FREE GUIDE · AIEWORKS · AIEWORKS.SUBSTACK.COM

The AI Engineering Quick-Start Guide

Everything a software engineer needs to know to successfully transition into AI engineering: the roles, the skills, the tools, the roadmap, and the honest truth about what it actually takes.

🎯 For Software Engineers | 🗓️ 20-Page Reference | ⚡ Roles · Skills · Roadmap · Tools

What's Inside

CHAPTER 01

What Is AI Engineering?

A precise definition of a role that barely existed five years ago.

AI Engineering is the discipline of **building production systems that use artificial intelligence and machine learning as core components**. It sits at the intersection of

software engineering and machine learning — covering everything from integrating LLM APIs to designing distributed training pipelines to deploying real-time inference services.

The term emerged organically around 2022–2023 as companies realized they needed engineers who could do more than call an OpenAI API or run a Jupyter notebook. They needed people who could architect reliable, scalable, observable AI-powered systems — and ensure those systems behave correctly when things go wrong.

An AI engineer is not a researcher inventing new algorithms. They are an architect making those algorithms work reliably at scale for real users.

> **THE CORE DISTINCTION**
>
> AI Research creates new algorithms. Data Science extracts insights from data. **AI Engineering ships systems** — production-ready, observable, maintainable software that uses ML as a component to create user value.

The field expanded dramatically in 2023–2024 with the rise of accessible LLMs. Suddenly, tasks that previously required PhD-level researchers (building a Q&A system, a code assistant, a document classifier) could be built by any competent software engineer with the right architectural knowledge. This democratization created enormous demand for AI engineers — people who understand both the software engineering side and enough ML to build these systems responsibly.

CHAPTER 02

AI Engineering vs. ML Engineering vs. Data Science

Three overlapping roles that companies often confuse — here's the honest difference.

AI ENGINEER

Builds AI-powered systems

Focuses on integrating AI models (pre-trained or fine-tuned) into production products. Deep software engineering + enough ML to

ML ENGINEER

Builds & trains models

Owns the full model lifecycle: data pipelines → training → evaluation → deployment. Stronger ML theory depth than AI Engineer. Feature

DATA SCIENTIST

Extracts insight from data

Statistical analysis, A/B testing, business metric analysis, ad-hoc ML for business questions. Strongest on statistics and

be dangerous. LLM integrations, RAG systems, agents, inference optimization.

engineering, custom model architectures, distributed training.

analytics. Less focus on production engineering.

HONEST REALITY

In practice, these roles blur significantly at smaller companies. Many startups hire one person who does all three. FAANG companies have clearly separated tracks. As a career move: **AI Engineering and ML Engineering have higher compensation ceilings** and more technical depth than Data Science tracks at most companies.

CHAPTER 03

The 8 Most Common AI Engineering Roles

The specific job titles you'll see, what they actually do, and which companies hire for each.

Most Common

ML Engineer (MLE)

Owns data pipelines, feature engineering, model training, and deployment. The most common AI role at tech companies. Requires both ML depth and strong software engineering.

High Growth

AI / LLM Engineer

Builds LLM-powered products: RAG systems, agents, evaluation pipelines. Less model training, more API integration, prompt engineering, and production reliability. Fastest-growing role in 2024–2025.

Deep Technical

Applied Research Scientist

Applies cutting-edge ML research to product problems. PhD often preferred. Sits between researcher and engineer. Heavy model architecture work and novel experiments.

Infrastructure Focus

ML Platform / Infrastructure Engineer

Builds the tools other ML teams use: feature stores, training clusters, model registries, serving infrastructure. Strong distributed systems background required.

Emerging Role

MLOps / AI Reliability Engineer

Ensures ML systems are observable, reliable, and reproducible. Monitors model drift, manages retraining pipelines, incident response for AI systems. DevOps-equivalent for ML.

Data-Heavy

Data Engineer (AI Focus)

Builds data pipelines that feed ML systems. ETL/ELT, Spark/Flink, streaming data, data quality frameworks. Less model work, critical infrastructure enabler.

Product-Adjacent

AI Product Engineer

Full-stack or backend engineers who specialize in shipping AI-powered product features. Strong at integrating ML into user-facing products, less depth on model internals.

Safety-Focused

AI Safety / Alignment Engineer

Ensures AI systems behave safely and reliably. Evaluations, red-teaming, Constitutional AI, RLHF pipelines. Growing rapidly at Anthropic, OpenAI, Google DeepMind.

CHAPTER 04

Where AI Engineering Is Going (2025–2027)

The honest outlook — what's growing, what's plateauing, and where to position yourself.

