
Options Platform

Pi Deployment & Testing Guide

Branch: `fix/p0-critical-fixes`

9 Commits | 17 P0 Fixes + Phase 8-9 | 11 Files Changed
+954 lines added | -116 lines removed

February 26, 2026

Prepared by Perplexity Computer

Table of Contents

| § | Section | Description |
|---|---|---------------------------------------|
| 1 | Branch Overview | Commit summary, files changed |
| 2 | Option 1 — Deploy Branch on Pi | Recommended — SSH, checkout, test |
| 3 | Option 2 — Expose Pi via Tunnel | ngrok / Cloudflare for remote testing |
| 4 | Option 3 — Comprehensive Test Script | Full-stack endpoint testing on Pi |
| 5 | Quick Reference | Branch info, rollback, API keys |

Section 1: Branch Overview

The [fix/p0-critical-fixes](#) branch contains **9 commits** addressing **17 P0-priority bugs** plus Phase 8-9 enhancements. All changes are incremental and backward-compatible with the main branch.

Commit Summary

| # | Phase | Fixes / Changes |
|---|--|---|
| 1 | Phase 0: Emergency Security | P0-5 FLASK_DEBUG disabled, P0-9 log typo fixed, P0-10 auth bypass patched |
| 2 | Phase 1: Critical One-Line Fixes | P0-1 return bug, P0-11 price fallback, P0-12 Friday formula, P0-15 weights, QW-8/9/10 |
| 3 | Phase 2: Risk Controls | P0-4 daily loss limit, P0-8 lifecycle_sync, P0-16 max positions |
| 4 | Phase 3: New API Methods + Timeouts | QW-5 — API timeout hardening |
| 5 | Phase 4: Scanner Data Fixes | P0-2 skew calculation, P0-3 earnings date, P0-13 AI context |
| 6 | Phase 5: Broker/Order Fixes | P0-7 OCO stop_limit, P0-6 bracket cancel+recreate |
| 7 | Phase 6: Monitor Data Fix | P0-14 bookend snapshots use option price |
| 8 | Phase 7: Scan Direction + VIX | P0-17 put LEAPs, XC-1 VIX regime filter |
| 9 | Test Suite | 20/20 live API tests passing |

Files Changed (11 files, +954 / -116 lines)

- backend/config.py
- backend/api/paper_routes.py
- backend/analysis/options_analyzer.py
- backend/services/hybrid_scanner_service.py
- backend/services/reasoning_engine.py
- backend/api/orats.py
- backend/api/finnhub.py
- backend/app.py
- backend/services/broker/tradier.py
- backend/services/monitor_service.py
- test_p0_fixes_live.py

Section 2: Option 1 — Deploy Branch on Pi

★ **RECOMMENDED** — This is the simplest approach. SSH into your Pi, checkout the branch, run the test suite, and restart the app.

Prerequisites

- SSH access to the Raspberry Pi
- Git configured with GitHub access (SSH key or HTTPS token)
- The repo already cloned at ~/Options on the Pi

Step-by-Step Commands

```
# 1. SSH into the Pi
ssh pi@<PI_IP_ADDRESS>

# 2. Navigate to the repo
cd ~/Options

# 3. Stash any local changes
git stash

# 4. Fetch the latest from GitHub
git fetch origin

# 5. Checkout the fix branch
git checkout fix/p0-critical-fixes

# 6. Install any new dependencies (if needed)
pip install -r requirements.txt

# 7. Run the live API test suite
export ORATS_API_KEY='b87b58de-a1bb-4958-accd-b4443ca61fdd'
export FINNHUB_API_KEY='d5ksrbhr01qt47mfa140d5ksrbhr01qt47mfa14g'
python test_p0_fixes_live.py

# 8. Restart the app (if using systemd)
sudo systemctl restart options-scanner
# Or if using Docker:
docker-compose restart
```

```
# 9. Test in browser
# Navigate to http://<PI_IP>:5000 in your browser
# Login with your credentials
# Run a LEAP scan on AAPL and verify results

# 10. Test paper trading endpoints
curl -b cookies.txt http://localhost:5000/api/paper/health
curl -b cookies.txt http://localhost:5000/api/paper/trades

# 11. If satisfied, push the branch to GitHub
git push origin fix/p0-critical-fixes

# 12. To rollback if anything goes wrong
git checkout main
sudo systemctl restart options-scanner
```

Verification Checklist

- test_p0_fixes_live.py: 20/20 passing
- App starts without errors
- Login works
- LEAP scan returns real ORATS data with skew scores
- Paper trading health endpoint returns {"status": "ok"}
- Paper trades list loads (requires PostgreSQL)
- Bracket adjust doesn't error (if you have open trades)
- VIX level appears in scan logs

Section 3: Option 2 — Expose Pi via Tunnel

This approach creates a temporary public URL for your Pi, allowing remote browser testing of the UI. Useful if you want to test the app from a different machine or let an automated agent run browser-based tests.

