
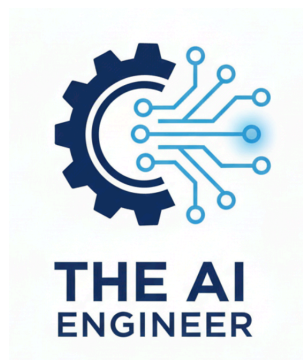


# AI, ML & Software Engineering

Building Intelligent Systems in Practice

Dr. Yves J. Hilpisch 

February 3, 2026



---

<sup>1</sup>Get in touch: <https://linktr.ee/dyjh>. Web page: <https://theaiengineer.dev>. Research, structuring, drafting, and visualizations were assisted by GPT 5.x as a co-writing tool under human direction. Comments and feedback are welcome.

# Preface

This book is about building. We focus on the engineering habits and minimal tooling that let you move from a promising idea to a dependable system—one you can run, explain, and improve. Each concept is grounded in code you can execute locally or in the cloud, favoring clarity over ceremony with deterministic seeds, concise logs, and tiny tests.

## Why These Topics Matter

We emphasise a handful of themes that show up in every system you ship.

- Software engineering for AI — without disciplined packaging, CLIs, and tests, good ideas remain one-offs. We establish habits that keep your work portable and reviewable.
- The ML lifecycle — configuration, tracking, and simple pipelines turn experiments into reproducible runs you can compare and ship.
- AI engineering beyond models — real applications combine retrieval, generation, and orchestration. You will learn to ground outputs with your own data, chain tools and steps, and balance latency, cost, and reliability.
- Responsible AI — ethics, trust, and governance are integrated from the start with safety checks and evaluation patterns.

## How to Use This Book

You can read linearly or dip into the workflows you need today; every chapter is paired with runnable code.

- Every chapter links to a runnable notebook and small Python modules in the companion repository. Start with the notebook to explore concepts, then adopt the scripts in your own projects.
- Use a local virtual environment or open the notebooks in Google Colab. Each example is designed to run on a laptop with modest resources.
- Treat outputs as contracts: we print verifiable, copy-pastable results so you can test quickly and catch regressions.

### Technical & Legal Note

This book and its code are provided for educational purposes only. Examples are simplified and may omit edge cases; verify results and assumptions for your context, especially in safety-, medical-, or financial-critical settings. The author and publisher provide the material “as is” without warranties and are not liable for losses arising from use; external dependencies and links can change over time.

# Contents

<b>Preface</b>	<b>i</b>
<b>I Software &amp; Systems Engineering Foundations</b>	<b>1</b>
<b>1 Engineering Mindset</b>	<b>3</b>
1.1 Why Engineering Patterns Matter . . . . .	3
1.2 A Tiny Running Example . . . . .	4
1.3 From Toy Script to System — A Roadmap . . . . .	5
1.4 Common Pitfalls . . . . .	5
1.5 Exercises . . . . .	6
1.6 Where We're Heading Next . . . . .	7
1.7 Further Sources . . . . .	7
<b>2 Software Engineering Essentials</b>	<b>8</b>
2.1 From Scripts to Packages . . . . .	8
2.2 Core Function in a Package . . . . .	9
2.3 A Thin CLI Over the Package . . . . .	9
2.4 Testing Basics — Confidence in Minutes . . . . .	10
2.5 Common Pitfalls . . . . .	10
2.6 Exercises . . . . .	11
2.7 Where We're Heading Next . . . . .	12
2.8 Further Sources . . . . .	12
<b>3 Infrastructure &amp; Deployment</b>	<b>13</b>
3.1 Containers in a Nutshell . . . . .	13
3.2 Dockerizing the Chapter 2 API . . . . .	14
3.3 Hardening: Non-root User Variant . . . . .	15
3.4 Local Orchestration with Docker Compose . . . . .	15
3.5 Common Pitfalls . . . . .	16
3.6 Exercises . . . . .	16
3.7 Where We're Heading Next . . . . .	17
3.8 Further Sources . . . . .	17
<b>II Machine Learning Engineering in Practice</b>	<b>18</b>
<b>4 The ML Lifecycle</b>	<b>20</b>
4.1 From Notebook Cells to a Script . . . . .	20
4.2 A Tiny Tracking Layer (No Heavy Framework) . . . . .	22
4.3 Compare Two Runs . . . . .	23
4.4 Common Pitfalls . . . . .	23

