NULLVECTOR
.monster

PHASE 05 — OPERATOR

DAYS 85–105 · FINE-TUNING · RAG · AWS ML SPECIALTY

Phase 4 taught you to build language models. Phase 5 teaches you to make them useful.

Pre-trained models are raw material. Fine-tuning shapes them to a specific task. RAG gives them access to information they were never trained on. Together these two techniques are how almost every production AI application is built today. Phase 5 ends with the AWS ML Specialty certification — the most recognized machine learning credential in the job market.

// PHASE 5 AT A GLANCE

Duration	21 days · 3.5 hours per day
Milestone	Day 105 — AWS ML Specialty MLS-C01 certification
Fine-tuning	HuggingFace PEFT — huggingface.co/docs/peft
RAG	LlamaIndex — docs.llamaindex.ai
Vector DB	ChromaDB — docs.trychroma.com
AWS cert	AWS Skill Builder — explore.skillbuilder.aws
AWS exam	aws.amazon.com/certification/certified-machine-learning-specialty/
Your library	LLM Engineer's Handbook + RAG-Driven Generative AI (your Humble Bundle)

WEEK 1 · DAYS 85–91 · FINE-TUNING LLMS

DAY 085 · FINE-TUNING FUNDAMENTALS

When and why to fine-tune.

■ INDUSTRY (15 MIN)

Search: 'when to fine-tune vs prompt engineer vs RAG — AI engineering decision'

■ STUDY (90 MIN) — Fine-tuning concepts

Full fine-tuning: update all model weights. Expensive. Best performance.

Parameter-efficient fine-tuning (PEFT): update only a small subset of weights.

LoRA: add small trainable matrices alongside frozen pre-trained weights.

When to fine-tune: consistent format, domain-specific knowledge, custom behavior.

When NOT to fine-tune: RAG is usually better for factual knowledge injection.

→ huggingface.co/docs/transformers/training

■ BUILD (90 MIN) — First fine-tune with HuggingFace

Fine-tune DistilBERT for text classification on a custom task:

Choose a dataset from HuggingFace Hub (datasets library)

from transformers import DistilBertForSequenceClassification, Trainer

Define training arguments: output_dir, epochs, batch_size, learning_rate

Train for 3 epochs. Evaluate. Compare to zero-shot performance.

✓ DONE WHEN: DistilBERT fine-tuned on custom task. Accuracy compared to zero-shot baseline.

DAY 086 · LORA — LOW-RANK ADAPTATION

Fine-tune a 7B model on a laptop.

■ INDUSTRY (15 MIN)

Search: 'LoRA fine-tuning explained why it works' — the mathematical intuition.

■ STUDY (90 MIN) — LoRA paper and implementation

Read LoRA paper abstract and Section 2 (Problem Statement) and Section 4 (Method).

Key insight: weight updates during fine-tuning have low intrinsic rank.

LoRA decomposes the weight update ΔW into two small matrices A and B.

Instead of training ΔW (large), train A and B (small). Same expressiveness, less compute.

→ arxiv.org/abs/2106.09685

→ huggingface.co/docs/peft/conceptual_guides/lora

■ BUILD (90 MIN) — LoRA fine-tuning with PEFT

Fine-tune a larger model using LoRA:

```
from peft import get_peft_model, LoraConfig, TaskType
config = LoraConfig(task_type=TaskType.SEQ_CLS, r=8, lora_alpha=32)
model = get_peft_model(base_model, config)
model.print_trainable_parameters() # should be < 1% of total
```

Compare trainable parameters: full fine-tune vs LoRA.

✓ DONE WHEN: LoRA fine-tuning working. Trainable parameters less than 1% of total model.

DAY 087 · QLORA — FINE-TUNING ON CONSUMER HARDWARE

A 7B model on your laptop.

■ INDUSTRY (15 MIN)

Search: 'QLoRA paper what it made possible for AI developers'

QLoRA democratized LLM fine-tuning. Before it, you needed a $10k GPU cluster.