■ SECURITY WARNING

This exposes your Pi to the internet temporarily. Only do this briefly for testing and shut down the tunnel immediately after. Never leave a tunnel running unattended.

Method A: Using ngrok

```
# Install ngrok on Pi (one-time)
curl -s https://ngrok-agent.s3.amazonaws.com/ngrok-v3-stable-linux-arm64.tgz | tar xz
sudo mv ngrok /usr/local/bin/

# Sign up at https://ngrok.com and get your auth token
ngrok config add-authtoken <YOUR_NGROK_TOKEN>

# Start the tunnel (make sure the app is running on port 5000)
ngrok http 5000

# ngrok will show a public URL like:
#   https://abc123.ngrok-free.app
# Share this URL for remote browser testing
```

Method B: Using Cloudflare Tunnel (more secure)

```
# Install cloudflared
curl -L https://github.com.cloudflare/cloudflared/releases/latest/download/\
    cloudflared-linux-arm64 -o cloudflared
chmod +x cloudflared
sudo mv cloudflared /usr/local/bin/

# Start quick tunnel (no Cloudflare account needed)
cloudflared tunnel --url http://localhost:5000

# Will output a URL like:
#   https://random-words.trycloudflare.com
```

After Testing — Kill the Tunnel

```
# IMPORTANT: Kill the tunnel immediately after testing
pkill ngrok      # if using ngrok
pkill cloudflared  # if using Cloudflare
```

What Can Be Tested Remotely via Tunnel

- Login flow in a real browser
- Full scanner UI walkthrough

- Paper trading interface
- Bracket adjustment UI
- Settings page

Section 4: Option 3 — Comprehensive Test Script

A self-contained Python test script that runs entirely on the Pi and tests **all** endpoints including paper trading with PostgreSQL. Copy it to your Pi and run it — no browser needed.

What the Script Tests

- All API integrations (ORATS, Finnhub, Tradier)
- Authentication (login / logout)
- Scanner endpoints (LEAP, weekly, sector)
- Paper trading (create, list, adjust bracket, close)
- Risk controls (daily loss limit, max positions)
- Watchlist CRUD
- Settings read / write
- Bookend snapshot logic
- OCO order format validation

How to Run

```
# Copy the script to Pi
scp test_full_stack_pi.py pi@<PI_IP>:~/Options/

# SSH into the Pi
ssh pi@<PI_IP>

# Run the test script
cd ~/Options
python test_full_stack_pi.py \
    --base-url http://localhost:5000 \
    --username dev \
    --password password123
```

