Phase 0 — Setup (2 weeks)

- Pick stack: **Python + PyTorch**, **FastAPI**, **PostgreSQL**, **Docker**, **GitHub Actions**.
- Read: systems design for data/ML (latency, throughput, SLOs, cost per request).
- Deliverable: repo template with linting, tests, Makefile, DevContainer.

Phase 1 — Core ML Engineering (0–2 months)

- Refresh math: linear algebra, prob/stats, optimization; implement logistic regression, tree, gradient boosting from scratch (numpy).
- Libraries: **scikit-learn**, **PyTorch Lightning**; experiment tracking with **MLflow**.
- Project 1: classic tabular ML (fraud/credit/churn). Ship a FastAPI inference service + Docker image.

Phase 2 — Data & MLOps Foundations (2–4 months)

- Data eng: **Airflow**/**Prefect**, feature stores (Feast), data contracts, CDC basics.
- CI/CD for ML: unit tests for data & models, model registry, canary/blue-green deploys.
- Infra: **AWS/GCP/Azure** (pick one); IaC with **Terraform**; container orch: **Kubernetes** (EKS/GKE/AKS).
- Observability: **Prometheus/Grafana**, **OpenTelemetry**, drift/quality monitoring (whylogs/evidently).
- Project 2: training pipeline + feature store + CI/CD to a k8s cluster with autoscaling.

Phase 3 — Generative AI & LLMOps (4–7 months)

- RAG system design: chunking, embeddings, vector DBs (**FAISS**, **Weaviate**, **pgvector**), retrieval evaluation.

- Prompt & policy: guardrails, PII filtering, prompt-injection defenses, evaluation harnesses (G-Eval style), offline & online metrics.

- Latency/cost control: quantization (**bitsandbytes**, **AWQ**), batching, GPU vs CPU trade-offs, caching (semantic & response).

- Project 3: production-grade RAG service with offline eval, A/B tests, and cost dashboards.

Phase 4 — Scalable Architectures (7–10 months)

- Patterns: event-driven (Kafka), **CQRS/Saga**, idempotency, exactly-once with outbox.

- Data @ scale: lakehouse (**Delta/Apache Iceberg**), streaming vs batch, feature freshness SLAs.

- Security & governance: IAM least privilege, **KMS**, secrets mgmt, model cards, audit trails; multi-tenant isolation.

- Project 4: multi-region, auto-failover AI API (CDN + WAF), RTO/RPO defined and tested.

Phase 5 — Platform & Enterprise Skills (10–14 months)

- Platform thinking: standardized **golden paths**, reusable templates, cost guardrails, shared feature store.

- Compliance: basic **ISO 27001/SOC 2** mapping for AI workflows; DPIA for sensitive data.

- Documentation: **Architecture Decision Records (ADRs)**, threat models, runbooks, postmortems.

- Project 5 (capstone): “AI Platform Starter Kit” repo—one-click deploy of a full stack (ETL → training → registry → serving → monitoring), with cost and reliability SLOs.

Certifications

- **AWS Solutions Architect (Assoc → Pro)** *or* **Azure Architect Expert (AZ-305)**;

- **CKA** (Kubernetes); **Terraform Associate**;
- Optional: **Databricks** or **GCP Professional ML Engineer** (if that's your stack).

Portfolio "Architect"

- 3 public reference architectures (diagrams + ADRs + costs + SLOs).
- A RAG system with robust eval + safety filters + latency/cost charts.
- A k8s-based serving stack with A/B & canary deploys, autoscaling, and rollback playbooks.
- Cost-optimization case study (e.g., quantized model cutting GPU spend 30% with equal quality).

Interview & Comp Playbook

- Systems-design drill: clarify reqs → constraints → candidate designs → trade-offs (latency, cost, privacy) → risks → mitigations. Whiteboard the data & control planes.
- Behavioral: STAR stories for incidents, migrations, and cost reductions.
- Negotiate total comp (base + bonus + RSUs). Leverage city bands (BLR/PNQ/HYD > others) and competing offers.

Quick start (next 14 days)

- Day 1–3: spin up a k8s cluster (managed), deploy a toy FastAPI model with autoscaling.
- Day 4–7: build a tiny RAG using pgvector + OpenAI/Local LLM; add offline eval harness.
- Day 8–14: wire MLflow + CI/CD; add drift monitors; publish an ADR + diagram for your setup.