🚀 Production Deployment Guide

Overview

This guide covers deploying the Automated Focused Data Collector in production with proper automation, monitoring, and reliability features.

📋 Prerequisites

1. System Requirements

- **OS**: Linux (Ubuntu 20.04+ recommended)
- **Python**: 3.8+
- **Memory**: 2GB+ RAM
- **Storage**: 10GB+ free space
- **Network**: Stable internet connection

2. Python Dependencies

```
pip install schedule pandas numpy pytz sqlite3
```

3. Alpaca API Access

- Valid Alpaca API credentials
- API key and secret configured
- Sufficient rate limits for your subscription

🔧 Installation & Setup

1. Clone and Configure

```
# Clone your project
git clone <your-repo>
cd finm-25000-project-alpaca

# Set up virtual environment
python3 -m venv alpaca_venv
source alpaca_venv/bin/activate

# Install dependencies
pip install -r requirements.txt

# Configure Alpaca credentials
cp Alpaca_API_template.py Alpaca_API.py
# Edit Alpaca_API.py with your actual credentials
```

2. Configuration File

The system will automatically create `collector_config.json` with defaults. Customize as needed:

```
{
  "collection": {
    "max_retries": 3,
    "retry_delay": 1,
    "batch_size": 5,
    "years_back": 7,
    "rate_limit_delay": 1,
    "batch_delay": 5
  },
  "scheduling": {
    "daily_update_time": "16:30",
    "weekly_full_collection": "Sunday 18:00",
    "check_interval_minutes": 15
  },
  "monitoring": {
    "enable_email_alerts": true,
    "smtp_server": "smtp.gmail.com",
    "smtp_port": 587,
    "email_from": "your-email@gmail.com",
    "email_to": "alerts@yourdomain.com",
    "email_password": "your-app-password"
  },
  "data_quality": {
    "min_records_per_symbol": 1500,
    "max_data_age_days": 2,
    "enable_validation": true
  }
}
```

🚀 Deployment Options

Option 1: Systemd Service (Recommended)

1. Create Service File

```
sudo nano /etc/systemd/system/alpaca-data-collector.service
```

```
[Unit]
Description=Alpaca Data Collector Service
After=network.target
Wants=network.target

[Service]
```

```
Type=simple
User=your-username
Group=your-username
WorkingDirectory=/path/to/finm-25000-project-alpaca/Step 4: Getting Market
Data from Alpaca
Environment=PATH=/path/to/finm-25000-project-alpaca/alpaca_venv/bin
ExecStart=/path/to/finm-25000-project-alpaca/alpaca_venv/bin/python
automated_focused_collector.py
Restart=always
RestartSec=10
StandardOutput=journal
StandardError=journal

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

2. Enable and Start Service

```
# Reload systemd
sudo systemctl daemon-reload

# Enable service (start on boot)
sudo systemctl enable alpaca-data-collector.service

# Start service
sudo systemctl start alpaca-data-collector.service

# Check status
sudo systemctl status alpaca-data-collector.service

# View logs
sudo journalctl -u alpaca-data-collector.service -f
```

Option 2: Cron Jobs

1. Daily Updates

```
# Edit crontab
crontab -e

# Add daily update at 4:30 PM (after market close)
30 16 * * 1-5 cd /path/to/finm-25000-project-alpaca/Step\ 4:\ Getting\ Market\
Data\ from\ Alpaca && /path/to/alpaca_venv/bin/python
automated_focused_collector.py --incremental
```

2. Weekly Full Collection

```
# Add weekly full collection on Sunday at 6:00 PM
0 18 * * 0 cd /path/to/finm-25000-project-alpaca/Step\ 4:\ Getting\ Market\
Data\ from\ Alpaca && /path/to/alpaca_venv/bin/python
automated_focused_collector.py --full
```

3. Data Quality Checks

```
# Add daily quality check at 9:00 AM
0 9 * * 1-5 cd /path/to/finm-25000-project-alpaca/Step\ 4:\ Getting\ Market\
Data\ from\ Alpaca && /path/to/alpaca_venv/bin/python
automated_focused_collector.py --quality
```

Option 3: Docker Container

1. Create Dockerfile

```
FROM python:3.9-slim

WORKDIR /app

# Install system dependencies
RUN apt-get update && apt-get install -y \
    cron \
    && rm -rf /var/lib/apt/lists/*

# Copy requirements and install Python dependencies
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

# Copy application code
COPY . .

# Create cron job
RUN echo "30 16 * * 1-5 cd /app/Step\ 4:\ Getting\ Market\ Data\ from\ Alpaca
&& python automated_focused_collector.py --incremental" > /etc/cron.d/alpaca-
collector
RUN echo "0 18 * * 0 cd /app/Step\ 4:\ Getting\ Market\ Data\ from\ Alpaca &&
python automated_focused_collector.py --full" >> /etc/cron.d/alpaca-collector
RUN chmod 0644 /etc/cron.d/alpaca-collector
RUN crontab /etc/cron.d/alpaca-collector

# Start cron and run application
CMD ["cron", "-f"]
```

2. Build and Run

```
# Build image
docker build -t alpaca-data-collector .

