

```
# src/api/main.py
import sentry_sdk
from sentry_sdk.integrations.fastapi import FastApiIntegration

sentry_sdk.init(
    dsn="https://your-sentry-dsn",
    integrations=[FastApiIntegration()],
    traces_sample_rate=0.1,
    environment="production"
)
```

Backup и восстановление

1. Автоматический backup PostgreSQL

Скрипт backup: `/home/telegram-shop/scripts/backup-db.sh`:

```
#!/bin/bash

# Конфигурация
BACKUP_DIR="/home/telegram-shop/backups"
DB_NAME="telegram_shop"
DB_USER="telegram_shop_user"
TIMESTAMP=$(date +"%Y%m%d_%H%M%S")
BACKUP_FILE="$BACKUP_DIR/${DB_NAME}_${TIMESTAMP}.sql.gz"
RETENTION_DAYS=7

# Создание директории
mkdir -p $BACKUP_DIR

# Создание backup
PGPASSWORD="your_password" pg_dump -U $DB_USER -h localhost $DB_NAME | gzip > $BACKUP_
FILE

# Проверка успешности
if [ $? -eq 0 ]; then
    echo "Backup успешно создан: $BACKUP_FILE"

    # Удаление старых backup'ов
    find $BACKUP_DIR -name "${DB_NAME}_*.sql.gz" -mtime +$RETENTION_DAYS -delete
    echo "Старые backup'ы удалены (старше $RETENTION_DAYS дней)"
else
    echo "Ошибка при создании backup!"
    exit 1
fi

# Опционально: загрузка в облако (AWS S3, Google Cloud Storage)
# aws s3 cp $BACKUP_FILE s3://your-bucket/backups/
```

Установка прав:

```
chmod +x /home/telegram-shop/scripts/backup-db.sh
```

2. Настройка cron для автоматического backup

```
# Редактирование crontab
crontab -e

# Добавление задачи (ежедневно в 2:00 AM)
0 2 * * * /home/telegram-shop/scripts/backup-db.sh >> /var/log/telegram-shop/
backup.log 2>&1
```

3. Восстановление из backup

```
# Распаковка и восстановление
gunzip -c /home/telegram-shop/backups/telegram_shop_20251130_020000.sql.gz | \
  PGPASSWORD="your_password" psql -U telegram_shop_user -h localhost telegram_shop

# Или в два шага:
gunzip /home/telegram-shop/backups/telegram_shop_20251130_020000.sql.gz
PGPASSWORD="your_password" psql -U telegram_shop_user -h localhost telegram_shop < tel
egram_shop_20251130_020000.sql
```

4. Backup файлов приложения

```
# Backup .env и secrets
tar -czf /home/telegram-shop/backups/app-config-$(date +%Y%m%d).tar.gz \
  /home/telegram-shop/telegram-shop-mvp/.env \
  /home/telegram-shop/telegram-shop-mvp/secrets/

# Backup загруженных файлов (если есть)
tar -czf /home/telegram-shop/backups/uploads-$(date +%Y%m%d).tar.gz \
  /home/telegram-shop/telegram-shop-mvp/uploads/
```

Troubleshooting

1. API не отвечает

```
# Проверка статуса сервиса
sudo systemctl status telegram-shop-api

# Проверка логов
sudo journalctl -u telegram-shop-api -n 100 --no-pager

# Проверка портов
sudo netstat -tulpn | grep 8000

# Проверка процессов
ps aux | grep uvicorn

# Тест API напрямую
curl http://localhost:8000/api/v1/products
```

2. Telegram bot не получает обновления

```
# Проверка webhook
curl "https://api.telegram.org/bot<TOKEN>/getWebhookInfo"

# Проверка логов бота
sudo journalctl -u telegram-shop-bot -f

# Проверка доступности webhook URL
curl -I https://api.telegram-shop.example/webhook

# Проверка SSL сертификата
openssl s_client -connect api.telegram-shop.example:443 -servername api.telegram-
shop.example
```

3. Ошибки подключения к PostgreSQL

```
# Проверка статуса PostgreSQL
sudo systemctl status postgresql

# Проверка соединения
PGPASSWORD="your_password" psql -U telegram_shop_user -h localhost -d telegram_shop -
c "SELECT 1;"

# Проверка логов PostgreSQL
sudo tail -f /var/log/postgresql/postgresql-15-main.log

# Проверка активных соединений
sudo -u postgres psql -c "SELECT * FROM pg_stat_activity WHERE dat-
name='telegram_shop';"
```

4. Высокая нагрузка на сервер

```
# Проверка CPU и памяти
htop

# Проверка дискового пространства
df -h

# Проверка I/O
iostat -x 1

# Проверка медленных запросов PostgreSQL
sudo -u postgres psql telegram_shop -c "
SELECT
    pid,
    now() - pg_stat_activity.query_start AS duration,
    query,
    state
FROM pg_stat_activity
WHERE state != 'idle'
ORDER BY duration DESC;
"
```

5. Nginx ошибки

```
# Проверка конфигурации
sudo nginx -t

# Проверка логов ошибок
sudo tail -f /var/log/nginx/telegram-shop-api-error.log

# Проверка access логов
sudo tail -f /var/log/nginx/telegram-shop-api-access.log

# Перезапуск Nginx
sudo systemctl restart nginx
```

6. Проблемы с SSL сертификатами

```
# Проверка срока действия сертификата
sudo certbot certificates

# Ручное обновление
sudo certbot renew --force-renewal

# Проверка SSL конфигурации
curl -vI https://api.telegram-shop.example 2>&1 | grep -i ssl
```

7. Ошибки импорта/экспорта

```
# Проверка прав на файлы
ls -la /home/telegram-shop/telegram-shop-mvp/secrets/

# Проверка Google Service Account
python3 << EOF
from google.oauth2 import service_account
credentials = service_account.Credentials.from_service_account_file(
    '/home/telegram-shop/telegram-shop-mvp/secrets/google-service-account.json'
)
print("Credentials loaded successfully")
print(f"Service account email: {credentials.service_account_email}")
EOF
```

Контрольный список развертывания

- [ ] Сервер подготовлен (Ubuntu 22.04, обновлен)
- [ ] Firewall настроен (UFW)
- [ ] Python 3.11+ установлен
- [ ] PostgreSQL 15+ установлен и настроен
- [ ] Redis установлен
- [ ] Nginx установлен
- [ ] База данных создана, пользователь настроен
- [ ] Миграции применены (Alembic)
- [ ] `.env` файл настроен со всеми переменными

- [ ] Секретные ключи сгенерированы
- [ ] Google Service Account настроен
- [ ] SSL сертификаты получены (Let’s Encrypt)
- [ ] Nginx конфигурация активирована
- [ ] Systemd сервисы созданы и запущены
- [ ] Telegram webhook установлен
- [ ] Логирование настроено
- [ ] Backup скрипты настроены и протестированы
- [ ] Мониторинг настроен
- [ ] Admin Panel собран и развернут
- [ ] Тестовый заказ успешно создан через бота
- [ ] Тестовый вход в Admin Panel выполнен

Дополнительные ресурсы

- FastAPI Deployment (https://fastapi.tiangolo.com/deployment/)
- Nginx Documentation (https://nginx.org/en/docs/)
- PostgreSQL Documentation (https://www.postgresql.org/docs/)
- Let’s Encrypt Documentation (https://letsencrypt.org/docs/)
- Telegram Bot API (https://core.telegram.org/bots/api)
- aiogram Documentation (https://docs.aiogram.dev/)

Примечание: Замените все placeholder значения (пароли, токены, домены) на реальные перед развертыванием в production.