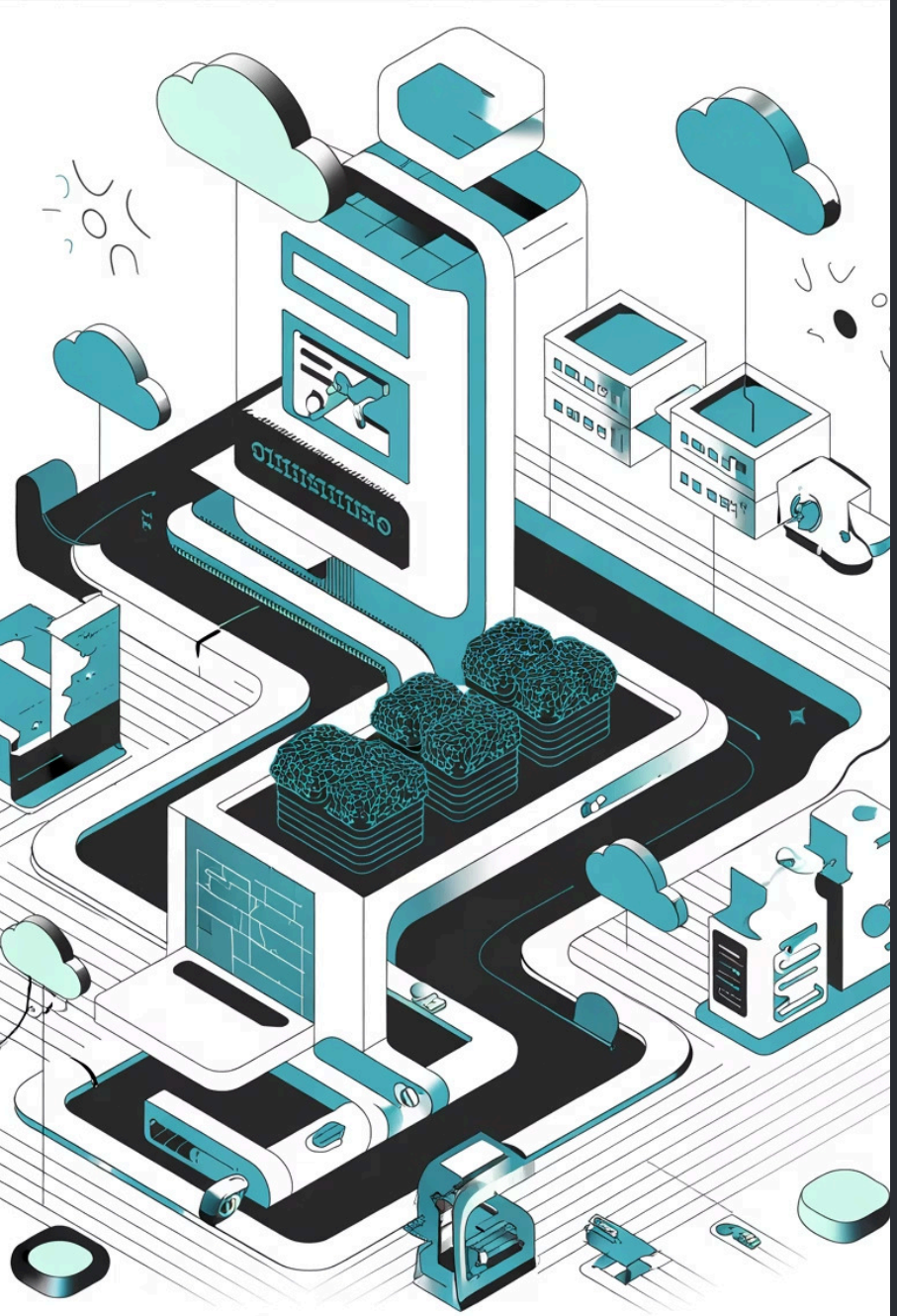


Deployment

Bringing Deep Learning Models to Life

Learn to deploy deep learning and RAG models into real-world environments using modern frameworks and services.