# Run container
docker run -d \
  --name alpaca-collector \
  -v /path/to/data:/app/data \
  -e ALPACA_KEY=your_key \
  -e ALPACA_SECRET=your_secret \
  alpaca-data-collector
```

📊 Monitoring & Maintenance

1. Log Monitoring

```
# View real-time logs
tail -f automated_collection.log

# View systemd logs
sudo journalctl -u alpaca-data-collector.service -f

# Search for errors
grep -i error automated_collection.log
```

2. Database Monitoring

```
# Check database size
du -h "../Step 5: Saving Market Data/market_data.db"

# Check data quality
sqlite3 "../Step 5: Saving Market Data/market_data.db" "SELECT COUNT(*) as
total_records FROM market_data;"
sqlite3 "../Step 5: Saving Market Data/market_data.db" "SELECT COUNT(DISTINCT
symbol) as total_symbols FROM market_data;"
```

3. Performance Monitoring

```
# Check system resources
htop
iotop
df -h

# Check process status
ps aux | grep automated_focused_collector
```

4. Health Checks

```
# Run manual health check
python automated_focused_collector.py --health

# Check data quality
python automated_focused_collector.py --quality

# View system status
python automated_focused_collector.py --status
```

🔄 Backup & Recovery

1. Database Backups

```
# Create backup script
cat > backup_database.sh << 'EOF'
#!/bin/bash
BACKUP_DIR="/path/to/backups"
DATE=$(date +%Y%m%d_%H%M%S)
DB_PATH="../Step 5: Saving Market Data/market_data.db"

mkdir -p $BACKUP_DIR
cp "$DB_PATH" "$BACKUP_DIR/market_data_$DATE.db"

# Keep only last 7 backups
ls -t $BACKUP_DIR/market_data_*.db | tail -n +8 | xargs rm -f

echo "Backup completed: market_data_$DATE.db"
EOF

chmod +x backup_database.sh

# Add to crontab for daily backups
0 2 * * * /path/to/backup_database.sh
```

2. Configuration Backups

```
# Backup configuration
cp collector_config.json collector_config.json.backup

# Backup logs
tar -czf logs_backup_$(date +%Y%m%d).tar.gz *.log
```

🚨 Troubleshooting

Common Issues

1. Service Won't Start

```
# Check service status
sudo systemctl status alpaca-data-collector.service

# Check logs
sudo journalctl -u alpaca-data-collector.service -n 50

# Check file permissions
ls -la /path/to/finm-25000-project-alpaca/Step\ 4:\ Getting\ Market\ Data\
from\ Alpaca/
```

2. Data Collection Failures

```
# Check Alpaca API status
curl https://status.alpaca.markets/

# Check rate limits
python automated_focused_collector.py --test-api

# Check network connectivity
ping api.alpaca.markets
```

3. Database Issues

```
# Check database integrity
sqlite3 "../Step 5: Saving Market Data/market_data.db" "PRAGMA
integrity_check;"

# Rebuild database if needed
sqlite3 "../Step 5: Saving Market Data/market_data.db" "VACUUM;"
sqlite3 "../Step 5: Saving Market Data/market_data.db" "REINDEX;"
```

Performance Optimization

1. Database Optimization

```
# Regular maintenance
sqlite3 "../Step 5: Saving Market Data/market_data.db" "ANALYZE;"
sqlite3 "../Step 5: Saving Market Data/market_data.db" "VACUUM;"
```

2. Memory Management

```
# Monitor memory usage
watch -n 1 'ps aux | grep automated_focused_collector'

# Set memory limits in systemd service
[Service]
MemoryMax=1G
```

📈 Scaling Considerations

1. Multiple Instances

- Run separate collectors for different asset categories
- Use load balancing for API requests
- Implement shared database with proper locking

2. High Availability

- Deploy on multiple servers
- Use database replication
- Implement health checks and auto-restart

3. Cloud Deployment

- AWS EC2 with auto-scaling
- Google Cloud Run for serverless
- Azure Container Instances

🔐 Security Best Practices

1. Credential Management

```
# Use environment variables
export ALPACA_KEY="your_key"
export ALPACA_SECRET="your_secret"

# Or use secrets management
sudo apt install pass
echo "your_secret" | pass insert alpaca/secret
```

2. File Permissions

```
# Restrict access to sensitive files
chmod 600 Alpaca_API.py
chmod 600 collector_config.json
chmod 700 ../Step\ 5:\ Saving\ Market\ Data/
```

3. Network Security

```
# Firewall rules
sudo ufw allow from 192.168.1.0/24 to any port 22
sudo ufw enable
```

📞 Support & Maintenance

1. Regular Maintenance Schedule

- **Daily**: Check logs and data quality
- **Weekly**: Full data collection and validation
- **Monthly**: Performance review and optimization
- **Quarterly**: Security audit and updates

2. Monitoring Alerts

- Set up email alerts for failures
- Monitor disk space usage
- Track API rate limit usage
- Alert on data quality issues

3. Update Procedures

```
# Update application
git pull origin main
pip install -r requirements.txt

# Restart service
sudo systemctl restart alpaca-data-collector.service

# Verify update
python automated_focused_collector.py --version
```

This production deployment guide ensures your automated data collector runs reliably, efficiently, and securely in a production environment.