Telegram Bot Deployment Guide

This guide walks you through setting up the Telegram bot for email and phone validation with cryptocurrency payment processing.

Prerequisites

- Python 3.11+
- PostgreSQL database
- Replit account (for hosting)
- Telegram account
- BlockBee account (for cryptocurrency payments)

Table of Contents

1. Creating a Telegram Bot
2. Setting Up BlockBee
3. Database Configuration
4. Environment Variables
5. Deployment on Replit
6. Testing the Setup
7. Troubleshooting

Creating a Telegram Bot

Step 1: Contact BotFather

1. Open Telegram and search for `@BotFather`
2. Start a conversation with BotFather
3. Send `/start` to begin

Step 2: Create New Bot

1. Send `/newbot` to BotFather
2. Choose a name for your bot (e.g., "Email Phone Validator Bot")
3. Choose a username for your bot (must end with 'bot', e.g., "emailphonevalidator_bot")
4. BotFather will provide you with a bot token

Important: Save this token securely - you'll need it for the `TELEGRAM_BOT_TOKEN` environment variable.

Step 3: Configure Bot Settings (Optional)

```
/setdescription - Set bot description
/setabouttext - Set about text
/setuserpic - Set bot profile picture
/setcommands - Set bot commands menu
```

Recommended commands to set:

```
start - Start using the bot
subscribe - View subscription options
status - Check your subscription status
help - Get help and support
```

Step 4: Get Your Admin Chat ID

1. Start your bot and send `/start`
2. Go to `https://api.telegram.org/bot<YOUR_BOT_TOKEN>/getUpdates`
3. Look for your chat ID in the response
4. Use this ID for the `ADMIN_CHAT_ID` environment variable

Setting Up BlockBee

Step 1: Create BlockBee Account

1. Visit BlockBee.io
2. Sign up for an account

3. Verify your email address

Step 2: Get API Key

1. Log into your BlockBee dashboard
2. Navigate to "API Keys" section
3. Generate a new API key
4. Copy the API key for the `BLOCKBEE_API_KEY` environment variable

Step 3: Configure Supported Cryptocurrencies

The bot supports these cryptocurrencies by default: - Bitcoin (BTC) - Ethereum (ETH) - USDT (Tether) - Litecoin (LTC) - Bitcoin Cash (BCH)

Ensure these are enabled in your BlockBee account.

Database Configuration

PostgreSQL Setup

1. **On Replit**: PostgreSQL is automatically provisioned
2. **Local Development**: Install PostgreSQL locally

The database URL will be automatically available in the `DATABASE_URL` environment variable on Replit.

Database Schema

The application automatically creates the following tables: - `users` - User information and Telegram chat IDs - `subscriptions` - Payment and subscription tracking - `validation_history` - Record of validation attempts

Environment Variables

Create these environment variables in your Replit project:

Required Variables

```
# Telegram Bot Configuration
TELEGRAM_BOT_TOKEN=your_bot_token_from_botfather

# Admin Configuration
ADMIN_CHAT_ID=your_telegram_chat_id

# BlockBee Payment Processing
BLOCKBEE_API_KEY=your_blockbee_api_key

# Database (automatically set on Replit)
DATABASE_URL=postgresql://username:password@host:port/database

# Application Configuration
WEBHOOK_URL=https://your-replit-app.replit.app
```

Optional Variables

```
# Rate Limiting (default values shown)
RATE_LIMIT_MESSAGES=10  # Messages per minute
RATE_LIMIT_WINDOW=60    # Time window in seconds

# Subscription Pricing (default values shown)
MONTHLY_PRICE_USD=10.00
SUBSCRIPTION_DAYS=30

# Validation Limits
FREE_VALIDATIONS_PER_DAY=5
```

Deployment on Replit

Step 1: Fork or Create Project

1. Fork this repository to your Replit account, or
2. Create a new Python Repl and upload the project files

Step 2: Install Dependencies

Dependencies are automatically managed through `pyproject.toml`. The following packages are required:

```
[project]
dependencies = [
    "python-telegram-bot>=21.0",
    "sqlalchemy>=2.0",
    "psycopg2-binary",
    "pandas",
    "openpyxl",
    "phonenumbers",
    "flask",
    "requests",
    "asyncio"
]
```