4.5	Optional: Hydra Configuration	23
4.6	Optional: Log to MLflow	24
4.7	Exercises	25
4.8	Where We're Heading Next	26
4.9	Further Sources	26
<b>5</b>	<b>Feature Engineering &amp; Pipelines</b>	<b>27</b>
5.1	A Modular, Scripted Pipeline	27
5.2	Orchestration Preview (Optional)	30
5.3	Common Pitfalls	32
5.4	Exercises	33
5.5	Where We're Heading Next	33
5.6	Further Sources	33
<b>6</b>	<b>MLOps at Scale</b>	<b>35</b>
6.1	Drift Demo — Detect a Simple Shift	35
6.2	Plan — What Happens When Drift Fires?	37
6.3	Exercises	38
6.4	Further Sources	38
<b>III</b>	<b>AI Engineering Beyond Models</b>	<b>40</b>
<b>7</b>	<b>AI Engineering Foundations</b>	<b>42</b>
7.1	The Systems View	42
7.2	A Tiny Retrieval Demo (Local Only)	43
7.3	Where Evaluation and Guardrails Fit	45
7.4	Common Pitfalls	45
7.5	Exercises	45
7.6	Further Sources	46
<b>8</b>	<b>Building with LLMs</b>	<b>47</b>
8.1	Prompts as Contracts	47
8.2	Structured Extraction (Local Demo)	48
8.3	A Tiny Oracle Set (Evaluation)	50
8.4	Schema-First Validation (Optional)	51
8.5	Common Pitfalls	52
8.6	Exercises	52
8.7	Further Sources	53
<b>9</b>	<b>Retrieval — Augmented Generation</b>	<b>54</b>
9.1	From Strings to Files	54
9.2	Common Pitfalls	57
9.3	Exercises	57
9.4	Further Sources	58
<b>10</b>	<b>AI Agents &amp; Orchestration</b>	<b>59</b>
10.1	Agents in One Paragraph	59
10.2	A Tiny Tool-Using Agent (Local)	60
10.3	Common Pitfalls	62
10.4	Exercises	62
10.5	Further Sources	62

<b>11 Scaling AI Systems</b>	<b>64</b>
11.1 Throughput vs. Latency	64
11.2 Batch Benchmark (CPU; GPU Optional)	65
11.3 Mixed Precision and Tensor Cores	67
11.4 Caching and KV Cache	67
11.5 Runtime Choices	68
11.6 Common Pitfalls	68
11.7 Exercises	68
11.8 Further Sources	68
<b>12 Ethics, Trust &amp; Governance</b>	<b>70</b>
12.1 Risk-First Framing	70
12.2 Measuring Fairness (Quick Pass)	70
12.3 Safety and Red-Teaming (LLMs)	71
12.4 Documentation and Governance	71
12.5 Example — Quick Fairness Metrics (Toy Data)	71
12.6 Example — Tiny Red-Team Harness (Local)	72
12.7 Example — Turn Metrics Into Tests (Thresholds)	74
12.8 Example — Read and Summarize the Report	75
12.9 Exercises	76
12.10 Further Sources (Original Papers & Standards)	76
<b>A Cheat Sheets</b>	<b>78</b>
A.1 What You Need (Quick)	78
A.2 Terminal Basics (First Steps)	78
A.3 Virtual Environments (Isolate Your Work)	78
A.3.1 Alternative: Conda Environments (If You Prefer Conda)	79
A.4 Your First Script (and How to Run It)	79
A.5 A Tiny Project Layout (Folders That Scale)	79
A.6 Testing (Confidence in Minutes)	80
A.7 Formatting and Linting (Clarity at Scale)	81
A.8 Optional: Static Types (Catch Mistakes Early)	81
A.9 Notebooks (Local and Colab)	81
A.10 Git (Save, Compare, Share)	81
A.11 Docker (Reproducible Runtime)	82
A.12 Makefile (Convenient Shortcuts)	82
A.13 Next Steps (Practice Loop)	83
A.14 Environment Variables (Configuration Without Code Changes)	83
A.15 Troubleshooting (Quick Fixes)	83
A.16 Further Sources	84
<b>B Engineering Tools</b>	<b>85</b>

# List of Figures

1.1 Clarify which dimension can move for this release. . . . .	4
2.1 Interfaces up top, domain in the middle, adapters at the edge. . . . .	9
3.1 Automate from commit to deploy; feed monitoring back into tests. . . . .	16
4.1 Make runs inspectable so you can review, rerun, and improve safely. . . . .	22
5.1 Each step produces artifacts and evidence that feed the next. . . . .	30
6.1 Feedback from monitoring informs the next training and release cycle. . . . .	37
7.1 Named stages make it easier to attach budgets, logs, and tests. . . . .	43
8.1 Treat prompt edits like code changes with measurable impact. . . . .	51
9.1 Each stage logs artifacts so you can inspect or re-run queries. . . . .	57
10.1 The loop stays observable when each arrow logs inputs and outputs. . . . .	59
11.1 Quantify savings before investing in new infrastructure. . . . .	67

# Contact

AI, ML & Software Engineering  
The AI Engineer

Get in touch:

<https://linktr.ee/dyjh>

<https://theaiengineer.dev>

© 2026 Dr. Yves J. Hilpisch — All rights reserved.