■ STUDY (90 MIN) — QLoRA and quantization

QLoRA = Quantized LoRA. Quantize base model to 4-bit. Fine-tune LoRA adapters in 16-bit.

Result: fine-tune a 7B parameter model on a single consumer GPU (16GB VRAM).

Or: use Google Colab free tier — they give you a T4 GPU.

Read: QLoRA paper abstract and introduction.

→ arxiv.org/abs/2305.14314

■ BUILD (90 MIN) — QLoRA fine-tune on Colab

Open Google Colab. Enable GPU runtime (Runtime → Change runtime type → T4 GPU).

Fine-tune a quantized LLM using QLoRA:

pip install bitsandbytes peft transformers accelerate

Load LLaMA 3.2 or Mistral 7B in 4-bit: load_in_4bit=True

Apply LoRA config. Train on a small instruction dataset.

Save the adapter weights.

→ colab.google.com

✓ DONE WHEN: QLoRA fine-tuning running on Colab. Adapter weights saved. Model responding to instructions.

DAY 088 · INSTRUCTION TUNING + DPO

How models learn to follow instructions.

■ INDUSTRY (15 MIN)

Search: 'RLHF explained simply how ChatGPT was trained to be helpful'

■ STUDY (90 MIN) — Alignment techniques

Instruction tuning: fine-tune on (instruction, response) pairs.

RLHF: train a reward model, then use RL to maximize reward. Expensive.

DPO (Direct Preference Optimization): simpler alternative to RLHF.

DPO uses preference pairs: (chosen response, rejected response) without a reward model.

Read: DPO paper — arxiv.org/abs/2305.18290 (abstract and intro).

→ huggingface.co/docs/trl/dpo_trainer

■ BUILD (90 MIN) — Instruction fine-tune

Fine-tune your Day 87 model on an instruction dataset:

Dataset: Alpaca, Dolly, or any instruction-response dataset from HuggingFace Hub

Format: {instruction: '...', input: '...', output: '...'}

Use the TRL library SFTTrainer: pip install trl

Before and after: give the same prompt. Does it follow instructions better?

✓ DONE WHEN: Model instruction fine-tuned. Before/after comparison shows improved instruction following.

DAY 089 · PUBLISH YOUR MODEL

Put it on HuggingFace for the world.

■ INDUSTRY (15 MIN)

Browse: huggingface.co/models — look at the most downloaded models. Read their model cards.

■ STUDY (45 MIN) — Model cards and responsible AI

A model card is a document that accompanies every published model. It includes:

Model description: what it is, what it does, who made it

Intended uses and limitations

Training data description

Evaluation results

Ethical considerations

This is standard practice. Every serious model has one.

→ huggingface.co/docs/hub/model-cards

■ BUILD (2.25 HRS) — Publish to HuggingFace Hub

Create a HuggingFace account if you haven't.

Push your fine-tuned model and adapter weights to the Hub:

model.push_to_hub('your-username/your-model-name')

tokenizer.push_to_hub('your-username/your-model-name')

Write a complete model card.

Test: can someone else load your model with from_pretrained('your-username/your-model-name')?

✓ DONE WHEN: Fine-tuned model published on HuggingFace Hub. Model card written. Anyone can download it.

DAY 090 · WEEK 1 CAPSTONE

A complete fine-tuning pipeline.

■ INDUSTRY (15 MIN)

Search: 'open source LLM fine-tuning use cases companies are using in 2025'

■ STUDY (30 MIN) — Identify your use case

The best fine-tuning projects solve real problems. Choose one:

Customer support bot for a specific domain you know well

Code assistant specialized in one language or framework

Writing assistant with a specific style or tone

Q&A; system for a specific knowledge domain

■ BUILD (2.75 HRS) — End-to-end fine-tuning pipeline

Build a complete pipeline for your chosen use case:

1. Curate or create a training dataset (at least 500 examples)

2. Format as instruction-response pairs

3. Fine-tune using QLoRA

4. Evaluate: does it perform better than the base model on your task?

5. Publish to HuggingFace Hub

6. Deploy as a HuggingFace Space — Portfolio Project #7

✓ DONE WHEN: Complete fine-tuning pipeline built. Custom model deployed. Portfolio Project #7 live.

