## PHASE 1 — IMMEDIATE READINESS (Weeks 1–4)

### 🎯 Narrative Focus

*"I'm a modern web architect who understands frontend at scale and is pragmatically leading the AI transition."*

### 🔷 What to KEEP (Validated)

| Topic | What to Focus On (Refined) | Resources |
|---|---|---|
| React as architecture | **Whiteboard-level understanding only** (don't read everything) | https://react.dev/learn/render-and-commit |
| Concurrent rendering | Why it exists, when it helps/hurts | https://react.dev/reference/react/useTransition |
| Micro-frontends | **Why they fail more often than they succeed** | https://martinfowler.com/articles/micro-frontends.html |
| Node event loop | **Diagram + explanation only** (most common senior gotcha) | https://nodejs.org/en/learn/asynchronous-work/event-loop |
| AI executive vocabulary | Cost, latency, hallucinations, guardrails | https://www.databricks.com/glossary/retrieval-augmented-generation |

---

### 🔷 NEW (Added from Feedback)

| Gap Filled | Why It Matters at Your Level | Resource |
|---|---|---|
| **LLM mechanics vocabulary** | Expert expect fluency, not math | https://platform.openai.com/docs/guides/text-generation |
| Context window / tokens | Cost & scalability conversations | https://www.anyscale.com/blog/llm-context-windows |
| Temperature & determinism | Risk & predictability in prod | https://docs.anthropic.com/claude/docs/temperature |

---

### 🔷 Project Step (Refined)

| Step | Action |
|---|---|
| Project Step 1 | **Node.js script calling OpenAI or AWS Bedrock → "Hello AI"** |
| Goal | Remove fear, gain vocabulary, prove hands-on credibility |

| Step | Action |
|---|---|
| Timebox | 1–2 evenings |

---

## PHASE 2 — SYSTEM DESIGN & CLOUD (Weeks 5–8)

### 🎯 Narrative Focus

*“I design systems that scale, are observable, and don’t bankrupt the company.”*

---

### 🔷 What to KEEP (Validated)

| Topic | Focus | Resources |
|---|---|---|
| Multi-region architecture | Conceptual design, not configs | https://aws.amazon.com/architecture/multi-region/ |
| CDN + caching | Draw cache layers | https://aws.amazon.com/caching/ |
| BFF pattern | Frontend-driven backend design | https://samnewman.io/patterns/architectural/bff/ |
| Rate limiting | Protecting systems at scale | https://cloud.google.com/architecture/rate-limiting-strategies-techniques |

---

### 🔷 NEW (Critical Gap Filled)

| Topic | Why Leaders Must Know This | Resource |
|---|---|---|
| **Infrastructure as Code (IaC)** | Modern cloud is unreadable without it | https://martinfowler.com/articles/infrastructure-as-code.html |
| Terraform / CDK (conceptual) | Team-scale infra management | https://developer.hashicorp.com/terraform/intro |
| Back-of-envelope math | System design | https://www.educative.io/blog/back-of-the-envelope-calculations |

---

### 🔷 Project Step (Optimized)

| Step | Action |
|---|---|
| Project Step 2 | **Add Vector DB (Pinecone / OpenSearch) → enable RAG** |
| Cloud tie-in | Deploy via AWS (Lambda / API Gateway) |
| Outcome | AI + Cloud + Node in one system |

---

## PHASE 3 — LEADERSHIP LAYER (Weeks 9–12)

### 🎯 Narrative Focus

*"I can lead teams, manage risk, and make AI decisions responsibly."*

---

### 🔷 What to KEEP (Validated)

| Topic | Focus | Resource |
|---|---|---|
| Monolith vs microservices | When NOT to split | https://martinfowler.com/articles/monolith-first.html |
| Schema evolution | Safe change management | https://www.confluent.io/blog/schema-evolution/ |
| Eventual consistency | Trade-offs | https://www.allthingsdistributed.com/2008/12/eventually_consistent.html |

---

### 🔷 NEW (Major Leadership Upgrade)

| Topic | Why This Matters | Resource |
|---|---|---|
| **End-to-End MLOps pipeline** | Leaders must visualize flow | https://cloud.google.com/architecture/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning |
| AI risk management | Regulatory & reputational risk | https://www.nist.gov/ai |
| OSS vs Proprietary models | Strategic decision-making | https://www.anyscale.com/blog/open-source-vs-closed-llms |

---

### 🔷 Behavioral Preparation (Added)

| Task | Details |
|---|---|
| STAR stories (5) | Conflict, failure, mentoring, influence, AI decision |
| Executive reflection | "What I'd do differently" (critical at senior level) |

