Stephan Paul · 1. Auflage, Juli 2026 · 3 Seiten

Vorwort

Dieses Buch behandelt systematisch, wie automatisierte Nachrichten- und Marktüberwachung mit Python aufgebaut, betrieben und abgesichert wird - von der Datenbeschaffung über die Filterung und KI-gestützte Klassifikation bis zur zuverlässigen Benachrichtigung und dem produktiven Dauerbetrieb. Als durchgehendes Referenzprojekt dient ein realer Anwendungsfall: ein Watch-System, das Finanznachrichten beobachtet und meldet, sobald ein bestimmtes Unternehmen Anstalten macht, an die Börse zu gehen.

Was das Buch besonders macht: Die Konzepte werden nicht nur benannt, sondern an einem vollständigen, lauffähigen System durchgespielt und zugleich in ihren breiteren Kontext eingeordnet. Zu jedem zentralen Entwurfsproblem - welche Architektur, welcher Speicher, welcher Klassifikationsansatz, welcher Benachrichtigungskanal, welche Automatisierung - werden Alternativen benannt und gegeneinander abgewogen, nicht nur die im Projekt gewählte Lösung beschrieben.

Was dieses Buch ist

Ein systematisch aufgebautes Fachbuch zu Architektur und Betrieb von Monitoring- und Alerting-Systemen in Python. Es behandelt Python, Web-APIs, Datenspeicherung, LLM-gestützte Klassifikation, Benachrichtigungsarchitekturen, Automatisierung, Testing und Betrieb - jeweils mit Entscheidungskriterien: warum dieser Ansatz, welche Alternativen es gibt, und wann sie sich eher lohnen. Jedes Kapitel verbindet die abstrakte Fragestellung mit ihrer konkreten Umsetzung im Referenzprojekt.

Was dieses Buch nicht ist

Es ersetzt nicht die Referenzdokumentation einzelner Bibliotheken oder eine Einführung in Datenbanktheorie im Detail - dafür wird auf die jeweilige Fachdokumentation verwiesen. Es ist aber als zusammenhängendes Nachschlagewerk zum Thema Watch-Systeme gedacht: Wer später nur ein einzelnes Kapitel braucht (etwa zur Auswahl einer Speicherlösung oder zur Automatisierung), soll es auch isoliert lesen können.

Für wen ist das Buch?

Grundlegende Computerkenntnisse werden vorausgesetzt: Dateien anlegen, Programme installieren, im Terminal Befehle eintippen. Vorerfahrung mit einer Programmiersprache ist von Vorteil, aber kein Muss - alle projektspezifischen Konzepte werden eingeführt. Das Buch richtet sich sowohl an Einsteiger, die ein vollständiges System einmal Schritt für Schritt durchbauen wollen, als auch an Praktiker, die einzelne Entwurfsentscheidungen (Architektur, Speicher, Automatisierung) nachschlagen möchten.

Wie ist das Buch aufgebaut?

Teil A (Kapitel 1-6) legt das Fundament: Werkzeuge, Linux-Grundlagen, Architektur, Konfiguration, Sicherheit sowie der erste eigene Watcher. Teil B (Kapitel 7-13) vertieft das Referenzprojekt Baustein für Baustein: Datenbeschaffung, Persistenz, LLM-Klassifikation, Filter-Pipelines, Benachrichtigungskanäle und

Automatisierung. Teil C (Kapitel 14-15) behandelt Testing sowie den Übergang vom Prototyp zum produktionsreifen, betriebenen System - Themen, die in reinen Einsteiger-Tutorials meist fehlen, für ein Fachbuch aber dazugehören. Teil D (Anhang) bündelt Glossar, Musterlösungen, Troubleshooting-Checkliste und weiterführende Ressourcen.

Jedes Kapitel folgt dem gleichen Muster: Lernziel, Einordnung, Konzept, Implementierung, Vertiefung, Vertiefungsaufgabe, Zusammenfassung. Die Vertiefungsabschnitte sind bewusst so geschrieben, dass sie über das Referenzprojekt hinausgehen und Alternativen sowie Abwägungen für den professionellen Einsatz benennen.

Die Fallstudie hinter dem Projekt

Im Sommer 2026 wollte ich in EngineAI Robotics investieren - ein chinesisches Humanoide-Roboter-Unternehmen, das kurz vor dem Börsengang stand. Das Problem: Es gab noch keine offizielle ISIN, keinen Handel an einer westlichen Börse, und die Nachrichten waren verstreut. Die Idee: ein eigener Agent, der die wichtigsten Quellen beobachtet und sofort Bescheid gibt, sobald EngineAI-Aktien tatsächlich kaufbar sind.

