Cloud Native Artificial Intelligence

What is cloud native intelligence

- Cloud-native AI development refers to the practice of building artificial intelligence (AI) applications and services using cloud-native principles and technologies. In essence, it involves developing AI solutions that are designed to fully leverage the capabilities of cloud computing infrastructure.

Key aspects of cloud-native AI development

- Scalability: Cloud-native AI applications are built to scale effortlessly based on demand. Cloud platforms provide resources like computing power, storage, and networking that can be dynamically adjusted to handle varying workloads.
- Microservices Architecture: Cloud-native AI applications are often structured as a collection of loosely coupled microservices. Each microservice is responsible for a specific function or component of the AI system, allowing for easier development, deployment, and maintenance.
- Containerization: Containers, such as those managed by Docker or Kubernetes, are used to package and deploy cloud-native AI applications. Containerization enables consistency across different environments and facilitates portability, making it easier to deploy AI solutions across various cloud providers or on-premises infrastructure.

- **Serverless Computing**:

Serverless computing, also known as Function as a Service (FaaS), is often utilized in cloud-native AI development. Serverless platforms abstract away infrastructure management tasks, allowing developers to focus solely on writing code. This model can be particularly beneficial for running AI inference tasks on-demand without the need to provision or manage servers.

- **Data Management**:

Cloud-native AI applications rely on cloud-based data storage and processing services for managing large volumes of data. Technologies such as cloud-based data lakes, data warehouses, and real-time data streaming platforms are commonly used to support AI workflows.

- **Agility:**

Cloud-native development practices promote faster development cycles and easier deployment of AI models.

- **Automation:**

Cloud platforms can automate tasks like model training and deployment, freeing up developers for more strategic work.

- **Resilience:**

Cloud-based infrastructure offers built-in redundancy and fault tolerance, making your AI applications more reliable.