---

## PHASE 4 — MASTERY & VISION (Months 3–4)

## 🎯 Narrative Focus

*"AI adoption is 80% systems and people, 20% models."*

---

### 🔷 What to KEEP (Validated)

| Topic | Focus | Resource |
|---|---|---|
| Strangler pattern | AI in legacy systems | https://martinfowler.com/articles/strangler-fig-application.html |
| Build vs buy | Long-term cost & control | https://martinfowler.com/articles/build-or-buy.html |
| AI observability | Trust & reliability | https://opentelemetry.io/docs/ |

---

### 🔷 NEW (Executive-Level Additions)

| Topic | Why Expert Care | Resource |
|---|---|---|
| AI governance | Board-level concern | https://cloud.google.com/architecture/ml-model-governance |
| Cost controls | CFO alignment | https://www.anyscale.com/blog/llm-cost-optimization |
| Regulation readiness | Future-proofing | https://www.oecd.org/ai/ |

---

## 🔑 FINAL REFINEMENT (Key Takeaways)

- ✅ The original plan was **correct**, but too dense
- ✅ This version:
  - Prevents burnout
  - Combines AI + AWS + Node into **one capstone**
  - Elevates you from *"hands-on senior"* to *"AI-capable technical leader"*
- ❌ You should **not** aim to read every link
- ✅ Aim to **explain, draw, and defend trade-offs**

---

## 🔥 Primary Capstone (Final Decision)

**Serverless RAG Bot on AWS**

- React chat UI
- Node.js Lambda backend
- AWS Bedrock / OpenAI

- Vector DB
- Observability + cost awareness

This single project **covers 80% of your surface area**.

---

| | |
|---|---|
| expectations | https://staffeng.com/guides/ |
| Engineering leadership vs IC depth | https://martinfowler.com/articles/leadership.html |
| Making trade-offs at scale | https://martinfowler.com/articles/decision-making.html |

---

# PHASE 1 — IMMEDIATE READINESS

## 🔷 React — Senior / Principal Level

| Topic | Learning Resources |
|---|---|
| React rendering architecture | https://react.dev/learn/render-and-commit |
| Concurrent rendering | https://react.dev/reference/react/useTransition |
| Suspense | https://react.dev/reference/react/Suspense |
| Streaming SSR | https://react.dev/reference/react-dom/server |
| Server vs Client Components | https://react.dev/learn/server-components |
| Web Vitals (LCP, CLS, TTI) | https://web.dev/vitals/ |
| Frontend performance at scale | https://web.dev/fast/ |
| Bundle splitting strategies | https://web.dev/code-splitting/ |
| Hydration cost | https://web.dev/rendering-on-the-web/ |
| Micro-frontends (when/why) | https://martinfowler.com/articles/micro-frontends.html |
| Module Federation | https://webpack.js.org/concepts/module-federation/ |
| State management at scale | https://react.dev/learn/scaling-up-with-reducer-and-context |
| Domain-driven frontend state | https://martinfowler.com/articles/modularizing-react-apps.html |
| Frontend observability | https://opentelemetry.io/docs/concepts/observability-primer/ |
| Error budgets | https://sre.google/sre-book/error-budgets/ |

---

## 🔷 Node.js — Senior Systems Thinking

| Topic | Learning Resources |
|---|---|
| Event loop (high level) | https://nodejs.org/en/learn/asynchronous-work/event-loop |

| Topic | Learning Resources |
|---|---|
| Async execution model | https://nodejs.org/en/docs/guides/event-loop-timers-and-nexttick/ |
| Backpressure & streams | https://nodejs.org/en/learn/modules/streams |
| CPU vs IO workloads | https://nodejs.org/en/learn/asynchronous-work/blocking-vs-non-blocking |
| Horizontal scaling | https://nodejs.org/en/learn/asynchronous-work/scaling-nodejs |
| API versioning | https://martinfowler.com/articles/api-versioning.html |
| Caching strategies | https://aws.amazon.com/caching/ |
| Failure modes & retries | https://martinfowler.com/articles/patterns-of-distributed-systems/retry.html |

---

## 🔷 AI/ML — Executive-Level Literacy

| Topic | Learning Resources |
|---|---|
| Supervised vs unsupervised | https://developers.google.com/machine-learning/intro |
| Training vs inference | https://cloud.google.com/architecture/ml-inference |
| Model drift | https://cloud.google.com/architecture/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning |
| Hallucinations | https://www.anthropic.com/research |
| Precision / Recall | https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall |
| RAG vs fine-tuning | https://www.databricks.com/glossary/retrieval-augmented-generation |
| Cost–latency–accuracy trade-offs | https://www.anyscale.com/blog |
| Guardrails | https://platform.openai.com/docs/guides/safety-best-practices |
| Human-in-the-loop | https://cloud.google.com/architecture/human-in-the-loop |