LLM Engineering is now table stakes. By 2025, any engineer who can't build a basic RAG pipeline or integrate an LLM API is behind. This is no longer a differentiating skill — it's baseline. The bar has moved to: production reliability, evaluation at scale, cost optimization, and multi-modal systems.

Agentic systems are the frontier. AI agents — systems where LLMs autonomously take multi-step actions — are moving from demos to production. Engineers who can build reliable, fault-tolerant agent architectures with proper observability are in extremely high demand. This requires deep understanding of tool use, memory systems, and failure modes.

ML infrastructure is a specialization with massive leverage. As companies scale, they need engineers who can make training and inference 10× faster and 5× cheaper. GPU optimization, quantization, speculative decoding, and efficient serving are niche skills with exceptional compensation.

Vertical AI is exploding. Healthcare, legal, finance, and manufacturing AI are seeing massive investment. Domain-specific AI engineers who understand both the ML and the industry context command premium compensation.

HIGH GROWTH (2025–2027)

- ✅ LLM/Agentic systems engineering
- ✅ ML infrastructure & GPU optimization
- ✅ AI evaluation & red-teaming
- ✅ Multimodal AI systems
- ✅ Vertical AI (healthcare, legal, fin)
- ✅ AI safety & alignment engineering

STABILIZING / COMMODITIZING

- ⚠️ Basic LLM API integration
- ⚠️ Simple RAG chatbots
- ⚠️ Classical ML (gradient boosting for tabular)
- ⚠️ SQL-heavy data science roles
- ⚠️ Basic dashboard and BI work
- ⚠️ Prompt engineering as a standalone skill

CHAPTER 05

Compensation Reality Check

Real numbers from 2024–2025 job postings and compensation databases (US market, unless noted).

ROLE	LEVEL	BASE SALARY (US)	TOTAL COMP (FAANG)
ML Engineer	L4 / Entry	$150K–$190K	$200K–$280K
ML Engineer	L5 / Mid	$190K–$230K	$280K–$420K
ML Engineer	L6 / Senior	$230K–$280K	$420K–$650K
ML Engineer	L7 / Staff	$280K–$350K	$650K–$1M+
Applied Research Scientist	L5	$200K–$260K	$350K–$550K
Applied Research Scientist	L6/Senior	$250K–$320K	$500K–$900K
AI/LLM Engineer	Mid–Senior	$180K–$260K	$250K–$500K

ROLE	LEVEL	BASE SALARY (US)	TOTAL COMP (FAANG)
ML Platform Engineer	Senior+	$230K–$310K	$400K–$750K

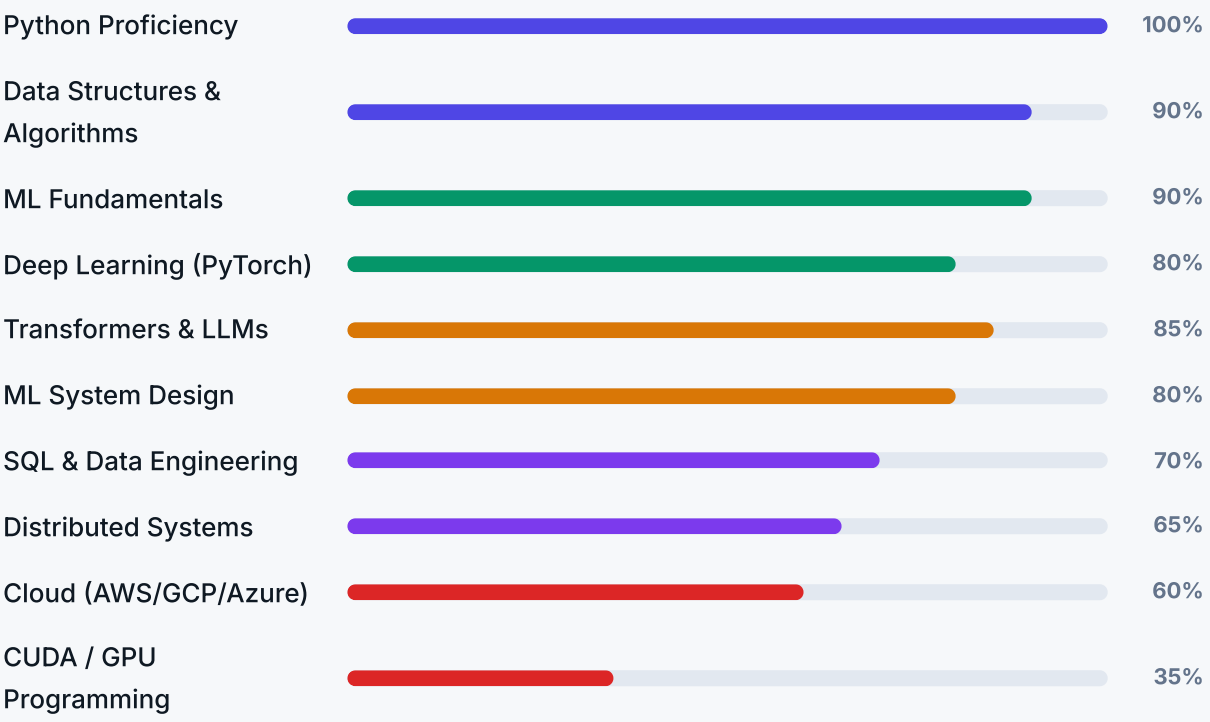
> 💡 **KEY INSIGHT**
>
> The gap between FAANG and startup compensation has narrowed significantly at the mid-level. Well-funded AI startups (Series B+) now compete with FAANG on cash comp. The FAANG advantage is primarily in **RSU/equity upside** — but AI startup equity can be more valuable if the company succeeds. Negotiate aggressively.