Made with GAMMA

Why Deployment Matters

Deployment transforms research artifacts into production systems. It bridges the gap between theoretical models and real-world applications, enabling organizations to automate workflows, reduce latency, and deliver AI capabilities to end-users at scale.

Bridge to Production

Connects research to real-world applications and business value.

Enable Automation

Powers integrated workflows and business process optimization.

Scale Access

Delivers AI capabilities to thousands of concurrent users globally.

Critical Requirements

Demands scalability, reliability, security, and monitoring.

Made with GAMMA

Model Serialization & Persistence

Serialization preserves trained models for later inference. TensorFlow and Keras provide multiple formats—each with trade-offs between size, speed, and compatibility. Proper versioning and compression strategies are essential for production environments.

Serialization Formats

- SavedModel: TensorFlow native format with full metadata
- HDF5 (.h5): Keras legacy format, compact and portable
- ONNX: Cross-framework standard for interoperability
- Pickle (.pkl): Python-specific, supports custom objects

Best Practices

- Use model.save() for comprehensive serialization
- Version models with timestamps or Git hashes
- Store backups in versioned artifact repositories
- Compress for efficient storage and transfer

Made with GAMMA

Serving Models with APIs

REST APIs provide stateless, scalable interfaces for model inference. TensorFlow Serving optimizes latency for production workloads, while FastAPI enables rapid development with automatic validation and documentation. Both support batching, versioning, and health monitoring.

1

Choose Framework

TensorFlow Serving for production scale; FastAPI for rapid iteration and flexibility.

2

Design Endpoints

Define REST routes with clear request/response schemas and input validation.

