# 🚀 Deployment Guide for V3.1 Whale Hunter

## Overview

This guide covers deployment strategies for different phases, with special considerations for users in Qatar.

---

## 📍 Latency Considerations for Qatar

### The Problem

Qatar → Solana mainnet has ~150-200ms latency. This puts you at a disadvantage for:
- Launch sniping (don't try)
- Mempool racing (not possible)

### The Solution

**Phase 0-1 (Harvesting):** Run locally - latency doesn't matter for data collection

**Phase 2+ (Live Trading):** Deploy to VPS near Solana validators
- **Recommended:** Hetzner Frankfurt (fsn1) - ~30ms to Solana
- **Alternative:** Tokyo/Singapore for Asia-Pacific routing

---

## 🏠 Phase 0-1: Local Setup

### Requirements

- Python 3.10+
- 4GB RAM minimum
- Stable internet connection

## Setup

```
# Navigate to project
cd /home/ubuntu/whale_hunter

# Create virtual environment
python3 -m venv venv
source venv/bin/activate

# Install dependencies
pip install -r requirements.txt

# Configure environment
cp config/.env.example config/.env
nano config/.env  # Add your API keys

# Initialize
python scripts/bootstrap.py

# Start logging
python scripts/phase0_logger.py
```

## Daily Operations

```
# Activate environment
cd /home/ubuntu/whale_hunter
source venv/bin/activate

# Morning: Harvest yesterday's winners
python scripts/harvester.py --tokens <TOKEN_1> <TOKEN_2>

# Check Black Book
python scripts/god_view.py

# Run decay (weekly)
python -c "from src.database import WalletGraphDB; db = WalletGraphDB();
print(db.apply_confidence_decay())"
```

## Cron Jobs for Automation

```
# Edit crontab
crontab -e

# Add these lines:

# Daily: Apply confidence decay at 00:00 UTC
0 0 * * * cd /home/ubuntu/whale_hunter && /home/ubuntu/whale_hunter/venv/bin/python -
c "from src.database import WalletGraphDB; db = WalletGraphDB();
db.apply_confidence_decay(); db.apply_edge_decay()" >> logs/cron.log 2>&1

# Weekly: Backup database (Sunday 01:00 UTC)
0 1 * * 0 cp /home/ubuntu/whale_hunter/data/wallet_graph.db /home/ubuntu/whale_hunter/
data/backups/wallet_graph_$(date +\%Y\%m\%d).db
```

---

# 🌐 Phase 2+: VPS Deployment

## Recommended: Hetzner Cloud (Frankfurt)

**Why Hetzner Frankfurt?**
- ~30ms to Solana mainnet validators
- €4.51/month for CX21 (2 vCPU, 4GB RAM)
- Good for Qatar users (routes through Europe)

## Step 1: Create VPS

1. Sign up at hetzner.com/cloud (https://www.hetzner.com/cloud)
2. Create new project “whale-hunter”
3. Add server:
   - Location: **Falkenstein (fsn1)** or **Nuremberg (nbg1)**
   - Image: Ubuntu 22.04
   - Type: CX21 (4GB RAM) or CX31 (8GB RAM)
   - Add SSH key

## Step 2: Initial Server Setup

```
# SSH to your server
ssh root@YOUR_SERVER_IP

# Create non-root user
adduser whaleuser
usermod -aG sudo whaleuser

# Setup firewall
ufw allow OpenSSH
ufw enable

# Switch to user
su - whaleuser
```

## Step 3: Install Dependencies

```
# Update system
sudo apt update && sudo apt upgrade -y

# Install Python 3.11
sudo apt install software-properties-common -y
sudo add-apt-repository ppa:deadsnakes/ppa -y
sudo apt install python3.11 python3.11-venv python3.11-dev -y

# Install other tools
sudo apt install git htop tmux -y
```

### Step 4: Deploy Application

```
# Clone/upload your code
cd ~
git clone YOUR_REPO whale_hunter
# OR upload via scp/rsync

# Setup environment
cd whale_hunter
python3.11 -m venv venv
source venv/bin/activate
pip install -r requirements.txt

# Configure
cp config/.env.example config/.env
nano config/.env  # Add your API keys

# Initialize
python scripts/bootstrap.py

# Test
python scripts/god_view.py
```

### Step 5: Setup Systemd Service

```
# Create service file
sudo nano /etc/systemd/system/whale-hunter.service
```

```
[Unit]
Description=Whale Hunter Trading System
After=network.target

[Service]
Type=simple
User=whaleuser
WorkingDirectory=/home/whaleuser/whale_hunter
Environment=PATH=/home/whaleuser/whale_hunter/venv/bin
ExecStart=/home/whaleuser/whale_hunter/venv/bin/python -m src.main
Restart=always
RestartSec=10

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

```
# Enable and start
sudo systemctl daemon-reload
sudo systemctl enable whale-hunter
sudo systemctl start whale-hunter

# Check status
sudo systemctl status whale-hunter

# View logs
sudo journalctl -u whale-hunter -f
```

### Step 6: Setup Monitoring

```
# Create health check script
nano ~/whale_hunter/scripts/health_check.sh
```

```
#!/bin/bash

# Check if process is running
if ! pgrep -f "whale_hunter" > /dev/null; then
    echo "$(date): Process not running, restarting..."
    sudo systemctl restart whale-hunter
fi

# Check database size
DB_SIZE=$(du -m /home/whaleuser/whale_hunter/data/wallet_graph.db | cut -f1)
if [ "$DB_SIZE" -gt 1000 ]; then
    echo "$(date): WARNING - Database size is ${DB_SIZE}MB"
fi

