```python
"""
幣安 5 分鐘漲幅監控（現貨 + 合約）+ Telegram 通知
====================================================
部署平台：Railway.app（免費）
手機只需安裝 Telegram 即可收到通知！

取得 Telegram Bot Token：
  Telegram 搜尋 @BotFather → /newbot → 複製 Token

取得 Chat ID：
  Telegram 搜尋 @userinfobot → 發任意訊息 → 複製 id 數字
"""

import requests
import time
import logging
import os
from datetime import datetime

# ============================================================
#  設定（Railway 上用環境變數，本地測試可直接填）
# ============================================================
TELEGRAM_BOT_TOKEN = os.environ.get("TELEGRAM_BOT_TOKEN", "YOUR_BOT_TOKEN")
TELEGRAM_CHAT_ID   = os.environ.get("TELEGRAM_CHAT_ID",   "YOUR_CHAT_ID")

ALERT_THRESHOLD  =  float(os.environ.get("ALERT_THRESHOLD", "3.0"))   # 漲幅門檻（
DROP_THRESHOLD   = -float(os.environ.get("DROP_THRESHOLD",  "3.0"))   # 跌幅門檻（
COOLDOWN_MINUTES =  int(os.environ.get("COOLDOWN_MINUTES",  "30"))    # 冷卻時間（
CHECK_INTERVAL   =  int(os.environ.get("CHECK_INTERVAL",    "60"))    # 掃描間隔（
KLINE_INTERVAL   =  os.environ.get("KLINE_INTERVAL", "5m")            # K線週期

MONITOR_SPOT     = True   # 監控現貨
MONITOR_FUTURES  = True   # 監控合約

# ============================================================
SPOT_BASE    = "https://api.binance.com"
FUTURES_BASE = "https://fapi.binance.com"

logging.basicConfig(
    level=logging.INFO,
    format="%(asctime)s %(message)s",
    datefmt="%H:%M:%S"
)
log = logging.getLogger(__name__)
```

```
alert_cooldown: dict = {}


# ──────────────────────────────────────────
#  取得交易對列表
# ──────────────────────────────────────────

def get_spot_symbols() -> list:
    resp = requests.get(f"{SPOT_BASE}/api/v3/exchangeInfo", timeout=15)
    resp.raise_for_status()
    symbols = [
        s["symbol"] for s in resp.json()["symbols"]
        if s["quoteAsset"] == "USDT" and s["status"] == "TRADING"
    ]
    log.info(f"[現貨] 共 {len(symbols)} 個 USDT 交易對")
    return symbols


def get_futures_symbols() -> list:
    resp = requests.get(f"{FUTURES_BASE}/fapi/v1/exchangeInfo", timeout=15)
    resp.raise_for_status()
    symbols = [
        s["symbol"] for s in resp.json()["symbols"]
        if s["quoteAsset"] == "USDT"
        and s["status"] == "TRADING"
        and s["contractType"] == "PERPETUAL"
    ]
    log.info(f"[合約] 共 {len(symbols)} 個 USDT 永續合約")
    return symbols


# ──────────────────────────────────────────
#  取得 K 線漲幅
# ──────────────────────────────────────────

def get_kline_change(base_url: str, path: str, symbol: str):
    try:
        resp = requests.get(
            f"{base_url}{path}",
            params={"symbol": symbol, "interval": KLINE_INTERVAL, "limit": 2},
            timeout=5
        )
        resp.raise_for_status()
        kline = resp.json()[-2]   # 已收盤的 K 線
        o = float(kline[1])
        c = float(kline[4])
        if o == 0:
            return None
        return round((c - o) / o * 100, 2)
    except Exception:
        return None


# ─────────────────────────────────────────
#  Telegram 通知
# ─────────────────────────────────────────

def send_telegram(message: str):
    try:
        resp = requests.post(
            f"https://api.telegram.org/bot{TELEGRAM_BOT_TOKEN}/sendMessage",
            json={"chat_id": TELEGRAM_CHAT_ID, "text": message, "parse_mode": "HT
