# SneakerDropBot - Render.com Deployment Guide

This guide will help you deploy a fully functional SneakerDropBot on Render.com with lightweight scraping capabilities.

## 🚀 Features

* **Real-time sneaker tracking** across Nike, Adidas, and StockX
* **Intelligent parsing** of user tracking requests
* **Background monitoring** with configurable intervals
* **Premium subscription support** via Stripe
* **Interactive Telegram bot** with search and tracking features
* **Lightweight scraping system** optimized for Render.com free tier

## 📋 Prerequisites

1. **Telegram Bot Token** - Create a bot via [@BotFather](https://t.me/BotFather)
2. **MongoDB Database** - Free cluster on [MongoDB Atlas](https://www.mongodb.com/atlas)
3. **Render.com Account** - Free tier supported
4. **Stripe Account** (optional) - For premium features

## 🔧 Environment Variables

Set these in your Render.com dashboard:

### Required

TELEGRAM\_BOT\_TOKEN=your\_bot\_token\_here  
MONGODB\_URI=mongodb+srv://username:password@cluster.mongodb.net/sneakerdropbot

### Optional

STRIPE\_SECRET\_KEY=sk\_test\_...  
STRIPE\_PUBLISHABLE\_KEY=pk\_test\_...  
ADMIN\_TELEGRAM\_CHAT\_ID=your\_telegram\_user\_id

## 🚀 Quick Deploy to Render.com

### Method 1: Using render.yaml (Recommended)

1. Fork this repository
2. Connect your GitHub to Render.com
3. Create a new “Web Service” and select your forked repo
4. Render will automatically detect the render.yaml configuration
5. Set the required environment variables in the Render dashboard
6. Deploy!

### Method 2: Manual Setup

1. Create a new “Web Service” on Render.com
2. Connect your GitHub repository
3. Configure build settings:
   * **Build Command**: pip install -r requirements-render.txt
   * **Start Command**: python main\_render.py
   * **Environment**: Python 3.11
4. Set environment variables
5. Deploy

## 🛠 Local Development

# Install dependencies  
pip install -r requirements-render.txt  
  
# Set environment variables  
export TELEGRAM\_BOT\_TOKEN="your\_token"  
export MONGODB\_URI="your\_mongodb\_uri"  
  
# Run locally  
python main\_render.py

## 📊 Monitoring & Health Checks

The bot includes several monitoring endpoints:

* **Health Check**: GET /health
* **Bot Stats**: GET /api/stats
* **Scraper Health**: GET /api/scraper/health
* **Search Test**: GET /api/search/{keyword}
* **Force Monitoring**: POST /api/monitor/force

## 🤖 Bot Commands

### User Commands

* /start - Welcome message and main menu
* /track - Start tracking a sneaker
* /status - View account status and tracked sneakers
* /premium - Upgrade to premium features
* /help - Show help and commands

### Example Usage

User: "Track Jordan 4 Bred size 10.5 under $220"  
Bot: ✅ Tracking added! I'll monitor major retailers and alert you when available.  
  
User: "Search Yeezy 350"  
Bot: 🔍 Search Results for 'Yeezy 350':  
 1. Yeezy 350 Cream  
 📍 StockX | $180 | ✅ In Stock

### Admin Commands (if ADMIN\_TELEGRAM\_CHAT\_ID is set)

* /admin - Show admin panel
* /admin stats - Detailed statistics
* /admin health - System health check

## 🕷️ Scraper System

The lightweight scraper system includes:

### Supported Retailers

* **Nike**: Product search and availability
* **Adidas**: Product search and pricing
* **StockX**: Resell market prices and availability
* **Mock Scrapers**: For other retailers (development/fallback)

### Features

* **Intelligent Parsing**: Multiple parsing strategies for website changes
* **Rate Limiting**: Respectful scraping with delays
* **Error Handling**: Circuit breaker patterns and retry logic
* **Health Monitoring**: Real-time scraper health tracking

### Configuration

The scraper system is optimized for Render.com free tier: - **15-minute monitoring cycles** (configurable) - **3 concurrent scrapers** maximum - **30-second timeout** per search cycle - **Minimal resource usage**

## 💾 Database Schema

### Collections

* **users**: User accounts and preferences
* **tracked\_sneakers**: User tracking requests
* **alerts**: Alert history and delivery status
* **analytics**: Daily usage statistics

### Indexes

Automatically created indexes for optimal performance: - users.telegram\_id (unique) - tracked\_sneakers.user\_id + sneaker\_name - alerts.user\_id + created\_at

## 💎 Premium Features

Premium users get: - **Unlimited tracking** (vs 1 for free) - **Instant alerts** (no delays) - **Flip opportunity analysis** - **Early access notifications** - **Priority customer support**

## 🔧 Troubleshooting

### Common Issues

**Bot not responding** - Check TELEGRAM\_BOT\_TOKEN is set correctly - Verify bot health at /health endpoint - Check Render logs for errors

**Database connection issues** - Verify MONGODB\_URI format and credentials - Check MongoDB Atlas network access settings - Ensure database is not paused (Atlas free tier)

**Scraping not working** - Check scraper health at /api/scraper/health - Verify no rate limiting from retailers - Check Render logs for scraping errors

**No alerts being sent** - Verify tracking is set up correctly with /status - Check monitoring cycle logs - Test with /api/monitor/force

### Performance Optimization

**For Render.com Free Tier:** - Monitoring cycles run every 15 minutes (optimized for free tier) - Limited to 3 concurrent scrapers - Database queries are optimized with proper indexing - Alert cooldown prevents spam

**For Paid Plans:** - Reduce monitoring interval to 5-10 minutes - Increase concurrent scraper limit - Add more retailer scrapers - Implement webhook notifications

## 🔒 Security

* Environment variables for sensitive data
* Input validation for all user inputs
* Rate limiting on API endpoints
* Secure webhook endpoints for Telegram and Stripe

## 📈 Scaling

The bot is designed to scale:

### Horizontal Scaling

* Stateless design allows multiple instances
* Database handles concurrent access
* Background monitoring can be distributed

### Vertical Scaling

* Add more scrapers for additional retailers
* Increase monitoring frequency
* Add real-time WebSocket alerts

## 🆘 Support

For issues: 1. Check the [troubleshooting section](#troubleshooting) 2. Review Render logs for specific errors 3. Test individual components using API endpoints 4. Check scraper health and database connectivity

## 📜 License

This project is for educational purposes. Please respect retailer websites and follow their terms of service when scraping.

**Happy sneaker hunting! 👟🔥**