3

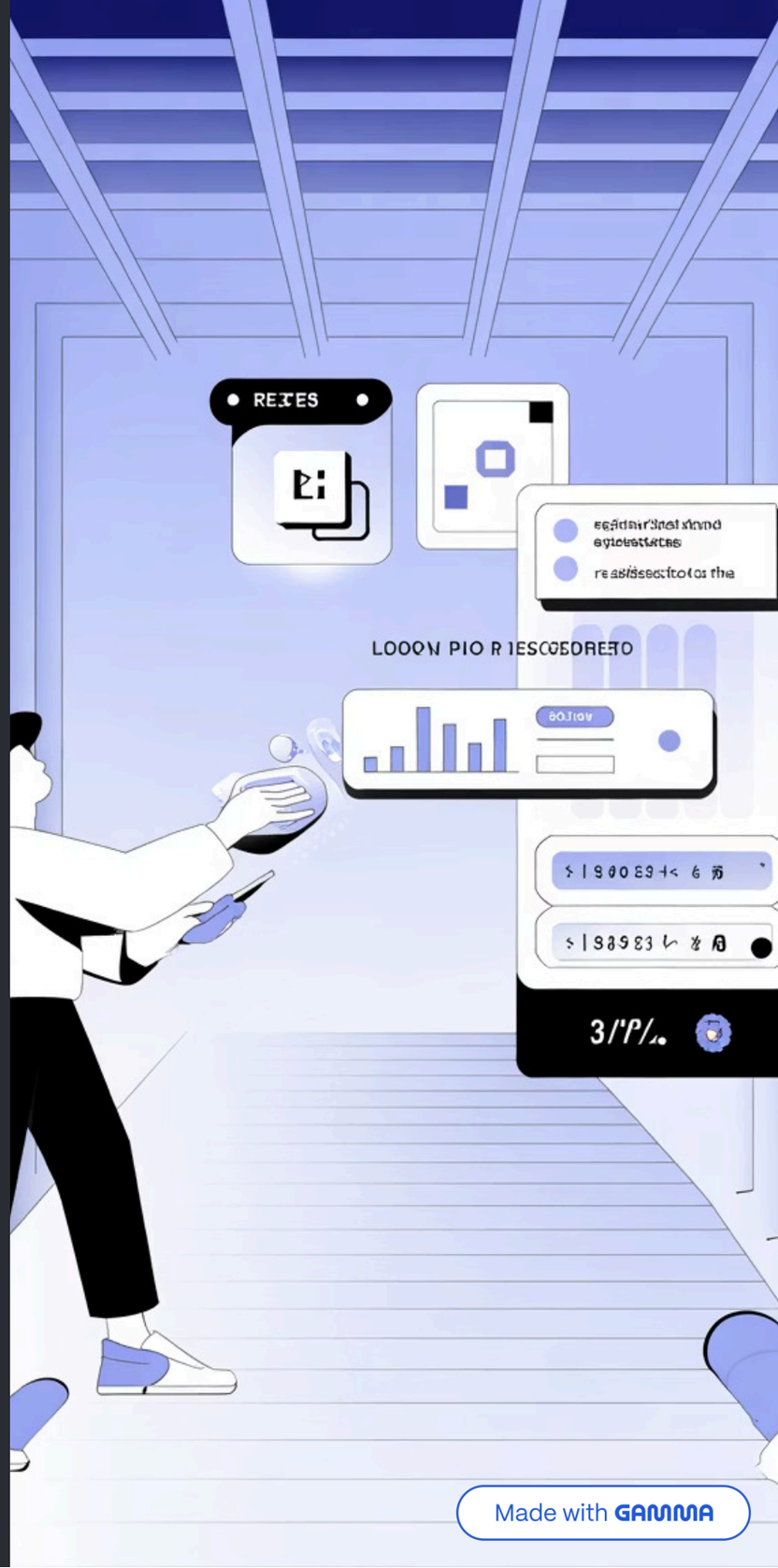
Handle Preprocessing

Embed normalization and feature engineering in the API layer for consistency.