            timeout=10
        )
        if resp.status_code != 200:
            log.warning(f"Telegram 發送失敗: {resp.text}")
    except Exception as e:
        log.warning(f"Telegram 錯誤: {e}")


def format_alert(market: str, symbol: str, change: float) -> str:
    icon = "🚀" if change > 0 else "🔻"
    tag  = "現貨" if market == "spot" else "合約"
    sign = "+" if change > 0 else ""
    now  = datetime.now().strftime("%H:%M:%S")
    return (
        f"{icon}【{tag}】<b>{symbol}</b>\n"
        f"📊 {KLINE_INTERVAL} 漲幅：<b>{sign}{change}%</b>\n"
        f"🕐 {now}"
    )


# ─────────────────────────────────────────
#  冷卻機制
# ─────────────────────────────────────────

def is_in_cooldown(key: str) -> bool:
    return key in alert_cooldown and (time.time() - alert_cooldown[key]) < COOLDO


# ─────────────────────────────────────────
#  掃描市場
# ─────────────────────────────────────────
```

```
def scan_market(market: str, base_url: str, path: str, symbols: list) -> int:
    triggered = []
    for symbol in symbols:
        change = get_kline_change(base_url, path, symbol)
        if change is None:
            continue
        key = f"{market}:{symbol}"
        if (change >= ALERT_THRESHOLD or change <= DROP_THRESHOLD) and not is_in_
            triggered.append((symbol, change))
            alert_cooldown[key] = time.time()
        time.sleep(0.05)

    triggered.sort(key=lambda x: abs(x[1]), reverse=True)
    for symbol, change in triggered:
        log.info(f"[{market}] {symbol} {change:+.2f}%")
        send_telegram(format_alert(market, symbol, change))

    return len(triggered)


# ──────────────────────────────────────────
#  主程式
# ──────────────────────────────────────────

def main():
    if "YOUR_" in TELEGRAM_BOT_TOKEN or "YOUR_" in TELEGRAM_CHAT_ID:
        print("❌ 請先設定 TELEGRAM_BOT_TOKEN 和 TELEGRAM_CHAT_ID！")
        return

    log.info("=" * 45)
    log.info(f"幣安漲幅監控啟動 | 週期：{KLINE_INTERVAL}")
    log.info(f"門檻：漲 +{ALERT_THRESHOLD}% / 跌 {DROP_THRESHOLD}%")
    log.info(f"冷卻：{COOLDOWN_MINUTES} 分鐘 | 掃描：每 {CHECK_INTERVAL} 秒")
    log.info("=" * 45)

    send_telegram(
        f"✅ <b>幣安漲幅監控已啟動</b>\n"
        f"📈 現貨 + 合約\n"
        f"⚡ 門檻：漲 +{ALERT_THRESHOLD}% / 跌 {abs(DROP_THRESHOLD)}%\n"
        f"⏱ K線週期：{KLINE_INTERVAL}"
    )

    # 取得交易對列表（只在啟動時取一次，每小時更新）
    spot_symbols    = get_spot_symbols()    if MONITOR_SPOT    else []
    futures_symbols = get_futures_symbols() if MONITOR_FUTURES else []
    last_refresh    = time.time()
    while True:
        # 每小時重新取得最新交易對列表
        if time.time() - last_refresh > 3600:
            spot_symbols    = get_spot_symbols()    if MONITOR_SPOT    else []
            futures_symbols = get_futures_symbols() if MONITOR_FUTURES else []
            last_refresh    = time.time()

        log.info(f"開始掃描...")

        total = 0
        if MONITOR_SPOT:
            total += scan_market("spot",    SPOT_BASE,    "/api/v3/klines",   spo
        if MONITOR_FUTURES:
            total += scan_market("futures", FUTURES_BASE, "/fapi/v1/klines",  fut

        log.info(f"掃描完成，共觸發 {total} 個通知。等待 {CHECK_INTERVAL} 秒...")
        time.sleep(CHECK_INTERVAL)


if __name__ == "__main__":
    main()
```