DAY 091 · WEEK 2 BEGINS — WHAT IS RAG

Give LLMs access to your documents.

■ INDUSTRY (15 MIN)

Search: 'what is RAG retrieval augmented generation explained simply'

■ STUDY (90 MIN) — RAG fundamentals

The core problem RAG solves: LLMs don't know about your private documents.

RAG pipeline: query → retrieve relevant documents → augment prompt → generate response.

Three components: document store, embedding model, LLM.

Why not just fine-tune? RAG is better for factual recall. Fine-tuning is better for behavior.

Do the LlamaIndex 5-minute quickstart. Get a RAG system running today.

→ docs.llamaindex.ai/en/stable/getting_started/starter_example/

■ BUILD (90 MIN) — Your first RAG system

Build a RAG system over a document you care about:

Any PDF, set of web pages, or collection of text files

```
from llama_index.core import VectorStoreIndex, SimpleDirectoryReader
documents = SimpleDirectoryReader('data').load_data()
index = VectorStoreIndex.from_documents(documents)
query_engine = index.as_query_engine()
```

Ask it 5 questions. Verify answers against the source document.

✓ DONE WHEN: RAG system built over personal document. Answers correctly traced to source text.

DAY 092 · VECTOR DATABASES

The memory system for AI applications.

■ INDUSTRY (15 MIN)

Search: 'vector databases explained simply why AI needs them'

■ STUDY (90 MIN) — Embeddings and vector search

Embedding: convert text to a dense vector (list of numbers) that captures meaning.

Similar texts have similar embeddings — close in vector space.

Vector database: stores embeddings, retrieves most similar ones quickly.

Why not just keyword search? Keywords miss synonyms and concepts.

Key vector DBs: ChromaDB (free, local), Pinecone (hosted), Weaviate (open source).

Work through the ChromaDB getting started guide.

→ docs.trychroma.com/getting-started

■ BUILD (90 MIN) — ChromaDB document store

Build a semantic search system using ChromaDB:

Embed 100 document chunks using sentence-transformers

Store in ChromaDB

Query: find the 5 most semantically similar chunks to any question

Compare: ChromaDB semantic search vs simple keyword grep

Test with 10 queries. Which misses things the other catches?

✓ DONE WHEN: ChromaDB semantic search built. 10 queries tested. Semantic vs keyword comparison documented.

DAY 093 · ADVANCED RAG

Make retrieval actually good.

■ INDUSTRY (15 MIN)

Search: 'RAG failure modes why RAG gets things wrong'

Understanding why RAG fails is as important as knowing how to build it.

■ STUDY (90 MIN) — RAG optimization

Chunk size: too large = irrelevant context. Too small = missing context.

Hybrid search: combine semantic + keyword search for better recall.

Re-ranking: after retrieval, re-rank by relevance using a cross-encoder.

Query expansion: rewrite the query to retrieve more relevant documents.

Read: 'Advanced RAG techniques' on the LlamaIndex blog

■ BUILD (90 MIN) — Improved RAG pipeline

Take your Day 91 RAG system. Add three improvements:

1. Try 3 different chunk sizes (256, 512, 1024 tokens). Measure retrieval quality.

2. Add a re-ranking step using a cross-encoder model

3. Add query expansion: rewrite the question before retrieving

For each improvement: test with the same 5 questions. Does accuracy improve?

✓ DONE WHEN: Three RAG improvements implemented. Retrieval quality improved. Changes documented.

DAY 094 · RAG EVALUATION

Measure whether your RAG is actually good.

■ INDUSTRY (15 MIN)

Search: 'RAGAS evaluation framework for RAG systems'

■ STUDY (90 MIN) — RAG evaluation metrics

Faithfulness: does the answer only use information from retrieved context?