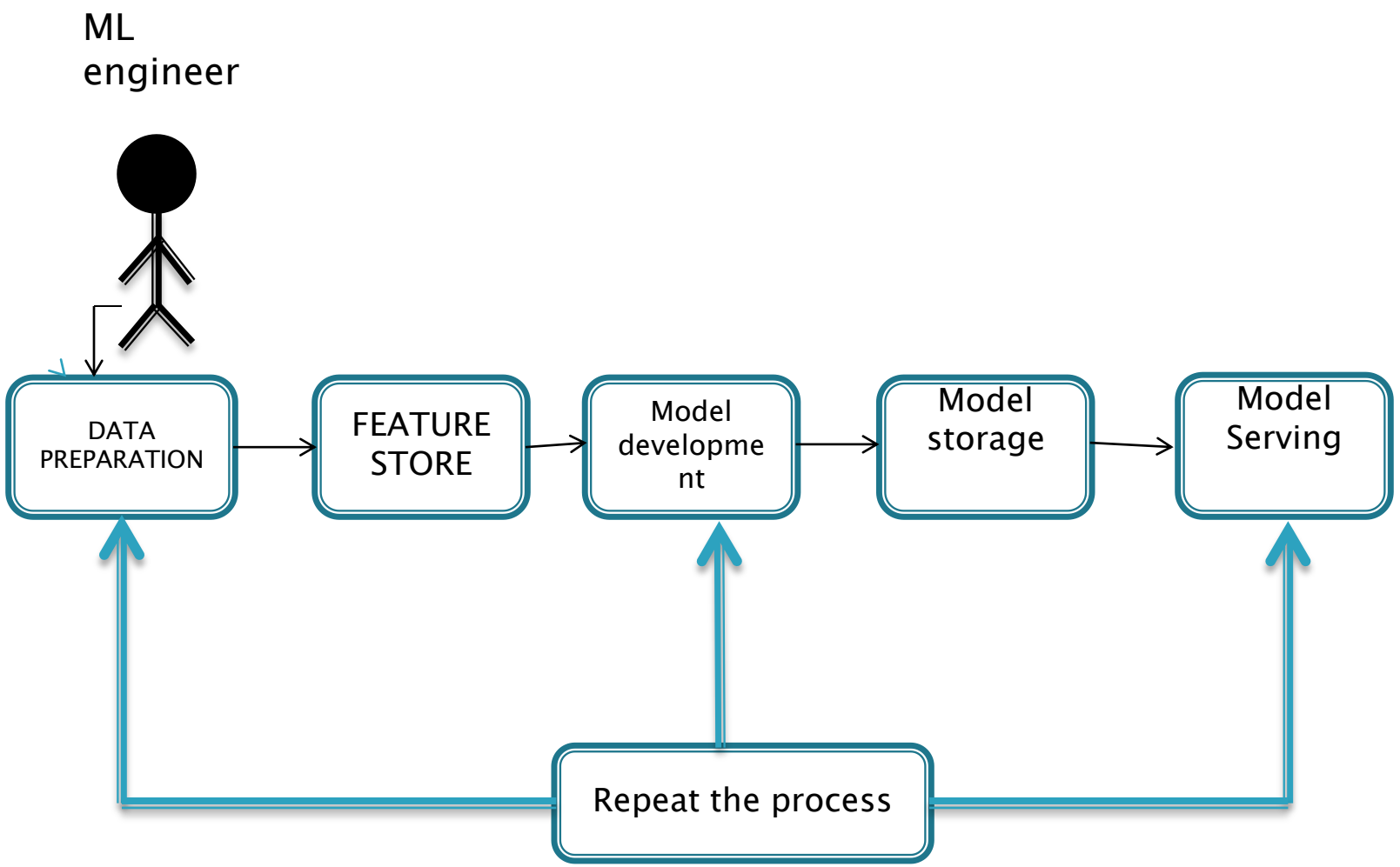
- **Monitoring and Observability:**

Cloud-native AI solutions incorporate robust monitoring and observability capabilities to track the performance, health, and usage of AI models and services. Logging, metrics, and tracing tools are employed to gain insights into system behavior and diagnose issues quickly.

CHALLENGES FOR CLOUD NATIVE AI

- t’s important to note that CNAI challenges will vary between the different personas.26 And, while Cloud
- Native’s flexible, scalable platform is a promising fit for AI workloads, AI’s scale and latency needs pose
- challenges and expose gaps in CN technologies while also presenting opportunities.

- We tease these out
- in the context of an end-to-end ML pipeline also referred to in the literature as MLOps Issues with the
- traditional trade-offs of time and space, parallelism, and synchronization all surface, exposing ease-of-use gaps.
- To summarize, the ML Lifecycle looks as follows:


ML
engineer
DATA
PREPARATION
FEATURE
STORE
Model
developme
nt
Model
storage
Model
Serving
Repeat the process

The typical ML pipeline is comprised of:

- Data Preparation (collection, cleaning/pre-processing, feature engineering)
- Model Training (model selection, architecture, hyperparameter tuning)
- CI/CD, Model Registry (storage)
- Model Serving
- Observability (usage load, model drift, security)

Data Preparation

- As the first phase in an AI/ML pipeline, data preparation can present various challenges. These can be
- broadly grouped into three main categories: managing large data sizes, ensuring data synchronization during development and deployment, and adhering to data governance policies.

- **Data Size:**

Distributed Cloud Native computing and efficient data movement and storage become essential to bridge the gap between these computational demands and hardware capabilities

- **Data Synchronization:**

Data may need to be sourced from multiple disparate locations in different formats; the developer and production environments, more often than not, are different, and all this is in addition to handling the increased complexity arising from distributed computing, such as partitioning and synchronization.

- **Data Governance:**

Data governance refers to the set of roles, processes, policies and tools which ensure proper data quality throughout the data lifecycle and proper data usage across an organization.

To establish a robust data governance framework, organizations often rely on four key pillars: **Data quality, data stewardship, data protection and compliance, and data management.**

CI/CD, Model Registry (storage)

- An ML model registry serves as a centralized repository, enabling effective model management and documentation. It allows for clear naming conventions, comprehensive metadata, and improved collaboration between data scientists and operations teams, ensuring smooth deployment and utilization of trained models

Model Training

Model training data volumes have risen exponentially, resulting in a need for distributed processing and accelerators to achieve even more parallelism. Further training is an iterative multi-step process, which makes scaling a complex multi-component coordinated task.

Model Serving

- Model serving refers to the process of deploying trained machine learning models into production environments, where they can receive input data, make predictions or inferences, and provide output to end-users or other systems. Model serving ensures that the trained models are operationalized and can be utilized by real-world applications to perform tasks such as classification, regression, clustering, or natural language processing.

PATH FORWARD WITH CLOUD NATIVE AI

- In this section provides a forward looking approach to taking the initiative to implement CNAI. We begin with recommendations (or actions), then enumerate existing yet evolving solutions (i.e., CNAI software), and finally consider opportunities for further development.

Recommendations

- Flexibility:

Sometimes, the variety of options regarding AI can become overwhelming. Fortunately, thanks to many, popular tools and techniques remain valid in this new world. From REST interfaces for interface to cloud based resources and services, CN technologies work well today and will continue to work well as new offerings evolve.

Sustainability

Improving the accountability of AI workload environmental impact is crucial for ecological sustainability, particularly in the cloud native landscape. This can be achieved by supporting projects, methodologies, and taxonomy that help clarify, classify, and catalyze AI workload on ecological sustainability.

Custom Platform Dependencies

We recommend ensuring the Cloud Native environment has the required GPU drivers and supports GPU acceleration for AI workloads. This is crucial as AI applications often depend on specific frameworks and library versions that may not be easily accessible or compatible with standard container images. This will help with the challenge of having various vendors and GPU architectures.

Industry Acceptance of Terminology

- As AI becomes ubiquitous, it becomes increasingly complex in some dimensions but simpler in others.
- For example, terminology evolves, providing businesses with more effortless conversations about AI (e.g., terms such as “repurpose” to reuse existing content). This also applies to more technical terms, such as RAG, Reason, and Refinement.