4

Test Thoroughly

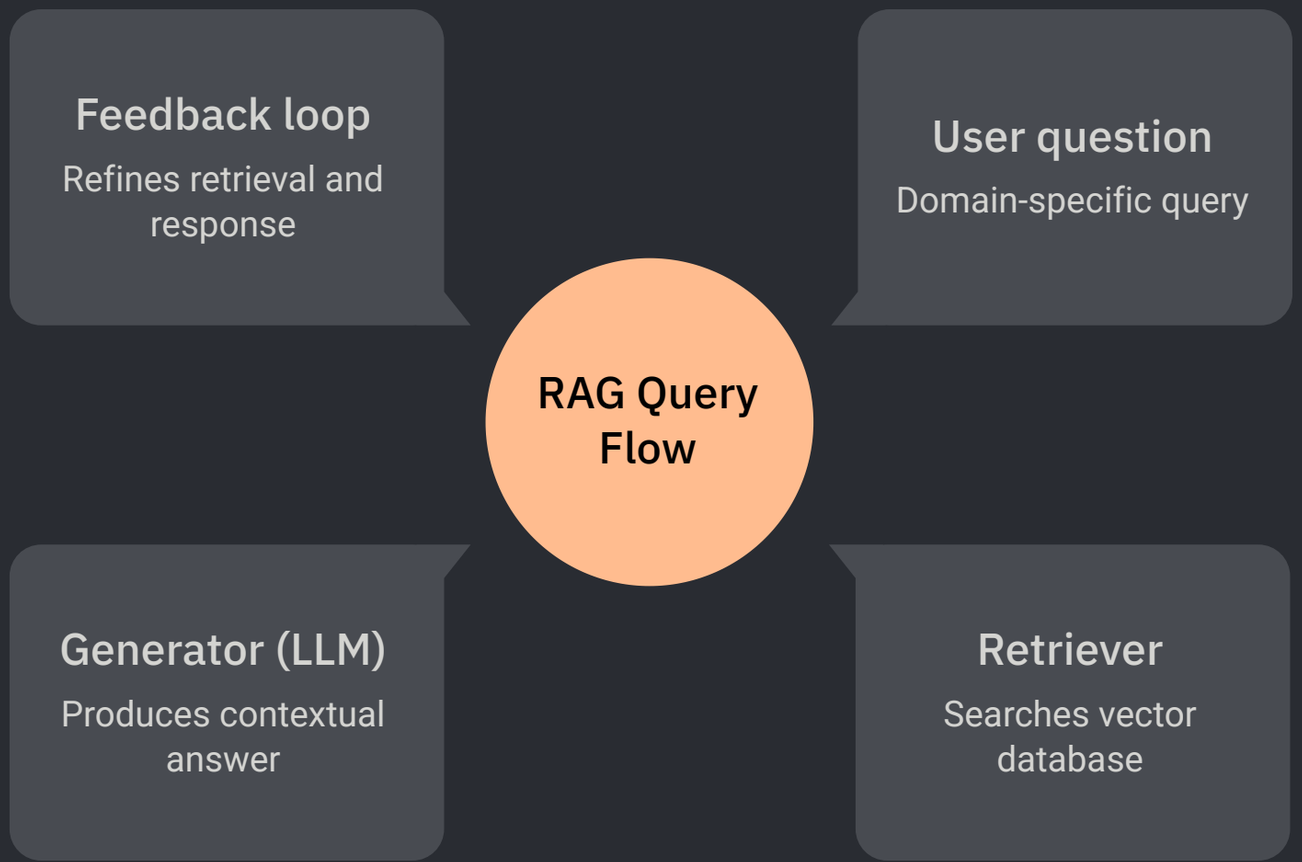
Use Postman, cURL, or pytest to validate endpoints before production deployment.