CHAPTER 06

The Skills You Need — Honestly Assessed

What to build, in what order, based on what actually matters in interviews and on the job.

Skill Priority by Role Type



> **SKILL PRIORITY PRINCIPLE**
>
> Get to **100% proficiency in Python and DSA first** — these are tested in every interview. Then ML fundamentals (deep theory, not just sklearn calls). Only then specialize in LLMs

> or infrastructure. Many candidates make the mistake of learning the shiny new thing (like LangChain) before they can explain backpropagation from first principles.

CHAPTER 07

The Canonical AI Engineering Tech Stack

The tools that appear most commonly in job descriptions and production AI systems in 2025.

CATEGORY	PRIMARY TOOLS	WHY IT MATTERS
Deep Learning	`PyTorch`, `JAX`, `TensorFlow`	PyTorch is the industry standard. 80%+ of AI roles use it.
LLM APIs	`OpenAI API`, `Anthropic API`, `Hugging Face`	Every AI product uses at least one. Know the auth, rate limits, and pricing models.
LLM Orchestration	`LangChain`, `LlamaIndex`, `DSPy`	RAG and agent pipelines. Know the patterns, not just the APIs.
Vector Databases	`Pinecone`, `Weaviate`, `FAISS`, `pgvector`	Core for semantic search and RAG. Understand ANN algorithms (HNSW).
Experiment Tracking	`MLflow`, `W&B`, `Comet`	Reproducibility and experiment management are essential in any ML role.
Feature Store	`Feast`, `Tecton`, `Vertex Feature Store`	Prevents train/serve skew. Critical for senior roles.
Model Serving	`TorchServe`, `Triton`, `vLLM`, `FastAPI`	Inference optimization is a major differentiator. vLLM is growing fast.
Data Processing	`Spark`, `Beam`, `dbt`, `Pandas`	Large-scale data transformation. Spark is required for senior/staff ML roles.
Orchestration	`Airflow`, `Prefect`, `Dagster`	Production pipelines need scheduling and observability.
Containers/Infra	`Docker`, `Kubernetes`, `Terraform`	Every ML system runs in containers. K8s knowledge becomes essential at L5+.

CATEGORY	PRIMARY TOOLS	WHY IT MATTERS
Cloud ML	`SageMaker`, `Vertex AI`, `Azure ML`	Know at least one. Amazon/AWS roles require SageMaker depth.
Observability	`Prometheus`, `Grafana`, `Arize`, `LangSmith`	Model monitoring is a rapidly growing skill requirement for ML SRE and senior roles.

CHAPTER 08

Your 90-Day Transition Plan

A concrete day-by-day roadmap from software engineer to AI engineer, whether you're starting from scratch or upskilling.

Days 1–10

Foundations Assessment

Run an honest skill audit against the skill bar chart in Chapter 6. Identify your top 3 gaps. Begin CS229 (Andrew Ng, free on YouTube) for ML theory. Do 2 LeetCode problems daily — focus on Arrays and HashMaps first. Read Chapter 1–3 of "Mathematics for Machine Learning" (free PDF at mml-book.com).

Days 11–25

ML Theory Sprint

Complete Weeks 1–4 of CS229. Implement linear regression, logistic regression, decision trees, and k-NN from scratch in NumPy. Run each with a real dataset (use UCI ML Repository). Read "Hands-On ML" Chapters 1–6. Continue LeetCode: add Trees and Binary Search.