---

## 🔷 Low-Lift ML Project — Content Classification

| Topic | Learning Resources |
|---|---|
| Text classification basics | https://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html |
| Transformers API (simple) | https://huggingface.co/docs/transformers/tasks/sequence_classification |
| Model lifecycle | https://ml-ops.org/content/model-lifecycle |
| Serving ML via API | https://fastapi.tiangolo.com/ |

| Topic | Learning Resources |
|---|---|
| Node ↔ Python integration | https://martinfowler.com/articles/patterns-of-distributed-systems/api-gateway.html |

## PHASE 2 — CLOUD & SYSTEM DESIGN

### 🔷 AWS + Cloud Architecture

| Topic | Learning Resources |
|---|---|
| Multi-region architecture | https://aws.amazon.com/architecture/multi-region/ |
| Caching layers | https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/Introduction.html |
| Event-driven systems | https://aws.amazon.com/event-driven-architecture/ |
| API Gateway vs LB | https://aws.amazon.com/compare/the-difference-between-api-gateway-and-load-balancer/ |
| Async workflows | https://aws.amazon.com/serverless/event-driven-architecture/ |
| Cost optimization | https://aws.amazon.com/aws-cost-management/ |
| Security boundaries | https://docs.aws.amazon.com/wellarchitected/latest/security-pillar/welcome.html |

### 🔷 Frontend + Cloud System Design

| Topic | Learning Resources |
|---|---|
| Global frontend architecture | https://web.dev/distribute-content-efficiently/ |
| Backend-for-Frontend (BFF) | https://samnewman.io/patterns/architectural/bff/ |
| AI feature rollout & fallback | https://martinfowler.com/articles/feature-toggles.html |

### 🔷 High-ROI AI Project #1 — RAG System

| Topic | Learning Resources |
|---|---|
| RAG architecture | https://www.pinecone.io/learn/retrieval-augmented-generation/ |
| LangChain | https://python.langchain.com/docs/ |
| Vector databases | https://www.pinecone.io/learn/vector-database/ |

| Topic | Learning Resources |
|---|---|
| Evaluation of RAG | https://www.anyscale.com/blog/evaluating-llm-applications |
| Source attribution | https://docs.langchain.com/docs/use_cases/question_answering/citations |

---

## PHASE 3 — SYSTEM DESIGN DEPTH

| Topic | Learning Resources |
|---|---|
| Monolith vs microservices | https://martinfowler.com/articles/monolith-first.html |
| SQL vs NoSQL | https://aws.amazon.com/nosql/ |
| Data modeling | https://martinfowler.com/articles/evodb.html |
| Eventual consistency | https://www.allthingsdistributed.com/2008/12/eventually_consistent.html |
| Rate limiting | https://cloud.google.com/architecture/rate-limiting-strategies-techniques |
| Backward compatibility | https://martinfowler.com/articles/consumerDrivenContracts.html |
| Schema evolution | https://www.confluent.io/blog/schema-evolution/ |

---

## PHASE 4 — MASTERY & VISION

### 🔷 AI in Legacy Systems

| Topic | Learning Resources |
|---|---|
| Incremental adoption | https://martinfowler.com/articles/strangler-fig-application.html |
| API-first AI | https://cloud.google.com/architecture/api-first |
| Risk mitigation | https://www.nist.gov/ai |

---

### 🔷 Leading AI Teams

| Topic | Learning Resources |
|---|---|
| AI team composition | https://www.thoughtworks.com/insights/articles/building-effective-ai-teams |
| Model governance | https://cloud.google.com/architecture/ml-model-governance |
| Cost controls | https://www.anyscale.com/blog/llm-cost-optimization |
| Evaluation frameworks | https://www.evidentlyai.com/ |

---

### 🔷 Long-Term AI Strategy

| Topic | Learning Resources |
|---|---|
| Build vs buy | https://martinfowler.com/articles/build-or-buy.html |
| Data moat | https://stratechery.com/ |
| AI observability | https://opentelemetry.io/docs/ |
| Regulation readiness | https://www.oecd.org/ai/ |

---

## CRITICAL AREAS (DO NOT SKIP)

| Topic | Learning Resources |
|---|---|
| DSA (reasoning focus) | https://neetcode.io/roadmap (conceptual use only) |
| OOAD / UML | https://www.uml-diagrams.org/ |
| Database design | https://use-the-index-luke.com/ |
| Concurrency concepts | https://queue.acm.org/detail.cfm?id=2745385 |
| Security basics | https://owasp.org/www-project-top-ten/ |
| Cost modeling | https://aws.amazon.com/well-architected/ |
| Trade-off articulation | https://martinfowler.com/articles/architecture-decision-records.html |