Made with GAMMA

Deploying NLP & RAG Pipelines

Retrieval-Augmented Generation (RAG) combines embedding-based document retrieval with language models to answer domain-specific questions. LangChain orchestrates these components, while vector databases enable efficient semantic search at scale.



LangChain Integration

Orchestrates retriever, embeddings, and LLM components in unified workflows.

Vector Databases

Store and search embeddings for efficient semantic retrieval (Pinecone, Weaviate, Milvus).

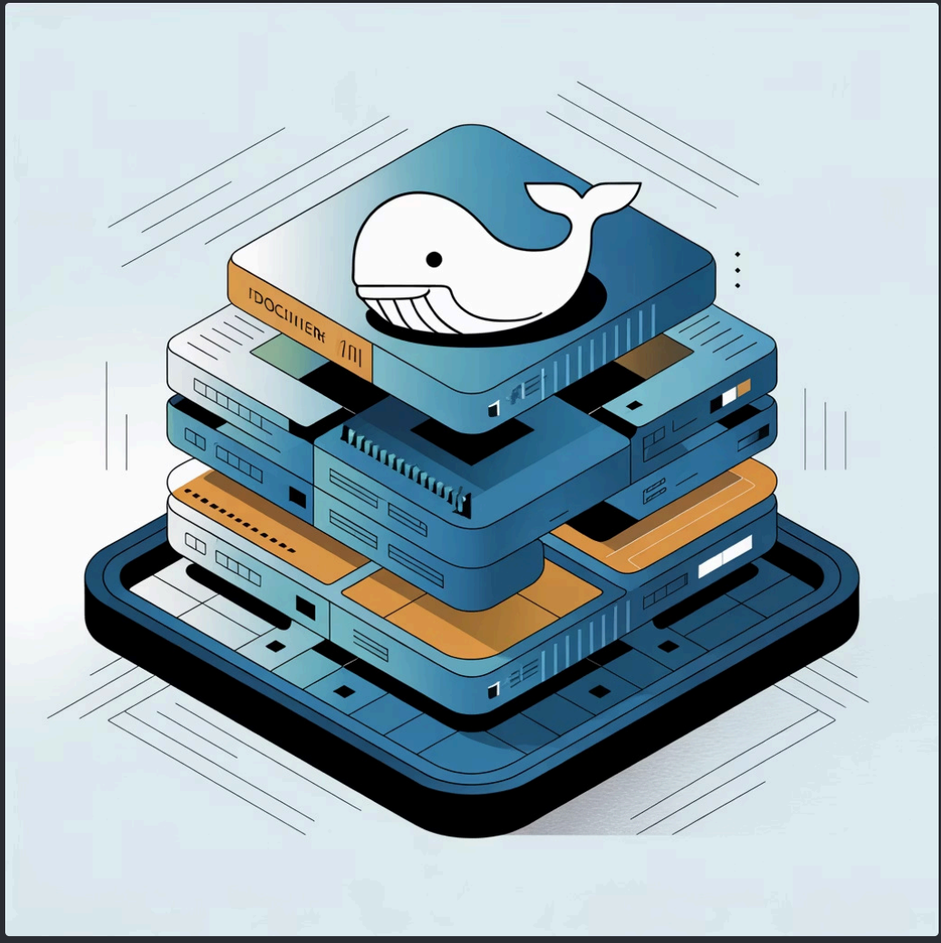
User Interfaces

Build interactive chatbots with Streamlit or Gradio for immediate user feedback.

Made with GAMMA

Containerization with Docker

Docker packages models, dependencies, and runtime into portable containers. This ensures consistency across development, testing, and production environments while simplifying deployment to cloud platforms and edge devices.



01

Write Dockerfile

Define base image, install dependencies, copy model and code.

02

Build Image

Use docker build to create reproducible container image with fixed versions.

03

Push Registry

Upload to Docker Hub, ECR, or private registry for team access.

04

Deploy Container

Run locally, on Kubernetes, or cloud services with minimal configuration.

Made with GAMMA

Real-Time Inference & Monitoring

Production systems require sub-millisecond latency, high throughput, and continuous monitoring. REST and WebSocket APIs support both request-response and streaming patterns. Observability tools track performance degradation and data drift in real time.

Latency Optimization

Minimize request time through batching, caching, and model quantization.

Integration Points

Connect mobile apps, web services, and microservices seamlessly via APIs.

Reliability

Implement circuit breakers, fallbacks, and graceful degradation for robustness.

Observability

Monitor throughput, latency, error rates, and data drift metrics continuously.

Made with GAMMA

Hands-On Lab: Deploy a Service

This lab guides you through containerizing and deploying a sentiment analysis model as a FastAPI service. You will build a Dockerfile, test locally, and optionally deploy to Streamlit Cloud or Hugging Face Spaces for public access.

Build FastAPI App

Create endpoints for model inference with input validation and error handling.