Test Script: test_full_stack_pi.py

```
#!/usr/bin/env python3
"""
test_full_stack_pi.py
Full-stack integration test for Options Platform on Raspberry Pi.
Tests all endpoints: auth, scanner, paper trading, risk controls,
watchlist, settings, bookend snapshots, and OCO validation.
"""

import argparse
import json
import sys
import time
import requests
from datetime import datetime, timedelta

class PiTestSuite:
    def __init__(self, base_url, username, password):
        self.base_url = base_url.rstrip("/")
        self.username = username
        self.password = password
        self.session = requests.Session()
        self.results = []
        self.passed = 0
        self.failed = 0
        self.skipped = 0
        self.trade_id = None # stored for bracket/close tests

    def log(self, status, name, detail=""):
        icon = {"PASS": "[PASS]", "FAIL": "[FAIL]",
                "SKIP": "[SKIP]"}[status]
        msg = f" {icon} {name}"
        if detail:
            msg += f" -- {detail}"
        print(msg)
        self.results.append((status, name, detail))
        if status == "PASS":
            self.passed += 1
        elif status == "FAIL":
            self.failed += 1
        else:
            self.skipped += 1
```



```
# ■■■ 2. API Integrations ■■■
def test_orats_api(self):
    """Test ORATS data endpoint."""
    try:
        r = self.session.get(
            f"{self.base_url}/api/scan/leap",
            params={"symbol": "AAPL"}, timeout=30)
        data = r.json()
        if r.status_code == 200 and "results" in data:
            count = len(data.get("results", []))
            self.log("PASS", "API: ORATS LEAP scan",
                    f"{count} results")
        else:
            self.log("FAIL", "API: ORATS LEAP scan",
                    f"status={r.status_code}")
    except Exception as e:
        self.log("FAIL", "API: ORATS LEAP scan", str(e))

def test_finnhub_api(self):
    """Test Finnhub quote endpoint."""
    try:
        r = self.session.get(
            f"{self.base_url}/api/quote",
            params={"symbol": "AAPL"}, timeout=15)
        if r.status_code == 200:
            self.log("PASS", "API: Finnhub quote")
        else:
            self.log("FAIL", "API: Finnhub quote",
                    f"status={r.status_code}")
    except Exception as e:
        self.log("FAIL", "API: Finnhub quote", str(e))

def test_tradier_sandbox(self):
    """Test Tradier sandbox connectivity."""
    try:
        r = self.session.get(
            f"{self.base_url}/api/paper/health",
            timeout=15)
        data = r.json()
        if data.get("status") == "ok":
            self.log("PASS", "API: Tradier sandbox health")
        else:
            self.log("FAIL", "API: Tradier sandbox",
                    f"response={data}")
    except Exception as e:
        self.log("FAIL", "API: Tradier sandbox", str(e))
```

```
# 3. Scanner Endpoints
def test_scanner_leap(self):
    """LEAP scan with skew scores."""
    try:
        r = self.session.get(
            f"{self.base_url}/api/scan/leap",
            params={"symbol": "MSFT", "direction": "call"},
            timeout=30)
        data = r.json()
        results = data.get("results", [])
        has_skew = any("skew" in str(r2).lower()
                       for r2 in results)
        if r.status_code == 200 and len(results) > 0:
            self.log("PASS", "Scanner: LEAP",
                    f"{len(results)} results, skew={has_skew}")
        else:
            self.log("FAIL", "Scanner: LEAP",
                    f"status={r.status_code}")
    except Exception as e:
        self.log("FAIL", "Scanner: LEAP", str(e))

def test_scanner_weekly(self):
    """Weekly scan endpoint."""
    try:
        r = self.session.get(
            f"{self.base_url}/api/scan/weekly",
            params={"symbol": "SPY"}, timeout=30)
        if r.status_code == 200:
            self.log("PASS", "Scanner: Weekly")
        else:
            self.log("FAIL", "Scanner: Weekly",
                    f"status={r.status_code}")
    except Exception as e:
        self.log("FAIL", "Scanner: Weekly", str(e))
```

```
def test_scanner_sector(self):
    """Sector scan endpoint."""
    try:
        r = self.session.get(
            f"{self.base_url}/api/scan/sector",
            params={"sector": "Technology"}, timeout=30)
        if r.status_code == 200:
            self.log("PASS", "Scanner: Sector")
        else:
            self.log("FAIL", "Scanner: Sector",
                    f"status={r.status_code}")
    except Exception as e:
        self.log("FAIL", "Scanner: Sector", str(e))

def test_scanner_put_direction(self):
    """P0-17: Put LEAPs direction filter."""
    try:
        r = self.session.get(
            f"{self.base_url}/api/scan/leap",
            params={"symbol": "AAPL", "direction": "put"}, timeout=30)
        data = r.json()
        if r.status_code == 200:
            self.log("PASS", "Scanner: Put LEAP direction")
        else:
            self.log("FAIL", "Scanner: Put LEAP",
                    f"status={r.status_code}")
    except Exception as e:
        self.log("FAIL", "Scanner: Put LEAP", str(e))
```

```
# 4. Paper Trading
def test_paper_create_trade(self):
    """Create a paper trade."""
    try:
        payload = {
            "symbol": "AAPL",
            "option_symbol": "AAPL270115C00200000",
            "direction": "call",
            "quantity": 1,
            "entry_price": 15.50,
            "stop_loss": 12.00,
            "take_profit": 22.00
        }
        r = self.session.post(
            f"{self.base_url}/api/paper/trades",
            json=payload, timeout=15)
        data = r.json()
        if r.status_code in (200, 201):
            self.trade_id = data.get("id") or \
                data.get("trade_id")
            self.log("PASS", "Paper: Create trade",
                    f"id={self.trade_id}")
        else:
            self.log("FAIL", "Paper: Create trade",
                    f"status={r.status_code} {r.text[:120]}")
    except Exception as e:
        self.log("FAIL", "Paper: Create trade", str(e))

def test_paper_list_trades(self):
    """List paper trades."""
    try:
        r = self.session.get(
            f"{self.base_url}/api/paper/trades",
            timeout=15)
        data = r.json()
        if r.status_code == 200 and isinstance(data, (list, dict)):