Der Agent ist nicht spektakulär: kein neuronales Netz, keine Magie. Aber er funktioniert, läuft seit Wochen zuverlässig, hat ein paar Stunden Entwicklung und ein paar Hundert Zeilen Code gebraucht. Das ist die Art von Werkzeug, die dir das Leben leichter macht, ohne dass du ein Data-Scientist sein musst.

Aktueller Stand (Juli 2026): EngineAI hat im Juni 2026 vertraulich einen Börsengang in Hongkong beantragt (Quelle: Bloomberg, "Humanoid Robot Manufacturer EngineAI Is Said to File for Hong Kong IPO", 12. Juni 2026) - Größe und Zeitpunkt der eigentlichen Notierung stehen aber noch nicht fest, gehandelt wird die Aktie noch nirgends. Dieses Buch ist damit kein nachträglich erzähltes Beispiel, sondern beschreibt ein Werkzeug, das im Moment, in dem du liest, noch echten Nutzen hat: Der Watcher, den du in diesem Buch baust, würde genau in den kommenden Wochen den entscheidenden Moment melden, sobald er eintritt.

Den vollständigen, lauffähigen Code aus diesem Buch gibt es zusätzlich zum Anhang auch als Open-Source-Repository: github.com/watch-alert-book/engineai-monitoring. Der Tag v1.0-buch entspricht exakt dem Stand, der in Anhang A.6 abgedruckt ist - das Repository selbst kann sich seither weiterentwickelt haben.

KAPITEL 1

Python installieren und einrichten

Werkzeuge, virtuelles Environment, erste Skripte

Lernziel

Nach diesem Kapitel hast du eine funktionierende Python-Installation, kannst Skripte ausführen, verstehst virtuelle Umgebungen (venv) und kannst Pakete mit pip installieren.

Exkurs: Windows-Nutzer - WSL2 einrichten

Dieses Buch arbeitet durchgehend mit Linux-Terminal-Befehlen, Cron und systemd. Nutzt du Windows, ist der einfachste Weg nicht, jeden Befehl zu übersetzen, sondern eine echte Linux-Umgebung direkt unter Windows laufen zu lassen: das Windows Subsystem for Linux (WSL2). Das ist kein Emulator, sondern ein vollwertiger Linux-Kernel neben Windows - jeder Befehl in diesem Buch funktioniert danach exakt so, wie er dasteht.

So richtest du es ein:

Öffne PowerShell als Administrator und führe wsl --install aus. Das installiert WSL2 mit Ubuntu als Standard-Distribution.

Starte den Rechner neu, wenn du dazu aufgefordert wirst.

Beim ersten Start von Ubuntu (Startmenü → "Ubuntu") legst du einen Benutzernamen und ein Passwort an - das ist von nun an dein Linux-Login, unabhängig vom Windows-Login.

Ab hier bist du in einer echten Ubuntu-Umgebung. Alle Terminal-Befehle aus Kapitel 2 funktionieren identisch zu einem nativen Linux-Rechner.

Zwei Dinge, die unter WSL2 anders laufen als unter nativem Linux:

Erstens, venv-Aktivierung (Kapitel 1.4): Innerhalb von WSL2 identisch zu Linux (source .venv/bin/activate). Nur falls du Python zusätzlich nativ unter Windows installierst (für dieses Buch nicht empfohlen), unterscheidet sich das (.venv\Scripts\activate) - bleib für dieses Buch komplett innerhalb von WSL2, dann stellt sich die Frage nicht.

Zweitens, systemd (Kapitel 12): WSL2 startet standardmässig ohne systemd - der in Kapitel 12.6-12.8 gezeigte systemd-Timer-Ansatz läuft dort ohne eine zusätzliche Einstellung nicht. Mehr dazu direkt in Kapitel 12.

1.1 Warum Python?

Python ist für unser Vorhaben ideal aus drei Gründen:

Die Standardbibliothek ist riesig. Du brauchst für die meisten Aufgaben keine zusätzlichen Pakete.

Es gibt für fast jeden Spezialfall ein gut gepflegtes Paket: HTTP-Anfragen (requests), RSS-Parsing (feedparser), Telegram-Bots (python-telegram-bot), Datenbanken (sqlite3 ist eingebaut).

Python-Code ist lesbar. Wenn du ein Skript in sechs Monaten wieder anschaust, verstehst du es ohne grosses Nachdenken.

Klingt nach Werbung. Ist aber ehrlich. Python ist nicht die schnellste Sprache, aber für die meisten Aufgaben - auch unsere - ist es irrelevant. Schnell genug ist es in den allermeisten Fällen.

1.2 Installation unter Linux