Step 3: Configure Environment Variables

1. Go to your Repl's "Secrets" tab
2. Add all required environment variables listed above
3. Ensure `WEBHOOK_URL` matches your Repl's domain

Step 4: Set Up Workflows

The project includes these workflows:

1. **Bot Server** (`python main.py`)
2. Runs the main Telegram bot
3. Handles user interactions and validation
4. **Payment API Server** (`python payment_api.py`)
5. Handles BlockBee webhook callbacks
6. Processes payment confirmations

Step 5: Configure Webhook

The webhook URL should be:
`https://your-repl-name.your-username.repl.co/webhook`

BlockBee will automatically use this URL when processing payments.

Testing the Setup

Step 1: Basic Bot Test

1. Start both workflows (Bot Server and Payment API Server)
2. Open Telegram and find your bot
3. Send `/start` - you should receive a welcome message
4. Try `/help` to see available commands

Step 2: Validation Test

1. Send a document with email addresses or phone numbers
2. The bot should process and validate the data
3. Check that validation limits are enforced for free users

Step 3: Payment Test

1. Try to subscribe using `/subscribe`
2. Select a cryptocurrency
3. Use BlockBee's testnet or send a small amount
4. Verify the webhook processes the payment correctly

Step 4: Webhook Test

Monitor the Payment API Server logs to ensure: - Webhook URLs are being called - Payment confirmations are processed - User subscriptions are activated

File Structure

```
├── main.py                    # Main bot application
├── payment_api.py             # Payment webhook handler
├── config.py                  # Configuration settings
├── database.py                # Database connection and setup
├── models.py                  # SQLAlchemy models
├── handlers/                  # Bot command handlers
│   ├── start.py               # Start and help commands
│   ├── subscription.py        # Payment and subscription
│   ├── validation.py          # File validation logic
│   ├── admin.py               # Admin commands
```

```
│   └── dashboard.py         # User dashboard
├── services/
│   └── blockbee_service.py  # BlockBee API integration
├── email_validator.py       # Email validation logic
├── phone_validator.py       # Phone validation logic
├── file_processor.py        # Document processing
├── subscription_manager.py  # Subscription handling
├── rate_limiter.py          # Rate limiting
└── keyboards.py             # Telegram inline keyboards
```

Monitoring and Logs

Application Logs

Monitor these logs for issues: - Bot Server workflow logs (user interactions) - Payment API Server logs (payment processing) - Database connection errors - Rate limiting violations

Key Metrics to Monitor

- Daily active users
- Validation requests per day
- Payment success rate
- Error rates and types
- Database performance

Security Considerations

Bot Token Security

- Never commit bot tokens to version control
- Use Replit Secrets for all sensitive data
- Rotate tokens if compromised

Webhook Security

- Validate webhook signatures (BlockBee provides this)
- Use HTTPS only for webhook URLs

- Implement rate limiting on webhook endpoints

Database Security

- Use strong database passwords
- Regularly backup user data
- Implement proper access controls

Troubleshooting

Common Issues

Bot not responding: - Check TELEGRAM_BOT_TOKEN is correct - Verify Bot Server workflow is running - Check network connectivity

Payments not processing: - Verify BLOCKBEE_API_KEY is valid - Check Payment API Server is running on correct port - Confirm webhook URL is accessible

Database errors: - Check DATABASE_URL format - Verify PostgreSQL service is running - Review connection pool settings

Validation errors: - Check file format support - Verify email/phone validation logic - Review rate limiting settings

Debug Mode

Enable debug logging by setting:

```
logging.basicConfig(level=logging.DEBUG)
```

Health Checks

The application provides these health check endpoints: - `GET /` - Basic API health check - `GET /health` - Detailed system status

Support

For technical support: 1. Check the logs in both workflows 2. Review this documentation 3. Check BlockBee API documentation 4. Review Telegram Bot API documentation

Production Checklist

Before going live: - [ ] All environment variables configured - [ ] Bot commands properly set with BotFather - [ ] Payment webhooks tested - [ ] Database backups configured - [ ] Monitoring and alerting set up - [ ] Rate limits properly configured - [ ] Admin functions tested - [ ] Error handling verified

---

Last Updated: August 2025 **Version**: 1.0