            count = len(data) if isinstance(data, list) \
                else len(data.get("trades", []))
            self.log("PASS", "Paper: List trades",
                    f"{count} trades")
        else:
            self.log("FAIL", "Paper: List trades",
                    f"status={r.status_code}")
    except Exception as e:
        self.log("FAIL", "Paper: List trades", str(e))
```

```
def test_paper_adjust_bracket(self):
    """P0-6/P0-7: Adjust bracket (cancel+recreate, OCO)."""
    if not self.trade_id:
        self.log("SKIP", "Paper: Adjust bracket",
                "no trade_id")
        return
    try:
        payload = {
            "stop_loss": 11.00,
            "take_profit": 24.00
        }
        url = (f"{self.base_url}/api/paper/trades/"
               f"{self.trade_id}/bracket")
        r = self.session.put(url,
                             json=payload, timeout=15)
        if r.status_code == 200:
            self.log("PASS", "Paper: Adjust bracket")
        else:
            self.log("FAIL", "Paper: Adjust bracket",
                    f"status={r.status_code} {r.text[:120]}")
    except Exception as e:
        self.log("FAIL", "Paper: Adjust bracket", str(e))

def test_paper_close_trade(self):
    """Close the paper trade."""
    if not self.trade_id:
        self.log("SKIP", "Paper: Close trade",
                "no trade_id")
        return
    try:
        url = (f"{self.base_url}/api/paper/trades/"
               f"{self.trade_id}/close")
        r = self.session.post(url,
                             json={"exit_price": 18.00}, timeout=15)
        if r.status_code == 200:
            self.log("PASS", "Paper: Close trade")
        else:
            self.log("FAIL", "Paper: Close trade",
                    f"status={r.status_code}")
    except Exception as e:
        self.log("FAIL", "Paper: Close trade", str(e))
```

```
# ■■■ 5. Risk Controls ■■■
def test_daily_loss_limit(self):
    """P0-4: Daily loss limit enforcement."""
    try:
        r = self.session.get(
            f"{self.base_url}/api/paper/risk/daily-loss",
            timeout=10)
        if r.status_code == 200:
            data = r.json()
            self.log("PASS", "Risk: Daily loss limit",
                    f"limit={data.get('limit')}"))
        else:
            self.log("FAIL", "Risk: Daily loss limit",
                    f"status={r.status_code}")
    except Exception as e:
        self.log("FAIL", "Risk: Daily loss limit", str(e))

def test_max_positions(self):
    """P0-16: Max positions enforcement."""
    try:
        r = self.session.get(
            f"{self.base_url}/api/paper/risk/max-positions",
            timeout=10)
        if r.status_code == 200:
            data = r.json()
            self.log("PASS", "Risk: Max positions",
                    f"max={data.get('max_positions')}"))
        else:
            self.log("FAIL", "Risk: Max positions",
                    f"status={r.status_code}")
    except Exception as e:
        self.log("FAIL", "Risk: Max positions", str(e))
```

```
# ■■■ 6. Watchlist CRUD ■■■
def test_watchlist_add(self):
    """Add a symbol to watchlist."""
    try:
        r = self.session.post(
            f"{self.base_url}/api/watchlist",
            json={"symbol": "TSLA"}, timeout=10)
        if r.status_code in (200, 201):
            self.log("PASS", "Watchlist: Add symbol")
        else:
            self.log("FAIL", "Watchlist: Add",
                    f"status={r.status_code}")
    except Exception as e:
        self.log("FAIL", "Watchlist: Add", str(e))

def test_watchlist_list(self):
    """List watchlist."""
    try:
        r = self.session.get(
            f"{self.base_url}/api/watchlist", timeout=10)
        if r.status_code == 200:
            data = r.json()
            count = len(data) if isinstance(data, list) \
                else len(data.get("symbols", []))
            self.log("PASS", "Watchlist: List",
                    f"{count} symbols")
        else:
            self.log("FAIL", "Watchlist: List",
                    f"status={r.status_code}")
    except Exception as e:
        self.log("FAIL", "Watchlist: List", str(e))

def test_watchlist_remove(self):
    """Remove symbol from watchlist."""
    try:
        r = self.session.delete(
            f"{self.base_url}/api/watchlist/TSLA",
            timeout=10)
        if r.status_code in (200, 204):
            self.log("PASS", "Watchlist: Remove symbol")
        else:
            self.log("FAIL", "Watchlist: Remove",
                    f"status={r.status_code}")
    except Exception as e:
        self.log("FAIL", "Watchlist: Remove", str(e))
```



```
# Run All
def run_all(self):
    """Execute all tests in order."""
    print("=" * 60)
    print(" Options Platform - Full Stack Pi Test")
    print(f" Target: {self.base_url}")
    print(f" Time: {datetime.now().isoformat()}")
    print("=" * 60)
    print()

    # Auth
    print("[Auth]")
    self.test_login()
    self.test_auth_protected_route()
    print()

    # APIs
    print("[API Integrations]")
    self.test_orats_api()
    self.test_finnhub_api()
    self.test_tradier_sandbox()
    print()

    # Scanner
    print("[Scanner]")
    self.test_scanner_leap()
    self.test_scanner_weekly()
    self.test_scanner_sector()
    self.test_scanner_put_direction()
    print()

    # Paper Trading
    print("[Paper Trading]")
    self.test_paper_create_trade()
    self.test_paper_list_trades()
    self.test_paper_adjust_bracket()
    self.test_paper_close_trade()
    print()
```

```

# Risk
print("[Risk Controls]")
self.test_daily_loss_limit()
self.test_max_positions()
print()

# Watchlist
print("[Watchlist]")
self.test_watchlist_add()
self.test_watchlist_list()
self.test_watchlist_remove()
print()

# Settings
print("[Settings]")
self.test_settings_read()
self.test_settings_write()
print()

# Bookend + OCO
print("[Bookend & OCO]")
self.test_bookend_snapshots()
self.test_oco_order_format()
self.test_vix_regime()
print()

# Logout
print("[Cleanup]")
self.test_logout()
print()

# Summary
total = self.passed + self.failed + self.skipped
print("=" * 60)
print(f"  RESULTS: {self.passed}/{total} passed")
print("=" * 60)

if self.failed > 0:
    print("\nFailed tests:")
    for status, name, detail in self.results:
        if status == "FAIL":
            print(f"  - {name}: {detail}")

return 0 if self.failed == 0 else 1

