

Deployment Guide

Quick Start

1. Environment Setup

```
# Clone or navigate to the project
cd trading-bot

# Create virtual environment
python -m venv venv
source venv/bin/activate # On Windows: venv\Scripts\activate

# Install dependencies
pip install -r requirements.txt
```

2. Configuration

API Keys Setup

Create a `.env` file in the project root:

```
# Exchange API Configuration
EXCHANGE_API_KEY=your_api_key_here
EXCHANGE_SECRET_KEY=your_secret_key_here
EXCHANGE_PASSPHRASE=your_passphrase_here # If required by exchange

# Environment
ENVIRONMENT=production # or 'development'
LOG_LEVEL=INFO
```

Strategy Configuration

Edit `config/strategy.yaml`:

```
# Key settings to customize:
exchange:
  testnet: false # Set to false for live trading

trading:
  initial_capital: 1000.0 # Your starting capital
  symbols: ["BTCUSDT", "ETHUSDT"] # Symbols to trade

strategy:
  confidence:
    min_threshold: 60 # Minimum confidence to trade (50-80)
```

3. Testing

```
# Run integration tests
python test_integration.py

# Run unit tests
python -m pytest tests/ -v

# Test configuration
python -c "from src.config import load_config; print('Config OK')"
```

4. Deployment Options

Option A: Direct Python Execution

```
# Start the trading bot
python src/main.py

# In another terminal, start the dashboard
streamlit run dashboard/app.py
```

Option B: Docker Deployment (if Docker available)

```
# Build image
docker build -t crypto-trading-bot .

# Run with docker-compose
docker-compose up -d
```

Option C: Background Service

```
# Using screen/tmux
screen -S trading-bot
source venv/bin/activate
python src/main.py

# Detach with Ctrl+A, D
# Reattach with: screen -r trading-bot
```

5. Monitoring

Dashboard Access

- Local: `http://localhost:8501`
- Remote: `http://your-server-ip:8501`

Log Files

- Main log: `logs/trading.log`
- Database: `data/trading.db`

Key Metrics to Monitor

- Account balance and P&L
- Active positions
- Win rate and profit factor
- Drawdown levels

- Crisis mode status

Production Checklist

Security

- [] API keys stored securely (environment variables)
- [] Database file permissions restricted
- [] Log files rotation configured
- [] Firewall rules for dashboard port

Risk Management

- [] Position sizing validated
- [] Stop losses configured
- [] Crisis mode thresholds set
- [] Daily loss limits defined

Monitoring

- [] Dashboard accessible
- [] Log monitoring setup
- [] Alert system configured
- [] Backup strategy in place

Performance

- [] Sufficient system resources
- [] Network connectivity stable
- [] Exchange API limits understood
- [] Latency requirements met

Troubleshooting

Common Issues

1. Configuration Errors

```
# Validate configuration
python -c "from src.config import load_config; load_config()"
```

2. Database Issues

```
# Check database
python -c "
import sqlite3
conn = sqlite3.connect('data/trading.db')
print('Tables:', [row[0] for row in conn.execute('SELECT name FROM sqlite_master WHERE
type="table"')])
conn.close()
"
```

3. Exchange Connection

```
# Test exchange connection
python -c "
import asyncio
from src.exchange import ExchangeClient
from src.config import load_config

async def test():
    config = load_config()
    client = ExchangeClient(config.exchange)
    await client.initialize()
    print('Exchange connection OK')
    await client.close()

asyncio.run(test())
"
```

4. Memory/Performance Issues

- Monitor system resources
- Reduce data lookback periods
- Limit number of symbols
- Increase update intervals

Log Analysis

```
# View recent logs
tail -f logs/trading.log

# Search for errors
grep -i error logs/trading.log

# Monitor specific symbol
grep BTCUSDT logs/trading.log
```

Scaling and Optimization

Performance Tuning

1. Data Management

- Optimize lookback periods
- Use data compression
- Implement data cleanup

2. Strategy Optimization

- Backtest parameter combinations
- Monitor performance metrics
- Adjust confidence thresholds

3. System Resources

- Monitor CPU/memory usage
- Optimize database queries
- Use connection pooling

Multi-Symbol Trading

1. Resource Allocation

- Balance symbols by volatility
- Adjust position sizes
- Monitor correlation

2. Risk Distribution

- Sector exposure limits
- Currency pair correlation
- Market cap weighting

Maintenance

Daily Tasks

- ☐ Check system health
- ☐ Review trading performance
- ☐ Monitor log files
- ☐ Verify API connectivity

Weekly Tasks

- ☐ Analyze strategy performance
- ☐ Review risk metrics
- ☐ Update configuration if needed
- ☐ Backup database

Monthly Tasks

- ☐ Performance review
- ☐ Strategy optimization
- ☐ System updates
- ☐ Security audit


Support and Updates

Getting Help

1. Check logs for error messages
2. Review configuration settings
3. Test individual components
4. Consult documentation

Updates

1. Test in development environment
 2. Backup current configuration
 3. Update dependencies carefully
 4. Monitor after deployment
-

 **Risk Warning:** This trading bot is for educational and research purposes. Cryptocurrency trading involves substantial risk of loss. Never trade with money you cannot afford to lose. Always test thoroughly before live trading.