Answer relevancy: is the answer actually relevant to the question?

Context precision: are the retrieved chunks actually relevant?

Context recall: did retrieval find all necessary information?

Install RAGAS: pip install ragas

→ docs.ragas.io

■ BUILD (90 MIN) — Evaluate your RAG system

Create a test set of 20 question-answer pairs from your document.

Run your RAG system on all 20. Collect: question, answer, retrieved_contexts.

Evaluate with RAGAS:

```
from ragas import evaluate
from ragas.metrics import faithfulness, answer_relevancy
result = evaluate(dataset, metrics=[faithfulness, answer_relevancy])
```

Document your scores. What's the weakest metric? Fix it.

✓ DONE WHEN: RAG system evaluated with RAGAS. Scores documented. Weakest metric identified and improved.

DAY 095 · PRODUCTION RAG APP

From notebook to live application.

■ INDUSTRY (15 MIN)

Search: 'companies using RAG in production examples 2025'

RAG is the most commonly deployed LLM pattern in enterprise.

■ STUDY (45 MIN) — Production RAG architecture

A production RAG app needs: document ingestion pipeline, vector store,

retrieval layer, LLM layer, conversation memory, and a user interface.

LangChain or LlamaIndex handle most of this — know which component does what.

■ BUILD (2.25 HRS) — Deploy your RAG app

Build a RAG application with a web interface:

Use Gradio or Streamlit for the UI

Allow file upload — users can upload their own PDF and ask questions

Show which source chunks were retrieved for each answer

Add conversation memory — the system remembers previous questions

Deploy to HuggingFace Spaces — Portfolio Project #8

✓ DONE WHEN: RAG app deployed with file upload, source citations, and memory. Portfolio Project #8 live.

DAY 096 · LANGCHAIN CHAINS AND AGENTS

Orchestrating multiple AI calls.

■ INDUSTRY (15 MIN)

Search: 'LangChain what it does and why developers use it'

■ STUDY (90 MIN) — LangChain fundamentals

Work through the LangChain quickstart tutorial.

Key concepts: chains, agents, tools, memory, prompts.

A chain: connect multiple LLM calls in sequence.

An agent: let the LLM decide which tools to call and in what order.

→ python.langchain.com/docs/tutorials/

■ BUILD (90 MIN) — Multi-step LangChain pipeline

Build a pipeline that combines RAG and generation in a chain:

Step 1: retrieve relevant context from ChromaDB

Step 2: summarize the retrieved context

Step 3: use summary + original question to generate a final answer

Step 4: verify the answer is grounded in the retrieved context

Each step is a separate LLM call. The chain connects them.

✓ DONE WHEN: Multi-step LangChain pipeline built. All 4 steps running in sequence.

DAY 097 · AWS ML SPECIALTY PREP BEGINS

The most valuable ML cert you will earn.

■ INDUSTRY (15 MIN)

Search: 'AWS machine learning services overview 2025 — SageMaker Bedrock Rekognition'

■ STUDY (2 HRS) — AWS ML Skill Builder

Go to AWS Skill Builder. Create a free account.

Find the Machine Learning Learning Plan. Start from the beginning.

Key AWS ML services to know:

SageMaker: train, deploy, and manage ML models at scale

Bedrock: access foundation models (Claude, Llama, etc.) via API

Rekognition: computer vision — faces, objects, text in images

Comprehend: NLP — sentiment, entities, key phrases

Translate: neural machine translation

Forecast: time series forecasting

→ explore.skillbuilder.aws/learn/public/learning_plan/view/28/machine-learning-learning-plan

■ BUILD (45 MIN) — AWS free tier exploration

Create an AWS free tier account if you don't have one.

Use Rekognition to analyze an image via the console — no code needed.

Use Comprehend to analyze sentiment on 5 different sentences.

See what these services can do before learning how they work internally.

✓ DONE WHEN: AWS Skill Builder started. Rekognition and Comprehend explored via console.

DAY 098 · AWS SAGEMAKER

How ML models are trained at scale.