Containerize

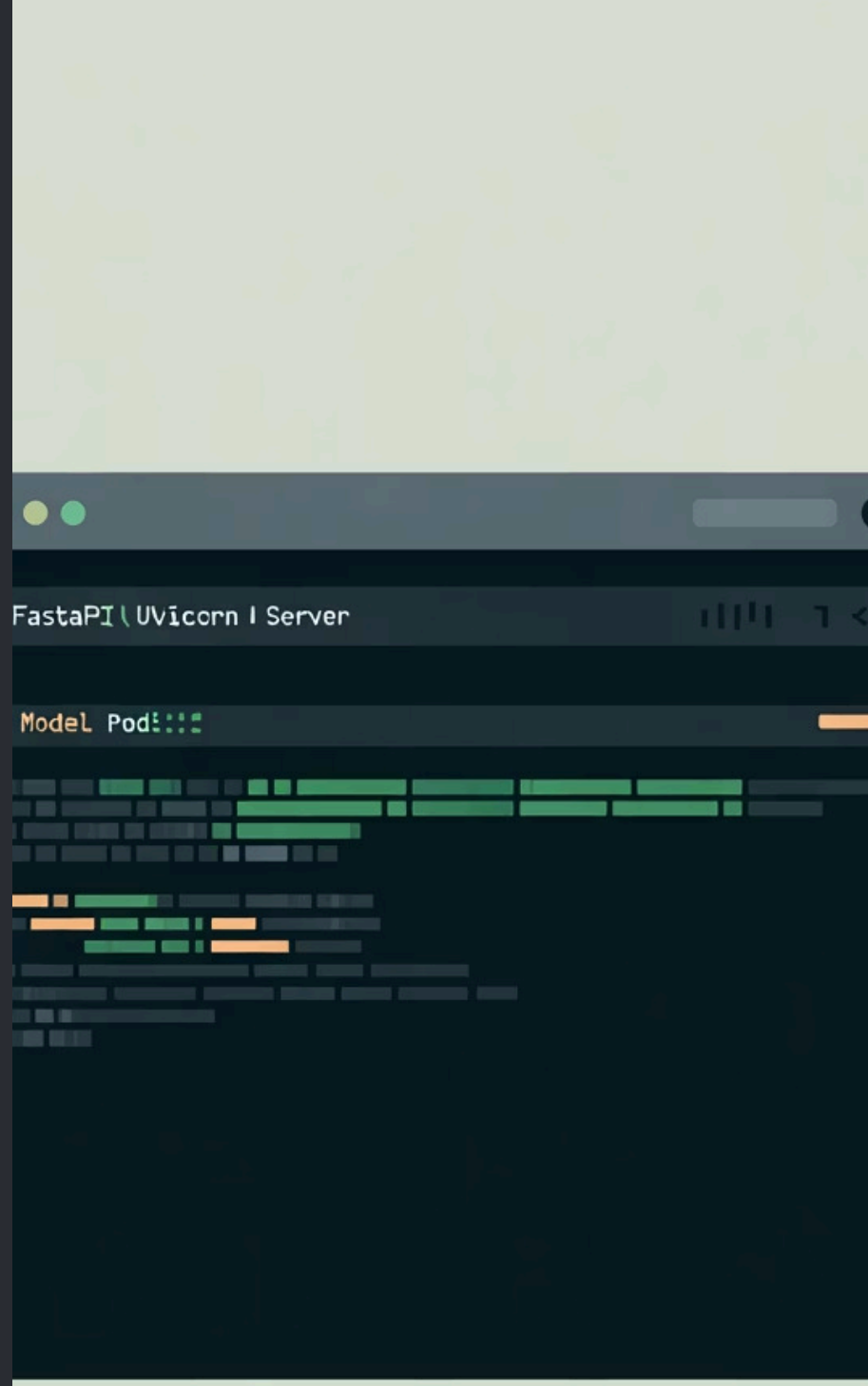
Write Dockerfile with dependencies, copy model weights, expose port 8000.