Days 26–45

Deep Learning Core

Complete Fast.ai Lesson 1–7. Implement a CNN from scratch in PyTorch (no torchvision). Fine-tune a pre-trained ResNet on a custom image dataset. Read "Hands-On ML" Chapters 10–15. Start Stanford CS224n on YouTube for NLP and Transformers. LeetCode: add Dynamic Programming.

Days 46–60

LLMs & Modern AI

Read "Attention Is All You Need" paper. Build a mini-Transformer from scratch using only PyTorch primitives. Complete HuggingFace NLP Course (free). Build a RAG system using LangChain + Pinecone + any LLM. Deploy it as a FastAPI endpoint. Study OpenAI API documentation completely.

Days 61–75

Production & MLOps

Read "Designing ML Systems" by Chip Huyen (buy it). Containerize your RAG system with Docker. Set up MLflow experiment tracking for your projects. Learn basic Kubernetes (just enough to deploy a pod and a service). Build a model monitoring dashboard with Prometheus + Grafana.

Days 76–90

Interview Prep Sprint

Target LeetCode NeetCode 150 completion. Practice 3 full ML system design problems using the framework in CS04 cheat sheet. Do 5 mock interviews on Pramp. Research each company you're applying to (read their engineering blogs). Prepare 8 STAR behavioral stories. Update LinkedIn with your project portfolio.

> **THE MOST IMPORTANT THING NO ONE TELLS YOU**
>
> Build one **real project that runs in production**. It doesn't need to serve millions of users — it needs to serve at least one real user (even yourself) via a public URL. A Heroku/Railway/Render deployment counts. This project is worth 10× any number of completed courses in an interview conversation.

CHAPTER 09

Building Your First Production AI System

A concrete recipe for a production-grade project that demonstrably proves you can build real AI systems.

The Project: Build a document Q&A system with semantic search, an LLM answer generator, source citations, an evaluation harness, and basic monitoring. Host it publicly. This project demonstrates the full AI engineering stack in one artifact.

ARCHITECTURE COMPONENTS

- 📥 **Ingest:** PDF/URL → text chunks (LangChain)
- 🧠 **Embed:** text-embedding-3-small (OpenAI)
- 💾 **Store:** Pinecone / FAISS index
- 🔍 **Retrieve:** semantic search + BM25 hybrid
- 🤖 **Generate:** GPT-4o / Claude 3.5 Sonnet
- 📊 **Evaluate:** RAGAS faithfulness + relevance
- 📡 **Serve:** FastAPI + Docker
- 👁️ **Observe:** LangSmith traces + custom metrics

WHAT IT PROVES TO INTERVIEWERS

- ✅ You understand RAG architecture end-to-end
- ✅ You can evaluate LLM output quality
- ✅ You know how to serve ML systems
- ✅ You care about observability
- ✅ You can containerize and deploy
- ✅ You handle chunking strategy trade-offs
- ✅ You've dealt with real-world failure modes

✅ You can discuss cost and latency optimization

GITHUB STRATEGY

Your repo README is the first thing interviewers see. Write a README that explains: (1) what problem it solves, (2) architecture diagram with components labeled, (3) setup instructions that actually work, (4) evaluation results with numbers, (5) known limitations and what you'd improve. A great README turns a side project into a portfolio piece.

CHAPTER 10

Resources & Next Steps

The exact resources — in priority order — that give you the highest return on study time.

📚 Books (in order)

- [ ] Hands-On ML — Aurélien Géron
  Start here. Best practical reference. Read Ch 1–15.
- [ ] Mathematics for ML — Deisenroth et al.
  Free PDF. Read when math comes up as a gap.
- [ ] Designing ML Systems — Chip Huyen
  Essential for production and system design rounds.
- [ ] ML System Design Interview — Aminian & Xu
  Case-study workbook. Read in final 4 weeks before interviews.

🎓 Courses (in order)

- [ ] Stanford CS229 (YouTube/Stanford online)
  Best ML theory. Free. Do the problem sets.
- [ ] Fast.ai Practical DL for Coders
  Top-down DL. Best for quick intuition building. Free.
- [ ] HuggingFace NLP Course
  Transformers and fine-tuning. Free. Do the exercises.
- [ ] Stanford CS224n (YouTube)
  Best NLP course. Focus on Transformer lectures.

Subscribe Free →

Browse Vault Products →

AIEWorks

The AI engineering newsletter trusted by 35,000+ engineers worldwide.