■ INDUSTRY (15 MIN)

Search: 'Amazon SageMaker explained simply what it does'

■ STUDY (2 HRS) — SageMaker deep dive

SageMaker components to understand for the exam:

Studio: web-based IDE for ML

Training jobs: managed training on any hardware

Endpoints: deploy models as REST APIs

Pipelines: automate the ML workflow

Feature Store: store and retrieve ML features

Model Monitor: detect data drift and model degradation

Work through the SageMaker getting started guide.

→ aws.amazon.com/sagemaker/getting-started/

■ BUILD (45 MIN) — Deploy a model to SageMaker

Deploy one of your trained models to a SageMaker endpoint:

Use the free tier where possible

Call the endpoint with a test input

Check the CloudWatch logs for the inference

Note: endpoints cost money when running. Delete when done.

✓ DONE WHEN: SageMaker study complete. Model deployed to endpoint (then deleted). Logs reviewed.

DAY 099 · AWS ML EXAM DOMAINS

Systematic exam preparation.

■ INDUSTRY (15 MIN)

Search: 'AWS ML Specialty MLS-C01 exam tips from someone who passed'

■ STUDY (2 HRS) — All 4 exam domains

Download the official exam guide. Study all 4 domains:

Domain 1 (24%): Data Engineering — S3, Glue, Kinesis, data pipelines

Domain 2 (26%): Exploratory Data Analysis — feature engineering, visualization

Domain 3 (28%): Modeling — algorithm selection, training, hyperparameter tuning

Domain 4 (22%): ML Implementation & Operations — SageMaker deployment, monitoring

→ aws.amazon.com/certification/certified-machine-learning-specialty/

■ BUILD (45 MIN) — Knowledge gaps

After reviewing the exam domains: identify your 3 weakest areas.

Write them down. Spend extra time on those over the next 3 days.

✓ DONE WHEN: All 4 exam domains reviewed. 3 weakest areas identified. Study plan adjusted.

DAY 100 · AWS PRACTICE EXAM + WEAK AREAS

Test yourself before the real test.

■ INDUSTRY (15 MIN)

100 days in. Search: 'what 100 days of consistent work produces' — any motivational piece.

You have been building for 100 days. That is not nothing.

■ STUDY (2 HRS) — Practice exam + focused study

Take the AWS ML Specialty practice exam on Skill Builder.

Target: 75% or above before booking the real exam.

Review every wrong answer. Understand why it was wrong.

Spend remaining time on your 3 weakest domains from Day 99.

→ explore.skillbuilder.aws

■ BUILD (45 MIN) — Data pipeline

Build a simple data pipeline using AWS S3 and Lambda (or local simulation):

Upload a CSV to S3

Process it (clean, transform)

Save processed version back to S3

Data pipelines are a major exam topic and a core ML engineering skill.

✓ DONE WHEN: Practice exam taken (target 75%+). Data pipeline built. Weakest domains studied.

DAY 101 · HYPERPARAMETER TUNING

Finding the best settings systematically.

■ INDUSTRY (15 MIN)

Search: 'hyperparameter tuning machine learning 2025'

■ STUDY (2 HRS)

SageMaker Hyperparameter Tuning Jobs: automated search over hyperparameter space.

Strategies: random search, grid search, Bayesian optimization.

Bayesian optimization: use results from previous trials to choose next trial.

Key hyperparameters for different algorithms:

Neural nets: learning rate, batch size, dropout, architecture

Tree methods: max depth, n_estimators, learning rate

SVM: C, kernel, gamma

■ BUILD (60 MIN)

Implement Bayesian hyperparameter optimization using Optuna:

pip install optuna

Define an objective function that trains your model and returns validation loss

Run 50 trials

Plot the optimization history

Compare best found vs your manual settings

✓ DONE WHEN: Optuna hyperparameter search running. Best parameters better than manual settings.

DAY 102 · ML SYSTEM DESIGN

Thinking at the system level.

