RADemics

Model Deployment and API Integration Using FastAPI and Docker for AI Services

Ashwini N. Tikle, Pranali L. Katore, S. Senthil Kumar
PRIYADARSHINI BHAGWATI COLLEGE OF ENGINEERING, KARPAGAM ACADEMY OF HIGHER EDUCATION

Model Deployment and API Integration Using FastAPI and Docker for AI Services

[1]Ashwini N. Tikle, Assistant Professor, Computer Science and Engineering, Priyadarshini Bhagwati College of Engineering, Nagpur, Mobile number: 824 800 2831, Mail id: ashwinitikle54@gmail.com.

[2]Pranali L. Katore, Assistant Professor, Computer Science and Engineering, Priyadarshini Bhagwati College of Engineering, Nagpur, Mobile number: 824 800 2831, Mail id: urbana23@gmail.com.

[3]S. Senthil Kumar, Assistant Professor, Artificial Intelligence and Data Science, Karpagam Academy of Higher Education, Deemed to be University, Coimbatore. Mail id: kumars12@gmail.com

Abstract

The deployment of artificial intelligence (AI) models into production environments presents significant challenges related to scalability, maintainability, and real-time service delivery. Traditional deployment methods often lack flexibility, leading to issues in versioning, resource isolation, and performance consistency across heterogeneous systems. This book chapter presents a comprehensive approach to AI model deployment by leveraging FastAPI, a modern asynchronous web framework, in conjunction with Docker, a widely adopted containerization platform. By encapsulating model logic within containerized microservices and exposing inference endpoints through RESTful APIs, the proposed methodology facilitates modular, reproducible, and cloud-agnostic deployment workflows. The chapter explores architectural principles, implementation strategies, and operational best practices for deploying AI services using FastAPI and Docker. It further addresses critical concerns such as API security, automated documentation generation using OpenAPI and Swagger, performance monitoring, and threat detection in production-grade environments. Real-world deployment scenarios and testing frameworks are discussed to validate the efficiency and robustness of the proposed model-serving architecture. Emphasis was placed on integrating model services within CI/CD pipelines to enable seamless updates and scalable service delivery. This contribution fills a significant gap in AI operationalization by providing a standardized, developer-friendly, and enterprise-ready framework for model deployment. The chapter offers valuable insights for researchers, developers, and system architects seeking to enhance the reliability, observability, and interoperability of AI-powered services in production settings.

Keywords: AI Model Deployment, FastAPI, Docker, Microservices, API Security, Continuous Integration.

Introduction

The practical application of artificial intelligence (AI) and machine learning (ML) models was no longer confined to research laboratories; instead, these models are now actively integrated into

large-scale production systems across diverse domains such as healthcare, finance, e-commerce, and autonomous systems [1]. While model training and evaluation remain integral to the development cycle, the deployment phase was increasingly recognized as a vital aspect of operationalizing AI [2]. A significant barrier to successful deployment lies in transforming trained models into reliable, maintainable, and scalable services capable of serving inference in real-time or batch-processing contexts [3], [4]. Without appropriate deployment strategies, even the most accurate models risk becoming obsolete due to inefficiencies in service delivery, integration failure, or inability to handle production-grade workloads [5].

Traditional deployment methods typically involve embedding models into monolithic systems or using generic web frameworks lacking support for concurrency and real-time responsiveness [6]. These approaches are often associated with challenges such as poor scalability, high latency, fragile dependency management, and manual configuration overhead [7]. Inconsistencies between development and production environments frequently result in software defects that are difficult to diagnose and resolve [8]. In response to these challenges, containerization and microservice-based architectures have emerged as transformative technologies that decouple model logic from the core application, streamline deployment, and enable elastic scalability [9]. Such architectural shifts are critical to enabling AI solutions that can evolve rapidly while maintaining robust performance and operational integrity [10].

FastAPI, a modern high-performance web framework for building APIs with Python, offers a compelling solution for exposing AI models as RESTful microservices [11]. It supports asynchronous programming natively, allowing concurrent request processing with minimal latency, which was essential for models serving large volumes of inference requests [12]. In addition, FastAPI leverages Python type hints to provide automatic request validation, response schema generation, and interactive documentation through OpenAPI and Swagger [13]. These capabilities enable the rapid development and testing of model APIs while maintaining code clarity and robustness [14]. FastAPI's modular structure facilitates integration with other microservices and databases, making it a suitable foundation for AI-driven systems that require flexibility, interoperability, and resilience in deployment [15].