```
#!/usr/bin/env python3
"""
HYDRA SENTINEL-X: ONE-CLICK DEPLOYMENT
Automated setup, dependency check, and configuration
"""

import os
import sys
import subprocess
import json
from pathlib import Path
from typing import Dict, List, Tuple

class HydraDeployer:
    """Automated deployment system"""

    def __init__(self):
        self.home = Path.home()
        self.hydra_dir = self.home / "hydra_sentinel"
        self.errors = []
        self.warnings = []

    def print_banner(self):
        print("""
╔═══════════════════════════════════════════════════════════╗
║   🐍 HYDRA SENTINEL-X DEPLOYMENT v4.5                     ║
║   Automated Setup & Configuration                         ║
╚═══════════════════════════════════════════════════════════╝
        """)

    def check_python_version(self) -> bool:
        """Verify Python 3.8+"""
        version = sys.version_info
        if version.major < 3 or (version.major == 3 and version.minor < 8):
            self.errors.append(f"Python 3.8+ required (you have {version.major}.{
            return False

        print(f"✓ Python {version.major}.{version.minor}.{version.micro}")
        return True

    def check_dependencies(self) -> Tuple[List[str], List[str]]:
        """Check which dependencies are installed"""
        required = [
            'aiohttp',
```

```
            'ccxt',
            'pandas',
            'pandas_ta',
            'numpy',
            'colorama',
            'python-dotenv'
        ]

        installed = []
        missing = []

        for package in required:
            try:
                __import__(package.replace('-', '_'))
                installed.append(package)
            except ImportError:
                missing.append(package)

        return installed, missing

    def install_dependencies(self, packages: List[str]) -> bool:
        """Install missing dependencies"""
        if not packages:
            print("✓ All dependencies installed")
            return True

        print(f"\n📦 Installing {len(packages)} packages...")
        print(f"   Packages: {', '.join(packages)}")

        try:
            cmd = [sys.executable, '-m', 'pip', 'install'] + packages

            # Add break-system-packages flag if needed (for system Python)
            try:
                subprocess.run(cmd, check=True, capture_output=True)
            except subprocess.CalledProcessError:
                cmd.append('--break-system-packages')
                subprocess.run(cmd, check=True, capture_output=True)

            print("✓ Dependencies installed successfully")
            return True

        except subprocess.CalledProcessError as e:
            self.errors.append(f"Failed to install dependencies: {e}")
            return False

    def create_directory_structure(self) -> bool:
```

```
        """Create necessary directories"""
        directories = [
            self.hydra_dir,
            self.hydra_dir / "logs",
            self.hydra_dir / "backups",
            self.hydra_dir / "data"
        ]

        print("\n📁 Creating directory structure...")

        for directory in directories:
            try:
                directory.mkdir(parents=True, exist_ok=True)
                print(f"   ✓ {directory}")
            except Exception as e:
                self.errors.append(f"Failed to create {directory}: {e}")
                return False

        return True

    def create_env_template(self) -> bool:
        """Create .env configuration template"""
        env_path = self.hydra_dir / ".env"

        if env_path.exists():
            print(f"\n⚠️  .env file already exists at {env_path}")
            return True

        template = """# HYDRA SENTINEL-X Configuration
# Copy this to .env and fill in your values

# Exchange API Keys (Coinbase)
COINBASE_API_KEY=your_api_key_here
COINBASE_API_SECRET=your_api_secret_here

# Profit Distribution Wallets
SOLANA_PROFIT_ADDR=AG7vMKGh25TUg6S6Sx8WmKwaEJvGNLUiLQwpqFMfST36
ETH_PROFIT_ADDR=0x51d045eb8a0e575d23f29683c821f0b382276bdc

# Risk Management
MAX_DRAWDOWN=0.05           # 5% maximum drawdown
RISK_PER_TRADE=0.02         # 2% risk per trade
MAX_POSITIONS=5             # Maximum concurrent positions
POSITION_SIZE_PCT=0.10      # 10% of portfolio per position

# System Configuration
EXCHANGE_ID=coinbase
```

```
SCAN_INTERVAL=30            # Seconds between market scans
PAPER_TRADING=true          # Set to 'false' for live trading

# Advanced (Optional)
LOG_LEVEL=INFO
"""

        try:
            with open(env_path, 'w') as f:
                f.write(template)
            print(f"✓ Created .env template at {env_path}")
            print(f"   ⚠️  IMPORTANT: Edit this file with your API keys before ru
            return True
        except Exception as e:
            self.errors.append(f"Failed to create .env: {e}")
            return False

    def create_launcher_script(self) -> bool:
        """Create convenient launcher script"""
        launcher_path = self.hydra_dir / "launch_hydra.sh"

        script_content = f"""#!/bin/bash
# HYDRA SENTINEL-X Launcher

cd {self.hydra_dir}

# Load environment
source .env 2>/dev/null || echo "⚠️  .env not found"

# Activate virtual environment if exists
if [ -d "venv" ]; then
    source venv/bin/activate
fi

# Run Hydra
echo "🐍 Launching HYDRA SENTINEL-X..."
python3 hydra_cognitive_v45_enhanced.py "$@"
"""

        try:
            with open(launcher_path, 'w') as f:
                f.write(script_content)

            # Make executable
            os.chmod(launcher_path, 0o755)

            print(f"✓ Created launcher script: {launcher_path}")
```

```
            return True
        except Exception as e:
            self.errors.append(f"Failed to create launcher: {e}")
            return False

    def create_quick_start_guide(self) -> bool:
        """Create Quick Start documentation"""
        guide_path = self.hydra_dir / "QUICKSTART.md"

        guide_content = f"""# 🚀 HYDRA SENTINEL-X QUICK START

## Installation Complete!

Your Hydra system is installed at: `{self.hydra_dir}`

## Next Steps

### 1. Configure API Keys (REQUIRED)

Edit your configuration file:
```bash
nano {self.hydra_dir}/.env
```

Add your Coinbase API credentials:
- `COINBASE_API_KEY=your_key_here`
- `COINBASE_API_SECRET=your_secret_here`

2. Start in Paper Trading Mode (RECOMMENDED)