■ INDUSTRY (15 MIN)

Search: 'ml system design machine learning 2025'

■ STUDY (2 HRS)

ML system design questions appear in senior ML engineer interviews.

Components of any ML system: data collection, feature engineering,

training pipeline, evaluation, serving, monitoring.

Common design questions: build a recommendation system, spam classifier,

fraud detection, search ranking, content moderation.

Read: 'Designing Machine Learning Systems' overview — chip huyen's blog at huyenchip.com

■ BUILD (60 MIN)

Design a complete ML system on paper for one of these problems:

A. News article recommendation for a media company

B. Fraud detection for an e-commerce platform

C. Content moderation for a social platform

Write a 400-word technical design document covering:

Data sources, features, model choice, training pipeline, serving, monitoring.

✓ DONE WHEN: 400-word ML system design document written. All components addressed.

DAY 103 · FINAL EXAM PREP

Everything before the test.

■ INDUSTRY (15 MIN)

Search: 'final exam prep machine learning 2025'

■ STUDY (2 HRS)

Full review of all 4 AWS ML Specialty domains.

Key algorithms and when to use them (this is heavily tested):

Linear regression: continuous output, linear relationship

XGBoost: tabular data, top Kaggle choice

K-means: clustering, unsupervised

Random cut forest: anomaly detection

BlazingText: text classification and word2vec

DeepAR: time series forecasting

■ BUILD (45 MIN)

Final preparation — no new building today. Focus entirely on the exam.

Make flashcards for every AWS ML service (physical or digital).

Take a second practice exam if available.

Book your exam for Day 104 or 105.

Passing score: 720/1000. Duration: 3 hours. Format: multiple choice and multiple response.

✓ DONE WHEN: Practice exam score above 75%. Exam booked. Every AWS service flashcard complete.

DAY 104 · AWS ML SPECIALTY EXAM

The certification milestone.

■ EXAM DAY

Sit the AWS Certified Machine Learning – Specialty exam.

3 hours. 65 questions. 720/1000 to pass.

If you pass: screenshot immediately. Add to GitHub README and LinkedIn.

If you need to retake: book again. Study your weakest domain.

→ aws.amazon.com/certification/certified-machine-learning-specialty/

■ AFTER THE EXAM (90 MIN)

Write your Phase 5 exam reflection in your log.

Update your GitHub README with Phase 5 section:

Fine-tuned model on HuggingFace Hub (link)

RAG app deployed (link)

AWS ML Specialty certification status

Phase 6 starts Day 106.

✓ DONE WHEN: Exam attempted. Result logged. GitHub updated. Phase 5 complete.

DAY 105 · PHASE 5 COMPLETE

You know how to make LLMs useful.

■ REVIEW (60 MIN)

Look at what you built in Phase 5:

Fine-tuned models using LoRA and QLoRA

RAG systems that answer questions from your documents

Production RAG app with file upload and memory

AWS services across the entire ML pipeline

The combination of fine-tuning and RAG covers 80% of enterprise AI applications.

■ PHASE 6 PREP (90 MIN) — Agentic AI setup

Phase 6 is about AI that acts, not just AI that answers.

pip install langchain-community langgraph

Read the LangChain agents tutorial.

Create an n8n account at n8n.io (free cloud version available).

Preview: tomorrow your AI will browse the web, write code, and manage files.

→ python.langchain.com/docs/tutorials/agents/

→ n8n.io

✓ DONE WHEN: Phase 5 complete. Phase 6 tools installed. Ready for agentic AI.

DAY 105 MILESTONE · AWS CERTIFIED MACHINE LEARNING — SPECIALTY MLS-C01

Your Fourth Credential — The Most Employable ML Cert on the Market

AWS ML Specialty validates your ability to design, implement, and maintain ML solutions on AWS. It is the certification that ML engineering job listings reference most often. You also have a fine-tuned model published on HuggingFace and a production RAG application deployed at a public URL. Phase 6 — AGENT — begins Day 106. You will build AI that acts autonomously.