Test Locally

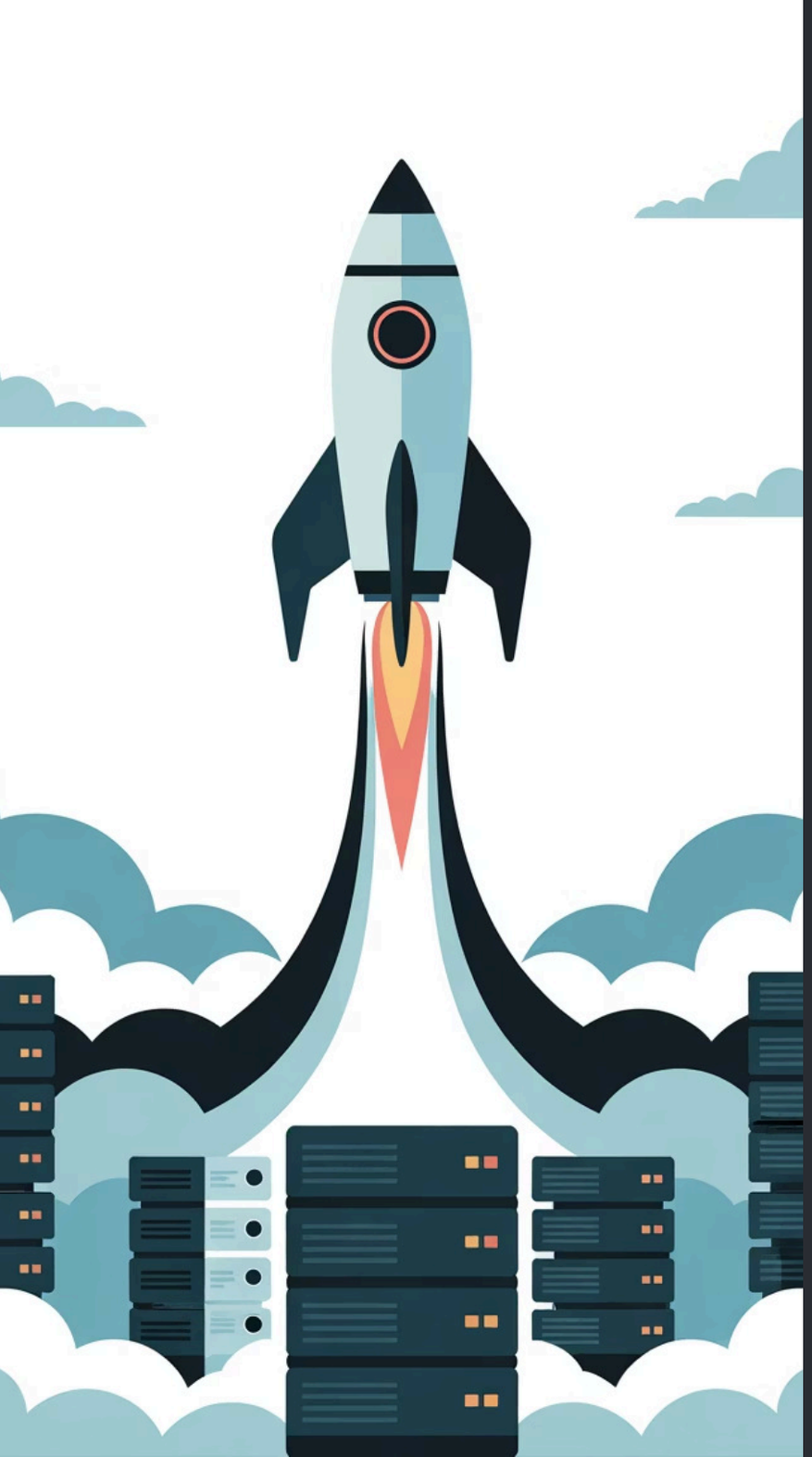
Run container locally with docker run, test endpoints using cURL or Postman.

Deploy & Share

Deploy to cloud service or Streamlit Cloud. Share URL with peers and gather feedback.



Made with GAMMA



Key Takeaways & Next Steps

Deployment transforms models from academic exercises into production systems serving real users. Mastering serialization, APIs, containerization, and monitoring unlocks the full potential of deep learning in industry.

Models → Products

Deployment converts trained models into customer-facing applications.

APIs as Bridge

REST and streaming APIs integrate AI into existing systems and workflows.

Docker Guarantees

Containerization ensures reproducibility and consistency across environments.

Scale with Confidence

Cloud tools and orchestration platforms enable deployment at any scale.

Made with GAMMA

Ready to Deploy

You now have the foundational knowledge to take trained deep learning models into production. Start with a simple FastAPI service, containerize it with Docker, and progressively adopt cloud deployment patterns as your systems scale. The deployment skills you develop here will distinguish you as a full-stack ML engineer.

Recommended Resources

- TensorFlow Serving documentation and tutorials
- FastAPI official guide and examples
- Docker for AI/ML applications best practices
- LangChain documentation for RAG systems

Made with GAMMA