```bash
cd {self.hydra_dir}
python3 hydra_cognitive_v45_enhanced.py
```

Or use the launcher:
```bash
{self.hydra_dir}/launch_hydra.sh
```

3. Monitor Performance

Watch the logs in real-time:
```bash
tail -f {self.hydra_dir}/logs/hydra_*.log
```
```

View memory and positions:
```bash
cat {self.hydra_dir}/cognitive_memory.json
cat {self.hydra_dir}/active_positions.json
```

Safety Checklist

- [ ] API keys configured in `.env`
- [ ] Paper trading mode enabled (`PAPER_TRADING=true`)
- [ ] Risk parameters reviewed and understood
- [ ] Test with small amounts before full deployment
- [ ] Backup `.env` file securely

Key Files

File	Purpose
`hydra_cognitive_v45_enhanced.py`	Main trading system
`.env`	Configuration (KEEP SECURE!)
`cognitive_memory.json`	Learning database
`active_positions.json`	Open trades
`logs/`	Operation logs

Commands

Start Hydra:
```bash
python3 hydra_cognitive_v45_enhanced.py
```

Query Memory:
```python
from hydra_cognitive_v45_enhanced import HydraCognitiveCore
core = HydraCognitiveCore()
core.mem.recall(["BTC", "CRASH"])
```

Check Performance:
```python
stats = core.positions.get_statistics()
print(stats)
```

Support

For issues or questions:

```
1. Check the logs in `logs/`
2. Review memory in `cognitive_memory.json`
3. Verify configuration in `.env`

## ⚠️ CRITICAL REMINDERS

- **NEVER** commit `.env` to version control
- **ALWAYS** start with paper trading
- **TEST** thoroughly before live trading
- **MONITOR** regularly for the first week
- **BACKUP** your configuration files

---

**Status:** Ready for deployment
**Mode:** Paper Trading (Safe Mode)
**Next Step:** Configure `.env` and launch

🐍 HYDRA SENTINEL-X v4.5 ENHANCED
"""

        try:
            with open(guide_path, 'w') as f:
                f.write(guide_content)
            print(f"✓ Created Quick Start guide: {guide_path}")
            return True
        except Exception as e:
            self.warnings.append(f"Failed to create guide: {e}")
            return False

    def run_deployment(self):
        """Execute full deployment"""
        self.print_banner()

        # Step 1: Python version check
        print("\n[1/6] Checking Python version...")
        if not self.check_python_version():
            self.print_errors()
            return False

        # Step 2: Dependencies
        print("\n[2/6] Checking dependencies...")
        installed, missing = self.check_dependencies()

        if installed:
            print(f"✓ Found {len(installed)} installed packages")
```

```
        if missing:
            print(f"⚠️  Missing {len(missing)} packages: {', '.join(missing)}")

            response = input("\nInstall missing dependencies? [Y/n]: ").strip().l
            if response != 'n':
                if not self.install_dependencies(missing):
                    self.print_errors()
                    return False

        # Step 3: Directory structure
        print("\n[3/6] Creating directories...")
        if not self.create_directory_structure():
            self.print_errors()
            return False

        # Step 4: Configuration
        print("\n[4/6] Setting up configuration...")
        self.create_env_template()

        # Step 5: Launcher
        print("\n[5/6] Creating launcher script...")
        self.create_launcher_script()

        # Step 6: Documentation
        print("\n[6/6] Generating documentation...")
        self.create_quick_start_guide()

        # Final summary
        self.print_summary()

        return len(self.errors) == 0

    def print_errors(self):
        """Print all errors"""
        if self.errors:
            print("\n❌ ERRORS:")
            for error in self.errors:
                print(f"   - {error}")

    def print_summary(self):
        """Print deployment summary"""
        print("\n" + "="*60)

        if self.errors:
            print("❌ DEPLOYMENT FAILED")
            self.print_errors()
        else:
```

```
            print("✅ DEPLOYMENT SUCCESSFUL")
            print(f"\nHydra Sentinel-X installed at: {self.hydra_dir}")
            print(f"\nNext steps:")
            print(f"1. Edit configuration:")
            print(f"   nano {self.hydra_dir}/.env")
            print(f"\n2. Launch Hydra:")
            print(f"   cd {self.hydra_dir}")
            print(f"   python3 hydra_cognitive_v45_enhanced.py")
            print(f"\n3. Read the Quick Start guide:")
            print(f"   cat {self.hydra_dir}/QUICKSTART.md")

        if self.warnings:
            print("\n⚠️  WARNINGS:")
            for warning in self.warnings:
                print(f"   - {warning}")

        print("="*60 + "\n")

if __name__ == "__main__":
    deployer = HydraDeployer()
    success = deployer.run_deployment()
    sys.exit(0 if success else 1)
```