# Check disk space
DISK_USE=$(df -h / | awk 'NR==2 {print $5}' | tr -d '%')
if [ "$DISK_USE" -gt 80 ]; then
    echo "$(date): WARNING - Disk usage at ${DISK_USE}%"
fi
```

```
chmod +x ~/whale_hunter/scripts/health_check.sh

# Add to crontab
crontab -e

# Run health check every 5 minutes
*/5 * * * * /home/whaleuser/whale_hunter/scripts/health_check.sh >> /home/whaleuser/
whale_hunter/logs/health.log 2>&1
```

---

## 🔒 Security Best Practices

### API Key Security

```
# Never commit .env files
echo "config/.env" >> .gitignore

# Set restrictive permissions
chmod 600 config/.env

# Consider using environment variables directly
export HELIUS_API_KEY="your_key"
```

### Firewall Rules

```
# Only allow SSH and necessary ports
sudo ufw default deny incoming
sudo ufw default allow outgoing
sudo ufw allow ssh
sudo ufw enable
```

### Wallet Security

```
# Never put main wallet private key on server
# Use dedicated trading wallet with limited funds
# Set max position limits in config

# Create trading wallet
solana-keygen new -o ~/.config/solana/trading_wallet.json

# Fund with only what you're willing to lose
```

### Backup Strategy

```
# Automated daily backups
mkdir -p ~/whale_hunter/data/backups

# Add to crontab
0 3 * * * cp ~/whale_hunter/data/wallet_graph.db ~/whale_hunter/data/backups/wal-
let_graph_$(date +\%Y\%m\%d).db

# Sync to local machine weekly
# On your local machine:
rsync -avz whaleuser@YOUR_SERVER_IP:~/whale_hunter/data/backups/ ~/
whale_hunter_backups/
```

---

## 📊 Performance Tuning

### Database Optimization

```
# Run periodically to optimize SQLite
import sqlite3
conn = sqlite3.connect('data/wallet_graph.db')
conn.execute("VACUUM")
conn.execute("ANALYZE")
conn.close()
```

### Memory Management

```
# Monitor memory usage
htop

# If running low on memory, increase swap
sudo fallocate -l 2G /swapfile
sudo chmod 600 /swapfile
sudo mkswap /swapfile
sudo swapon /swapfile
echo '/swapfile none swap sw 0 0' | sudo tee -a /etc/fstab
```

### Latency Testing

```
# Test latency to Solana RPC
ping -c 10 api.mainnet-beta.solana.com

# Test Helius RPC
curl -w "@-" -o /dev/null -s "https://mainnet.helius-rpc.com/?api-key=YOUR_KEY" <<'EOF
'
     time_namelookup:  %{time_namelookup}s\n
        time_connect:  %{time_connect}s\n
     time_appconnect:  %{time_appconnect}s\n
    time_pretransfer:  %{time_pretransfer}s\n
       time_redirect:  %{time_redirect}s\n
  time_starttransfer:  %{time_starttransfer}s\n
                     ----------\n
          time_total:  %{time_total}s\n
EOF
```

---

## 🔄 Migration from SQLite to Neo4j (Phase 3+)

When wallet count exceeds 5,000, consider migrating to Neo4j:

```
# Check if migration needed
python -c "from src.database import WalletGraphDB; db = WalletGraphDB();
print('Migrate:', db.should_migrate_to_neo4j())"

# If true, start Neo4j
docker-compose --profile neo4j up -d

# Run migration script (to be created based on your needs)
python scripts/migrate_to_neo4j.py
```

---

# 📱 Alerting (Optional)

## Discord Webhook

```
# Add to config/.env
DISCORD_WEBHOOK_URL=https://discord.com/api/webhooks/...

# In your code:
import aiohttp

async def send_discord_alert(message: str):
    async with aiohttp.ClientSession() as session:
        await session.post(
            os.getenv("DISCORD_WEBHOOK_URL"),
            json={"content": message}
        )
```

## Telegram Bot

```
# Add to config/.env
TELEGRAM_BOT_TOKEN=your_bot_token
TELEGRAM_CHAT_ID=your_chat_id

# In your code:
import aiohttp

async def send_telegram_alert(message: str):
    url = f"https://api.telegram.org/bot{os.getenv('TELEGRAM_BOT_TOKEN')}/sendMessage"
    async with aiohttp.ClientSession() as session:
        await session.post(url, json={
            "chat_id": os.getenv("TELEGRAM_CHAT_ID"),
            "text": message
        })
```

---

# 🆘 Emergency Procedures

## Kill Switch Activation

```
# Manual emergency stop
echo "EMERGENCY_STOP" > /home/whaleuser/whale_hunter/data/KILL_SWITCH

# Or via script
python -c "
from src.database import WalletGraphDB
db = WalletGraphDB()
db.set_state('risk_mode', 'EMERGENCY_STOP')
print('KILL SWITCH ACTIVATED')
"
```

### Recovery from Failure

```
# Check logs
sudo journalctl -u whale-hunter -n 100

# Restart service
sudo systemctl restart whale-hunter

# Restore from backup if needed
cp ~/whale_hunter/data/backups/wallet_graph_YYYYMMDD.db ~/whale_hunter/data/wal-
let_graph.db
```

---

## 📝 Checklist Before Going Live

- [ ] API keys configured and tested
- [ ] Database initialized with `bootstrap.py`
- [ ] Backtest shows positive results
- [ ] Risk limits set appropriately
- [ ] Kill switch tested manually
- [ ] Backup system in place
- [ ] Monitoring/alerting configured
- [ ] Capital loaded to trading wallet
- [ ] First 50 trades rules understood