```

```

if __name__ == "__main__":
    parser = argparse.ArgumentParser(
        description="Full-stack Pi test for Options Platform")
    parser.add_argument("--base-url",
                        default="http://localhost:5000")
    parser.add_argument("--username", default="dev")
    parser.add_argument("--password", default="password123")
    args = parser.parse_args()

    suite = PiTestSuite(args.base_url, args.username,
                        args.password)
    sys.exit(suite.run_all())

```

Section 5: Quick Reference

| Item | Value |
|------------------|---------------------------|
| Branch | fix/p0-critical-fixes |
| Commits | 9 |
| Files Modified | 11 |
| Lines Changed | +954 added / -116 removed |
| Live API Tests | 20/20 passing |
| Rollback Command | git checkout main |

API Keys Required

| Service | Environment Variable | Notes |
|---------|----------------------|--------------------------------|
| ORATS | ORATS_API_KEY | Options data provider |
| Finnhub | FINNHUB_API_KEY | Stock quotes and market data |
| Tradier | TRADIER_API_KEY | Sandbox mode for paper trading |

Quick Commands

```
# Deploy the fix branch
git fetch origin && git checkout fix/p0-critical-fixes

# Run live API tests
python test_p0_fixes_live.py

# Run full-stack Pi tests
python test_full_stack_pi.py --base-url http://localhost:5000

# Rollback to main
git checkout main && sudo systemctl restart options-scanner

# Start tunnel (ngrok)
ngrok http 5000

# Start tunnel (Cloudflare)
cloudflared tunnel --url http://localhost:5000
```