Docker AI

From Prototype to CI/CD Pipeline

There’s no shortage of press and hype about AI. Generative AI in particular has rapidly caught the attention of everyone in the development community as well the general public due to its accessibility via ChatGPT. The number of generative AI tutorials, how-to videos, podcasts, books, and online courses for setting up your first generative AI chatbot, AI/ML prototype application, or agent for self-learning and exploration seems to grow each day.

Hacking AI

With easy access to shareable, interactive prototyping environments like Jupyter Notebooks, local and cloud sandbox environments, GitHub code repositories filled with sample and tutorial code, and highly popular transformer models free on sites like huggingface.co, developers quickly began learning and experimenting with new and emerging generative AI technologies. Many have already learned machine language technologies using popular open source and free-to-use libraries and platforms like PyTorch, Google’s TensorFlow, NumPy, and JAX, just to name a few.

During this experimental generative AI phase, developers prototype with multiple large language models (LLMs), vector databases, ML libraries, datasets and much more. But that introduces a new challenge, as local machines and sandboxes quickly become cluttered with software, package managers, code, and base images that quickly become unwieldy.

Docker is the ideal technology to containerize developer and prototyping environments, tailoring the needs of each innovative idea with code, supporting software, configurations, and security needed for each particular development pursuit. Data scientists have long enjoyed Docker's ability to maintain precise and consistent environments that also make data science and machine learning work portable from project setup to production.

From AI Prototype to CI/CD in 5 Easy Steps

01 Establish precise and consistent environment →

02 Utilize internal container image and configuration standards →

03 Leverage generative AI stacks to streamline CI/CD integration →

04 Test your generative AI containers into CI/CD pipelines early →

05 Apply security and DevSecOps practices to secure AI containers into workflow pipeline

Developers are innovating, coming up with new ideas, and creating prototypes they may quickly use as part of larger applications and in production. The question is, how do we move generative AI apps from ideation to mainstream development, integrating generative AI apps and supporting technologies as part of a larger flow of work moving through DevOps pipelines and supporting toolchains? And how can Docker containers serve as an integral aspect of this revolutionary transformation?

The New GenAI Stack

01 Pre-configured LLMs

02 Ollama model management

03 Neo4j default database

04 Neo4j knowledge graphs

05 LangChain orchestration

06 Comprehensive support

Learn More

GenAI Stack on GitHub

AI is More Than Code

AI projects bring with them more than typical code and data. AI projects, particularly those involving generative AI's LLMs and vector databases, require frequent tuning and updates due to the continuous influx of new data. Docker enables rapid prototyping by utilizing containers that already have all of the libraries and dependencies installed. Early integration of containerization technologies with Docker can significantly expedite and streamline this process, offering low- or no-friction deployment, minimizing disruption to existing DevOps and CI/CD workflows. Docker simplifies moving containerized generative AI projects from laptops to public clouds like AWS, Azure, and GCP.

During the initial development phase of generative AI applications, Docker presents itself as a foundation for developing and transporting AI work, providing isolation and portability across dev, test, and production environments and their variations. Docker containerized environments are conducive to securing and transporting AI applications using combinations and versions of AI libraries, frameworks, LLMs, vector databases, data, and dependencies. Docker's containerized, isolated environment ensures seamless introduction of AI applications with various AI and ML libraries and software so AI applications can easily coexist and operate, minimizing incompatibilities, conflicts, and broader disruptions. Generative AI applications, especially those using LLMs and vector databases, often require access to and interaction with massive and continuously changing datasets. Docker's benefits are even greater as these applications integrate with DevOps CI/CD workflows.

Docker encapsulates the generative AI application along with its myriad dependencies into an immutable, reliable artifact. This artifact, once created, can be consistently and reliably deployed across a mix of environments. This consistent encapsulation guarantees a uniform level of behavior for generative AI applications across diverse environments.

Top 10 Docker Hub Image Downloads for AI in 2023

01. library/python	**06.** jypter/base-notebook
02. tensorflow/serving	**07.** pytorch/pytorch
03. jupyter/datascience-notebook	**08.** jupyter/all-spark-notebook
04. tensorflow/tensorflow	**09.** jupyter/tensorflow-notebook
05. tensorflow/build	**10.** library/spark

16,676,058

Total downloads within 6 months

A Day in the Life of an AI App

In the subsequent phase, where the generative AI application is meticulously prepared for testing, Docker facilitates the creation and deployment of scalable testing environments capable of closely mirroring the configurations and nuances of production settings. Automated test frameworks, essential to any robust CI/CD pipeline, can seamlessly leverage Docker containers and execute a suite of rigorous tests. This level of automation and efficiency in the testing process is pivotal for ensuring that generative AI applications, which inherently deal with large and constantly evolving data, are rigorously validated and vetted before being deployed into a live production environment.

In the post-deployment phase, generative AI applications enter a continuous cycle of improvement, refinement and evolution. In this phase, Docker facilitates a seamless and frictionless update process for these applications. Docker's architecture and workflow allow for easy and efficient incorporation of new data into the AI application. This supports the retraining of AI models and facilitates the redeployment of updated applications with less effort and downtime, helping generative AI applications maintain their accuracy, reliability, and effectiveness over time.

Docker's containerization technology also addresses and mitigates variations in the scalability and resource allocation requirements of AI applications. AI and ML applications have diverse and fluctuating computational demands and varying computational requirements in production and during the model training inference use cases.

Wrapping It Up — AI, ML, and GenAI in Containers

In summary, Docker's containerization technology stands out as a robust, reliable mechanism that reduces friction and aids the flow of AI, ML, and generative AI development projects into existing DevOps and CI/CD workflows. From the initial stages of development through to continuous updates and improvements, Docker provides a smooth, reliable, and efficient workflow for developers and IT professionals working on generative AI applications. Through its architecture and functionality, Docker allows developers to focus on refining and improving AI applications without having to contend with environmental inconsistencies or deployment-related challenges and issues.

Simplify AI development with automated guidance from Docker.
Get started today!